Polytech Sorbonne

Mathématiques appliquées et informatique

Algorithme de Lemke sur un problème d'optimisation quadratique

Auteur :
Felix Choi

Encadrant :
Hacène Ouzia

18 octobre 2019

1 Introduction au Problème posé

L'objectif est de résoudre un problème d'optimisation quadratique par l'algorithme de Lemke.

Considérons le problème d'optimisation quadratique suivant :

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} < x, Hx > + < c, x > \\ \text{s.c.} \quad & Ax \leq b, \\ & x \in \mathbb{R}^n_+ \end{aligned} \tag{1}$$

où :

- H est une matrice symétrique d'ordre n
- la matrice A appartient à $\mathcal{M}_{m\times n}(\mathbb{R})$,
- les vecteurs c et b appartiennent à $\mathbb{R}^n$ et $\mathbb{R}^m$ respectivement

1.1 Un problème de complémentarité linéaire

En posant les conditions KKT du problème (1) on obtient :

$$\begin{aligned} & w - Mz = 0 \\ & w^t z = 0 \\ & w, z \geq 0 \end{aligned} \tag{2}$$

avec :

$$M = \begin{pmatrix} 0 & -A \\ A^t & H \end{pmatrix}, q = \begin{pmatrix} b \\ c \end{pmatrix}, w = \begin{pmatrix} y \\ v \end{pmatrix}, z = \begin{pmatrix} u \\ x \end{pmatrix}$$

Cela revient donc à résoudre un Problème de complémentarité linéaire (LCP).

2 L'algorithme de Lemke

Nous posons le problème LCP sous forme de tableau et faisons des opérations de pivots sur ce tableau.

	w	z	z$_0$	
w	I$_d$	-M	-1$_n$	q

2.1 Première itération

Le premier pivot se fait sur Z_0 , la ligne est choisit par :

$$S = argmin\{q_k : k = 1, \dots, n\}$$

avec n, la taille de w. Le pivot est donc Z$_{S,0}$

2.2 itérations suivantes et critère d'arrêt

Choix du pivot :

Le pivot se fait sur la colonne du Z_s, S correspondant à l'indice de la ligne du pivot de l'itération précédente. On choisit alors la ligne par :

$$S' = argmin\{\frac{q_k}{Z_{k,S}} : k = 1, \dots, n\}$$

Le pivot se note alors $Z_{S',S}$. A chaque opération de pivot, la variable associée Z_S entre en base et prend la place de la variable correspondante à la ligne S'.

critère d'arrêt

L'algorithme s'arrête lorsque Z_0 sort en base.

2.3 Theoreme de convergence

Soit le système suivant :

$$\begin{aligned} &w - Mz - 1_n Z_0 = q \\ &w^t z = 0 \\ &w, z, Z_0 \geq 0 \end{aligned} \tag{3}$$

Si chaque solution de base presque complémentaire du système (3) est non dégénérée et que la matrice M est co-positive, alors l'algorithme de Lemke converge en un nombre fini d'itérations.

3 Comparaison des performances avec les autres méthodes d'optimisation quadratique proposées sous Matlab

La fonction sous matlab permettant de résoudre un problème d'optimisation quadratique est *quadprog* (pour Quadratic Programming).

Prenons le problème quadratique (1) avec :

$$H = \begin{pmatrix} 2 & -2 \\ -2 & 4 \end{pmatrix}, c = \begin{pmatrix} -2 \\ -6 \end{pmatrix}, A = \begin{pmatrix} 1 & 1 \\ -1 & 2 \end{pmatrix}, b = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

comparaison des performances :

	nb itérations	temps de calcul
Lemke	4	0.0443 s
quadprog	5	0.0301 s

On remarque que bien que l'algorithme de Lemke fait moins d'itérations, il utilise plus de temps que la méthode de quadprog. Ce résultat se répète sur plusieurs cas de même taille (petite).

L'algorithme de lemke n'est pas optimal sur des problèmes de grande taille car l'opération de pivot devient trop coûteuse, en effet il faut recalculer tout les éléments du tableaux a chaque itérations.

De plus l'algorithme de Lemke n'est pas non-plus optimale sur des problèmes "creux" (ou les matrices sont creuses) car l'opération de pivot enlève la nature "creuse" du problème.

