MATHÉMATIQUES ET INFORMATIQUE
Sciences
Université Paris Cité

MASTER INFO 2022–2023

Optimisation Algorithmique :

TP 3 : Méthode de descente de gradient.

Exercice

Un signal, émis par une source de position inconnue $x \in \mathbb{R}^n$, est recu par m récepteurs de positions connues $y_1, \ldots, y_m \in \mathbb{R}^n$. Grâce à l'intensité du signal, on obtient pour chaque récepteur y_j une estimation (bruitée) de la distance $\|x - y_j\|_2$.

On veut déterminer la position x de la source à partir des données observées (y_j, d_j), pour cela on introduit la fonction définie sur $\mathbb{R}^n$

$$f(x) = \sum_{j=1}^{k} \left(\|x - y_j\|_2^2 - d_j^2 \right)^2 .$$

1. Que peut-on dire du problème $x^\star = \arg\min_{x \in \mathbb{R}^n} f(x)$ (existence, unicité) ?
2. Calculer $\nabla f(x)$.
3. Écrire les fonctions Scilab
   `function [v]=objectif(x,Y,dist)` et `function [g]=gradient_obj(x,Y,dist)`
   pour calculer $f(x)$, reps. $\nabla f(x)$ La matrice `Y`, de taille `[n,m]`, contient les positions des récepteurs y_j et le vecteur *dist*, de taille `[1,m]`, les distances d_j.
   On tentera d'éviter les boucles.
4. Implémenter la méthode du gradient, `function [x_k,v_k]=methode_gradient(x0,c,rho,Y,dist)` en utilisant un backtracking de paramètres c et ρ et où la matrice `x_k` contient la suite des points $x^{(k)}$, $k \geq 0$ et le vecteur `v_k` les valeurs $f(x^{(k)})$, $k \geq 0$.
5. Pour $n = 2$ et $m = 5$ on se donne
   $$y_1 = \begin{pmatrix} 1.8 \\ 2.5 \end{pmatrix}, \; y_2 = \begin{pmatrix} 2.0 \\ 1.7 \end{pmatrix}, \; y_3 = \begin{pmatrix} 1.5 \\ 1.5 \end{pmatrix}, \; y_4 = \begin{pmatrix} 1.5 \\ 2.0 \end{pmatrix}, \; y_5 = \begin{pmatrix} 2.5 \\ 1.5 \end{pmatrix}$$
   et `dist`$= (2.00,\ 1.24,\ 0.59,\ 1.31,\ 1.44)$.
   Tracer le graphe de f sur le domaine $[-0.5, 3]^2$; dans une seconde fenêtre tracer les lignes de niveaux et les positions des récepteurs, y_j, $1 \leq j \leq m$.
   Pour diverses valeurs de $x^{(0)}$, tracer la suite des points $x^{(k)}$, calculée par la descente de gradient, dans la seconde fenêtre (*i.e.* sur les lignes de niveau).
6. Toujours pour $n = 2$, testez d'autres exemples de données (y_j, dj), $1 \leq j \leq m$.
   $$y_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \; y_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{avec } \texttt{dist} = (1\ 1)$$
   $$y_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \; y_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \; y_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \; y_4 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{avec } \texttt{dist} = (0.5\ 0.5\ 0.5\ 0.5)$$

```
% Définition de la fonction coût
function [v]=objectif(x,Y,dist)
[n,m] =  size(Y);
Yx = x*ones(1,m)-Y  ;
v = sum( (sum(Yx.^2,1) -dist.^2).^2 );
endfunction

% Gradient de l'objectif
function [g]=gradient_obj(x,Y,dist)
[n,m] =  size(Y)  ;
Yx =   x*ones(1,m)-Y  ;
err = sum(Yx.^2,1) -dist.^2;
g = 4*sum(Yx * diag(err),2) ;
endfunction

% Descente du gradient avec backtracking
function [x_k,v_k] = methode_gradient(x0,c,rho,Y,dist)
MAX_ITER=500;
precision=1e-7;

% initialisation
x_k = [ ]; % historique des points x
v_k = [ ]; % historique des valeurs v
x = x0;

for iter=1:MAX_ITER
  v = objectif(x,Y,dist)  ;     % valeur en x
  d = -gradient_obj(x,Y,dist) ; % direction de descente
  n = norm(d)                 ;
  x_k = [x_k , x] ;
  v_k = [v_k , v] ;
  if ( n<precision)   % test d'arret
    printf("\n   %i itérations \n",iter) ;
    return
  end
  s=1; % Backtracking
  while objectif( x+s*d, Y, dist) > v-c*s*n*n  , s=rho*s;
     end;

  printf("\n norme_grad=%2.5e , s=%2.5e , valeur=%2.5e ",n,s,v);
  x = x+s*d ;
end;
printf("\n   %i itérations \n",iter) ;
endfunction;
```