MATHÉMATIQUES ET INFORMATIQUE
Sciences
Université Paris Cité

MASTER INFO 2022–2023

Optimisation Algorithmique :

TD 5 : Méthode du gradient conjugué linéaire. Méthode de descente de gradient. Méthode de Newton. Méthode du gradient conjugué non linéaire.

Exercice 1

Comparer les méthodes du gradient, de Newton et gradient conjugué linéaire pour $b \in \mathbb{R}^n$ et A une matrice réelle, symétrique de taille (n, n).

Exercice 2

On s'intéresse à la minimisation de la fonctionnelle coût : $f(x) = < w, x > - \sum_{i=1}^{m} \log \Big(b_i - < a_i, x > \Big)$

où : x, w, $a_{i(1\leq i\leq m)} \in \mathbb{R}^n$, $b_{i(1\leq i\leq m)} \in \mathbb{R}$ et $< x, a_i >= \sum_{j=1}^{n} x_j (a_i)_j$.

On note $b = (b_1 \cdots b_m)^t \in \mathbb{R}^m$ et $A = (a_1^t \cdots a_m^t)$, matrice réelle (m, n), dont la i^{e}-ligne contient les coordonnées du vecteur a_i.

1. Calculer le gradient $\nabla f(x)$ et la matrice hessienne $H_f(x)$.
   On va illustrer les performances de la *méthode de descente du gradient, de la méthode de* Newton *et de la méthode du gradient conjugué non linéaire, version* Fletcher-Reeves *et* Polak-Ribière.
   Pour tenir compte du domaine de définition de f il suffit de modifier (légèrement) les fonctions qui effectuent la minimisation grâce à la méthode de descente du gradient par "backtracking" et la méthode de Newton.
2. Écrire la fonction `[x_h,v_h]=gc_non_lin(x0,flag_bet,A,b,w)` qui minimise la fonction f grâce à la méthode du gradient conjugué non linéaire avec un "backtracking" pour déterminer le pas de descente s.
   On prend β^{FR} de Fletcher-Reeves si `flag_bet=FR` et β^{PR} si `flag_bet=PR`.
   Pour toute la suite, on fixe $w = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, $A = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$ et $b = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$.
3. Écrire alors la fonction $f(x_1, x_2)$, déterminer $\text{dom}(f)$.
   Déterminer le minimum exact unique de f sur $\text{dom}(f)$: $v_m = f(x_1^*, x_2^*) = \min_{(x_1,x_2)\in\text{dom}(f)} f(x_1, x_2)$.
   Interpréter le problème de minimisation. Pourquoi les algorithmes de descente vont converger ?
4. Initialiser `x0` de façon aléatoire dans $\text{dom}(f)$ en évitant de partir trop près de (x_1^*, x_2^*).
   Pour différentes valeurs de `x0` :
   Comparer les performances et le comportement des quatre méthodes en affichant des lignes de niveau et le trajet de la suite des points minimisants pour différentes valeurs de `x0`.
   Tracer la précision du résultat ($-\log_{10} |$`v-v_min`$|$) en fonction du nombre d'itérations pour les quatre méthodes.
   Qu'est-ce que vous constatez ? Expliquez.

```
% Methode du GC linéaire
function [x_h,v_h]=gc_lin(x0,A,b,c)
MAX_ITER=500; precision=1e-8;
x_h=[ ]; v_h=[ ];
x = x0;  g = gradient_obj(x,A,b);% =r
d = -g;
for k=1:MAX_ITER
  v=objectif(x,A,b,c);     % valeur en x
  x_h=[x_h , x];  v_h=[v_h , v];
  n=norm(g);
  if(n<precision) % x n'est pas pt critique
     printf("\n   % i itérations \n",k-1); return;
  end;
  s = (g'*g)/(d'*A*d);
  x =  x + s*d ;
  ng = g+s*A*d;
  bet = ( ng'*ng) / (g'*g);
  d = -ng+bet*d;
  g = ng;
end
printf("\n   %i iterations \n",MAX_ITER);
endfunction

% Methode du GC non linéaire
function [x_h,v_h]=gc_non_lin(x0,flag_bet,c,rho,A,b,w)
MAX_ITER=500; precision=1e-8;
x_h=[ ]; v_h=[ ];
x = x0;  g = gradient_obj(x,A,b,w);
d = -g; g2 = g'*g;
for k=1:MAX_ITER
  v=objectif(x,A,b,w);     % valeur en x
  x_h=[x_h , x];  v_h=[v_h , v];
  n=norm(g);
  if(n<precision)   %  x n'est pas pt critique
     printf("\n  point critiqe \n");
     printf("\n   %i itérations \n",k); return;
  end;
  s=1;   % Verifier que x+s*d est dans dom(f)
  while min(b-A*(x+s*d))< precision,  s=0.9*s;  end;
  while objectif( x+s*d ,A,b,w) > v-c*s*n*n  , s=rho*s; end;
  x = x + s*d ;
  ng=gradient_obj(x,A,b,w);
   switch flag_bet   % Choix entre les deux methodes FR et PR
    case 'FR'
      bet=( ng'*ng)/g2;
    case 'PR'
      bet= (ng'*(ng-g))/g2;
      bet=max(bet,0);
    endswitch
  g = ng;  g2 = g'*g;
  d = -g+bet*d;
  if ( d'* g > -eps )  % Verifier que d est une direction des descente
    printf(" Plus de direction de descente ! Iteration %i \n",k); return
  end
end
printf("\n   %i iterations \n",MAX_ITER);
endfunction
```