Optimisation

Devoir 1
Régression quadratique

Charlier Maximilien

Corentin Ducruet

BAC 2 Science Informatique

Charlier Maximilien & Corentin Ducruet
Bac 2 Info
30 octobre 2013

Modélisation du problème sous forme d'un problème d'optimisation.

On part de la norme infinie d'un vecteur :

$$\min_{a,b,c \in R} \|Z\|_\infty$$

c'est-à-dire par définition de la norme infinie d'un vecteur :

$$\min_{a,b,c \in R} (\max(|Z_1|, |Z_2|, |Z_3|, \dots, |Z_n|))$$

Où $Z_i = y_i - (ax_i^2 + bx_i + c)$
Càd : (introduction d'un vecteur U pour enlever la valeur absolue)

$$\min_{\substack{a,b,c \in R \\ U \in R^n}} (\max(U))$$

$$tq \begin{cases} U_i \geq y_i - (ax_i^2 + bx_i + c) \\ U_i \geq -y_i + (ax_i^2 + bx_i + c) \\ \forall i \epsilon \{1, \dots, n\} \end{cases}$$

On doit retirer le max de la fonction objectif, pour cela on introduit un réel d.
Le problème devient :

$$\min_{\substack{a,b,c,d \in R \\ U \in R^n}} d$$

$$tq \begin{cases} U_i \geq y_i - (ax_i^2 + bx_i + c) \\ U_i \geq -y_i + (ax_i^2 + bx_i + c) \\ d > U_i \\ \forall i \epsilon \{1, \dots, n\} \end{cases}$$

On peut donc se passer du vecteur U, le problème peut se réécrire :

$$\min_{a,b,c,d \in R} d$$

$$tq \begin{cases} d \geq y_i - \left(ax_i^2 + bx_i + c\right) \\ d \geq -y_i + \left(ax_i^2 + bx_i + c\right) \\ \forall i \epsilon \{1, \dots, n\} \end{cases}$$

On peut maintenant écrire correctement ce problème.

$$\min_{a,b,c,d \in R} d$$

$$tq \begin{cases} -ax_i^2 - bx_i - c - d \leq -y_i \\ ax_i^2 + bx_i + c - d \leq y_i \end{cases}$$

Charlier Maximilien & Corentin Ducruet
Bac 2 Info
30 octobre 2013

Modélisation pour sous forme matricielle

$$\min f$$

$$tq\ Ax \leq b$$

Avec $f = [0\ 0\ 0\ 1]$

$$A = \begin{bmatrix} -x_i^2 & -x_i & -1 & -1 \\ x_i^2 & x_i & 1 & -1 \\ \vdots & \vdots & \vdots & \vdots \\ -x_n^2 & -x_n & -1 & -1 \\ x_n^2 & x_n & 1 & -1 \end{bmatrix}$$

$$x = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \quad b = \begin{bmatrix} -y_1 \\ y_1 \\ \vdots \\ -y_n \\ y_n \end{bmatrix}$$

Charlier Maximilien & Corentin Ducruet
Bac 2 Info
30 octobre 2013

Code Matlab

quadratique_regression_n_linprog.m

```
%**************************************************************************
% Devoir Optimisation
% Par Maximilien Charlier & Corentin Ducruet
% Umons 2013 - 2014
%**************************************************************************

clc                     % efface tout ce qui a dans le command window
clear all               % efface toutes les variables du workspace
close all               % ferme toutes les fenêtres de plot


% Génération de n points aléatoire dans le plan proche de la droite y = x²
n = 20;              %initialisation de la variable n
P = genpoints(n);  %on génère les points aléatoires
epsilon = 0.001; %on définit un epsilon

%**************************************************************************
% CVX %
%**************************************************************************

cvx_begin quiet
    % variables:
    variable cvx_a(1)
    variable cvx_b(1)
    variable cvx_c(1)
    variable cvx_t(n)
    % objectif:
    minimize( max(cvx_t) )
    % contraintes:
    subject to
        for i = 1 : n
            cvx_t(i) >= (cvx_a*P(1,i)^2 + cvx_b*P(1,i) + cvx_c) - P(2,i);
            cvx_t(i) >= - (cvx_a*P(1,i)^2 + cvx_b*P(1,i) + cvx_c) + P(2,i);
        end
cvx_end


%**************************************************************************
% LINPRO %
%**************************************************************************

%on génère A
for i=1:n
    A((2*i)-1, 1) = - (P(1, i)^2);
    A((2*i)-1, 2) = - P(1, i);
    A((2*i)-1, 3) =  -1;
    A((2*i)-1, 4) =  -1 ;
    A((2*i), 1) = P(1, i)^2;
    A((2*i), 2) = P(1, i);
    A((2*i), 3) =  1;
    A((2*i), 4) =  -1;
end

%On génère b
for i=1:n
    b((2*i)-1, 1) = -P(2, i);
```

```
    b((2*i), 1) = P(2, i);
end

%fonction objective
f =  [0 0 0 1];
%calcul de linprog
[res] = linprog(f,A,b);
lin_a = res(1);
lin_b = res(2);
lin_c = res(3);
lin_d = res(4);

%************************************************************************
% Résultat %
%************************************************************************

disp('******************************************'); % équivalent du
cout en c
fprintf('La meilleure droite est y = %0.2f x² + %0.2f x + %0.2f  \n',cvx_a,
cvx_b, cvx_c); % équivalent du cout en c avec les f qui servent à sortir
les valeurs
fprintf('La plus grande erreur (distance point courbe) est de %0.2f \n',
res(4));
disp('******************************************');

%comparaison CVX et linprog
if ( abs(lin_a - cvx_a) <= epsilon && abs(lin_b - cvx_b) <= epsilon &&
abs(lin_c - cvx_c) <= epsilon)
    fprintf('Linprog et CVX ont les même résultats.\n');
end


%************************************************************************
% PLOT %
%************************************************************************

%mise en forme du plot
title('Régression quadratique');
xlabel('x');
ylabel('y');
hold on; % pour pouvoir redessiner sur la même figure sinon matlab supprime
par défaut l'ancien dessin


%affiche les traits entre les points et la courbe
    for i = 1 : n
        z = res(1) * P(1,i).^2 + res(2) * P(1,i) + res(3) ;
        plot(P(1,i), min(P(2,i),z):0.001:max(P(2,i),z), 'k-');
    end

% Affichage des points
plot(P(1, :), P(2,:),'bo');


%dessine la courbe
minx = min(P(1,:));
maxx = max(P(1,:));
x= minx:0.0001:maxx;
y = lin_a * x .^2 +lin_b * x + lin_c ;
% Affichage de la régression quadratique
```

