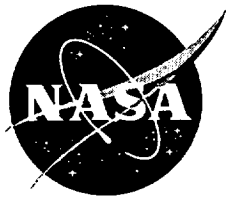
NASA/TM—1998-112225



Toward a More Robust Pruning Procedure for MLP Networks

Slawomir W. Stepniewski
Recom Technologies, Inc., Ames Research Center, Moffett Field, California

Charles C. Jorgensen
Ames Research Center, Moffett Field, California

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

April 1998

Available from:

NASA Center for AeroSpace Information
800 Elkridge Landing Road
Linthicum Heights, MD 21090-2934
Price Code: A03

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Price Code: A03

Abstract

Choosing a proper neural network architecture is a problem of great practical importance. Smaller models mean not only simpler designs but also lower variance for parameter estimation and network prediction. The widespread utilization of neural networks in modeling highlights an issue in human factors. The procedure of building neural models should find an appropriate level of model complexity in a more or less automatic fashion to make it less prone to human subjectivity. In this paper we present a Singular Value Decomposition based node elimination technique and enhanced implementation of the Optimal Brain Surgeon algorithm. Combining both methods creates a powerful pruning engine that can be used for tuning feed-forward connectionist models. The performance of the proposed method is demonstrated by adjusting the structure of a multi-input multi-output model used to calibrate a six-component wind tunnel strain gage.

1. Introduction

In nonlinear regression problems employing feed-forward neural networks, it is important to reduce model complexity in order to decrease the parameter variance. This may be at the expense of a slower growing model bias. It is commonly argued in the neural network community that neural network size reduction may improve model generalization, i.e., its response accuracy to unseen stimulations.

Optimal Brain Surgeon (OBS), proposed by Hassibi and Stork (ref. 1), is an excellent framework for removing redundant weights. It relies on a local approximation of the learning error increase when the current weight settings $\mathbf{w}$ are changed by some $\Delta\mathbf{w}$:

$$\Delta E \approx \frac{1}{2}\Delta\mathbf{w}^{\mathrm{T}}\mathbf{H}\Delta\mathbf{w} \qquad (1)$$

where $\mathbf{H} = \partial^2 E/\partial\,\mathbf{w}^2$ is a symmetric matrix of second order derivatives (Hessian) or its suitable approximation. Elements of $\mathbf{H}$ are constant for models that are linear in parameters. For feed-forward neural networks considered here we assume the elements are slowly changing around local minima. Imposing a single equality constraint, $\Delta w_i + w_i = 0$, which forces the i-th weight to vanish, an estimation of the minimal impact of removing a single connection $S_i = \min_{\Delta\mathbf{w}}(\Delta E)$ may be found. S_i is called a weight saliency coefficient. Using the analytical approach of Lagrange multipliers to find the minimum of this quadratic programming problem, subject to the above constraint, a fairly simple solution is obtained (ref. 1):

$$S_i = \frac{1}{2}\frac{w_i^2}{\mathbf{e}_i^{\mathrm{T}}\mathbf{H}^{-1}\mathbf{e}_i}, \quad \Delta\mathbf{w} = -\frac{w_i}{\mathbf{e}_i^{\mathrm{T}}\mathbf{H}^{-1}\mathbf{e}_i}\mathbf{H}^{-1}\mathbf{e}_i \qquad (2)$$

where $\mathbf{e}_i$ is the i-th unit vector. The saliency coefficient S_i is calculated for all the weights, and the connection having smallest S_i is selected for elimination using the appropriate weight update $\Delta\mathbf{w}$. The paradigm based on equation (1) has been extended to approximate the removal of some other meaningful combinations of connections, e.g., entire unit deletion (ref. 2). Pedersen (ref. 3) proposes another extension that estimates changes in the network generalization rather than the learning error.

The approximation (eq. 1) comes from the expansion of the Taylor series in which all terms except the second order are dropped. It is usually a valid assumption that the first-order term (gradient) is negligible for locally identified neural models. Besides this assumption, the matrix $\mathbf{H}$ in equation (1) should be conditioned so that it is positive definite. This recommendation comes from the observation that for a reasonably trained network and large changes to the weight vector we should not expect the error to decrease. For the indefinite Hessian, a locally negative curvature will suggest, however, that the error could be substantially lowered given sufficiently large steps. In such a

case, we can discard too "optimistic," negative saliency values or try to modify the Hessian matrix to make it positive definite in the first place. This can be achieved, for example, by using methods reviewed in reference 4.

For an error function of the form

$$E(\mathrm{w}) = \frac{1}{2}\sum_j s_j^2(\mathrm{w}) \tag{3}$$

an approximation of the Hessian matrix that is guaranteed to be positive semidefinite (and works well in practice) could be obtained from $\mathbf{H} = \mathbf{J}^{\mathrm{T}}\mathbf{J}$, where $\mathbf{J} = [J_{ji}] = \partial s_j / \partial w_i$ is the Jacobian matrix. Instead of inverting this matrix, the positive definite inverse $\mathbf{H}^{-1} = \left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \mu\mathbf{I}\right)^{-1}$, $\mu > 0$ (e.g., $\mu = 10^{-3}$), may be calculated using a recursive formula similar to the RLS (recursive least squares method) with infinite memory, i.e. (ref. 5)

