AN IMPROVED SEQUENTIAL BACKWARD SELECTION ALGORITHM FOR LARGE-SCALE OBSERVATION SELECTION PROBLEMS

Stanley J. Reeves

Department of Electrical Engineering
Auburn University
Auburn, AL 36849
(334) 844-1821
sjreeves@eng.auburn.edu

ABSTRACT

Some signal reconstruction problems allow for flexibility in the selection of observations and hence the signal formation equation. In such cases, we have the opportunity to determine the best combination of observations before acquiring the data. We analyze the computational complexity of various forms of sequential backward selection (SBS) to select observations. In light of this analysis, we present a computationally improved algorithm for large-scale observation selection problems.

1. INTRODUCTION

We consider a signal y, which is a linearly transformed version of x observed in the presence of additive noise. This signal is described by

$$y = Ax + u, \tag{1}$$

where u is additive noise. Suppose that $A \in \mathbb{C}^{m \times n}$, where $m \geq n$. The goal is to reconstruct a good estimate of x given the observed signal y.

In many applications, the matrix A is known a priori, but the elements of y are not. Observing the elements of y may be expensive, time-consuming, or risky; in such cases we desire to limit the number of observations. This is the case in certain problems in magnetic resonance (MR) imaging and MR spectroscopic imaging (MRSI) [1–3]. The same concept is applicable in placing sensors in control problems [4] and remote sensing problems [5] and in determining the geometry of antenna arrays. The process is related to statistical experiment design, in which the experimental data are chosen to provide the best information about the unknown regression parameters given a specific regression model [6].

We would like to observe the k of m elements of y that provide the best possible reconstruction of x, using only the information from the A matrix to make the selection of observations. This is equivalent to choosing the rows of A that correspond to the best observations to acquire. We call this problem observation selection [7]. We must define a criterion for the optimal choice of rows and deal with the resulting combinatoric optimization problem.

In previous work, we derived a sum of squared errors (SSE) criterion as a function of the rows of A under the assumption of zero-mean i.i.d. noise and a least-squares reconstruction [7]. We also proved that the criterion (2) increases monotically as rows are removed from A. Establishing this property allowed us to apply branch-and-bound (B&B) to the optimization problem to determine an optimal combination of observations (rows). B&B is much more efficient than exhaustive search and also yields an optimal result. However, B&B can still be computationally prohibitive for even moderately sized problems. Therefore, we also proposed the use of sequential backward selection (SBS). SBS sequentially eliminates one row at a time until k rows remain. Although this approach is suboptimal, it eliminates the combinatoric problem. Furthermore, we have shown that a certain level of performance can be guaranteed if SBS is used [8].

In the next section, we derive efficient implementations of the SBS algorithm and analyze the computational requirements of these implementations. In Section 3, we propose an improvement for the case where the matrix A is too large to be stored or inverted directly. In Section 4, we demonstrate the algorithm with simulations.

This work was supported by a Biomedical Engineering Research Grant from the Whitaker Foundation

2. SBS ALGORITHMS

2.1. Selection Criteria

If the noise u is i.i.d. and the reconstruction of x is performed via least squares, we have shown [7] that the SSE in the reconstruction is proportional to

$$E(A) = \operatorname{tr}\left(A^H A\right)^{-1} \tag{2}$$

A more general form of the criterion above is

$$E(A) = \operatorname{tr}\left(A^H A + K\right)^{-1} \tag{3}$$

where K is Hermitian and nonnegative definite. This form covers two cases:

1. In certain applications, some of the observations may not be optional. For example, in a control sensor selection problem, some of the sensors may be pre-existing; therefore, the corresponding observations are available at no extra cost. In other applications, a previously selected subset of observations may be available that is known to be a good starting point for selecting further observations. In still other applications, a subset of observations may already have been acquired before selecting subsequent observations for acquisition.
2. In the case of reconstruction with a Wiener filter, the selection criterion will have the form of (3), where $K = \sigma_u^2 R_x^{-1}$ [9].

