MULTIPLE SUBSPACE ULV ALGORITHM AND LMS TRACKING

S. HOSUR, A. H. TEWFIK, D. BOLEY
University of Minnesota
200 Union St. S.E.
Minneapolis, MN 55455
U.S.A
{*hosur@ee,tewfik@ee,boley@cs*}*.umn.edu*

ABSTRACT. The LMS adaptive algorithm is the most popular algorithm for adaptive filtering because of its simplicity and robustness. However, its main drawback is slow convergence whenever the adaptive filter input auto-correlation matrix is ill-conditioned i.e. the eigenvalue spread of this matrix is large [2, 4].

Our goal in this paper is to develop an adaptive signal transformation which can be used to speed up the convergence rate of the LMS algorithm, and at the same time provide a way of adapting only to the strong signal modes, in order to decrease the excess Mean Squared Error (MSE). It uses a data dependent signal transformation. The algorithm tracks the subspaces corresponding to clusters of eigenvalues of the auto-correlation matrix of the input to the adaptive filter, which have the same order of magnitude. The algorithm up-dates the projection of the tap weights of the adaptive filter onto each subspace using LMS algorithms with different step sizes. The technique also permits adaptation only in those subspaces, which contain strong signal components leading to a lower excess Mean Squared Error (MSE) as compared to traditional algorithms. The transform should also be able to track the signal behavior in a non-stationary environment. We develop such a data adaptive transform domain LMS algorithm, using a generalization of the rank revealing ULV decomposition, first introduced by Stewart [5]. We generalize the two-subspace ULV updating procedure to track subspaces corresponding to three or more singular value clusters.

KEYWORDS. Adaptive filtering, least mean squares, subspace tracking, singular value decomposition.

1 INTRODUCTION

The LMS adaptive algorithm is the most popular algorithm for adaptive filtering because of its simplicity and robustness. However, its main drawback is slow convergence whenever the adaptive filter input auto-correlation matrix is ill-conditioned i.e. the eigenvalue spread of this matrix is large [2, 4]. A class of adaptive filters known as the transform domain filters have been developed for the purpose of convergence rate improvement [4]. All transform domain adaptive filters try to approximately de-correlate and scale the input to the adaptive filter in the transform domain, in order to obtain an autocorrelation matrix with zero eigen value spread in that domain.

The convergence rate of the Least Mean Squares (LMS) algorithm is poor whenever the adaptive filter input auto-correlation matrix is ill-conditioned. In this paper we propose a new LMS algorithm to alleviate this problem. It uses a data dependent signal transformation. The algorithm tracks the subspaces corresponding to clusters of eigenvalues of the auto-correlation matrix of the input to the adaptive filter, which have the same order of magnitude. The algorithm up-dates the projection of the tap weights of the adaptive filter onto each subspace using LMS algorithms with different step sizes. The technique also permits adaptation only in those subspaces, which contain strong signal components leading to a lower excess Mean Squared Error (MSE) as compared to traditional algorithms.

Our goal in this paper is to develop an adaptive signal transformation which can be used to speed up the convergence rate of the LMS algorithm, and at the same time provide a way of adapting only to the strong signal modes, in order to decrease the excess MSE. The transform should also be able to track the signal behavior in a non-stationary environment. We develop such a data adaptive transform domain LMS algorithm, using a generalization of the rank revealing ULV decomposition, first introduced by Stewart [5].

The ULV updating procedure [5] maintains and updates only two groups of singular values: the large ones and the "ones close to zero." This is suitable if the input autocorrelation matrix has eigenvalues which could be so classified.

In this paper, we generalize the two-subspace ULV updating procedure to track subspaces corresponding to more than two singular value clusters. Each step of the generalized procedure may be viewed as a recursive application of the ULV decomposition on the upper triangular matrix R computed at the previous stage within the same step.

2 THE ULV DECOMPOSITION

The SVD is typically used to isolate the smallest singular values, and the success of any method based on the SVD depends critically on how that method decides which singular values are "small" enough to be isolated. The decision as to how many singular values to isolate may be based on a threshold value (find those values below the threshold), by a count (find the last k values), or by other considerations depending on the application. However, in extracting singular values one often wants to keep clusters of those values together as a unit. For example, if all values in a cluster are below a given threshold except one, which is slightly above the threshold, it is often preferable to change the threshold than split up the

cluster. In the SVD, this extraction is easy. Since all the singular values are "displayed", one can easily traverse the entire sequence of singular values to isolate whichever set is desired. In this section we present a set of primitive procedures to provide these same capabilities with the less computationally expensive ULV Decomposition.

