Master MVA - Convex Optimization - Homework 3

ZEDDOUN Adnan

Novembre 2019

1 Mise sous forme duale du problème LASSO

Question 1

On étudie le problème d'optimisation convexe suivant :

$$\min_w \frac{1}{2}\|Xw - y\|_2^2 + \lambda\|w\|_1$$

où $w \in \mathbf{R}^d$, $X \in \mathbf{R}^{n\times d}$, $y \in \mathbf{R}^n$ et $\lambda > 0$, $n \ll d$

En posant $r = Xw - y$, on est ramené à résoudre le problème sous contrainte :

$$\min_{w,r} \frac{1}{2}\|r\|_2^2 + \lambda\|w\|_1$$

$$\text{s.t} \quad r = Xw - y$$

où $r \in \mathbf{R}^n$

Le lagrangien associé au problème d'optimisation sous contraint s'écrit :

$$L(w, r, \nu) = \frac{1}{2}\|r\|_2^2 + \lambda\|w\|_1 + \nu^T(Xw - y - r)$$

où $\nu \in \mathbf{R}^n$ est la variable duale

Par suite on a :

$$\min_{w,r} L(w, r, \nu) = \min_r(\frac{1}{2}\|r\|_2^2 - \nu^T r) + \min_w(\lambda\|w\|_1 + \nu^T Xw) - \nu^T y$$

Or $f_1 = \frac{1}{2}\|r\|_2^2 - \nu^T r$ possède un unique minimum atteint en $r = \nu$.
On définit $f_2 = \lambda\|w\|_1 + \nu^T Xw = (\nu^T X)w - (-\lambda\|w\|_1)$.
Pour f_2, on reconnaît la fonction conjuguée avec l'expression suivante :

$$\min_w f_2 = \sup_w (-X^T\nu)^T w - \lambda\|w\|_1 = \begin{cases} 0 \text{ si } \|X^T\nu\|_\infty \leq \lambda \\ -\infty \text{ sinon} \end{cases}$$

On en déduit que :

$$\min_{w,r} L(w, r, \nu) = -\nu^T y - \frac{1}{2}\|\nu\|_2^2$$

Le problème duale est donc :

$$\min_{\nu} \nu^T y + \frac{1}{2}\|\nu\|_2^2$$

$$\text{s.t} \quad \|X^T \nu\|_\infty \leq \lambda$$

où $\nu \in \mathbf{R}^n$

La forme standard demandée dans l'énoncée est :

$$\min_{\nu} \nu^T Q \nu + p^T \nu$$

$$\text{s.t} \quad A\nu \leq b$$

où $\nu \in \mathbf{R}^n$ avec $p = y, Q = \frac{1}{2}\mathbf{I}_n, A = [X^T; -X^T] \in \mathbf{R}^{2d \times n}, b = (\lambda, ...\lambda)^T \in \mathbf{R}^{2d}$

2 A la pratique

Question 2 et 3

Pour programmer la fonction *centering_step(Q,p,A,b,t,v0,eps)*, nous avons défini la fonction à minimiser *fonctionf*, son gradient *gradientf*, son hessien *hessienf* et la fonction *line_search* permettant d'utiliser la méthode du backtracking lors d'une itération de Newton.

```
function v_seq = centering_step(Q,p,A,b,t,v0,eps)
itmax = 1000; %on définit une itération maximale pour éviter une boucle infinie
it = 0;
alpha = 0.1;
beta = 0.7;
lambda_sq = inf;
v = v0;
v_seq = [v0];
fonc = @(v) fonctionf(t,Q,p,v,A,b); %fonction f de notre problème duale
grad = @(v) gradientf(t,v,p,A,b); %gradient de f
while it<=itmax && lambda_sq >2*eps %méthode de Newton
    gradfv = gradientf(t,v,p,A,b);
    hessfv = hessienf(t,v,A,b);
    delta = hessfv\-gradfv;
    lambda_sq = transpose(gradfv)*-delta;
    mu = line_search(fonc, v, delta , grad, alpha , beta); %line_search utilise le backtracking
    v = v + mu*delta; %mise à jour
    v_seq = [v_seq, v];
    it = it + 1;
end
```