Our results are equally applicable for this general case.

2.2. SBS Options

Sequential backward selection begins with a candidate matrix and sequentially eliminates the least important row at each step until the desired number of rows remain. A derivation of the algorithm is given in [7], but we sketch it again here using the more general form of the criterion. Let a_i represent row i of A, and $B = (A^H A + K)^{-1}$. If we eliminate a_i from A, the modified B is given by the Sherman-Morrison matrix inversion formula [10] as

$$\tilde{B} = B + \frac{B a_i^H a_i B}{1 - a_i B a_i^H} \tag{4}$$

Taking the trace of both sides gives

$$\operatorname{tr}\tilde{B} = \operatorname{tr} B + \frac{\operatorname{tr} B a_i^H a_i B}{1 - a_i B a_i^H} \tag{5}$$

and using the property that $\operatorname{tr} CDE = \operatorname{tr} DEC$, we obtain

$$\operatorname{tr}\tilde{B} = \operatorname{tr} B + \frac{a_i B B a_i^H}{1 - a_i B a_i^H} \tag{6}$$

Therefore, the criterion is minimized at each step by eliminating the row (observation) that minimizes

$$\frac{a_i B B a_i^H}{1 - a_i B a_i^H} \tag{7}$$

We now consider some options for incorporating the simplified criterion into an SBS algorithm.

2.2.1. No Matrix Storage

If the candidate matrix is too large to be stored, we must solve a linear system to compute $(A^H A)^{-1} a_i$ for each row in the candidate matrix. While some matrices may have a special structure that can be exploited for greater efficiency, we consider the general case here. If matrix storage is not possible, the only alternative in general is an iterative solution. Since conjugate gradients converges in n iterations for n unknowns (assuming infinite precision), we consider the use of conjugate gradients here. The problem can be evaluated by

$$\phi(v_i) = \|e_i - A v_i\|^2 \tag{8}$$

where e_i is the ith column of an $m \times m$ identity matrix. The SBS criterion (7) can then be computed as

$$\frac{v_i^H v_i}{1 - a_i v_i} \tag{9}$$

since $(A^H A)^{-1} = (A^H A)^{-H}$.

With this strategy, we need $n(2n^2+4n)$ flops for the conjugate gradients algorithm. Then for $m-j$ rows in the candidate matrix, the strategy requires a total of $(m-j)(2n^3+4n^2)$ flops. If we sum over all eliminated rows and retain only the highest-order terms, the approximate flop count for the algorithm is $(m^2-k^2)n^3$, where k is the final number of rows.

2.2.2. Storage of $(A^H A)^{-1}$

If a matrix can be stored, the computation can be reduced dramatically. First, the computation of the criterion for each candidate row reduces to a matrix-vector multiply to obtain v_i, and then we can use (9). Second, once the row to be eliminated has been determined, we can use (4) to update B. Retaining only the highest-order terms, the approximate flop count for the algorithm is $(m^2-k^2)n^2$.

2.2.3. Storage of $(A^H A)^{-1} A^H$

Alternatively, BA^H can be updated efficiently by using a slightly modified version of (4):

$$\tilde{B}A^H = BA^H + \frac{B a_i^H a_i B A^H}{1 - a_i B a_i^H} \tag{10}$$

and then deleting column i from the result. Note that in this approach no matrix-vector multiplies are required, since Ba_i^H is simply a column of BA^H. Retaining only the highest-order terms, the approximate flop count for the algorithm is $4(m^2-k^2)n+\frac{2}{3}n^3+3mn^2$.

This is generally more computationally efficient than the other algorithms. However, more memory is required to store BA^H.

3. IMPROVED ALGORITHM FOR LARGE-SCALE PROBLEMS

If we observe (7), we see that the quantities $a_iBa_i^H$ and $a_iBBa_i^H$ are needed for each row i to determine which row to eliminate at each step. For the case where A is too large to store, we observed that the quantity Ba_i^H can be computed for each remaining row at each elimination step by minimizing (8). From this, the criterion can be computed for each row. Unfortunately, if A is large, then minimizing (8) for every remaining row at every step may require a tremendous amount of computation!

