Mathematical Programming 18 (1980) 146–154.
North-Holland Publishing Company

A NOTE ON SOLUTION OF LARGE SPARSE MAXIMUM ENTROPY PROBLEMS WITH LINEAR EQUALITY CONSTRAINTS

Jan ERIKSSON

Linköping University, Sweden

Received 26 October 1978
Revised manuscript received 25 June 1979

This paper describes a method to solve large sparse maximum entropy problems with linear equality constraints using Newtons and the conjugate gradient method. A numerical example is given to introduce the reader to possible applications of entropy models and this method. Some experience from large scale problems is also reported.

Key words: Convex Programming, Maximum Entropy Problem, Forecasting Models.

1. Introduction

In this paper we will describe an algorithm for solving the maximum entropy problem with linear equality constraints

$$\begin{aligned} &\text{minimize} \quad \sum_{j=1}^{n} x_j \ln(x_j/x_j^0), \\ &\text{subject to} \quad Ax = g, \quad x \geq 0, \end{aligned} \tag{1}$$

where x^0 is non-negative and A is an $m \times n$ matrix. The algorithm is particularly efficient when the matrix A is sparse (i.e. contains many zeroes) and m is much less than n. This problem arises from a minimum information principle [10].

Large sparse problems of this type occur e.g. in the following situation (cf. section 4). We know a flow table x^0 for a certain period $[T - \Delta T, T]$ and want to compute a flow table x for the period $[T, T + \Delta T]$. We have some information for the new period (exact and/or forecasts), which can be expressed as a linear system $Ax = g$. It can then be shown that the solution of (1) will give us the most probable solution. For the concept of entropy, see e.g. [5].

To solve the entropy problem we transform (1) to a system of non-linear equations, which is solved by Newtons method. In each step of Newtons method the resulting linear system is solved by a scaled version of the conjugate gradient method. This approach has several advantages as will be explained in the following sections.

2. Application of Newtons method

The Lagrangian for the problem (1) is

$$L(x, \beta, \gamma) = \sum_{j=1}^{n} x_j \ln(x_j/x_j^0) + \beta^T(g - Ax) - \gamma^T x,$$

where $\beta \in \mathbf{R}^m$ and $\gamma \in \mathbf{R}^n$. This gives us the Kuhn–Tucker conditions

$$\ln(x_j/x_j^0) + 1 - \beta^T a_{\cdot j} - \gamma_j = 0, \quad j = 1, 2, \dots, n,$$
$$g - Ax = 0, \qquad \gamma^T x = 0, \quad \gamma \geq 0,$$

where $a_{\cdot j}$ denotes the jth column of the matrix A.

Since $\sum_{j=1}^{n} x_j \ln(x_j/x_j^0)$ is a strictly convex function and x and $g - Ax$ are linear it follows that a necessary and sufficient condition for the existence of a unique solution of the minimization problem (1) is that there is a solution of the Kuhn–Tucker conditions (see e.g. [4]). The first Kuhn–Tucker condition can be written

$$x_j = x_j^0 \exp(\beta^T a_{\cdot j} - 1 + \gamma_j), \quad j = 1, 2, \dots, n \tag{2}$$

from which it follows that the condition $x \geq 0$ is satisfied if and only if $x^0 \geq 0$, i.e. the constraint $x \geq 0$ disappears. Therefore, substitution of the first Kuhn–Tucker condition into the second gives us

$$P_i(\beta) = \sum_{j=1}^{n} a_{i,j} x_j^0 \exp\{\beta^T a_{\cdot j} - 1\} - g_i = 0, \quad i = 1, 2, \dots, m \tag{3}$$

and the problem to be solved is now $P(\beta) = 0$. This system of non-linear equations can also be viewed as a condition for a stationary point for the dual formulation (minimize $\sum_{j=1}^{n} x_j(\beta) - \beta^T g$).

To apply Newtons method to the system of non-linear equations (3) we determine the related Jacobian matrix $P'(\beta)$. We have

$$\frac{\partial P_i}{\partial \beta_k} = \sum_{j=1}^{n} a_{i,j} x_j^0 \exp\{\beta^T a_{k,j} - 1\} a_{k,j} = \sum_{j=1}^{n} a_{i,j} x_j a_{k,j}, \quad 1 \leq i, k \leq m$$

and thus the Jacobian matrix can be expressed as

$$P'(\beta) = \left(\frac{\partial P_i}{\partial \beta_k}\right) = AXA^T, \quad \text{where } X = \operatorname{diag}(x_1, x_2, \dots, x_n).$$

The idea to apply Newtons method to entropy problems is due to Erlander [3].

