RO (lovely) TP 1 : La méthode des moindres carrés

Stéphane Canu

Novembre 2016, ASI, INSA Rouen

Le but du TP est de résoudre un problème d'optimisation en utilisant deux méthodes différentes : le logiciel CVX et une forme fermée de la solution. Nous allons voir comment d'approcher une fonction f quelconque par un polynôme estimé à partir d'un échantillon bruité, c'est-à-dire un ensemble de couples $(x_i, y_i), i = 1, n$ vérifiant

$$y_i = f(x_i) + \varepsilon_i$$

où ε_i est un bruit gaussien de variance σ^2. Afin d'illustrer la méthode, nous allons prendre la fonction cosinus comme fonction cible en prétendant que nous ne la connaissons pas. Nous allons l'estimer ne utilisant un polynôme de degré 12. Pour faire réaliser ce TP vous êtes supposé avoir installé CVX sous matlab (à télécharger depuis `http://cvxr.com/cvx/`).

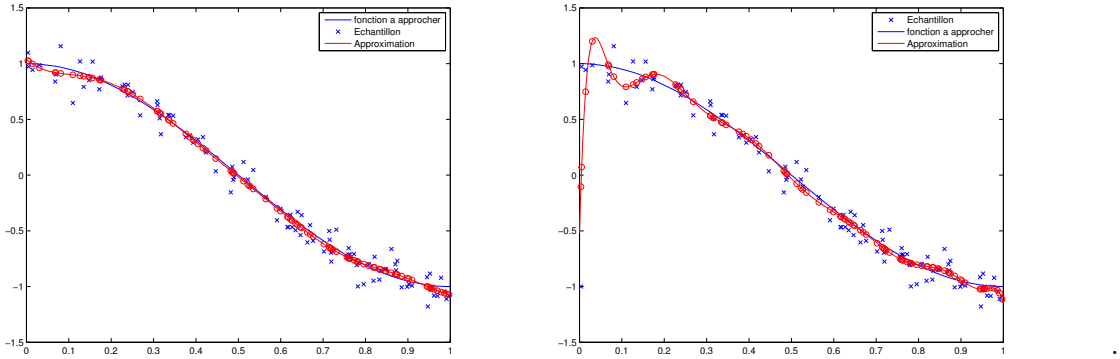

FIGURE 1 – Résultat du TP1

Ex. 1 — La méthode des moindres carrés comme un problème d'optimisation

1. Questions stratégiques :
   a) quelles sont les variables V, l'objectif O et les contraintes C du problème ?
   b) ce problème admet il une solution unique ?
   c) comment peut on s'assurer qu'un polynôme est bien la solution du problème ?
   d) quel outil vous propose t'on d'utiliser pour résoudre ce problème ?
   e) comment (re) formuler le problème
2. Générez les données du problème
   a) Générez $n = 100$ points $x_i, i = 1, n$ aléatoirement entre 0 et 1 suivant la distribution uniforme. Ordonnez ces points.

```
n = 100;
x = sort(rand(n,1));
```

   b) Pour chaque point x_i calculez $f(x_i) = \cos(\pi x_i)$

```
f = cos(pi*x);
```

   c) 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 $\sigma^2 = 0.01$

```
sig = 0.1;
y = f+sig*randn(size(f));
```

   d) Dessinez sur la figure (1) la fonction cosinus

```
nt = 1000; xt = linspace(0,1,nt)';
yt = cos(pi*xt);
figure(1);
plot(xt,yt); hold on;
```

e) Affichez sur la même figure les couples $(x_i, y_i), i = 1, n$ à partir desquels nous allons approcher la fonction cosinus.

```
plot(x,y,'x');
```

3. Dessinez la fonction $g(x) = 1 + x - 3x^2 + x^3 + x^4 + x^5 - x^6$ pour $x \in [0, 1]$ sur la figure 2.

```
Xd = [ones(size(xt)) xt xt.^2 xt.^3 xt.^4 xt.^5 xt.^6 ];
aa = [1 1 -3 1 1 1 -1];
plot(xt,Xd*aa')
```

4. La méthode des moindres carrés consiste à 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 les carrés des erreurs sur les points $(x_i, y_i), i = 1, n$ dont nous disposons. Mathématiquement le problème s'écrit :

$$\min_{\alpha_j, j=0,p} \sum_{i=1}^{n} (\hat{f}(x_i) - y_i)^2 \qquad \text{avec} \qquad \hat{f}(x_i) = \sum_{j=0}^{p} \alpha_j x_i^j \tag{1}$$

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

$$\min_{\alpha} \ \|X\alpha - y\|^2 \tag{2}$$

avec y, α deux vecteurs et X une matrice dont on précisera la dimension. Calculez la matrice X avec $p = 12$.

```
X = [ones(size(x)) x x.^2 x.^3 x.^4 x.^5 x.^6 x.^7 x.^8 x.^9 x.^10 x.^11 x.^12];
```

b) résolvez le problème des moindres carrés à l'aide de CVX

```
cvx_begin
   variables ac(size(X,2))
   minimize( norm(y - X*ac,2) )
cvx_end
```

c) calculez la valeur du gradient de la fonction cout 1

$$\nabla_\alpha(\alpha) = X^t(X\alpha - y)$$

à la solution proposée par CVX et commentez le résultat

```
grad = X'*(X*ac-y)
```

d) calculez une estimation en forme close de la solution

$$\alpha_m = (X^t X)^{-1} X^t y$$

```
am = (X'*X)\(X'*y);
```

e) vérifiez que les solutions de CVX et de la forme fermée sont les mêmes

```
[ac am]
```

f) visualisez votre approximation au sens des moindres carrés $\hat{f}$.

```
Xt = [ones(size(xt)) xt xt.^2 xt.^3 xt.^4 xt.^5 xt.^6 xt.^7 xt.^8 xt.^9 xt.^10 xt
    .^11 xt.^12];
plot(xt,Xt*am,'r');
plot(x,X*am,'or');hold off;
```

g) que pensez vous du résultat ?

5. Nous allons maintenant tester la robustesse de notre solution.
   a) recalculez et visualisez votre approximation lorsque une erreur est introduite de sorte que $y_1 = -1$.

```
 y(1) = -1;
a = X\y;

figure(2)
plot(x,y,'x');hold on
plot(x,y,'b');
plot(xt,Xt*a,'r');
plot(x,X*a,'or');hold off;
```

   b) que pensez vous du résultat ?
   c) que ce passe t'il lorsque que l'on introduit une discontinuité dans la fonction f à approcher ? Par exemples en considérant la fonction

$$f(x) = \begin{cases} \cos(x) & \text{si } x < .5 \\ \cos(x) + 2 & \text{sinon} \end{cases}$$

```
ind = find(x>.5);
y(ind) = y(ind) + 2;
f(ind) = f(ind) + 2;
```

```
p = size(X,2);
l = 10^-9;
a = (X'*X + l*eye(p))\(X'*y)
```

```
figure(3)
plot(x,y,'x');hold on
plot(xt,yt,'b');
plot(xt,Xt*a,'r');
plot(x,X*a,'or');hold off
```

6. Ecrire deux fonctions matlab *polyapprox* et *polyval* avec l'entête suivante

```
function a = polyapprox(x,y,p)
% a = argmin (y_i - f_i)^2 avec f_i = sum_{j=0}^p a_j x_i^j
%       a
```

```
function f = polyaval(x,a)
%      p
% f = sum  a_j x^j
%     j=0
```