2.1 DATA STRUCTURE

The ULV Decomposition of a real $n \times p$ matrix A (where $n \geq p$) is a triple of 3 matrices U, L, V plus a rank index r, where $A = ULV^T$, V is $p \times p$ and orthogonal, L is $p \times p$ and lower triangular, U has the same shape as A with orthonormal columns, and the leading $r \times r$ part of L has a Frobenius norm approximately equal to the norm of a vector of the r leading singular values of A. That is, $A = ULV^T$ with

$$L = \begin{pmatrix} C & 0 \\ E & F \end{pmatrix}$$

where $\|C\|_F^2 \approx \sigma_1^2(A) + \cdots + \sigma_r^2(A)$ encapsulates the "large" singular values of L. This implies that (E, F) (the trailing $p - r$ rows of L) approximately encapsulate the $p - r$ smallest singular values, and the last $p - r$ columns of V encapsulate the corresponding trailing right singular vectors.

In the data structure actually used for computation, L is needed to determine the rank index at each stage as new rows are appended, but the U is not needed to obtain the right singular vectors. Therefore, a given ULV Decomposition can be represented just by the triple $[L, V, r]$.

2.2 PRIMITIVE PROCEDURES

We partition the ULV updating process into a five primitive procedures. The first three procedures are designed to allow easy updating of the ULV Decomposition as new rows are appended. Each basic procedure costs $O(p^2)$ operations and consists of a sequence of plane (Givens) rotations [1]. By using a sequence of such rotations in a very special order, we can annihilate desired entries while filling in as few zero entries as possible, and then restoring the few zeroes that are filled in. We show the operations on L, partitioned as in (2.1). Each rotation applied from the right is also accumulated in V, to maintain the identity $A = ULV^T$, where the U is not saved. The last two procedures use the first three to complete a ULV update.

- `Absorb_One`: Absorb a new row. The matrix A is augmented by one row, obtaining

$$\begin{pmatrix} A \\ \mathbf{a}^T \end{pmatrix} = \begin{pmatrix} U & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} L \\ \mathbf{a}^T V \end{pmatrix} V^T.$$

  Then the L, V are updated to restore the ULV structure, and the rank index r is incremented by 1. No determination is made if the rank has really increased by 1; this is done elsewhere. The process is sketched as follows, where `C` denotes large entries, `e`,`f` denote small entries in the ULV partitioning, `R` denotes an entry of the new row, `+` a temporary fill, and `.` a zero entry:

```
C....     apply       C+...     apply       C....
CC...     rotations   CC+..     rotations   CC...
CCC..     from        CCC+.     from        CCC..
eeef.     right       eeef+     left        CCCC.
eeeff     to get      eeeff     to get      eeeef
RRRRR     ---->       R....     ---->       .....
```

- `Extract_Info`: The following information is extracted from the ULV Decomposition: (a) the Frobenius norm of (E, F) (i.e., the last $p-r$ rows of L), (b) an approximation of the last singular value of C (i.e., the leading $r \times r$ part of L), and (c) a left singular vector of C corresponding to this singular value. These are computed using a *condition number estimator* [3].

- `Deflate_One`: Deflate the ULV Decomposition by one (i.e., apply transformation and decrement the rank index by one so that the smallest singular value in the leading $r \times r$ part of L is "moved" to the trailing rows). Specifically, transformations are applied to isolate the smallest singular value in the leading $r \times r$ part of L into the last row of this leading part. The transformations are constructed using item (c) from `Extract_Info`. Then the rank index is decremented by 1, effectively moving that smallest singular value from the leading part to the trailing part of L. This operation just moves the singular value without checking whether the singular value moved is close to zero or any other singular value.