$$\mathbf{H}_0^{-1} = (1/\mu)\mathbf{I}$$

$$\mathbf{H}_{j+1}^{-1} = \mathbf{H}_j^{-1} - \frac{1}{1 + \mathbf{J}_{j\bullet}^{\mathrm{T}}\mathbf{H}_j^{-1}\mathbf{J}_{j\bullet}}\left[\mathbf{H}_j^{-1}\mathbf{J}_{j\bullet}\right]\left[\mathbf{H}_j^{-1}\mathbf{J}_{j\bullet}\right]^{\mathrm{T}} \qquad j = 1,\ldots,N\times P \tag{4}$$

where $\mathbf{J}_{j\bullet}$ is the j-th row of the Jacobian matrix (treated as a column vector), N is the size of the training sample, and P is the number of network outputs. Hassibi and Stork (ref. 1) suggested using the gradient, $\mathbf{g}_j = \partial E_j / \partial \mathbf{w}$ ($j = 1,\ldots,N$) in place of $\mathbf{J}_{j\bullet}$. This in fact produces a formally different criteria for connection pruning that considers, under certain simplifying conditions, the ratio of the squared weight value to the weight variance estimation (refs. 6 and 7).

2. Efficient OBS Implementation

In the original OBS algorithm $\mathbf{H}^{-1}$ is evaluated after one or a fixed number of weights are removed. When more than one weight is removed between independent inverse Hessian evaluations, the current $\mathbf{H}^{-1}$ should be corrected to reflect the exclusion of the i-th weight (i-th row and column) from $\mathbf{H}$. The corresponding operation on $\mathbf{H}^{-1}$ is (ref. 3)

$$\mathbf{H}_{k+1}^{-1} = \mathbf{H}_k^{-1} - \frac{1}{\mathbf{e}_i^{\mathrm{T}}\mathbf{H}_k^{-1}\mathbf{e}_i}\left(\mathbf{H}_k^{-1}\mathbf{e}_i\right)\left(\mathbf{H}_k^{-1}\mathbf{e}_i\right)^{\mathrm{T}} \tag{5}$$

followed by the deletion of the zeroed i-th column and row in $\mathbf{H}_{k+1}^{-1}$ (matrix dimensions are reduced by 1).

Frequent evaluation of $\mathbf{H}^{-1}$, especially after removing every individual weight, may be unnecessary; this tends to make the pruning procedure rather inefficient. On the other hand, it is hard to determine in advance the number of weights that could be removed between subsequent evaluations of $\mathbf{H}^{-1}$. Certainly, in the beginning the pruning process will tend to eliminate more weights than at later stages. It is the pruning algorithm itself which should decide how many weights could be removed and when to retrain the network and evaluate a new approximation of $\mathbf{H}^{-1}$.

Equation (5) does not take into account that the remaining weights are modified by the vector $\Delta\mathbf{w}$. As pruning progresses, larger weight changes may violate the assumption of a constant or slowly changing Hessian. In such a case it may be worthwhile to correct $\mathbf{H}^{-1}$ by applying the Broyden family update (ref. 8), e.g., the well known BFGS (Broyden-Fletcher-Goldfarb-Shanno) formula. In the later stages of the pruning process, the BFGS update typically contributes to a higher number of weights being removed and/or a smaller number of overall iterations (new $\mathbf{H}^{-1}$ evaluations).

A reliable pruning algorithm should exhibit limited trust in the estimations provided by equation (1). Even in the early stages of pruning, it could happen that some values of S_i may be incorrect (there exists another weight producing a smaller error increase) or $\Delta\mathbf{w}$ may be inaccurate, even erroneous (a few training cycles are able to compensate connection removal that $\Delta\mathbf{w}$ cannot achieve).

Application of this rather straightforward approach to the construction of the OBS based pruning routine produces an algorithm that outperforms the original one in both speed and efficiency. The following steps outline an efficient modification to the basic OBS paradigm:

1. Calculate a positive definite approximation of $\mathbf{H}^{-1}$; set the number of failures to zero (the number of consecutive events when the training error cannot be reduced below a certain threshold).

2. Calculate new saliency values based on $\mathbf{H}^{-1}$.

3. Try to delete the feasible weight with the smallest salience using the OBS update (eq. 2) only.

4. If the error increase rate for the new weights is below a certain threshold (e.g., 3%) null the number of failures, correct $\mathbf{H}^{-1}$ for the deleted variable (eq. 5), optionally update $\mathbf{H}^{-1}$ using the BFGS formula, and then go to step 2. If the error is above the threshold, increment the number of failures, remove the current salience from consideration, and record which weight caused the smallest increase of error in the recent series of failures.

5. If the number of failures reaches a limit, attempt to remove the weight which most recently caused the smallest increase in error; use a regular OBS update followed by a short retraining. If no success was achieved since the last extensive retraining abort the pruning procedure.

6. If the number of failures exceeds the limit, perform an extensive network training.

7. Go to step 1.

The above algorithm has several distinct advantages in comparison to the basic OBS method (ref. 1). The changes made to the original paradigm make the pruning procedure less sensitive to imperfect estimation of saliency values. Moreover, the modifications attempt to reduce such CPU intensive tasks as full retraining and $\mathbf{H}^{-1}$ evaluation.

