UMONS

Université de Mons

UMONS

Optimisation linéaire
Récupération d'une image floutée (deblurring)

Projet d'optimisation linéaire

Élèves :
Nicolas Gatta
Arnaud Schellekens

Enseignant :
Nicolas Gillis

23 décembre 2022

Table des matières

1 Introduction

L'objectif de ce projet est l'étude du problème de minimisation $\min_{0\leq x\leq 1} \|Ax - \tilde{x}\|_1 + \lambda\|x\|_1$ qui permettra de reconstruire une image non floutée et non bruitée à partir d'une image floutée et bruitée.

2 Question 1 : Modélisation du problème

Il faut écrire le problème ci-dessus sous forme d'un problème d'optimisation linéaire :

$$\min_{0\leq x\leq 1} \|Ax - \tilde{x}\|_1 + \lambda\|x\|_1 \tag{1}$$

Nous savons que la norme 1 d'un vecteur est la somme des valeurs absolues de ses composantes, on applique cela dans la formule (1) ce qui nous donne :

$$\min_{0\leq x\leq 1} \sum_{i=1}^{n} |a_i x - \tilde{x}_i| + \lambda|x_i| \tag{2}$$

Grâce au a_i la n-ième ligne de A, x_i la n-ième ligne de x et n le nombre de lignes de A et $\tilde{x}$. En effet, ces matrices doivent avoir le même nombre de lignes et de colonnes dans le cas d'une addition/soustraction et doivent avoir le même nombre de lignes pour une multiplication pour que celles-ci puissent se produire. Enfin, en traitant les valeurs absolues, on obtient :

$$\begin{aligned} \min_{xi} \quad & \sum_{i=1}^{n} t_i + \lambda|x_i| \\ \text{t.q.} \quad & t_i \geq a_i x_i - \tilde{x_i} \\ & t_i \geq -a_i x_i + \tilde{x_i} \\ & x_i \geq 0 \\ & x_i \leq 1 \end{aligned} \tag{3}$$

3 Question 2 : Forme standard

Nous devons écrire le problème d'optimisation linéaire formulé à la question 1 sous forme standard.

$$\begin{aligned} \min_{xi} \quad & \sum_{i=1}^{n} t_i + \lambda x_i \\ \text{t.q.} \quad & t_i - s_1 = a_i x - \tilde{x}_i \\ & t_i - s_2 = -a_i x + \tilde{x}_i \\ & x_i + s_3 = 1 \\ & x_i \geq 0 \\ & s_1, s_2, s_3 \geq 0 \end{aligned} \tag{4}$$

4 Question 3 : Déflouter une image

Pour déflouter une image, on utilise comme demander dans la question la fonction ***glpk*** de Octave. Cette fonction va nous permettre de résoudre le problème d'optimisation linéaire suivant :

$$\begin{aligned} \min_{0\leq x\leq 1} \quad & \sum_{i=1}^{n} t_i + \lambda \tilde{x}_i \\ \text{t.q.} \quad & -t_i + a_i x \leq \tilde{x}_i \\ & -t_i - a_i x \leq -\tilde{x}_i \end{aligned} \tag{5}$$

Pour faire fonctionner la fonction de ***glpk***, il a fallu remplacer les deux contraintes $x \geq 0$ et $x \leq 1$ par des matrices d'une colonne et de n lignes (correspondant à la taille de $\tilde{x}$) remplies de 0 pour **lb** (borne inférieure) et remplies de 1 pour **ub** (borne supérieure).

En plus de cela, il a fallu créer une nouvelle variable contenant les types de contraintes utilisés pour chaque ligne. Vu que nous avons pris le temps de convertir toutes les contraintes « $\leq$ », la variable **cty** (type de contrainte) va être remplie de « U » correspondant aux contraintes avec des bornes supérieures.

Enfin, les n derniers éléments de la solution renvoyés par la fonction ***glpk*** vont être sélectionnés pour afficher l'image défloutée et débruitée, car les n premiers éléments correspondent aux t_i et les n derniers correspondent aux x_i

5 Question 4 : Le polyèdre

Il faut déterminer si la solution obtenue est un sommet du polyèdre c'est-à-dire si la solution est une solution admissible de base. La solution x est une solution admissible de base s'il y a n contraintes actives et linéairement indépendantes en x. Comme vu en cours, la solution du problème est un sommet du polyèdre. Cette réponse peut être obtenue via l'algorithme **estSommet** fonctionnant de la manière suivante :

- Il crée une matrice vide (**actives**) qui regroupera, les contraintes actives.
- Pour chaque contrainte, il vérifie si elle est active en x et si elle l'est, il la rajoute dans la matrice.
- Il calcule le rang de la matrice qui est le nombre de contraintes linéairement indépendantes. On a alors le nombre de contraintes actives et linéairement indépendantes.
- On a besoin de trouver $2n - m$ contraintes actives et linéairement indépendantes. Nous cherchons ces contraintes dans les contraintes de positivité et de $x_i \leq 1$.
- Si le nombre de contraintes actives et linéairement indépendantes est plus grand ou égal à $2n$ alors la solution est donc un sommet du polyèdre.