- `Deflate_To_Gap`: This procedure uses a heuristic to try to move the rank boundary, represented by the rank index r, toward a gap among the singular values. Let s be the smallest singular value of C and let f be the Frobenius norm of $[E, F]$. Then we use the heuristic that a gap exists if $s > df$, where d is a user chosen `Spread`. In order to allow for round-off or other small noise, we pretend that the trailing part has an extra $p+1$-th singular value equal to a user chosen `Zero_Tolerance` b. Then the heuristic actually used is $s^2 > d^2(f^2 + b^2)$. If this condition fails, `Deflate_One` is called repeatedly until this condition is satisfied. Hence, any singular value that is below b or within a cluster of b will be treated as part of the trailing part. The only two user defined parameters needed for this heuristic are the `Spread` d and the `Zero_Tolerance` b.

- `Update`: This procedure encompasses the entire process. It takes an old ULV Decomposition and a new row to append, and incorporates the row into the ULV Decomposition. The new row is absorbed, and the rank is deflated if necessary to find a gap among the singular values.

3 GENERALIZED ULV UPDATE

The idea of a generalized ULV decomposition, which divides the singular values into more than two clusters can be introduced with the simple example where there are three groups of singular values. Now there are two singular value boundaries, r_1 & r_2, which have to be maintained and updated properly. We have the following primitive procedures which are all implemented by calling the ordinary procedures discussed above with either the data structure $[L, V, r_1]$ or $[L, V, r_2]$, depending on which boundary must be updated.

- `Generalized_Absorb_One`. Add a new row and update the two boundaries. This procedure just calls `Absorb_One` using the second boundary, i.e. with the data structure $[L, V, r_2]$. This has the effect of incrementing r_2, But the resulting rotations have the effect of expanding the top group of singular values by one extra row, hence the first boundary, r_1 is incremented by one.

- `Generalized_Deflate_One`. This procedure deflates the lower singular value boundary using `Deflate_One` applied to $[L, V, r_2]$. But as in `Generalized_Absorb_One`, the upper boundary must be incremented by one. In order to restore the separation between the first and second groups of singular values that existed before application of these update procedures, the upper boundary must be repeatedly deflated until a gap is found. This process is accomplished using `Deflate_To_Gap` on $[L, V, r_1]$, which does not affect the boundary r_2 at all.

Using the generalized ULV decomposition, we can group the singular values of any matrix into an arbitrary number of groups. The number of groups or clusters is determined automatically by the largest condition number that can be tolerated in each cluster. This implies that if one chooses the clustering to be done in such a way that each cluster has singular values of the same order of magnitude, the condition number in each cluster is improved which in turn implies a faster convergence of the LMS filter applied to a projection of weights in the corresponding subspace. The largest condition number is the maximum of the ratio of the largest singular value in each cluster to its smallest singular value. This value depends on the `Spread` and `Zero_Tolerance`, specified by the user.

4 THE ULV-LMS ALGORITHM

Let the input signal vector at time n be given as

$$\mathbf{x}_n = [x(n), x(n-1), \cdots, x(n-N+1)]^T$$

and let the weight vector at this time be h_n. The corresponding filter output is

$$z_n = \mathbf{x}_n^T \mathbf{h}_n,$$

and the output error e_n is given as the difference of the desired response $d(n)$ and the output z_n of the adaptive filter at time n:

$$e_n = d(n) - z_n.$$

The LMS algorithm tries to minimize the mean squared value of the output error with each new data sample received as

$$\mathbf{h}_{n+1} = \mathbf{h}_n + 2\mu \mathbf{x}_n e_n,$$

where $0 < \mu < 1$ is the step size.

The convergence of the LMS algorithm depends on the condition number of the input autocorrelation matrix

$$\mathbf{R}_x = \mathbf{X}\mathbf{X}^T \stackrel{\triangle}{=} E[\mathbf{x}_n \mathbf{x}_n^T].$$

If the input vector $\mathbf{x}_n$ is transformed to $\mathbf{u}_n = \mathbf{E}^T\mathbf{x}_n$, where E is the unitary eigenvector matrix of $\mathbf{R}_x$, then the output process $\mathbf{z}_n$ would be de-correlated. However, this implies that we need to perform an eigen decomposition of the autocorrelation matrix or a singular value decomposition of the data matrix at every adaptation, implying a computational computational complexity of $O(N^3)$ for every adaptation. One could replace $\mathbf{E}$ with $\mathbf{V}$, where $\mathbf{V}$ is any unitary matrix which block diagonalizes $\mathbf{R}_x$, separating the signal subspaces from the noise subspace, but this still takes $O(N^3)$ operations to compute.