Through direct network testing, the modified method verifies whether the OBS weight update is correct and does not lead, by chance, to an excessive increase in error (step 4). Our observations suggest that especially for partially pruned networks, the OBS weight update may be occasionally inappropriate even when it is calculated based on a newly evaluated $\mathbf{H}^{-1}$. If deletion of a particular link is unsuccessful, the algorithm does not instantly stop. It attempts to remove several other weights having the smallest saliences using the OBS formula only. If this in turn becomes insufficient, the algorithm undertakes to find the correct weight update using a previously applied vector $\Delta\mathbf{w}$ followed by brief retraining, e.g., 5–10% of iterations as in the full training phase (step 5). The connection selected for elimination at this stage is the one that caused the smallest error increase using the pure OBS update in the previous attempts. The extensive network retraining (step 6) is the last resort. It is applied only when all the previously described efforts have failed.

Figure 1 presents a performance comparison between the simple (curve A) and the modified (curve B) OBS based pruning algorithms. The feedforward neural network, having initial architecture 6-36-6 (474 adjustable parameters) was trained on 1373 data points to calibrate the six-component strain-gage balance used in the wind tunnel experiments. It is apparent that the proposed modifications indeed demonstrate a positive impact on the pruning performance which tends to be shorter and capable of removing more weights in total, 154 versus 174 (see also table 1 in section 5).

Equations (2) assume that only one weight is eliminated at a time while all others, in general, remain active. In certain situations, especially when the interconnection pattern is sparse or the network has a single output, some nodes hold only one incoming or outcoming connection (note that biases are treated as incoming synapses). Elimination of these weights inactivates the entire network node and consequently disables more adjacent connections. In such cases, forced deletion of multiple weights could be estimated by a straightforward extension of the basic OBS approach as discussed in reference 2. The Lagrangian function associated with the problem of minimizing (eq. 1), subject to one or more equality constraints $\mathbf{M}_e^{\mathrm{T}}(\mathbf{w}+\Delta\mathbf{w})=0$, is given by

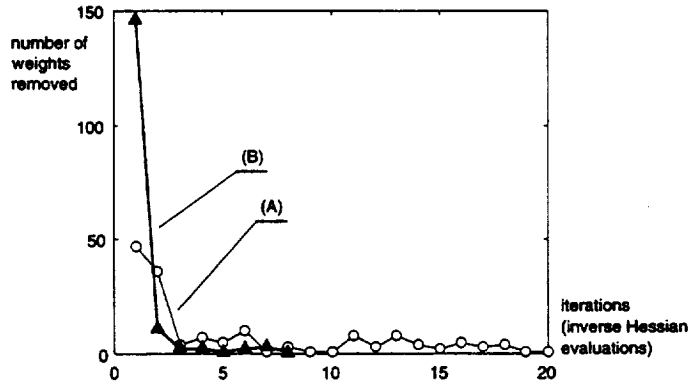


Figure 1. Performance comparison of the simple (curve A) and the modified (curve B) OBS pruning algorithms.

$$L(\Delta\mathbf{w}, \lambda) = \frac{1}{2}\Delta\mathbf{w}^{\mathrm{T}}\mathbf{H}\Delta\mathbf{w} + \lambda^{\mathrm{T}}\mathbf{M}_e^{\mathrm{T}}(\mathbf{w} + \Delta\mathbf{w}) \quad (6)$$

where λ is the vector of Lagrange multipliers and $\mathbf{M}_e = [e_i, e_j, e_k, \ldots]$ is the selection matrix obtained by combining (in any order) unit vectors that correspond to the connections that will be deleted. The solution for $\Delta\mathbf{w}$ should satisfy the following conditions:

$$\begin{cases} \dfrac{\partial L}{\partial \Delta\mathbf{w}} = \mathbf{H}\Delta\mathbf{w} + \mathbf{M}_e\lambda = 0 \\ \dfrac{\partial L}{\partial \lambda} = \mathbf{M}_e^{\mathrm{T}}(\mathbf{w} + \Delta\mathbf{w}) = 0 \end{cases} \quad (7)$$

Elimination of λ from equation (7) yields $\Delta\mathbf{w}$, which in turn allows us to calculate the group saliency coefficient S_M using equation (1). Explicit expressions for $\Delta\mathbf{w}$ and S_M are

$$\begin{aligned} \Delta\mathbf{w} &= -\mathbf{H}^{-1}\mathbf{M}_e\left(\mathbf{M}_e^{\mathrm{T}}\mathbf{H}^{-1}\mathbf{M}_e\right)^{-1}\mathbf{M}_e^{\mathrm{T}}\mathbf{w} \\ S_M &= \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{M}_e\left(\mathbf{M}_e^{\mathrm{T}}\mathbf{H}^{-1}\mathbf{M}_e\right)^{-1}\mathbf{M}_e^{\mathrm{T}}\mathbf{w} \end{aligned} \quad (8)$$

When inversion of the $\mathbf{M}_e^{\mathrm{T}}\mathbf{H}^{-1}\mathbf{M}_e$ matrix in equations (8) is a concern, we suggest to find λ and unknown elements of the $\Delta\mathbf{w}$ vector (packed into one vector $\mathbf{x}$) from the equation

$$\left(\mathbf{H}^{-1}\mathbf{M} + (\mathbf{I} - \mathbf{M})\right)\mathbf{x} = \mathbf{M}\mathbf{w} \quad (9)$$

where $\mathbf{M} = \mathbf{M}_e\mathbf{M}_e^{\mathrm{T}}$ is a diagonal matrix having $m_{ii} = 1$ for those weights that will be deleted and $m_{ii} = 0$ otherwise. Equation (9) was obtained by rearranging unknown variables in equation (7). The final solution could then be extracted form $\mathbf{x}$, i.e.

