Discrete Applied Mathematics 6 (1983) 49–53
North-Holland Publishing Company

AN ALGORITHM FOR COLLAPSING SIGN ALTERNATING SEQUENCES OF REAL NUMBERS

Daniel J. KLEITMAN*
Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Robert A. LEW
Departments of Nutrition and Biostatistics, Harvard School of Public Health, Boston, MA, USA

Received 25 May 1981
Revised 20 January 1982

A table with two rows and n columns may be thought of as two vectors with n components. The distance between the two rows then corresponds to the norm of the difference between the rows. We examine the problem of how to collapse the adjacent columns of the table while keeping the norm of the difference as large as possible. First a stepwise algorithm is given which achieves this end with respect to the norm of the vector of differences. After proving the optimality of the stepwise solution we extend the result to the norm which arises from minimizing the number of persons misclassified. The same algorithm suffices.

1. Origin of problem

Statisticians often must collapse large frequency tables into smaller ones in order to summarize data in a simple manner.

For example consider the comparison of survival by age in the following table.

	Age Category				
	0–20	21–40	41–60	61–80	81–100
Alive	0.4	0.3	0	0.3	0
Dead	0.1	0.2	0.4	0.1	0.2
Difference	+0.3	+0.1	−0.4	+0.2	−0.2

Each row total is one. The first line of the table is read "of the persons alive, 40% are ages 0–20, 30% ages 21–40, and 30% ages 61–80." We wish to group this data into three categories that preserve large differences between rows.

Note that in the row of differences $0.3+0.1+0.2=-(-0.4-0.2)$, the sum of positive differences equals minus the sum of the negative differences. Any collapse

*Supported in part by ONR Contract No. 00014-76-C-0366.

of the table preserves this property. One such collapse is:

	Age Category		
	0–40	41–60	61–100
Alive	0.7	0	0.3
Dead	0.3	0.4	0.3
Difference	+0.4	−0.4	0

With five categories the sum of positive differences is 0.6; with three categories the sum is 0.4. Note that merging 41–60 with 61–100 we obtain two categories 0–40 and 41–100 for which the sum is 0.4.

Our objective is to find an algorithm that collapses adjacent categories of a table with n categories into a table with k categories (k given, less than n) while keeping the sum of positive differences as large as possible. Merging adjacent categories with like signs leaves this positive sum unchanged. The problem becomes harder when the signs alternate and mergers bring about cancellation.

2. The algorithm

The 'greedy' algorithm for this problem proceeds by a sequence of mergings of neighboring categories, at each step merging a pair that reduces the sum of positive differences least. We shall show that, while the greedy algorithm itself will not always produce best solutions, one small modification of it will always produce an optimal solution here.

Consider a sequence of n real numbers $\Delta_1, \ldots, \Delta_n$ with zero sum. We seek a partition of the indices into k consecutive blocks that maximizes, over all such partitions, the absolute value of the sum of the Δ's within each new block.

We distinguish four cases as follows:

Case 1. For at least one index j, Δ_j and $\Delta_j + 1$ have the same sign. Excluding $j=1$ and $j=k$, let r be the index $2 \le j \le n-1$ with minimum $|\Delta_j|$.

Case 2. Not Case 1, $|\Delta_r| \le |\Delta_1|$ and $|\Delta_r| \le |\Delta_n|$.

Case 3. Not Case 1, $|\Delta_1| + |\Delta_n| \le |\Delta_r|$.

Case 4. Not Case 1, $|\Delta_1| + |\Delta_n| > |\Delta_r|$ and either $|\Delta_1| \le |\Delta_r|$ or $|\Delta_n| \le |\Delta_r|$.

We define the cost of merging j with $j+1$ as

$$C_{j+1} \equiv |\Delta_j| + |\Delta_{j+1}| - |\Delta_j + \Delta_{j+1}|.$$

We define algorithm G, the greedy algorithm, and the optimal algorithm 0, as follows:

Algorithm $G(n,k)$. If $n>k$ choose any index m such that the cost $C_{m+1} = \mathrm{Min}_{j=1,2,\ldots,n-1}[C_{j+1}]$ is minimal. For Case 2 $m=r$ suffices; for Cases 3 and 4

$m=1$ or $m=n-1$. Merge m with $m+1$, so that $\Delta'_m=\Delta_m+\Delta_{m+1}$; $\Delta'_j=\Delta_{j+1}$ for $j>m$, $\Delta'_j=\Delta_j$.

If $n=k+1$ stop; otherwise apply Algorithm $G(n-1,k)$ to this sequence.

Algorithm $O(n,k)$. Perform the first step of $G(n,k)$ unless the sequence lies in Case 4, and $n\equiv k \pmod 2$. In that case set $n=r$ and merge Δ's as in Algorithm G. If $n=k+1$ stop; otherwise apply $O(n-1,k)$ to the Δ sequence.

At first glance the difference between Algorithms O and G may appear peculiar. it can be understood as follows: when two adjacent Δ's alternate in sign, any merging of successive internal indices (not involving 1 or n) necessarily produce two adjacent Δ's with the same sign; whenever this happens the next merge is free. Thus, for $n-k=2$, an end merge for which this doesn't happen will be wasted, unless, as in Case 3, the other end merger when added to that of the first is cheaper than any alternative. This exceptional situation propagates by induction to whenever $n\equiv k \pmod 2$.

