RO – TP 5 : Programmation linéaire et reformulation matricielle

Stéphane Canu

Décembre 2016, ASI, INSA Rouen

Le but du TP est d'identifier, de reformuler et de résoudre des programmes linéaires à l'aide de logiciels standard parmi les plus efficaces (comme par exemple Cplex).

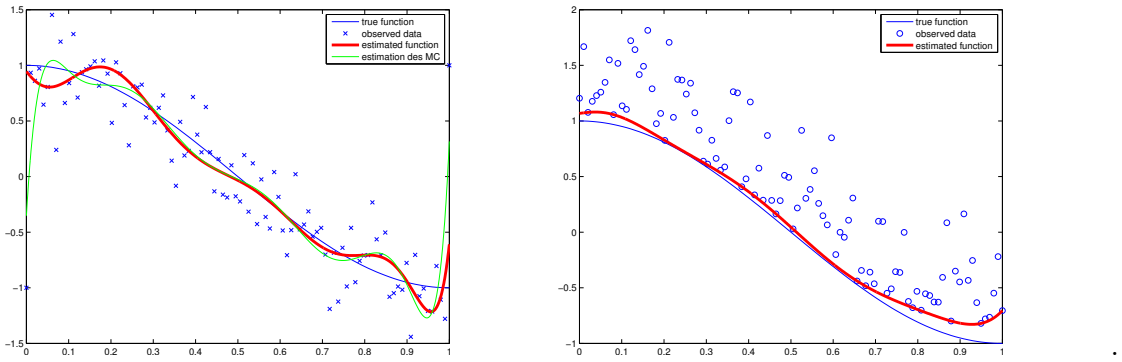

FIGURE 1 – Résultats du TP5

Ex. 1 — programmation linéaire et la régression MAD

1. Générez les mêmes données que lors du TP1 ($n = 100$ points $x_i, i = 1, n$ aléatoirement entre 0 et 1 suivant la distribution uniforme. Ordonnez ces points. Pour chaque point x_i générez une observation bruitée $y_i = f(x_i) + \varepsilon_i$, avec ε_i suivant une loi normale de variance $\sigma^2 = sig = 0.0625$;
   a) résoudre le programme linéaire suivant :

```
randn('seed',1);
rand('seed',1);
n = 100;
x = sort(rand(n,1));
nt = 1000;
xt = linspace(0,1,nt)';
y = cos(pi*x);
yt = cos(pi*xt);
sig = 0.25;
ya = y+sig*randn(size(y));
```

   b) Ajoutez deux points aberrants à chaque extrémité $y_1 = -1$ et $y_n = 1$.

```
ya(1) = -1;
ya(n) = 1;
```

   c) Visualisez les données ainsi générées et la fonction cible (la vérité terrain).

```
figure(1);
tt = plot(x,y); hold on;
oo = plot(x,ya,'x');
```

2. La méthode des moindres absolues déviations (MAD : *mean absolute deviation*) consiste estimer la fonction f supposée inconnue par une fonction connue $\hat{f}$ (ici un polynôme de degré p) dont on va trouver les coefficients $\alpha_j, j = 1, p$ en minimisant la somme des valeurs absolues des erreurs sur les points $(x_i, y_i), i = 1, n$ dont nous disposons. Mathématiquement le problème s'écrit :

$$\min_{\alpha_j, j=0,p} \sum_{i=1}^{n} |\hat{f}(x_i) - y_i| \qquad \text{avec} \qquad \hat{f}(x_i) = \sum_{j=0}^{p} \alpha_j x_i^j \tag{1}$$

a) reformulez matriciellement l'équation 1, c'est à dire écrivez la sous la forme

$$\min_{\alpha \in \mathbb{R}^{p+1}} \|X\alpha - y\|_1 \tag{2}$$

avec y, α deux vecteurs et X une matrice dont on précisera la dimension. Calculez la matrice X.

```
X = [ones(size(x)) x x.^2 x.^3 x.^4 x.^5 x.^6 x.^7 x.^8 x.^9 ];
Xt = [ones(size(xt)) xt xt.^2 xt.^3 xt.^4 xt.^5 xt.^6 xt.^7 xt.^8 xt.^9 ];
[n,p] = size(X);
```

