Incremental Learning of Auto-Association Multilayer Perceptrons Network

Essam Al-Daoud
Faculty of Science and Information Technology, Zarqa Private University, Jordan

Abstract: *This paper introduces a new algorithm to reduce the time of updating the weights of auto-association multilayer perceptrons network. The basic idea is to modify the singular value decomposition which has been used in the batch algorithm to update the weights whenever a new row is added to the input matrix. The computation analysis and the experiments show that the new algorithm speeds up the implementation about 5-8 times.*

Keywords: *Neural networks, auto-association multilayer perceptrons, singular value decomposition.*

Received October 10, 2004; accepted December 16, 2004

1. Introduction

Neural Networks (NN) adapt to changing environments and afford the possibility of relatively easy hardware implementation. They can even overcome the drawbacks of classical algorithms or enhance the performance of the classification [5]. The Multilayer Perceptrons (MLP) and the Auto-Association Multilayer Perceptrons (AAMLP) respectively embed supervised and unsupervised mappings of the input space in their hidden layers. An NN implementation of Sammon's mapping which has been suggested by Mao and Jain, and Kohonen's Self-Organizing Map (SOM) are other examples of neural networks [1, 6, 9].

AAMLP is used to provide pattern completion to produce a pattern whenever a portion of it or a distorted pattern is presented. In the second case, the network actually stores pairs of patterns building an association between two sets of patterns. AAMLP is forced to perform an identity mapping through a small hidden layer; in other words the target output pattern is identical to the input pattern. Hence, an AAMLP has a configuration of M:P:M with M units in both the input and output layers and $P < M$ hidden units in the hidden layer [3].

The remainder of this paper is organized as follows. Section 2 presents the basic equations of auto-association multilayer perceptrons. Section 3 introduces the AAMLP Batch Algorithm. Section 4 updates the components of the singular value decomposition. In section 5 we introduce a new algorithm to update the weights whenever a new row in the input matrix is added. Section 6 discusses the improvement of using the suggested algorithm numerically and analytically.

2. Auto-Association Multilayer Perceptrons

The AAMLP consists of an input layer with M input units, a hidden layer with P units and M output unites. Let X be an $M \times N$ real input matrix formed by N input vectors and let H and Y be the $P \times N$ and $M \times N$ matrices formed by the hidden and the output vectors respectively. Subsequently the output matrix Y of the AAMLP is obtained as the result of the following operations:

$$B = W_1X + w_1u^t \quad (1)$$
$$H = F\,(B) \quad (2)$$
$$Y = W_2H + w_2u^t \quad (3)$$

Where W_1 is the input-to-hidden $P \times M$ weight matrix, W_2 is the hidden-to-output $M \times P$ weight matrix, w_1 *and* w_2 are P-vectors of biases and u is N-vector of ones. AAMLP training problem is to find optimal weight matrices W_1, W_2 and bias vectors w_1, w_2 minimizing the mean square error:

$$J = \| X\text{-}Y \|^{2}$$

Where $\|.\|$ is Euclidean matrix norm. This problem can be solved by the usual Error Back-Propagation (EBP) algorithm as described by Rumelhart *et al.* [1]. However Bourlard and Kamp propose a new fast algorithm based on standard linear algebra and the Singular Value Decomposition (SVD). Moreover, they show that the nonlinear functions at the hidden layer completely unnecessary [3, 4]. Therefore; we will restrict our study on the linear function $F\,(B) = B$.

3. AAMLP Batch Algorithm

Bourlard and Kamp derive a new algorithm using the SVD. The following discussion summarizes their work [3]:

- Using (3) the mean square error can be rewritten as:

$$J = || X - W_2H + w_2u^t ||^2 \quad (4)$$

- Minimization of J with respect to w_2 yields

$$w_2 = (X - W_2H)u / N \quad (5)$$

- Substituting (5) in (4) we get:

$$J = ||X (I - uu^t / N) - W_2H (I - uu^t / N)||^2$$

Let $X' = X (I - uu^t / N)$ *and* $H' = H (I - uu^t / N)$, *then*

$$J = || X' - W_2 H' ||^2$$

J can be minimized if we found the best rank P approximation of X', this is a standard problem can be solved by SVD as follows:

$$SVD (X') = U_p \Sigma_p V_p^t \quad (6)$$

Where U_p and V_p are $M \times P$ matrix associated with the eigenvectors of $X' X'^t$ and $X'^t X'$ respectively, and Σ_p is a diagonal matrix formed by the roots of the largest P eigenvalues, for more information see [8]. Thus:

$$W_2 H' = U_p \Sigma_p V_p^t$$

this implies to:

$$W_2 = U_pT^{-1} \quad (7)$$

and

$$H' = T \Sigma_p V_p^t$$

Where T is an arbitrary non singular $P \times P$ matrix (to reduce the calculations we will use sparse or diagonal matrix), Since $B = H$ then we have

$$T \Sigma_p V_p^t = W_1X' + w_1u^t (I - uu^t / N)$$

But $u^tu = N$, therefore w_1 is arbitrary and

$$T \Sigma_p V_p^t = W_1X'$$

This implies to

$$W_1 = TU_p^t \quad (8)$$

Finally by replacing H and W_2 in (3) we get

$$w_2 = (I - U_p U_p^t)Xu / N - U_p T^{-1} w_1 \quad (9)$$

Algorithm 1. AAMLP Batch Algorithm

Input: *The number of input units M, the number of input vectors N, the number of the hidden units P, The input matrix X, an arbitrary non singular matrix T, a unit victor u, and an arbitrary vector* w_1.