Fortunately, the no-matrix-storage case can be made much more efficient by recursively computing the quantities $a_jBa_j^H$ and $a_jBBa_j^H$. Using (4), we can write the first quantity recursively as

$$a_j\tilde{B}a_j^H = a_jBa_j^H + \sigma_i a_jBa_i^H a_iBa_j^H \qquad (11)$$

where $\sigma_i = \frac{1}{1-a_iBa_i^H}$. Likewise, the second quantity can be written as

$$\begin{aligned} a_j\tilde{B}\tilde{B}a_j^H &= a_jBBa_j^H + 2\sigma_i Re\{a_jBBa_i^H a_iBa_j^H\} \\ &\quad +\sigma_i^2 a_jBa_i^H a_iBBa_i^H a_iBa_j^H \qquad (12)\end{aligned}$$

To compute these quantities, we must have available the vector quantities Ba_i^H and BBa_i^H. With these quantities available, we can compute all of the necessary quantities for updating (11) and (12) for each remaining row.

The quantity $v_i = Ba_i^H$ must be computed iteratively if B is too large to store. This can be accomplished by minimizing (8). The quantity $w_i = BBa_i^H$ can then be computed by iteratively minimizing

$$\phi(w_i) = \|v_i - Bw_i\|^2 \qquad (13)$$

with respect to w_i. With these quantities in hand, we can rewrite (11) and (12) as

$$a_j\tilde{B}a_j^H = a_jBa_j^H + \sigma_i|a_jv_i|^2 \qquad (14)$$

where $\sigma_i = \frac{1}{1-a_iv_i}$, and

$$\begin{aligned} a_j\tilde{B}\tilde{B}a_j^H &= a_jBBa_j^H + 2\sigma_i Re\{(a_jw_i)(v_i^Ha_j^H)\} \\ &\quad +\sigma_i^2|a_jv_i|^2(a_iw_i) \qquad (15)\end{aligned}$$

Note that v_i and w_i only need to be computed once per elimination step, since every remaining row can be evaluated from the recursively computed terms in (14) and (15).

This strategy requires two conjugate-gradient solutions per elimination step as well as m solutions to initialize the recursively computed values. (For some problems, the initial values may be known and do not have to be computed.) Thus, after retaining only the highest-order terms, the total flop count is $2(3m-2k)n^3$, which represents an order reduction as compared to the direct method.

4. NUMERICAL RESULTS

We considered four different candidate matrices to illustrate the computational complexity of the algorithms. The matrices were chosen to be 100×10, 1000×100, 100×20, and 1000×200. Flop counts for each of these are shown in Table 2. the The results of the SBS algorithm for row sizes from m down to n are shown in Figure 1 for a 100×10 matrix formed by selecting each element from a normal random number generator. The curve represents actual SSE achieved with SBS as a function of the number of rows remaining. The SSE increases modestly as rows are removed, and the upper bound is fairly tight.

The computational complexity requirements for the various forms of the SBS algorithm are shown in Table 2, assuming the number of rows chosen equals the number of columns. Note that the form of the algorithm that stores $(A^HA+K)^{-1}A^H$ generally requires the least computation. On the other hand, the memory requirements for this algorithm are significantly higher than for the other algorithms.

5. CONCLUSION

We have developed several sequential observation selection algorithms and analyzed their computational complexity. We found that there is a clear tradeoff between memory and computational requirements among the algorithms. In some cases, the memory requirements may be prohibitive, but this means that the computational burden may be significantly higher. Fortunately, our improved no-storage algorithm allows for only a modest increase in computation over storage of $(A^HA)^{-1}$. By exploiting the specific structure of the matrix A, it may be possible to reduce the complexity even more. We are continuing to investigate this possibility for applications of interest.