$$\begin{aligned} \Delta\mathbf{w} &= (\mathbf{I} - \mathbf{M})\mathbf{x} - \mathbf{M}\mathbf{w} \\ S_M &= \frac{1}{2}(\mathbf{M}\mathbf{w} - (\mathbf{I} - \mathbf{M})\mathbf{x})^{\mathrm{T}}\mathbf{M}\mathbf{x} \end{aligned} \quad (10)$$

Our implementation of the OBS based pruning algorithm does not exploit direct topological dependencies between weights for saliency estimation. We have found that the $\Delta\mathbf{w}$ vector, determined from equations (8) or (10), is usually not sufficient to counteract simultaneous elimination of several weights. Although the saliency calculated according to equations (2) would be, to a certain extent, less accurate (and smaller or, unlikely, equal for positive definite Hessian) in comparison to equations (8) or (10), the structure of the proposed weight pruning algorithm has no problem recovering from incorrect or imprecise inputs. Instead, we have used an additional node pruning method (as discussed in the next section) to improve the efficiency and robustness of the designing process.

3. Node Pruning

Connection pruning and node pruning are usually considered as two alternative techniques for simplifying the neural network structure. In our opinion,

however, both procedures may serve complementary functions in the task of neural network complexity reduction, each one having its own strengths and weaknesses. Our experience with the OBS algorithm (implemented as discussed in the previous section) is that for nontrivial regression problems this method tends not to remove entire networks units (hidden nodes). Connection cancellation that leads to node elimination produces a much more destructive impact on the network that is often not accurately characterized by the local approximation (eq. 1). Also, one-shot retraining prescribed by the OBS algorithm often seems to be insufficient to compensate for the effects of node removal, especially when the acceptable margin for the error increase is quite narrow. As a result, the algorithm refuses to remove such weights in favor of others that are easier to compensate.

Linear algebra analysis provides the necessary foundation to construct a node pruning algorithm that detects exact or nearly linear dependencies between neuron outputs of the same hidden layer. In our approach we use the Singular Value Decomposition (SVD) as the diagnostic tool. We utilize a heuristic strategy to choose which hidden node may potentially be removed, as described in Jolliffe (ref. 9), in the context of variable set reduction. Furthermore, we calculate the first approximation of the new weight values for the remaining links between two neighboring layers to compensate for node elimination. In the retraining stage that follows, these values are used as a starting point for the training procedure.

The SVD based node pruning algorithm focuses on each hidden layer sequentially. To some extent it also considers nodes that receive signals from the current layer. Of course, in case of the strictly layered, fully connected architecture all receiving nodes will be located in the next hidden layer (closer to the network output) or in the output layer. Output signals of the current hidden layer may be stored in a matrix $\mathbf{A}$ of the form

$$\mathbf{A} = \underbrace{\begin{bmatrix} 1 & a_1^{(1)} & a_2^{(1)} & \cdots & a_M^{(1)} \\ 1 & a_1^{(2)} & a_2^{(2)} & \cdots & a_M^{(2)} \\ 1 & a_1^{(3)} & a_2^{(3)} & \cdots & a_M^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a_1^{(N)} & a_2^{(N)} & \cdots & a_M^{(N)} \end{bmatrix}}_{\text{bias and } M \text{ hidden nodes}} \Bigg\} N \text{ data points} \qquad (11)$$

The signals $\mathbf{B}$ received by nodes of the subsequent layer (before being transformed by activation functions) may be evaluated using the linear equation $\mathbf{B} = \mathbf{AW}$, where $\mathbf{W}$ is the matrix of connection weights linking the two layers. The first row of $\mathbf{W}$ corresponds to the biases of the receiving nodes. Linear equations[1] $\mathbf{AW} = \mathbf{B}$ could be singular or close to singular due to row or column degeneration of the matrix $\mathbf{A}$. Since the number of data points N used for training is higher than the number of hidden nodes M in the layer under consideration, $N >> M$ (i.e., we deal with an overdetermined system of equations), any linear dependency between columns in $\mathbf{A}$ will cause rank deficiency. This indicates that there possibly exist redundant nodes in the neural network structure whose outputs may be substituted by a linear combination of responses from other nodes. From the point of view of connection weights, singularity implies that the solution $\mathbf{W}$ to the problem $\mathbf{AW} = \mathbf{B}$ is not unique. We are interested in finding weight settings in which one weight in each column of $\mathbf{W}$, say the i-th row, would be exactly zero. This corresponds to the elimination of one network node. The new solution, $\mathbf{W}_0$, also satisfies the equation $\mathbf{AW}_0 = \mathbf{B}$ so the final network performance is not affected. Of course, in practice, instead of an exact linear dependency, a near-linear relationship could be encountered. In such a case $\mathbf{AW}_0 \approx \mathbf{B}$ and new weight settings will cause the network error to increase. To regain the previous network performance additional retraining would be required after node removal.

The analysis of $\mathbf{A}$ may be performed by decomposing the cross-product matrix

[1] Since the next layer would likely contain more than one node it would, rather, be several sets of linear equations, each one having different right hand side arguments.

$$\Sigma = \mathbf{A}^T\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^T \qquad (12)$$

or equivalently by performing the SVD factorization of the original matrix **A**, i.e.

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T \qquad (13)$$

In equation (12), **V** is the $(M+1)\times(M+1)$ orthogonal matrix ($\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$) of Σ eigenvectors, Λ is the same size diagonal matrix containing Σ eigenvalues. In equation (13), **U** is the $N\times(M+1)$ matrix with orthonormal columns, i.e., $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, and $\mathbf{S} = \mathrm{diag}(s_1, s_2, ..., s_{M+1})$ is a non-negative, diagonal matrix such that $\Lambda = \mathbf{S}\mathbf{S}$. The diagonal elements of **S** are called singular values of the **A** matrix.