Ouput: *The optimal Weights* W_1, W_2, *and* w_2.

1. *Xp = X - ((X * u) * u') / N*
2. *[U, S, V] = svd (Xp);*
3. *Up = U (:,1:P)*
4. *W1 = T * Up'*
5. *W2 = Up * inv (T)*
6. *Mx = (1 / N) * X * u*
7. *w2 = (Mx – Up * (Up' * Mx)) - (W2 * w1))*

4. Updating the SVD

Let X be an $M \times N$ real matrix, SVD of $X = U \ \Sigma \ V$ and let C be a new column, then we can update SVD as follows [2, 7]:

$$SVD ([X\ C]) = U'' \ \Sigma'' \ V''^t$$

But the previous components can be computed as follows:

$$U'' = [U\ J]\ U'$$

$$\Sigma'' = \Sigma'$$

$$V'' = \begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix} V'$$

where

$$F = C - U * U^t * C$$
$$J = F / ||F||$$
$$L = U^t * C$$
$$Q = \begin{bmatrix} \Sigma & L \\ 0 & || F || \end{bmatrix}$$
$$SVD (Q) = U' \ \Sigma' \ V'^t.$$

5. Updating the Weights of AAMLP Network

Suppose that after we have found the weights of AAMLP network; we like to add a new element at the end of each input vector, this means a new row must be added to the end of the input matrix X. hence the previous weights must be updated. Let U_p, Σ_p and V_p be the singular value decomposition of the input matrix X, and let R be the new row, thus the new input matrix is:

$$\begin{bmatrix} X \\ R \end{bmatrix}$$

Let $\begin{bmatrix} X' \\ R' \end{bmatrix} = \begin{bmatrix} X \\ R \end{bmatrix} * (I - uu^t / N)$ and $p' = p + 1$ be the number of hidden units in the new network, this implies to

$$X' = X (I - uu^t / N)$$
$$R' = R \ (I - uu^t / N)$$

and

$$SVD \left(\begin{bmatrix} X' \\ R' \end{bmatrix}\right) = SVD ([X' \ R']^t)$$

$$= SVD\,([X' \;\; R'])^t$$
$$= (U''_{p'} \; \Sigma''_{p'} \; V''^{t}_{p'})^t$$
$$= V''_{p'} \; \Sigma''_{p'} U''^{t}_{p'}$$

But, the first component of $SVD\left(\begin{bmatrix} X' \\ R' \end{bmatrix}\right)$ can be computed as follows:

$$V''_{p'} = \begin{bmatrix} V_p & 0 \\ 0 & I \end{bmatrix} V'_{p'}$$
$$= V_{p^*} \; V'_p + V' \; (p+1, p+1) \qquad (10)$$

Where

$$F = R' - U_p * U_p^{\,t} * R' \qquad (11)$$
$$L = U_p^{\,t} * R' \qquad (12)$$
$$Q = \begin{bmatrix} \Sigma_p & L \\ 0 & \| F \| \end{bmatrix}$$
$$SVD\,(Q) = U'_{p'} \; \Sigma'_{p'} \; V''_{p'} \qquad (13)$$

So the new weights are

$$W_1 = T \; V''^{t}_{p'} \qquad (14)$$
$$W_2 = V''_{p'} \; T^{-1} \qquad (15)$$
$$w_2 = (I - V''_{p'} \; V''^{t}_{p'} \,)\, Xu / N) - V''_{p'} \; T^{1} \, w_1 \qquad (16)$$

Algorithm 2. Increment AAMLP algorithm

Input: *The number of input units M, the number of input vectors N, the number of the hidden units $P' = P + 1$, the component of the input matrix X (U_p, $\sum_p$,* and *V_p), an arbitrary non singular matrix T, a unit victor u, an arbitrary vector w_1, and a new row R.*

Ouput: *The optimal Weights W_1, W_2, and w_2.*

1. *Xp = X - ((X * u) * u') / N;*
2. *Rp = R - ((R * u) * u') / N;*
3. *F = norm(Rp - Up * (Up' * Rp));*
4. *L = Up' * Rp;*
5. *Z (1:1, 1:P) = 0;*
6. *Q = [S L; Z F];*
7. *[U2, S2, V2] = svds (Q);*
8. *V3 = Vp * V2 (1:P, 1:P) + V2 (P + 1, P + 1);*
9. *W1 = T * V3'*
10. *W2 = V3 * inv (T)*
11. *Mx = (1 / N) * X * u*
12. *w2 = (Mx - V3 * (V3' * Mx)) - (W2 * w1))*

In algorithm 2 we have used the sparse matrices procedure *svds ()* which is considered much faster than the procedure *svd ()*.