We assume in the following that the system $AX^0 y = g$, $y \geq 0$, is consistent. Then there is a solution x^* to the minimization problem (1) and for some corresponding β^* the Kuhn–Tucker conditions are satisfied and $P(\beta^*) = 0$. Note that if β^* is a solution to $P(\beta) = 0$ then $AX^0 y = g$, $y \geq 0$, is consistent because of (3). Thus our assumption is necessary and sufficient for (1) to have a unique solution.

Suppose first that the rows in the matrix AX^0 are linearly independent, i.e. $\text{rank}(AX^0) = m$. Then, from (2) it follows that the Jacobian AXA^T is positive definite and Newtons method converges (Newton–Kantorovich theorem [6]) if the initial value of β, β^1, is sufficiently close to the solution β^*. We write Newtons method as follows:

Take

$$x_j^1 = x_j^0 \exp\{a_{\cdot j}^T \beta^1 - 1\}, \quad j = 1, 2, \ldots, n, \tag{4}$$

and for $\nu = 1, 2, 3, \ldots$ compute $x^{\nu+1}$ from

$$AX^\nu A^T \Delta\beta^\nu = -(Ax^\nu - g), \tag{5}$$

$$x_j^{\nu+1} = x_j^\nu \exp\{a_{\cdot j}^T \Delta\beta^\nu\}, \quad j = 1, 2, \ldots, n. \tag{6}$$

If $\text{rank}(AX^\nu A^T) = \text{rank}(AX^0) < m$, then (5) has not a unique solution $\Delta\beta^\nu$. However, $x^{\nu+1}$ is uniquely determined, because the component of $\Delta\beta^\nu$ in the null-space of $AX^\nu A^T$ is annihilated in (6). (Note that if $AX^\nu A^T y = 0$, then we have $x_j^\nu a_{\cdot j}^T y = 0$, $j = 1, 2, \ldots, n$, and $x_j^\nu \exp\{a_{\cdot j}^T \Delta\beta^\nu\} = x_j^\nu \exp\{a_{\cdot j}^T (\Delta\beta^\nu - y)\}$ because either $a_{\cdot j}^T y$ or x_j^ν is equal to zero). In this case we can eliminate equations in the system $Ax = g$ in such a way that we get a new system $Mx = h$ with $\text{rank}(AX^0)$ equations. If we insert M and h instead of A and g in (4)–(6) and choose a corresponding initial value of β we will get the same sequence x^ν, $\nu = 1, 2, 3, \ldots$, as we get for the original system. We conclude that the sequence x^ν, $\nu = 1, 2, 3, \ldots$, in (6) converges to the solution x^* independent of the rank of AX^0, if the initial value β^1 is chosen sufficiently close to a solution β^*.

3. Solution of the systems of linear equations

If the matrix A is sparse, it also can happen that the Jacobian AXA^T becomes sparse. If also the Cholesky factor of this matrix is sparse, then the system (5) can be solved efficiently by a direct method. However, in many practical applications AXA^T is not sparse, and then iterative methods are advantageous to use. Note also that we do not need high relative accuracy in the solution to (5).

To solve the system of linear equations (5) the conjugate gradient method (including scaling [1], [9]) is used. The following algorithm corresponds to a symmetric diagonal scaling of $AX^\nu A^T$ to have unit diagonal elements.

$$\begin{aligned}
&r := -(Ax^\nu - g), \quad r_0 := r^T r \\
&D = \text{diag}(d_1, d_2, \ldots, d_m), \quad \text{where } d_i := 1 \Big/ \Big(\sum_{j=1}^{n} a_{i,j}^2 x_j^\nu\Big) \\
&p := Dr, \quad r_2 := r_1 := r^T p, \quad \Delta\beta^\nu := 0 \\
&\text{for } k := 1, 2, \ldots, MCG \text{ do} \\
&\quad q := AX^\nu A^T p, \quad q_1 := q^T p \\
&\quad s := r_2/q_1, \quad r_3 := r_2
\end{aligned} \tag{7}$$