Instead of transforming the input using the eigen matrix, we could transform the input using the unitary matrix V obtained by the generalized ULV decomposition, which *approximately* block diagonalizes $\mathbf{R}_x$. This would imply a savings in the computational costs as the ULV decomposition can be updated with each new data at a relatively low computational cost. We note that V almost block diagonalizes $\mathbf{R}_x$ in the sense that it exactly block diagonalizes a small perturbation of it. If $\mathbf{X}^T = ULV^T$ with L defined by (2.1) so that

$$\mathbf{R}_x = VL^TLV^T,$$

then V exactly block diagonalizes $\mathbf{R}_x - \Delta$ as follows:

$$V(\mathbf{R}_x - \Delta)V^T = \begin{pmatrix} C^TC & 0 \\ 0 & F^TF \end{pmatrix} \text{ where } \Delta = V^T \begin{pmatrix} E^TE & E^TF \\ F^TE & 0 \end{pmatrix} V.$$

So $\|\Delta\|_F \leq f^2$ is small, where $f = \|[E, F]\|_F$ defined above. For a more detailed analysis of the generalized ULV and the subspace tracking LMS algorithm refer to [6, 7].

The input data vector $\mathbf{x}_n$ is transformed into the vector

$$\mathbf{y}_n = V^T\mathbf{x}_n.$$

These transformed coefficients are then weighed using the subspace domain adaptive filter coefficient vector $\mathbf{g}_n$. The output signal z_n is given as

$$z_n = \mathbf{g}_n^T\mathbf{y}_n,$$

and the LMS weight update equation is given by

$$\mathbf{g}_{n+1} = \mathbf{g}_n + 2\mathbf{M}e_n\mathbf{y}_n,$$

where e_n is the corresponding output error and $\mathbf{M}$ is a diagonal matrix of the step sizes used. The diagonal elements of $\mathbf{M}$ can usually be clustered into values of equal step sizes, corresponding to the subspaces isolated using the generalized ULV. This clustering is due to the fact that each subspace is selected to minimize the condition number in that subspace. Hence adaptation of all the projected tap weights within each subspace has nearly the same convergence speed and one only needs to match the convergence speeds of the slow converging subspace projections of the tap weights to those of the fast converging subspace projections. This can be done by using larger step sizes for those subspace projections of the tap weights which converge slowly, to increase their convergence speed. Also the clusters obtained using the generalized ULV are very well organized, with the largest singular value cluster first, making construction of $\mathbf{M}$ is very straightforward. The diagonal values of the upper triangular matrix generated in the generalized ULV decomposition reflect the average magnitude of the singular values in each cluster. This information can also be used in the selection of the step sizes and hence in the construction of $\mathbf{M}$

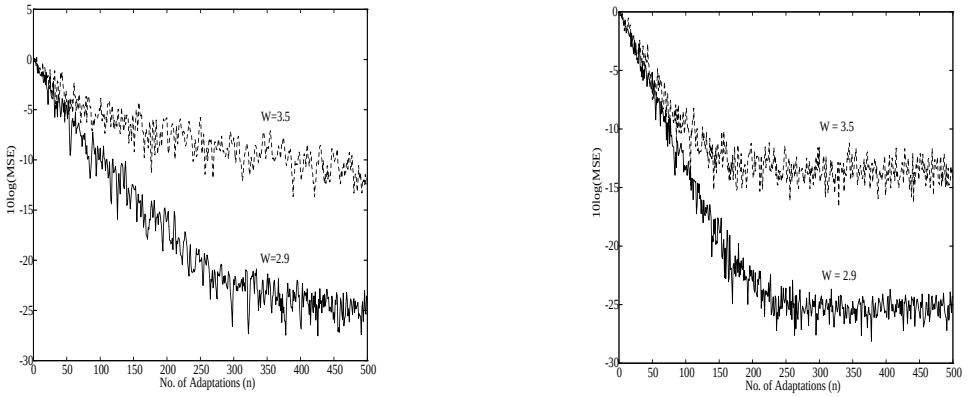

Figure 1: Learning curves with two values of W for the LMS algorithm (left) and for the ULV-LMS algorithm (right) (Curves are averages of 20 runs). ©1994 IEEE

An increase in step size usually implies an increase in the misadjustment error. The subspaces which belong to small singular values are dominated by noise and would tend to increase the noise in the solution. Thus by not adapting in those subspaces, we can reduce the misadjustment error. This can be simply done by setting those diagonal entries of $\mathbf{M}$, which correspond to projections of the tap weights onto these subspaces, to zero.