b) résolvez le problème des moindres absolues déviations (MAD) à l'aide de CVX

```
cvx_begin
    variables beta_cvx(p)
    minimize( sum(abs(X*beta_cvx - ya)) )
cvx_end
```

c) reformulez le problème MAD comme un problème d'optimisation sous contraintes en posant $e_i = \hat{f}(x_i) - y_i$

$$\begin{cases} \min\limits_{\alpha \in \mathbb{R}^{p+1},\, \mathbf{e} \in \mathbb{R}^n} & \sum\limits_{i=1}^{n} |e_i| \\ \text{avec} & \mathbf{e} = X\alpha - y \end{cases} \tag{3}$$

d) reformulez le problème MAD comme un programme linéaire en posant $\mathbf{e} = \mathbf{e}^+ - \mathbf{e}^-$ avec $0 \leq e^+$ et $0 \leq e^-$.

$$\begin{cases} \min\limits_{\alpha \in \mathbb{R}^{p+1},\, \mathbf{e}^+ \in \mathbb{R}^n,\, \mathbf{e}^- \in \mathbb{R}^n} & \sum\limits_{i=1}^{n} e_i^+ + e_i^- \\ \text{avec} & \mathbf{e}^+ - \mathbf{e}^- = X\alpha - y \\ \text{et} & 0 \leq e^+, 0 \leq e^- \end{cases} \tag{4}$$

e) résolvez le problème des moindres absolues déviations (MAD) comme un LP à l'aide de CVX

```
e = ones(n,1);
cvx_begin
    variables beta_cvx2(p) ep(n) em(n)
    dual variable d
    minimize( e'*ep + e'*em)
    subject to
        d : X*beta_cvx2 - ya == ep-em;
        ep >= 0;
        em >= 0;
cvx_end
```

f) Reformulez le problème sous a forme d'un programme linéaire matriciel de la forme

$$\begin{cases} \min\limits_{\mathbf{z} \in \mathbb{R}^q} & \mathbf{f}'\mathbf{z} \\ \text{et} & A\mathbf{z} = \mathbf{b} \\ & lb \leq \mathbf{z} \leq ub \end{cases}$$

```
f = [zeros(2*p,1) ; ones(2*n,1)];
I = eye(n);
A = [X -X -I I];
b = ya;
lb = zeros(2*p+2*n,1);
ub = [];
```

g) Résolvez le à l'aide de CVX

```
cvx_begin
    cvx_quiet(true)
   cvx_precision best
   variables xl(2*p+2*n)
   dual variable d
   minimize( f'*xl )
   subject to
      d : A*xl == b;
      xl >= lb;
cvx_end
```

h) Ecrire le lagrangien et les KKT du problème. Montrez que le dual est bien

$$\begin{cases} \max\limits_{\lambda \in \mathbb{R}^n} & \mathbf{b}^\top \lambda \\ \text{avec} & A^\top \lambda \leq \mathbf{f} \end{cases} \tag{5}$$

Comment retrouver les variables primales connaissant les variables duales ?

i) Résoudre le dual du MAD à l'aide de CVX

```
cvx_begin
    cvx_quiet(true)
   cvx_precision best
   variables yl(n)
   dual variable dd
   maximize( b'*yl )
   subject to
      dd : A'*yl <= f;
cvx_end
```

j) Résoudre le même problème MAD en utilisant linprog

```
x_lin = linprog(f,[],[],A,b,0*f);
```

k) Résoudre le même problème MAD (le primal) en utilisant un autre solveur que vous aurez installé (ici CPLEX à `/nfs/opt/CPLEX/cplex/matlab` en utilisant la fonction `addpath`)

```
addpath('/nfs/opt/CPLEX/cplex/matlab');
x_cpl = cplexlp(f,[],[],A,b,0*f);
```

