Université Paris Descartes
UFR de Mathématiques et Informatique
45, rue des Saints-Pères 75270 Paris cedex 06

PARIS DESCARTES

MASTER MISV 2007–2008

Optimisation et Algorithmique

Projet Scilab surveillé, durée 2h

Polycopié de cours et notes de TD/TP autorisés, l'utilisation de tout appareil électronique, hormis les ordinateurs, est interdit.

Problème

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, $a_{i(1 \le i \le m)} \in \mathbb{R}^n$, $b_{i(1 \le i \le m)} \in \mathbb{R}$ 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 $\nabla^2 f(x)$.
   Adapter les fonctions Scilab définies en TD : `objectif(x,A,b,w)` qui calcule $f(x)$; `gradient_obj(x,A,b,w)` qui calcule $\nabla f(x)$; `hessienne_obj(x,A,b,w)` qui calcule $\nabla^2 f(x)$.
2. On pose $\phi(s) = f(x + sd)$, où $x, d \in \mathbb{R}^n$ et $s \in \mathbb{R}_+$. Que vaut $\phi'(s)$?
   Écrire la fonction Scilab `d_phi(s)` qui évalue $\phi'(s)$.

   On veut comparer 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 cela il suffit de *modifier* (légèrement) les fonctions Scilab suivantes, adaptées à f et vues en TP : `[x_h,v_h]=methode_gradient(x0,c,rho)` qui effectue la minimisation grâce à la méthode de descente du gradient par "backtracking" et la fonction `[x_h,v_h]=methode_newton(x0)`.
3. *Écrire* la fonction Scilab `[x_h,v_h]=gc_non_lin(x0,flag_bet)` qui minimise la fonction f grâce à la méthode du gradient conjugué non linéaire avec un *pas de descente s optimal* et en prenant β^{FR} de Fletcher-Reeves si `flag_bet=FR` et β^{PR} si `flag_bet=PR`.
   Note : pour résoudre l'équation $\phi'(s)$ sur l'intervalle $[0, s_M]$, $s_M > 0$, utiliser l'instruction suivante :
   `s=fsolve(s_M/2,d_phi);` où `d_phi()` a été définie en 2.

   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 $b = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$.
4. Écrire alors la fonction $f(x_1, x_2)$, déterminer $\text{dom}(f)$.
   En utilisant la symétrie et le fait que f est convexe (pourquoi ?) 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 géométriquement le problème de minimisation. Pourquoi les algorithmes de descente vont converger ?
5. 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.

<u>À la fin de la séance :</u>
(1) remettre un rapport avec : les réponses, commentaires
(2) envoyer les fichiers sources, commandes et les résultats éventuels à `gk@math-info.univ-paris5.fr`