Factorizations defined by equation (12) or (13) are explored in principal component analysis (PCA) to diagnose multicollinearity. This approach is particularly useful when more than two regressors are expected to form a near-linear relationship. Before performing decomposition (eq. 12 or 13), in the standard PCA analysis, variables (columns of **A**) are typically centered by subtracting from each one its average values ($\Sigma/(N-1)$ then becomes the covariance matrix). When the range/units of the variables are substantially different, by further scaling each column of **A** (typically, dividing it by the standard deviation) we obtain standardized variables ($\Sigma/(N-1)$ is then called the correlation matrix). In our approach, similarly with reference 10, we analyze "raw" data stored in **A**. To some extent, the lack of centering is compensated for by adding one more degree of freedom to **A** through the first column of ones. Signal shifting—such as when a particular node is generating a sequence $\eta(k)$ ($k = 1,..., N$) while another neuron response is $c_1\eta(k) + c_2$ (c_1, c_2 are constant values)—does not interfere with the correct diagnosis of a linear relationship between those nodes.

Our node pruning algorithm starts from the SVD factorization of the **A** matrix (eq. 11). The SVD method is known to be numerically stable and well suited for diagnostics of pathological cases, such as near rank deficiency, associated with solving least squares (LS) problems. In the SVD factorization $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, the matrix **V** defines a linear transformation (rotation) of coordinates which, when applied to **A**, creates a new set of orthogonal variables $\tilde{\mathbf{A}}$, obtained as linear combinations of the original signals. Utilizing the property $\mathbf{V}\mathbf{V}^T = \mathbf{I}$ we can write

$$\mathbf{B} = \mathbf{A}\mathbf{W} = \mathbf{A}\mathbf{V}\mathbf{V}^T\mathbf{W} = \tilde{\mathbf{A}}\tilde{\mathbf{W}} \qquad (14)$$

The operation defined by equation (14) is visualized in figure 2. Conceptually, an additional layer of so called "singular nodes" is introduced after the current layer. The new variables stored in columns of $\tilde{\mathbf{A}}$ are linearly independent, and $\mathbf{S}^{-1}$ contains scaling factors that normalize column vectors of $\tilde{\mathbf{A}}$ to the unit length. From equations (13) and (14) we have

$$\tilde{\mathbf{A}} = \mathbf{U}\mathbf{S} \xrightarrow[\text{if } \mathbf{S}^{-1} \text{ exists}]{} \tilde{\mathbf{A}}\mathbf{S}^{-1} = \mathbf{U} \qquad (15)$$

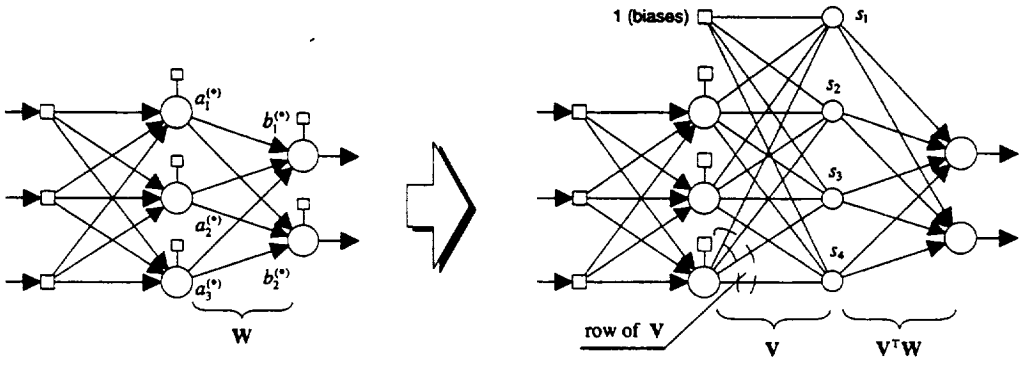


Figure 2. Graphical interpretation of the transformation (eq. 14).

The diagonal values of **S** may be treated as some sort of indicators of the "singular node" strength.[2] A relatively small diagonal value of **S** (in comparison to the biggest singular value) implies that there exists some subset of the original signals which are close to being linearly dependent. In such a case we could try to remove one of the signals (network node) associated with the weakest (last) "singular node," hoping that this will not cause unrecoverable deterioration of the network performance. Figure 3 shows singular values for the hidden layer of the neural network 6-36-6 examined in section 2. Clearly, the last singular values are small and elimination of one or more hidden nodes may be possible.

The key issue of the algorithm is how to make inferences about the importance of the original nodes based on the singular values. Our task is not to remove a "singular node" (this will leave the number of original nodes intact but will disturb connection weights between layers) but somehow guess which of the original nodes may be eliminated. Jolliffe (ref. 9, chap. 6.3) discusses several heuristic approaches in the context of covariance or correlation matrices that we have adopted in our algorithm.

Basically, the problem may be approached from two opposite directions. One method is to indicate which of the original nodes *should not* be removed by associating strong "singular nodes" with the real counterparts and then remove one of the remaining nodes that may potentially be redundant. Another method is to choose (or, more appropriately, guess) directly which node may be unnecessary by associating the real nodes with the weak "eigennodes." The association is based on the strength of connection between the network layer and the conceptual layer of "singular nodes" (matrix **V**). In other words, for the latter approach, we would first select the smallest singular value (the weakest "singular node") and then search for the biggest (in the sense of absolute value) element in the corresponding column of **V** (strongest connection). Next, we zero the entire row of **V** in which the element was found. This is equivalent to the deletion of the genuine node in the neural network structure.