Charlier Maximilien & Corentin Ducruet
Bac 2 Info
30 octobre 2013

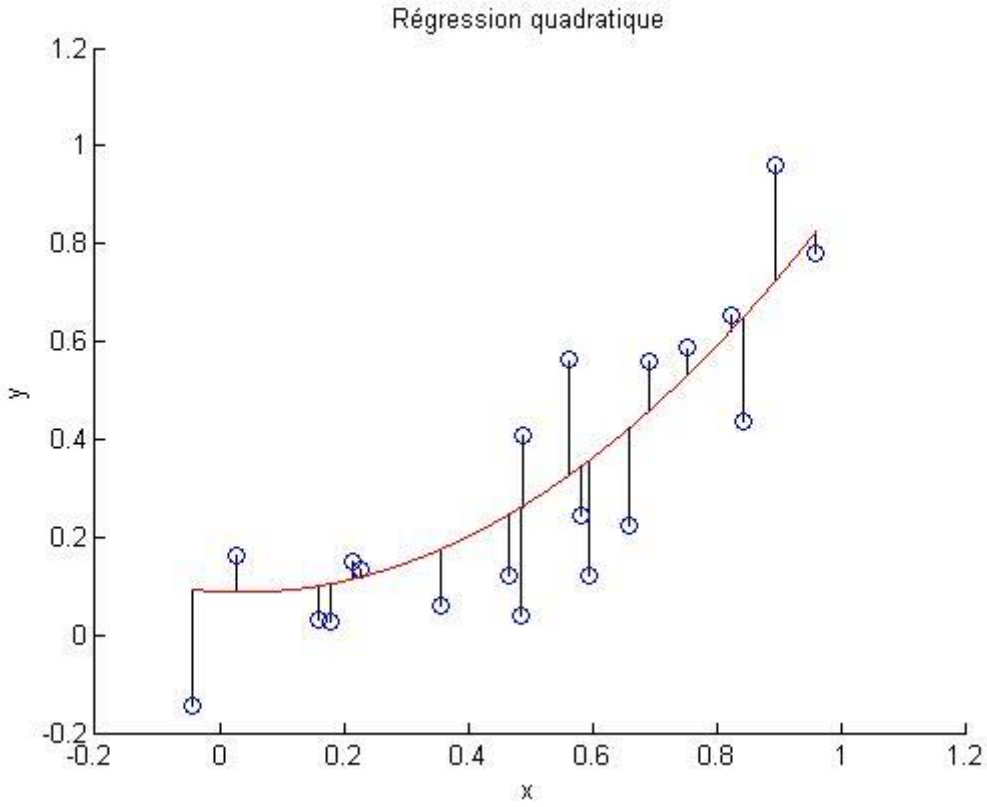
```
plot(x, y,'r-');

%*************************************************************************
% Sommet ? %
%*************************************************************************


disp('*********************************************');
%Calcule du nombre de contrainte sérée
% on a 4 variables de contrainte, il nous faut donc 4 contraintes pour
avoir
% un sommet (non dégénéré)
% plus de 4 contraintes équivaut à avoir un sommet dégénéré
%moins de 4 contrainte aucune solution
cont = 0;
for i=1:(2*n)
    if abs((lin_a * A(i, 1) + lin_b * A(i, 2) + lin_c * A(i,3) + lin_d *
A(i, 4)) - b(i,1)) <= epsilon
        cont = cont + 1;
        %on rempli une matrice des contraintes serrées
        contM(cont, 1) = A(i, 1);
        contM(cont, 2) = A(i, 2);
        contM(cont, 3) = A(i, 3);
        contM(cont, 4) = A(i, 4);
    end
end
fprintf('Nombre de contrainte sérée est de %i \n', cont);
rang = rank(contM);
fprintf('Le rang de la matrice des contraintes serrées est de %i\n', rang);
if (abs(rang-4) <= 0.001 && abs(cont-4) <= epsilon)
    fprintf('La solution est le sommet du polyèdre des contraintes !\n');
end
```

Charlier Maximilien & Corentin Ducruet
Bac 2 Info
30 octobre 2013

Plot



J'ai fait une boucle qui permettait de compter le nombre de contrainte serrée, elle en comptait chaque fois 4, ce qui correspond au quadriennal du vecteur de la fonction objectif, la solution obtenue via linprog est donc bien un somment du polyèdre correspondant.

Charlier Maximilien & Corentin Ducruet
Bac 2 Info
30 octobre 2013