6. REFERENCES

[1] S. J. Reeves, "Selection of k-space samples in localized spectroscopy of arbitrary volumes of interest," Journal of Magnetic Resonance Imaging, vol. 5, pp. 245–247, March/April 1995.

[2] S. K. Plevritis and A. Macovski, "Alternative k-space sampling distributions for MR spectroscopic imaging," in Proceedings of the 1994 IEEE International Conference on Image Processing, pp. 11–14.

[3] Y. Cao and D. N. Levin, "Using an image database to constrain the acquisition and reconstruction of MR images of the human head," IEEE Transactions on Medical Imaging, vol. 14, pp. 350–361, June 1995.

[4] D. E. Reeves, A Comprehensive Approach to Control Configuration Design for Complex Systems. PhD thesis, Georgia Institute of Technology, 1991.

[5] N. T. Middleton and M. T. Harman, "A preprocessor for geotomographic imaging of irregular geometric scans," IEEE Transactions on Industry Applications, vol. 28, pp. 1148–1153, September/October 1992.

[6] S. D. Silvey80, Optimal Design: An Introduction to the Theory for Parameter Estimation. Chapman and Hall, 1980.

[7] S. J. Reeves and L. P. Heck, "Selection of observations in signal reconstruction," IEEE Transactions on Signal Processing, vol. 43, pp. 788–791, March 1995.

[8] S. J. Reeves and Z. Zhao, "New results on observation selection in signal reconstruction," in Proceedings of the 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. III, pp. 1676–1679, 1996.

[9] S. J. Reeves, "Selection of observations in magnetic resonance spectroscopic imaging," in Proceedings of the 1995 IEEE International Conference on Image Processing, vol. I, pp. 641–644, 1995.

[10] G. H. Golub and C. Van Loan, Matrix Computations. Baltimore, MD: Johns Hopkins University Press, 2nd ed., 1989.

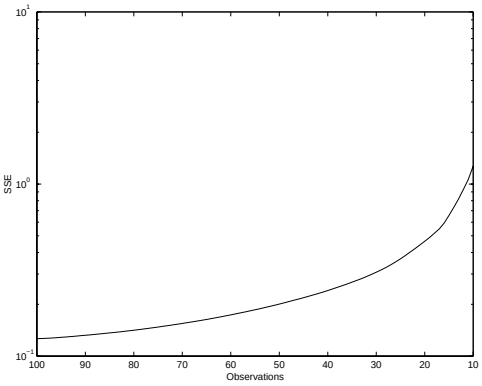

Figure 1: SSE as a function of remaining rows using the SBS algorithm

Table 1: Flops Required in Sequential Algorithms

No matrix storage	$(m^2-k^2)n^3$
Improved no storage	$2(3m-2k)n^3$
Storage of $(A^HA)^{-1}$	$(m^2-k^2)n^2$
Storage of $(A^HA)^{-1}A^H$	$4(m^2-k^2)n+\frac{2}{3}n^3+3mn^2$

Table 2: Computational Complexity Comparison

Matrix Size	Algorithm	Flops
100x10	No matrix storage	2.1e+06
	Improved no matrix storage	5.6e+05
	Storage of $(A^HA)^{-1}$	1.9e+05
	Storage of $(A^HA)^{-1}A^H$	9.9e+05
1000x100	No matrix storage	1.9e+11
	Improved no matrix storage	5.6e+09
	Storage of $(A^HA)^{-1}$	1.9e+09
	Storage of $(A^HA)^{-1}A^H$	1.1e+08
100x20	No matrix storage	3.1e+07
	Improved no matrix storage	4.2e+06
	Storage of $(A^HA)^{-1}$	1.4e+06
	Storage of $(A^HA)^{-1}A^H$	4.0e+05
1000x200	No matrix storage	2.9e+12
	Improved no matrix storage	4.2e+10
	Storage of $(A^HA)^{-1}$	1.4e+10
	Storage of $(A^HA)^{-1}A^H$	4.1e+08