Projet d'optimisation linéaire

Récupération d'une image floutée (deblurring)

Professeurs :
Nicolas Gillis
Arnaud Vandaele

Auteur :
Arnaud Palgen

UMONS
Université de Mons

Année académique 2017-2018

1 Intro

L'objectif de ce projet est l'étude du problème $\min_{0\leq x\leq 1} \|Ax-\tilde{x}\|_1+\lambda\|x\|_1$ afin de reconstruire de manière approchée une image défloutée et débruitée à partir de l'image floutée et bruitée ainsi que de la matrice de floutage associée.

2 Question 1

Nous devons écrire le problème suivant comme un problème d'optimisation linéaire :

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

Sachant que la norme 1 d'un vecteur est la somme des valeurs absolues de ses composantes, on a alors :

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

Avec a_i la ième ligne de A, x_i la 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 pour que les opérations de produit, somme et différence soient possibles. Enfin en traitant les valeurs absolues, on obtient :

$$\min_{x_i} \sum_{i=1}^{n} t_i+\lambda x_i$$

$$t_i \geq a_i x-\tilde{x}_i$$

$$t_i \geq -a_i x+\tilde{x}_i$$

$$x_i \leq 1$$

$$x_i \geq 0$$

Remarquons que la valeur absolue de $\lambda|x_i|$ a été supprimée. En effet, par l'avant dernière contrainte, x_i est positif.

3 Question 2

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

$$\min_{x_i} \sum_{i=1}^{n} t_i + \lambda x_i$$

$$t_i - s_1 = a_i x - \tilde{x}_i$$

$$t_i - s_2 = -a_i x + \tilde{x}_i$$

$$x_i + s_3 = 1$$

$$x_i \geq 0$$

$$s_1, s_2, s_3 \geq 0$$

4 Question 3

Pour déflouter l'image, la fonction linprog() de Matlab à été utilisée. Cette fonction résout les problèmes d'optimisation linéaires de la forme :

$$\min_{x} f' * x \; tel \; que \; A * x \leq b$$

Nous utilisons donc ce modèle pour la résolution du problème :

$$\min_{0 \leq x \leq 1} \sum_{i=1}^{n} t_i + \lambda \tilde{x}_i$$

$$-t_i + a_i x \leq \tilde{x}_i$$

$$-t_i - a_i x \leq -\tilde{x}_i$$

Pour des raisons d'efficacité, les contraintes $0 \leq x$ et $x \leq 1$ on été remplacées par l'utilisation des paramètres lb et ub de la fonction linprog qui délimitent respectivement les bornes inférieures et supérieures du domaine admissible. Toujours pour des questions d'efficacité, l'option *optimoptions('linprog','Algorithm','interior-point')*, force linprog à utiliser la méthode des points intérieurs qui, dans le cadre de ce projet, est plus efficace que l'algorithme du simplex utilisé par défaut.
Les n derniers éléments de la solution ont été sélectionnés pour afficher l'image défloutée et débruitée car il y a 2n éléments dans la solution et que les n premiers sont les t_i et donc les n derniers sont les x_i.

5 Question 4

Nous devons 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 (avec $x \in R^{2n}$) est une solution admissible de base s'il y a n contraintes actives et linéairement indépendantes en x. La solution du problème est un sommet du polyèdre. Cette réponse est obtenue via un algorithme implémenté dans matlab. Cet algorihme fonctionne de la manière suivante :

- Il crée une matrice vide (appelons la " Ac") qui regroupera, les unes en dessous des autres les contraintes actives.
- Pour chaque contrainte, il vérifie si elle est active en x . Si elle l'est, il la rajoute dans la matrice Ac
- Il calcule le rang de la matrice. Le rang de la matrice est donc le nombre de contraintes linéairement indépendantes. On a alors le nombre de contraintes actives et linéairement indépendantes.
- A ce stade nous avons m contraintes actives et linéairement indépendantes. Comme on en a besoin de 2n , il nous reste à trouver 2n-m contraintes actives et linéairement indépendantes à trouver. Nous allons chercher ces contraintes dans les contraintes de positivité et de $x_i \leq 1$. C'est pourquoi l'algorithme somme ensuite le nombre de composantes de la solution qui sont égales à zero ou à un. Ces contraintes sont forcément linéairement indépendantes.
- Donc si le nombre de contraintes actives et linéairement indépendantes est plus grand ou égal à 2n (car x appartient àR^{2n}) la solution est donc un sommet du polyèdre.

6 Question 5

Nous devons étudier la sensibilité de la solution en fonction de la valeur de λ pour les images Example0 et Example1.
Les valeurs des erreurs relatives en fonction de la valeur de λ sont reprises dans les tableaux ci-dessous pour les deux images.
Pour Example0, on peut remarquer que la valeur de λ qui convient le mieux est 1e-2 et que la valeur qui convient le moins est 1. Pour toutes les autres valeurs, la valeur de l'erreur relative de reconstruction est 0.0004% .

Pour Example1, on peut remarquer que les valeurs de λ qui semblent fonctionner le mieux sont 0 et toutes les valeurs plus petites ou égales à 1e-7. On peut remarquer également que les valeurs de λ qui conviennent le moins sont 1 et 0.1 .

On remarque également que les valeurs de l'erreur relative (sauf pour $\lambda = 1$) sont plus basses pour l'example0 que l'example1.

Valeurs de lambda	Erreur relative de reconstruction
1	82.5729%
0	0.0004%
1e-1	0.0004 %
1e-2	0.0003%
1e-3	0.0004%
1e-4	0.0004%
1e-5	0.0004%
1e-6	0.0004%
1e-7	0.0004%
1e-8	0.0004%
1e-9	0.0004%
1e-10	0.0004%

TABLE 1 – Example 0

Valeurs de lambda	Erreur relative de reconstruction
1	64.9233%
0	4.0045%
1e-1	11.0311%
1e-2	4.5324%
1e-3	4.0357%
1e-4	4.0126%
1e-5	4.0059%
1e-6	4.0046%
1e-7	4.0045%
1e-8	4.0045%
1e-9	4.0045%
1e-10	4.0045%

TABLE 2 – Example1

7 Utilisation

— Dans *MainFile* Vous pouvez utiliser la fonction *testLambda* pour afficher les valeurs des erreurs relatives de reconstruciton en fonction des valeurs de lambda (voir testLambda.m).
— Dans *deburr* vous pouvez utiliser la fonction *estSommet* pour afficher dans la console si la solution est un sommet du polyèdre correspondant.

8 Conclusion

Ce projet nous aura permis de mettre en pratique la matière vue au cours. Avec le dernier exemple, cela nous a permis d'apprendre comment optimiser le mieux possibles nos modèles pour qu'ils soient le plus efficace possible.