With such an approach, the algorithm may occasionally be tempted to remove all biases of the nodes in the subsequent layer. Since at this stage of pruning it would be rather beneficial to leave biases intact, the algorithm may try to remove another node associated with the weakest "singular node" or focus on the second weak "singular node." The same strategy may be exercised if node removal was unsuccessful. Failure to remove another node could serve as a stopping criterion which indicates the fact that no more redundant nodes could be identified in a given hidden layer using the present algorithm.

4. Node Post-Removal Retraining

Node removal is an event that usually has a strong effect on the network performance. Additional weight adjustment which takes into account target values (matrix **B**) and/or the whole network structure is almost always necessary. In our approach, full network retraining is always applied after each node elimination. The elimination is pronounced successful if the error level prior to node deletion was regained (or some preset threshold for the error increase was not exceeded). To help the training procedure, supplementary, local (in the topological sense) weight correction is implemented. Hopefully, this weight correction will place the modified weight settings closer to the desired solution. For the local weight update, to compensate node deletion, we assume that the new weights $\mathbf{W}_0$ should minimize the norm

$$\min_{\mathbf{W}_0} \left(\left\| \mathbf{A}_0 \mathbf{W}_0 - \mathbf{B} \right\|_2 \right) \tag{16}$$

where $\mathbf{A}_0$ is the matrix derived from **A** by zeroing its column associated with the deleted node. The criterion (eq. 16) ignores nonlinear transformations performed by the subsequent hidden layers. For the $\mathbf{A}_0$ matrix we can write based on equation (13)

$$\mathbf{A}_0 = \mathbf{U}\mathbf{S}\mathbf{V}_0^{\mathrm{T}} \tag{17}$$

[2] In the case of a covariance or correlation matrix, these values will be proportional to standard deviation of the transformed variables (principal components).

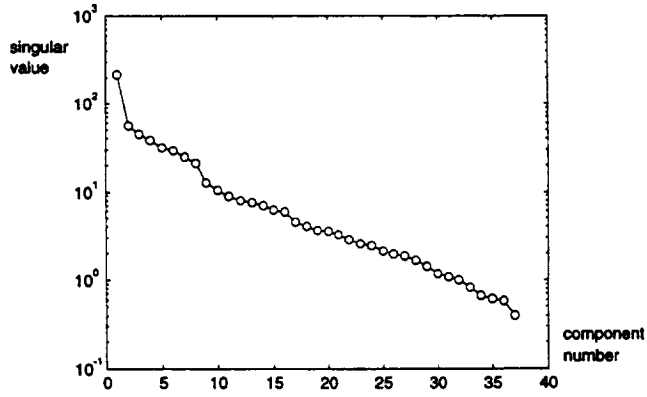


Figure 3. Singular values of the matrix (eq. 11) obtained from hidden node responses of the unpruned, fully trained neural network 6-36-6 used to model the six-component wind tunnel strain gage.

where $\mathbf{V}_0$ is obtained from $\mathbf{V}$ by zeroing one of its rows, say the j-th one, that corresponds to the deleted node. Note that $\mathbf{V}_0$ is in general no longer orthogonal ($\mathbf{V}_0^T\mathbf{V}_0 \neq \mathbf{I}$). The loss of this property prohibits us from employing decomposition (eq. 17) directly to find the new weights $\mathbf{W}_0$. Unfortunately, the correction of the SVD decomposition to compensate for the deleted or zeroed column in $\mathbf{A}$ is not computationally simple (ref. 11). The LS problem (eq. 16) may be solved using other algorithms, such as QR, carefully implemented to handle the singularities which are possible even after removing the weights (unknowns) associated with the deleted node (the obvious source of singularity).

In our approach we have decided to employ another SVD decomposition, but applied to a substantially smaller matrix than $\mathbf{A}$. By substituting equation (17) in the well known equation $\mathbf{A}_0^T\mathbf{A}_0\mathbf{W}_0 = \mathbf{A}_0^T\mathbf{B}$ that defines the solution to the LS problem, we obtain

$$\left(\mathbf{V}_0\mathbf{S}\mathbf{S}\mathbf{V}_0^T\right)\mathbf{W}_0 = \mathbf{V}_0\mathbf{S}\mathbf{U}^T\mathbf{B} \qquad (18)$$

Factorizing the $\mathbf{S}\mathbf{V}_0^T$ matrix using additional SVD decomposition, i.e., $\mathbf{S}\mathbf{V}_0^T = \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^T$, the solution $\mathbf{W}_0$ may be expressed as

$$\mathbf{W}_0 = \mathbf{V}_1\mathbf{S}_1^{-1}\mathbf{U}_1^T\mathbf{U}^T\mathbf{B} \qquad (19)$$

where $\mathbf{S}_1^{-1}$ is calculated in such a way that in the case of true or numerical singularities ($s_{ii} \approx 0$), $1/s_{ii} \rightarrow 0$. No prior elimination of weights (unknowns) associated with the deleted node in $\mathbf{W}_0$ is necessary.