l) vérifiez que toues les méthodes donnent bien le même résultat.

```
[beta_cvx beta_cvx2 xl(1:p) - xl(p+1:2*p) x_lin(1:p) - x_lin(p+1:2*p) x_cpl(1:p) -
    x_cpl(p+1:2*p) dd(1:p) - dd(p+1:2*p)]
```

m) visualisez votre estimation et celle des moindres carrés.

```
ee = plot(xt,Xt*a,'r');
set(ee,'LineWidth',3);
ee2 = plot(xt,Xt*(X\ya),'g');
legend([tt, oo, ee , ee2],'true function','observed data','estimated function','
    estimation des MC');
```

n) Sur ma machine, j'ai obtenu les temps de calcul suivants :

```
MAD CVX: Elapsed time is 0.1839 seconds.
MAD CVX: Elapsed time is 0.1004 seconds.
MAD CVX: Elapsed time is 0.1090 seconds.
MAD dual: Elapsed time is 0.0975 seconds.
MAD linprog: Elapsed time is 0.0429 seconds.
MAD cplex: Elapsed time is 0.0020 seconds.
```

Qu'en pensez vous ? Pourquoi le premier programme CVX (celui de la question (b)) est-il plus lent ? Pourquoi le dernier est-il plus rapide ? Que recommandez vous ?

3. Le but de cet exercice est d'utiliser la méthode présenté dans la question 2 ci-dessous pour écrire une fonction matlab `a = mon_enveloppe(x,ya)` permettant d'estimer l'enveloppe inférieure d'un ensemble de point à l'aide d'un polynôme de degré neuf. Il s'agit, à partir d'un ensemble de couples $(x_i, y_i), i = 1, n$ d'estimer les coefficients $a \in \mathbb{R}^{10}$ d'un polynôme $p(x) = \sum_{i=0}^{9} a_j x^j$ (en bleu sur la figure) au plus prêt des observations mais toujours en dessous. Afin de tester votre procédure nous allons générer n observations bruitées $y_i = f(x_i) + \varepsilon_i$, avec ε_i une variable aléatoire positive générée à partir de la valeur absolue d'une loi normale de variance $\sigma^2 = 0.25$;

```
sig = .5;
yb = y+sig*abs(randn(size(y)));
```

   a) L'enveloppe inférieure est définie par la solution du problème suivant :

$$\begin{cases} \min\limits_{a\in\mathbb{R}^{10}} & \sum\limits_{i=1}^{n} |f(x_i) - y_i| \\ \text{avec} & f(x_i) \leq y_i \qquad i = 1, n \\ \text{et} & f(x) = \sum\limits_{j=0}^{9} a_j x^j \qquad \forall x \in \mathbb{R} \end{cases}$$

   Donner un code matlab permettant de le résoudre et reformuler ce problème comme un programme linéaire standard (on pourra effectuer des simplifications).

```
cvx_begin
    cvx_quiet(true)
   cvx_precision best
   variables beta_cvx(p)
   dual variable d
   minimize( sum(abs(X*beta_cvx - yb)) )
   subject to
      d : X*beta_cvx <= yb;
cvx_end
```

   b) Visualisez les données, la fonction cible (la vérités terrain et votre estimation)

```
set(gcf,'Color',[1,1,1])
oo = plot(x,yb,'ob');hold on
tt = plot(x,y,':c');
ee = plot(xt,Xt*beta_cvx,'r');
legend([tt, oo, ee],'true function','observed data','estimated function')
```

   c) Reformulez le problème sous a forme d'un programme linéaire matriciel de la forme

$$\begin{cases} \min\limits_{\mathbf{z}\in\mathbb{R}^q} & \mathbf{f}'\mathbf{z} \\ \text{avec} & A\mathbf{z} \leq \mathbf{b} \\ \text{et} & A_{eq}\mathbf{z} = \mathbf{b}_{eq} \\ & lb \leq \mathbf{z} \leq ub \end{cases}$$

   d) Validez à l'aide de CVX votre solution
   e) Résoudre le même problème en utilisant Cplexlp.
   f) Écrire une fonction matlab `a = mon_enveloppe(x,ya)` permettant d'estimer l'enveloppe inférieure d'un ensemble de point à l'aide d'un polynôme de degré neuf