Conclusion

Bien que l'algorithme de Lemke résout les problèmes d'optimisation quadratiques sous les conditions de convergences, elle n'est pas la plus rapide. Les algorithmes utilisés par matlab sont en effet plus performant en utilisant des algorithmes tels que "La méthode des points intérieurs" ou "Algorithme à régions de confiance"

1 Code Matlab de l'algorithme de Lemke

```
%Resolution de probleme quadratiques par l'algorithme de Lemke
%
% Felix CHOI, 2019
%
% le probleme est suppose de forme : | min (1/2)<Ax,x> + <b,x>
%                                    |       <H,x> ≤ B
%L'algorithme sortira 2 vecteurs z=[u; x] et w=[y; v]

%%%% definition du probleme %%%%

A= [6 3 ; 3 4]
b= [-4; 2]
H= [3 2; -1 2]
B= [6 ;4]



%exemple du cours
%A= [2 -2 ; -2 4]
%b= [-2; -6]
%H= [1 1; -1 2]
%B= [2 ;2]


%%%% Reecriture sous forme LCP
[mA,nA] = size(A); %taille de A
[mH,nH] = size(H); %taille de H

%%% M
zero = zeros(nA);
M = [zero -H; H' A]
[mM,nM] = size(M);

%%% q
q= [B; b]
[mq,nq] = size(q);

%variable a trouver
w = zeros(mq,1);
z = zeros(mq,1);


%-----------------------------------%
%%%% Algorithme de Lemke %%%%
%-----------------------------------%
tstart = tic;

z0 = -1 * ones(mq);
Z = -M;

W = eye(mq);

%matrice indiquant les entrees et sorties en base
temp = zeros(mq,1);
```

```
for i=1:mq
    temp(i) = i;
end
base = [ones(mq,1)  temp] %colonne 1 pour le vecteur correspondant
                           %(1 pour w et 0 pour z)
                           %colonne 2 pour l'indice


tableau = [ W Z z0 q]

%-----------------------------------%
%%%%%%% iteration 1: %%%%%%%
k = find( q == min(q(:)))  %ligne du pivot
k0 = k;                    %ligne initial du pivot -> utilise ...
   pour critere
                           %d'arret

                           % pivot -> z0(k)
base(k,:) = [0 0];
% -- operation pivot -- %

%reduction de la ligne du pivot
a = 1/z0(k);         % a =-1
W(k,:) = a*W(k,:);
Z(k,:) = a*Z(k,:);
z0(k)  = a*z0(k);
q(k)   = a*q(k);

for i=1:mq
    if i ≠ k                  % ≠ -> !=
        a = -z0(i)/z0(k) ; %on doit avoir "z0(i) + a*z0(k) = 0"
        W(i,:) = W(i,:) + a*W(k,:);
        Z(i,:) = Z(i,:) + a*Z(k,:);
        z0(i)  = z0(i)  + a*z0(k);
        q(i)   = q(i)   + a*q(k);
    end
end

tableau = [ W Z z0 q];


%-----------------------------------%
% iterations jusqu'a condition d'arret -> z0 sortant, donc si on ...
   tombe sur
% k == k0

k_mem = k;        %on garde en memoire la ligne du pivot precedent
k=0    ;          %on entre dans la boucle k0 ne vaut jamais 0

nb_iter = 1;
while k ≠ k0,
    vec_temp = q./Z(:,k_mem);
    vec_temp(vec_temp  ≤ 0) = inf  ;  % on ecarte les valeurs ≤ 0
    k = find( vec_temp == min(vec_temp(:)));%on cherche le ...
       pivot=Z(k,k_mem)
    Z(k,k_mem);
    % -- operation pivot -- %

```

```
        %reduction de la ligne du pivot
        a = 1/Z(k,k_mem);        %on doit avoir "a*Z(k,k_mem) = 1"
        W(k,:) = a*W(k,:);
        Z(k,:) = a*Z(k,:);
        z0(k)  = a*z0(k);
        q(k)   = a*q(k);

        for i=1:mq
            if i ≠ k
                a = -Z(i,k_mem)/Z(k,k_mem) ; %on doit avoir
                                             %"Z(i,k_mem) + ...
                                                a*Z(k,k_mem) = 0"
                W(i,:) = W(i,:) + a*W(k,:);
                Z(i,:) = Z(i,:) + a*Z(k,:);
                z0(i)  = z0(i)  + a*z0(k);
                q(i)   = q(i)   + a*q(k);
            end
        end
        base(k,:) = [0, k_mem];

        tableau = [ W Z z0 q];




        k_mem = k;

        %%% limiteur d'iteration %%%
        nb_iter = nb_iter + 1;
        if nb_iter > 10
            k=k0;
            disp("maximum iteration reached.")
        end
    end
    telapse = toc(tstart);


    %interpretation de la matrice artificielle "base" pour ...
        determiner w et z
    for i=1:mq
        x = base(i,:);
        if x(1) == 0
            z(x(2)) = q(i);
        elseif x(1) == 1
            w(x(2)) = q(i)   ;
        end
    end
    disp("RESULT :")
    z
    w

    nb_iter
    telapse
```