5. Computational Experiments

The performance of the four pruning methods discussed above (three variations of the OBS algorithms and one hybrid SVD/OBS method) has been evaluated by adjusting the interconnection pattern of the 6-36-6 feedforward neural network (6 input sensors, 6 outputs, 474 weights). The network was trained on 1373 data points to perform the calibration task of the six-component wind tunnel strain gage. Table 1 summarizes the results obtained. Application of the SVD based node elimination algorithm, before the OBS connection pruning, produced further reduction in the network size in comparison with the OBS method alone. The modeling problem was so specific that the simple OBS method and its modifications did not remove any network nodes. The SVD method, however, was able to identify and successfully delete 5 network nodes in a somewhat restrictive mode that did not allow the learning error to increase.

To visualize the effects of pruning, we have chosen a simpler nonlinear modeling problem:

$$y(k) = 2.5y(k-1)\sin\left(\pi\exp\left(-u^2(k-1)-y^2(k-1)\right)\right) + u(k-1)\left(1+u^2(k-1)\right) \tag{20}$$

which could be represented in a form of a 3-D plot as shown in figure 4(A). The system (eq. 20) was excited by uniformly distributed white noise in the range $(-2, 2)$. An unpruned, feedforward neural network model 2-8-8-1 (2 input sensors, 1 output, 105 weights) was trained using 250 data samples and the Levenberg-Marquardt algorithm (refs. 8 and 12).

The performance of the fully connected model was compared with its trimmed counterpart produced by the hybrid SVD/OBS pruning procedure. Remarkably, our method was able to remove 75 weights out of 105 holding the average training error (mean-squared error) below 0.002. The SVD approach led to the removal of 50 weights and 4 nodes in the first hidden layer. A subsequent application of the modified OBS pruning method eliminated an additional 25 model parameters using only three inverse Hessian evaluations. Figure 5 compares the error surface between the desired and actual network outputs before and after pruning. Clearly, for sufficiently probed problems and appropriately adjusted error levels, the pruning procedure results in better approximating properties of models having a smaller number of irrelevant parameters, so the overfitting could be avoided. Figure 4(B) presents the response surface of the reduced model.

Table 1. Final results of different pruning algorithms

	Simple OBS	Modified OBS	Modified OBS+BFGS update	Modified OBS+SVD node pruning
Number of iterations	20	8	9	9
Total number of weights removed	154	168	174	199
Number of nodes removed	0	0	0	5

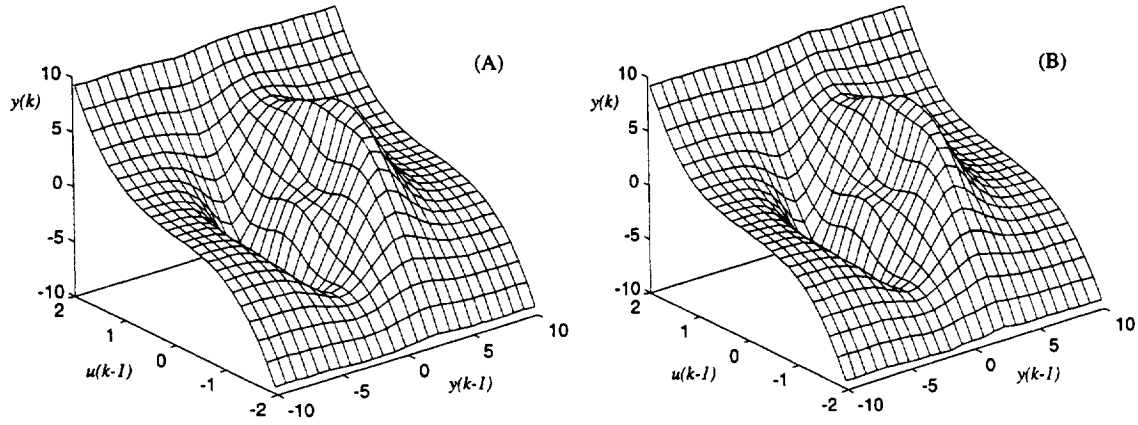


Figure 4. Desired response surface (A) of the nonlinear system (eq. 20) and the approximating output (B) of the neural model tuned using the hybrid SVD/OBS pruning method.

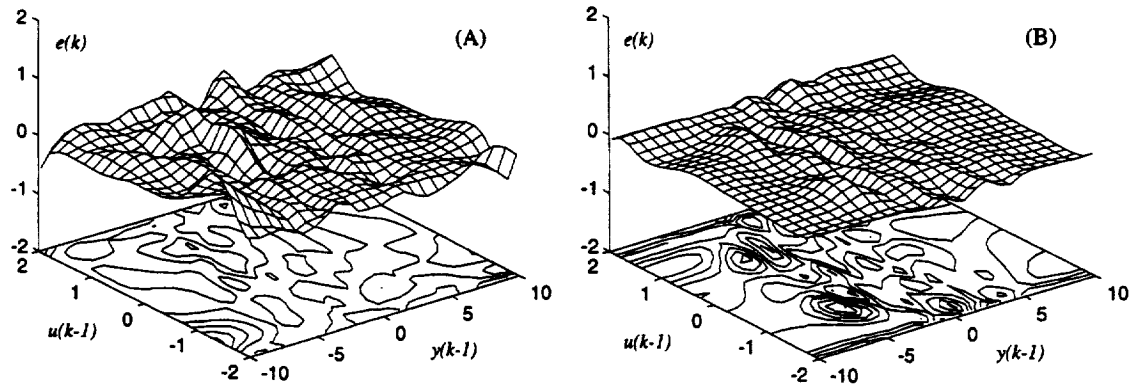


Figure 5. The error surface of the unpruned neural network model (A) and the trimmed counterpart (B).