6 Question 5 : Sensibilité de la solution

Pour cette question, il faut étudier la sensibilité de la solution en fonction de la valeur de λ pour les images Example0.mat et Example1.mat. Les valeurs des erreurs relatives en fonction de la valeur de λ sont reprises dans les tableaux ci-dessous :

Pour « Example0.mat », on remarque que la valeur de λ qui convient le mieux sont les valeurs égales ou plus petites que 0 alors que la valeur qui convient le moins est 1 :

Valeurs de lambda	Erreur relative de reconstruction
1	82.5734 %
0	1.23e-12 %
1e-1	3.81e-12 %
1e-2	9.75e-12 %
1e-3	2.24e-12 %
1e-4	1.31e-12 %
1e-5	1.23e-12 %
1e-6	1.23e-12 %
1e-7	1.23e-12 %
1e-8	1.23e-12 %
1e-9	1.23e-12 %
1e-10	1.23e-12 %

Pour « Example1.mat », il est malheureusement impossible d'utiliser la méthode vu que «Example1» ne possède aucune variable xtrue étant la variable permettant de recréer l'image de base et ainsi pouvoir la comparer au résultat obtenu grâce à l'algorithme.

A	double	2304x2304	[0.09375, 0.0833...
L	double	1x1	48
l	double	1x1	48
xtilde	double	2304x1	[0.90171568627...

7 Annexe

7.0.1 deblurr

```
% Deblurring algorithm
%
% Given A and xtilde, solve the linear optimization problem:
%
% min_{0 <= x <= 1} ||A*xtilde - x||_1 + lambda ||x||_1
function x = deblurr(A, xtilde, lambda)

  n = length(xtilde);

  one = eye(n,n);
  oneCol(1:n,1:1) = 1;
  lambdaCol(1:n,1:1) = lambda;
  infini(1:n,1:1) = Inf;

  A=[-one,A; -one,-A];

  C = [oneCol;lambdaCol];

  B = [xtilde;-xtilde];

  LB = [oneCol * 0; oneCol * 0];%borne inf

  UB = [infini ; oneCol];%borne sup

  CTY = "";
  for i = 1 : 2*n
    CTY = strcat(CTY,"U");
  endfor

  sol = glpk(C, A, B, LB, UB, CTY);
  x = sol(n + 1:length(sol),1:1);

  %affiche dans la console si la solution est un sommet du polyÃÍdre
  estSommet(A, sol, B, n);

end
```

7.0.2 main

```
clear all; close all; clc;
format longG;

% Utiliser la fonction test ou imageTest ou lambda.
function x = main(xtilde, l, L, A)
  figure;
  imshow(reshape(xtilde,l,L));
  title('Image floutee et bruitee');

  % Choose parameter and solve problem
  lambda = 1e-4;
  x = deblurr(A,xtilde,lambda);

  % Display solution
  figure;
  imshow(reshape(x,l,L) );
  title('Image defloutee et debruitee');
  endfunction

function test(file)

  load(file);
  main(xtilde, l, L, A);
  endfunction

function imageTest()

  load Example0;
  x = main(xtilde, l, L, A);
  figure;
  imshow(reshape(xtrue,l,L));
  title('Image fournie');
  fprintf('L''erreur relative de reconstruciton L1 est de %2.2f %% +//0-n',
          norm(x-xtrue,1)/norm(xtrue,1)*100);

  endfunction

function lambda(file)
  load(file);
  errors = testLambda(A,xtilde,xtrue);
  display(errors)
  endfunction

test("Example0")
```

7.0.3 testLambda

```
% Retourne une matrice contenant les valeurs des erreur
% relative de reconstruciton en fonction des valeurs de lambda
%
% Premiere colonne: valeur de lambda
% Deuxieme colonne: erreur relative de reconstruciton

function errors = testLambda (A, xtilde, xtrue)

    errors=[0,0;1,0;1e-1,0;1e-2,0;1e-3,0;1e-4,0;1e-5,0;1e-6,0;1e-7,0;1e-8,
            0;1e-9,0;1e-10,0];

    for i=1: length(errors)
        lambda=errors(i,1);
        x = deblurr(A, xtilde, lambda);
        erreur=(norm(x-xtrue,1)/norm(xtrue,1))*100;
        errors(i,2)=erreur;
    end
end
```

7.0.4 estSommet

```
%Affiche dans la console si la solution est un sommet
%du polyedre de contraintes A

function estSommet(A,sol,B,n)
  actives=[];
  prod=abs(A * sol - B);
  for i=1:length(sol)
    if prod(i) <= 0.0001
        actives = [actives;A(i:i,:)];
    end
  end
  activesEtLinInd=sum(sol < 0.0001) + sum(abs(sol(1:n) - 1) < 0.0001);

  if rank(actives) + activesEtLinInd >= 2*n
      disp("La solution est un sommet du polyedre");
  else
      disp("la solution n'est pas un sommet du polyedre");
  end

end
```