$\Delta\beta^\nu := \Delta\beta^\nu + s \times p, \quad r := -s \times q, \quad r_2 := r^T Dr$
if $r_2/r_1 < \epsilon^2$ then stop
$t := r_2/r_3, \quad p := Dr + t \times p$

This method works well even when the matrix $AX^\nu A^T$ is only positive semidefinite (i.e. $\text{rank}(AX^0) < m$) and then $\Delta\beta^\nu$ converges to the solution of minimum Euclidian norm [1]. The value of *MCG* should only be considered as a protection and can be set equal to e.g. $m+2$ [8]. The active termination criterion in (7) is normally the check if the Euclidian norm of the scaled residual in the conjugate gradient method has decreased to at least ϵ times the initial value.

Tests have been made with different choices of ϵ. In Fig. 1, the number of matrix by vector multiplications for the whole algorithm, NG, is given as a function of ϵ. The numbers on the curve are the required number of Newton steps. The example is derived from the first problem given in this paper where only ϵ is changed. For this and also some other examples there were nearly no changes in computing time for the whole algorithm when ϵ varied in the range [0.05, 0.2]. The reason for this is that the updating of x^ν (6), which also gives the new Jacobian, is nearly as fast as one step of the conjugate gradient method. Other termination criterias (e.g. the one described in [7]) have also be considered, but they are inefficient to use for the same reason. The following three advantages of the scaled conjugate gradient method are important: it converges to a minimum norm solution, the required memory space is not more than $2n+5m$ words and it requires only the products Ax, A_2x and A^Tz, where A_2 is the matrix $(a_{i,j}^2)$. A faster algorithm than the conjugate gradient method can be more expensive to use if it requires more memory space. Note that it is often possible to write efficient codes for the products Ax, A_2x and A^Tz even when we

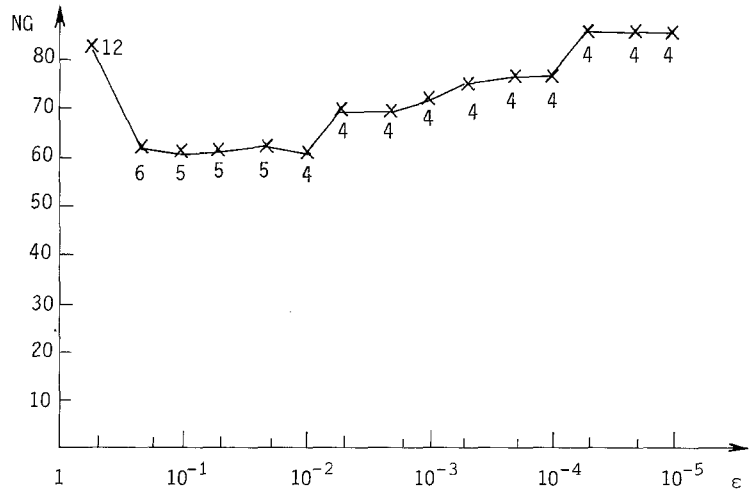

Fig. 1.

have a very compact but complicated description of the constraints. For an example see [2].

We remark that it is possible to apply a conjugate gradient method directly to the non-linear problem. However, this is not efficient because twice as much matrix by vector multiplications are required in each step compared to the conjugate gradient method applied to the linear problem. Besides, in Newtons method it is easy to implement accuracy criteria and we do not need to give a restart condition for the conjugate gradient method.

4. Numerical examples

The algorithm described here has been implemented as a FORTRAN program in [2]. We first describe a small numerical example to introduce the reader to possible application of this algorithm. In this example we have a model of household changes in five years periods for the capital of Sweden, Stockholm. The changes can be described in a flow table:

	b_1	b_2	b_3		b_n	
a_1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$\cdots$	$x_{1,n}$	x_{1*}
a_2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$\cdots$	$x_{2,n}$	x_{2*}
a_3	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$\cdots$	$x_{3,n}$	x_{3*}
$\vdots$						
a_m	$x_{m,1}$	$x_{m,2}$	$x_{m,3}$	$\cdots$	$x_{m,n}$	x_{m*}
	x_{*1}	x_{*2}	x_{*3}		x_{*n}	x_{**}

where $x_{i,j}$ means the number of individuals which change from category a_i to category b_j during the five years period. We have also introduced the notations

$$x_{i*} = \sum_{j=1}^{n} x_{i,j}, \quad i = 1, 2, \ldots, m,$$

$$x_{*j} = \sum_{i=1}^{m} x_{i,j}, \quad j = 1, 2, \ldots, n,$$

and

$$x_{**} = \sum_{j=1}^{n} x_{*j} \quad \left(\text{or } x_{**} = \sum_{i=1}^{m} x_{i*}\right).$$