6. Complexity Analysis and Experiments

The complexity of the batch algorithm equals to the total complexity of the equations (6-9), it is clear that: The most expensive equation is (6) which takes $O\,(M * N * C)$ operations where $C = \min\,(M, N)$, and since the matrix T can be diagonal or sparse the other equations takes $O\,(M * P)$ or $O\,(N * P)$ operations. On other hand; the complexity of the new algorithm equals the total complexity of the equations (10-16). Equation (10) is the most expensive equation and takes $O\,(M * P^2)$ operations, but equation (13) takes $O\,(P^2)$ operations because Q is bordered diagonal (remember that: To reduce the number of multiplications in the other equations we have to multiply matrix_vector before matrix_matrix). Subsequence the new algorithm is faster than the batch algorithm because $P \leq \min\,(M, N)$, furthermore; matrices multiplications can be done faster by using Winograd method.

For our experiments, we use random input matrices for training and testing. The machine had 1.2 GH Intel Pentium 4 CPU with 256 MB and the code implemented by using MATLAB 6.5. Table 1 and Figure 1 show the advantage of using the new algorithm over the batch algorithm if a new row is added to the input matrix *X*.

Table 1. Comparison between the Batch algorithm and the new algorithm.

M+1	N	Batch Algorithm (Seconds)	The New Algorithm (Seconds)	Error $J = \|X - Y\|^2$
301	300	3.02	0.67	$0.1*10^{-10}$
401	450	9.04	1.21	$0.1*10^{-10}$
501	550	17.34	3.64	$0.4*10^{-11}$
601	500	30.33	6.12	$0.4*10^{-11}$
701	700	49.50	8.5	$0.3*10^{-11}$
801	900	85.96	13.94	$0.1*10^{-11}$
1201	1100	258.41	43.18	$0.2*10^{-12}$
1601	1600	873.91	143.68	$0.2*10^{-12}$
2001	2100	3665.4	513.08	$0.1*10^{-12}$

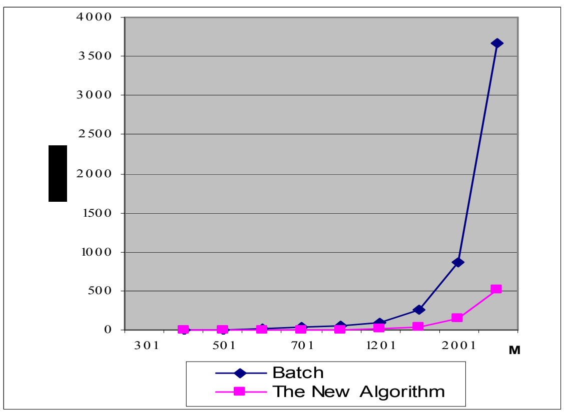

Figure 1. Comparison between the batch algorithm and the new algorithm.

7. Conclusion

We have shown that the implementation of the new algorithm reduces the time of updating the weights of auto-association multilayer perceptrons whenever a new row is added to the input matrix. The comparison shows that the new algorithm speeds up the computation about 5-8 times. This work can be extended easily if a new input vector (or column) is added to the input matrix.

References

[1] Bishop C. M., *Neural Networks for Pattern Recognition*, Oxford Press, 1995.

[2] Brand M., "Fast Online SVD Revisions for Lightweight Recommender Systems," *in Proceedings of the SIAM International Conference on Data Mining (SDM)*, TR2003-014, May 2003.

[3] Bourland H. and Kamp Y., "Auto-Association by Multilayer Perceptrons and Singular Value Decomposition," *Biological Cybernetics*, vol. 59, pp. 291-294, 1988.

[4] Japkowicz N., Hanson S. J., and Gluck M. A., "Nonlinear Autoassociation is not Equivalent to PCA," *Neural Computation*, vol. 12, no.3, pp. 531-545, March 2000.

[5] Lerner B. H., Guterman M., Aladjem I., Dinstein, and Y. Romem, "On Pattern Classification with Sammon's Nonlinear Mapping: An Experimental Study," *Pattern Recognition*, vol. 31, pp. 371-381, 1998.

[6] Mao J. and Jain A. K., "Artificial Neural Networks for Feature Extraction and Multivariate Data Projection," *IEEE Transactions on Neural Networks*, vol. 6, pp. 296-317, 1995.

[7] Matthew B., "Incremental Singular Value Decomposition of Uncertain Data with Missing Values," *in Proceedings of the 7th European Conference on Computer Vision-Part I*, pp. 707-720, 2002.

[8] Strang G., *Introduction to Linear Algebra*, Wellesley, MA, Wellesley-Cambridge Press, 1998.

[9] Valentin N. and Denoeux T., "Neural Network-Based Software Sensor for Coagulation Control in a Water Treatment Plant," *Intelligent Data Analysis*, vol. 5, pp. 23-39, 2001.

Essam Al-Daoud is an assistant professor at the Department of Computer Science, Zarqa Private University, Jordan. He received his PhD from University Putra Malaysia, Malaysia, in 2001. His research interests include cryptography, data mining, quantum computing, neural networks, and Singular Value Decomposition (SVD).