Fig. 2.1: Essentielle de la fonction *centering_step(Q,p,A,b,t,v0,eps)*

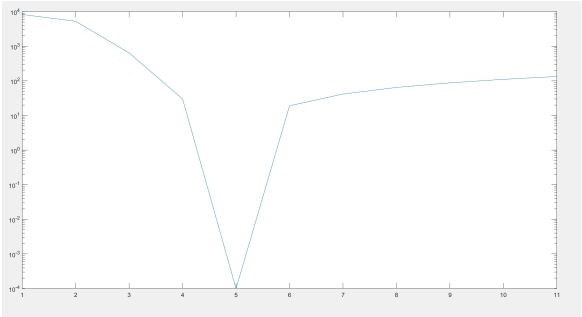
Pour programmer la fonction *barr_method(Q,p,A,b,v0,eps)*, nous avons utiliser la fonction précédente *centering_step(Q,p,A,b,t,v0,eps)*.

```
function vseq = barr_method(Q,p,A,b,v0,eps,mu)
m = length(b(:,1));
t = 1;
vseq = [v0];
x = v0;
while m/t > eps
    res = centering_step(Q,p,A,b,t,x,eps);
    x = res(:,length(res(1,:)));
    vseq = [vseq, x];
    t = mu*t;
end
```

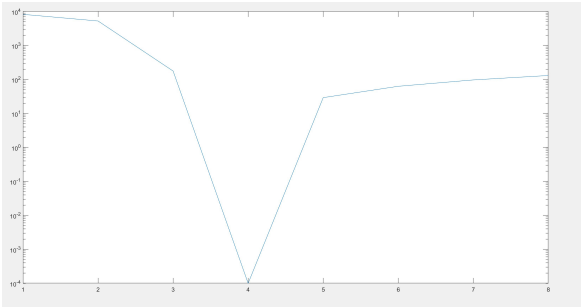
Fig. 2.2: Essentielle de la fonction *barr_method(Q,p,A,b,v0,eps)*

Commentaires

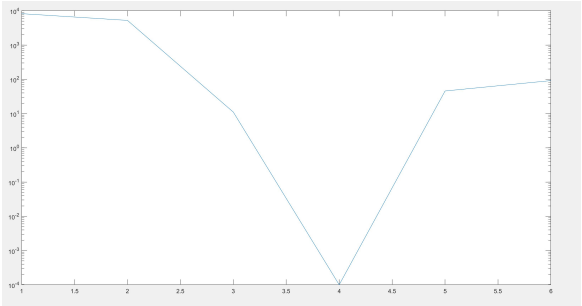
Nous avons généré X et y avec des entiers pris aléatoirement entre 0 et 10. Nous avons choisi une précision $\epsilon = 10^{-8}$, $\lambda = 10$. Nous constatons que pour $\mu \geq 10$, l'algorithme *barr_method(Q,p,A,b,v0,eps)* est très rapide et converge en 4 ou 5 itérations. Lorsque l'on augmente μ, le nombre d'itération diminue légèrement. Lorsque $\mu < 10$, l'algorithme est beaucoup plus lent, d'où l'intérêt de choisir judicieusement sa valeur. Enfin, pour $\mu = 10; 30; 100$, la valeur w^* varie très peu, l'algorithme est robuste sur cette plage de valeurs et donne bien une solution "sparse" comme nous le promet la méthode lasso.



(a) $f(v_t) - f^*$ en fonction du nombre d'itérations, $\mu = 10$



(b) $f(v_t) - f^*$,en fonction du nombre d'itérations, $\mu = 30$



(c) $f(v_t) - f^*$, en fonction du nombre d'itérations, $\mu = 100$