In this example we have $n = 14$ and $m = 13$. The categories are

a_1 individuals born in Stockholm during the period,

a_2 in-migrators during the period not older than 44 years,

a_3 in-migrators during the period older than 44 years,

b_1 individuals not older than 44 years that have died during the period,

b_2 individuals older than 44 years that have died during the period,

b_3 out-migrators during the period not older than 44 years,

b_4 out-migrators during the period older than 44 years

Table 1
The period 1970–1975

772	0	11359	0	0	0	3235	0	26299	197	32676	936	18459	1717	95650
796	163	39420	367	33116	19298	16623	7287	15094	3261	13667	6359	5035	4462	164948
0	1330	0	3265	0	6179	453	2479	342	690	414	721	22	405	16300
247	89	37192	1147	1680	733	6089	1632	1996	265	865	130	68	18	52151
0	11589	0	8212	0	89589	397	435	451	195	408	284	450	400	112410
358	341	28689	481	2239	1011	31607	2873	12451	1822	5010	916	394	117	94309
140	24408	509	3334	263	27505	205	165583	13	134	15	10	16	14	222199
423	243	23110	274	1507	859	4815	1489	57444	9606	23512	4722	8413	2154	138571
122	11345	1555	3937	3779	1447	3742	54088	2224	63969	243	1193	18	238	147900
456	327	24080	368	2672	1931	334	182	17746	4025	89106	24655	28970	10270	205122
304	6944	1659	4080	9777	172	27030	323	4862	49358	809	11301	223	198	117040
370	212	2289	82	3119	3005	756	355	123	5	37549	13517	54941	25199	141522
170	2851	1038	2204	9831	289	7899	90	5473	54	3190	34193	125	14173	81580
4150	59842	170900	27801	67983	152018	103185	242816	144518	133581	207464	98937	117134	59365	1589702

Table 2
The period 1975–1980

769	0	8437	0	0	0	3095	0	25593	192	32072	919	18229	1696	91002
878	170	32423	316	33328	19421	17611	7720	16267	3514	14855	6912	5506	4880	163801
0	1374	0	2787	0	6168	476	2605	366	738	446	777	24	439	16200
401	137	45050	1454	2490	1086	9500	2546	3168	421	1385	208	110	29	67985
0	16303	0	9545	0	121767	568	622	656	284	599	417	665	591	152017
436	393	26097	458	2492	1125	37033	10396	14840	2172	6023	1101	477	142	103185
161	26528	437	3039	276	28883	227	183037	15	151	17	11	18	16	242816
470	255	19139	237	1527	870	5136	1588	62335	10424	25733	5168	9264	2372	144518
115	10098	1093	2895	3250	1244	3388	48966	2048	58910	226	1108	17	222	133580
482	327	19008	304	2581	1865	340	185	18354	4163	92952	25719	30406	10779	207465
268	5795	1093	2813	7883	139	22942	274	4198	42616	704	9841	195	173	98934
312	169	1439	54	2400	2312	612	288	101	4	31204	11233	45937	21069	117134
123	2031	584	1298	6768	199	5724	65	4035	40	2372	25424	94	10603	59365
4420	63580	154800	25200	62995	185079	106652	258292	151976	123629	208588	88838	110942	53011	1598002

and for $k = 1, 2, 3, 4$ and 5:

$a_{2k+2} = b_{2k+3}$ k-person-households, household head not older than 44 years,

$a_{2k+3} = b_{2k+4}$ k-person-households, household head older than 44 years.