6. Conclusions

We have proposed an enhanced strategy for selection of candidates to be eliminated using the OBS framework. A careful modification of the basic OBS algorithm allows its efficiency to increase. We claim that it is beneficial to combine the OBS connection pruning with a node removal technique. We propose applying the SVD based method to accomplish this task since both algorithms utilize different (also not conflicting) assumptions and measurements of network redundancy. The SVD based pruning algorithm is relatively easy to implement for fully connected networks and it is reasonable to execute it before the OBS routine. The SVD based pruning algorithm tries to select redundant nodes based on the matrix of node outputs to which an additional unit column should be appended. It utilizes the heuristics similar to those used in linear regression analysis for variable subset selection. We have also proposed a simple-to-implement, local correction of the network weights after node removal.

The pruning method presented requires an efficient training algorithm since the gradually trimmed network has to be periodically retrained. In our experiments we have used the Levenberg-Marquardt algorithm with the predicted error reduction as described in references 8 and 12.

It is an open question (that will be addressed in the further studies) which method should be used for selecting possibly redundant nodes. This issue is concerned with a strategy of associating the weakest "singular node" and possibly irrelevant network neurons. Also, using cross-product, correlation, or covariance matrices would lead to different eigenvalues and different transformations of the original coordinate system. As pointed out by Jolliffe in reference 9, no simple relation exists between eigenvectors and eigenvalues of these matrices. Consequently, applying various modifications of the basic procedure described, one may obtain algorithms that differ in their efficiency and robustness. Despite these obstacles, however, it is apparent that the SVD pruning algorithm provides a valuable (if not indispensable) supplement to the fine weight pruning performed by the OBS procedure.

References

1. Hassibi, B.; Stork, D.; and Wolff, G. J.: Optimal Brain Surgeon and General Network Pruning. *IEEE Int. Conf. Neural Networks*, San Francisco, 1993, pp. 293–299.

2. Stahlberger, A.; and Riedmiller, M.: Fast Network Pruning and Feature Extraction using Unit-OBS Algorithm. *Advances in Neural Information Processing Systems*, vol. 9, Morgan Kaufmann, 1997, pp. 655–661.

3. Pedersen, M. W.; Hansen, L. K.; and Larsen, J.: Pruning with Generalization Based Weigh Saliencies: γOBD, γOBS. *Advances in Neural Information Processing Systems*, vol. 7, Morgan Kaufmann, 1995, pp. 521–527.

4. Battiti, R.: First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method. *Neural Computation*, vol. 4, 1992, pp. 141–166.

5. Ästrom, K. J.; and Wittenmark, B.: *Computer Controlled Systems—Theory and Design.* Prentice-Hall, 1984.

6. Seber, G. A. F.; and Wild, C. J.: *Nonlinear Regression.* John Wiley & Sons, 1989.

7. Golden, R. M.: *Mathematical Methods for Neural Network Analysis and Design.* The MIT Press, 1996.

8. Fletcher, R.: *Practical Methods of Optimization.* John Wiley & Sons, 1987.

9. Jolliffe, I. T.: *Principal Component Analysis.* Springer-Verlag, 1986.

10. Psichogios, D. C.; and Ungar, L. H.: SVD-NET: An Algorithm that Automatically Selects Network Structure. IEEE Trans. *Neural Networks*, vol. 5, no. 3, 1994, pp. 513–515.

11. Bunch, J. R.; and Nielsen, C. P.: Updating the Singular Value Decomposition. *Numer. Math.*, vol. 31, 1978, pp. 111–129.

12. Norgaard, P. M.; Jorgensen, C. C.; and Ross, J.: Neural Network Prediction of New Aircraft Design Coefficients. NASA TM-112197, May 1997.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1998	Technical Memorandum

4. TITLE AND SUBTITLE

Toward a More Robust Pruning Procedure for MLP Networks

5. FUNDING NUMBERS

519-30-12

6. AUTHOR(S)

Slawomir W. Stepniewski* and Charles C. Jorgensen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Ames Research Center
Moffett Field, CA 94035-1000

8. PERFORMING ORGANIZATION REPORT NUMBER

A-98-10243

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA/TM—1998-112225

11. SUPPLEMENTARY NOTES

Point of Contact: Charles C. Jorgensen, Ames Research Center, MS 269-1, Moffett Field, CA 94035-1000
(650) 604-6725
*Recom Technologies, Inc., Ames Research Center.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified — Unlimited
Subject Category 31

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

Choosing a proper neural network architecture is a problem of great practical importance. Smaller models mean not only simpler designs but also lower variance for parameter estimation and network prediction. The widespread utilization of neural networks in modeling highlights an issue in human factors. The procedure of building neural models should find an appropriate level of model complexity in a more or less automatic fashion to make it less prone to human subjectivity. In this paper we present a Singular Value Decomposition based node elimination technique and enhanced implementation of the Optimal Brain Surgeon algorithm. Combining both methods creates a powerful pruning engine that can be used for tuning feedforward connectionist models. The performance of the proposed method is demonstrated by adjusting the structure of a multi-input multi-output model used to calibrate a six-component wind tunnel strain gage.

14. SUBJECT TERMS

Neural networks, Pruning, Optimal Brain Surgeon, Principal Component Analysis, Singular Value Decomposition

15. NUMBER OF PAGES

16

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		