5 SIMULATION RESULTS

We illustrate the performance of our procedure with a simple example in which a white noise random sequence $a(n)$ that can take the values ± 1 with equal probability is filtered with a 3 tap FIR filter whose impulse response is a raised cosine $h(n) = (1+\cos(2\pi(n-2)/W))/2$, $n = 1, 2, 3$. White Gaussian noise is added to the output and an 11 tap equalizer is adaptively constructed using the LMS and ULV-LMS algorithms (Fig. 1). Note that whereas the speed of convergence of the traditional LMS algorithm depends heavily on the eigenvalue spread of the input covariance matrix as determined by W, the ULV-LMS algorithm has no problem adapting to the environment even when W is large ($W = 3.5$) and the condition number of the input covariance matrix is correspondingly large ($\lambda_{\max}/\lambda_{\min} = 47.4592$).

An Adaptive Line Enhancer (ALE) experiment was also conducted to illustrate the performance of the algorithm when the adaptation is done only in the signal subspaces. The input to the ALE was chosen to be $0.1\cos(\frac{\pi}{15}n) + \cos(\frac{5\pi}{16}n)$ corrupted by white Gaussian noise of variance 0.0001. The autocorrelation matrix of the input to the ALE has only four significant eigenvalues, which could be grouped into two clusters. The ALE was adapted using both the LMS and the ULV-LMS algorithms. The ULV-LMS algorithm was adapted only in the subspaces corresponding to the two large singular value clusters. The superior performance of the ULV-LMS algorithm can be seen from the learning curves are plotted in Fig. 2.

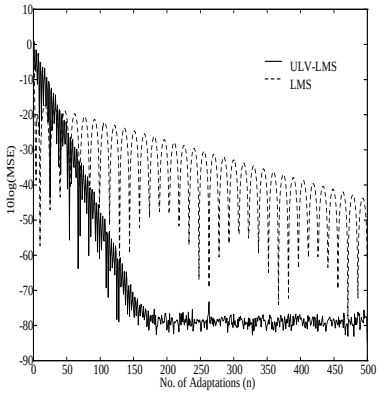

Figure 2: Learning curves for the ALE experiment (both methods averaged over 20 runs). ©1994 IEEE

Acknowledgements

This work was supported in part by ONR under grant N00014-92-J-1678, AFOSR under grant AF/F49620-93-1-0151DEF, DARPA under grant USDOC/60NANB2D1272, and NSF under grant CCR-9405380. Figures 1, 2 from [6] are used by permission.

References

[1] G.H. Golub, C.F. Van Loan. *Matrix computations.* Johns Hopkins Univ. Press, 1988.

[2] S. Haykin, *Adaptive Filter Theory*, 2nd ed., Prentice Hall, 1991.

[3] N. J. Higham, *A survey of condition number estimators for triangular matrices*, SIAM Rev. 29:575-596, 1987.

[4] D.F. Marshall, W.K. Jenkins, J.J. Murphy, "The Use of Orthogonal Transforms for Improving Performance of Adaptive Filters," IEEE Trans. Circ. & Sys. 36:474-483, 1989.

[5] G.W. Stewart, "An Updating Algorithm for Subspace Tracking," IEEE Trans. Signal Proc. 40:1535-1541, 1992.

[6] S. Hosur, A. H. Tewfik and D. Boley, "Generalized URV Subspace Tracking LMS Algorithm," ICASSP-94 III:409-412, Adelaide, Australia, 1994.

[7] S. Hosur, A. H. Tewfik and D. Boley, "Generalized ULV Subspace Tracking LMS Algorithm," Under Preparation.