Now a flow table is wanted for the period 1975–1980 (Table 2). First we need an a priori flow table (i.e. x^0) and as such we take the flow table for the period 1970–1975 (Table 1). From this table we also get some of the new constraints. The new values x_{i*}, $i = 4, 5, \ldots, 13$, in Table 2 are set equal to the old values x_{*j}, $j = 5, 6, \ldots, 14$, in Table 1. Further constraints are obtained from forecasts, namely the values for x_{i*}, $i = 1$, 2 and 3, x_{*j}, $j = 1$, 2, 3 and 4, (see Table 2) and $x_{*5} + x_{*6} + \frac{1}{2}(x_{*7} + x_{*8}) + \frac{1}{3}(x_{*9} + x_{*10}) + \frac{1}{4}(x_{*11} + x_{*12}) + (x_{*13} + x_{*14})/5.25 =$ 62800 = the sum of households. This problem has 18 constraints and $13 \times 14 =$ 182 unknowns. Note that m and n have different meanings here than in earlier sections.

All larger households are included in five-person-households and because of this we get the value 5.25. We also remark that we can use $x^1 = x^0$ as an initial vector in Newtons method (4) because there is a solution to $a_{\cdot j}^{\mathrm{T}} \beta^1 = 1$, $j = 1, 2, \ldots, n$. Then the program in [2] was used to compute Table 2.

The size of this example would increase considerably if we refine the partitions with respect to the age of the household head and insert the size of dwelling and living areas in the categories.

The second example is a test of the numerical behaviour of the algorithm for some large scale problems. The structure of the test problems can be described as (for more detailed information, see [2])

$$\text{minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} x_{i,j} \ln(x_{i,j}/x_{i,j}^0),$$

$$\text{subject to} \quad \sum_{j=1}^{n} x_{i,j} = x_{i*}, \quad i = 1, 2, \ldots, m$$

$$\sum_{i=1}^{m} x_{i,j} = x_{*j}, \quad i = 1, 2, \ldots, n.$$

Then the corresponding constraint matrix A in (1) has $m + n$ rows, mn columns and $2mn$ non-zero elements. For different values of m and n we get the results shown in Table 3.

Table 3
Test on different sizes of large scale problems

Problem number	1	2	3	4	5	6	7
Number of constraints	850	425	850	425	850	425	212
Number of unknowns	40000	39900	19824	20034	10056	10000	10011
Total number of matrix by vector multiplications	44	44	46	44	50	50	44
CPU-time in seconds on a DEC10-machine	86	88	48	44	25	25	22

We note that the number of matrix by vector multiplications is nearly independent of the size of the problem. This is not true when we use the unscaled conjugate gradient method, i.e. when we let $d_i = 1$, $i = 1, 2, \ldots, m$, in (7). For this example we get the large improvement from scaling for the first problem of Table 3, for which without scaling the number of matrix by vector multiplications increased to 202 and the CPU-time to 407 seconds.

Acknowledgment

The author is grateful for stimulating discussions during this work with Åke Björck, Tommy Elfving and Sven Erlander of Linköping University, Sweden.

This research has been supported by the Swedish Institute of Applied Mathematics.

References

[1] Å. Björck and T. Elfving, "Accelerated projection methods for computing pseudo-inverse solutions of systems of linear equations", Report LiTH-MAT-R-78-5, Linköping University, Linköping (1978).

[2] J. Eriksson, "Solution of large sparse maximum entropy problems with linear equality constraints", Report LiTH-MAT-R-78-2, Linköping University, Linköping (1978).

[3] S. Erlander, "Entropy in linear programs—an approach to planning", Report LiTH-MAT-R-77-3, Linköping University, Linköping (1977).

[4] G. Hadley, *Nonlinear and dynamic programming* (Addison-Wesley, London, 1970).

[5] S. Kullback, *Information theory and statistics* (Wiley, New York, 1959).

[6] J.M. Ortega and W.C. Rheinboldt, *Iterative solution of nonlinear equations in several variables* (Academic Press, New York, 1970).

[7] V. Pereyra, "Convergence of discretization algorithms", *SIAM Journal on Numerical Analysis* 4 (1967) 508–533.

[8] J.K. Reid, "On the method of conjugate gradients for the solution of large sparse systems of linear equations", in: J.K. Reid, ed., *Large sparse sets of linear equations* (Academic Press, New York, 1971) pp. 231–254.

[9] A. van der Sluis, "Condition numbers and equilibration of matrices", *Numerische Mathematik* 14 (1969) 14–23.

[10] F. Snickars and W. Weibull, "A minimum information principle—theory and practice", *Regional Science and Urban Economics* 7 (1977) 137–168.