Theorem 1. *Algorithm $O(n,k)$ will always provide a sequence $\Delta'_1\cdots\Delta'_k$ the sums of the absolute values of which are maximal over all partitions possible here.*

Proof. It is straightforward, as already noted, that this algorithm is optimal for $n-k\leq 2$. The order in which mergers are performed in forming a partition is irrelevant, and after any one merger, the remaining ones must be optimal starting from the resulting $n-1$ length sequence. By induction therefore, we need only show that for $n-k\geq 3$ there is an optimal partition with parameters k,n for the given sequence for which the pair of indices merged in the first step lie in the same block.

In Case 1 the first merge involves j and $j+1$ with Δ_{j+1} having the same sign. If they are in separate blocks in an optimal solution which blocks have Δ's the same sign, we can merge these blocks at no cost and make an arbitrary split elsewhere to get an optimal solution containing the first merged pair together in a block. If one block has changed sign, by separating the Δ in it from the rest and putting it with the Δ_{j+1} block one actually improves the solution.

In the remaining cases the Δ's alternate in sign. Suppose now that $j,j+1$ form a recommended merge according to Algorithm O, and suppose that optimal partition P, which by induction we can assume is formed by use of Algorithm O after the first merging, has j and $j+1$ in separate blocks.

In Case 2 (or 4 with $n\equiv k \pmod 2$) suppose further that $|\Delta_j|\leq|\Delta_{j+1}|$; choose any merge compatible with P other than an end merge, if possible merging j with $j-1$. If the latter merge is chosen, then $O(n-1,k)$ will in its first step merge j with $j+1$, so that j and $j+1$ may be in the same block of an optimal partition. Otherwise, if the first merge chosen was $s,s\pm 1$ with $|\Delta_s|<|\Delta_{s\pm 1}|$ the succeeding free merge will link $s\pm 1$ with these 2 and the Δ value of the resulting block will have the same sign as $\Delta_{s\pm 1}$ and exceed it in magnitude. Thus the $j,j+1$ merge will again

be optimal according to O, and by the induction hypothesis will appear in some optimal partition.

In the remaining case the Δ's alternate in sign. Suppose now that $m, m+1$ form a recommended merge according to Algorithm O while no optimal partition has m and $m+1$ in the same block. By induction we can assume that an optimal partition P is formed by use of Algorithm O after the first merging.

Let s and $s+1$ be a pair of indices which belong to the same block of P. Since $k \geq n-3$ we can assume that

$$1 < s \quad \text{and} \quad s+1 < n.$$

Perform the merging of s with $s+1$ and the following free merging of these two with either $s-1$ or $s+2$. After this operation we get an alternate sequence of $n-2$ Δ's each of which is in canonical correspondence with an initial Δ. Note that the absolute values of the new Δ's are not smaller than those of the initial ones so that either m and $m+1$ are already merged or are a recommended merge in $O(n-2, k)$. A contradiction.

It is easy to verify that this algorithm can be implemented in a number of steps proportional to $n \log n$ by partially sorting the Δ's by absolute value and updating the sorted list with each pairwise merger.

3. Extensions

The problem spawns a variety of other questions, some relating to combinatiorics, others to statistics. As an example of the former, consider collapsing n categories into $k < n$ categories while *minimizing* the sum of positive differences. This is a version of the knapsack problem that can be handled by a dynamic programming algorithm [1]; it seems to require on the order of n^3 steps. When the categories have no intrinsic order so that adjacency need not be preserved, then the original problem becomes trivial. Suppose instead of age categories, we have color categories. Then merge together all colors with positive Δ and, separately, all colors with negative Δ. All such mergers are free and only two categories result. The problem is solved.

For statisticians, questions arise when we alter the cost of mergers. In the age category problem suppose a loss results from misclassifying a person. Suppose for example we have the following table.

	Age Category		
	0-40	41-60	61-100
Alive	10	3	8
Dead	2	4	5
Difference	8	−1	3
Minimum	2	3	5

In terms of the 'raw' frequencies above, the classification which minimizes loss declares persons 0–40 and 61–100 'alive', and persons 41–60 'dead'. If errors are of equal weight in either direction the total loss is the sum of the minima, $2+3+5=10$. Merging all three categories results in 21 alive and 11 dead so that declaring all to be alive yields a loss of 11. One easily shows that in terms of this loss function free mergers arise when adjacent Δ_j have the same sign. Also on any step the minimum loss occurs when the category with the minimum $|\Delta_j|$ is merged with its adjacent categories. it follows that in all aspects the problem of reducing n categories to k categories while minimizing the loss due to misclassification is in all ways analogous to our original problem except that the cost of merger differs. Thus for the 'misclassification problem', we have

Theorem 2. *Algorithm $O(n,k)$ applied to a $2\times n$ table of raw frequencies with loss equal to the number of misclassified persons (the sum of the k resulting category minima) yields the k category table with minimum loss.*

The result extends to asymmetric loss functions; that is, a penalty of h units for declaring a dead person to be alive and one unit for declaring a living person to be dead. The h units lost for a single person misclassified is equivalent to one unit lost on h persons. Thus multiplying the number of dead persons by h in the raw frequency table restores equal unit losses, the form of the problem resolved by Theorem 2.

Acknowledgement

The authors would like to thank the referee for helpful suggestions.

References

[1] M.R. Garey and D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness (Freeman, San Francisco, 1980).