Enoncé du projet pour le cours d'optimisation

Juillet 2021
Ecole d'Eté en Intelligence Artificielle
fondation Vallet
Cotonou, Bénin

1 Prise en main du code, intuition sur les fonctions

Aller dans le répertoire `Project`.

1. Ouvrir `3Dplots`. C'est un fichier pour dessiner des fonctions en `d<-2` dimensions ("contour plots" et "plot 3D"). Dessiner plusieurs des fonctions données dans `test_functions`[1]. Il suffit de changer le champ `fun<-` et mettre le nom de la fonction à dessiner (par exemple `fun<-quadratic` ou `fun<-rosen` ou `fun<-ackley`, ...).

   (a) Repérer quelles fonctions sont multimodales (i.e., ont plusieurs optima locaux différents).

   (b) Changer la position de l'optimum `glob_xstar` et le conditionnement `cond.no` de la fonction `quadratic`. Regarder l'effet sur les dessins. La fonction `quadratic` est une fonction test fondamentale pour le développement théorique et pratique des optimiseurs: c'est la fonction la plus simple qui possède un minimum, et on peut considérer que localement autour des optima locaux toutes les fonctions sont quadratiques (d'après le développement de Taylor à l'ordre 2).

[1] Les fonctions de `test_functions` marchent avec des dimensions arbitraires

2. Ouvrir `mainOptim`. Il s'agit du programme principal qui permet de

   - formuler le problème (liste `PbFormulation`): choisir une fonction (champ `fun`), son nombre de dimensions (champ `d`), ses bornes (champs `LB,UB`)
   - choisir les paramètres de l'algorithme d'optimisation (liste `optAlgoParam`). Ici ce sont les paramètres des variantes de la méthode de descente vues en cours: des critères d'arrêt (champs `budget`, `minGradNorm`, `minStepSize`), l'activation de la recherche en ligne (`linesearch_type <- "armijo"`) ou non (`linesearch_type <- "none"`), la méthode d'estimation de la direction de recherche (`direction_type <- "gradient"` ou `"momentum"` ou `"NAG"`), le facteur de taille de pas quand la recherche en ligne n'est pas active (`stepFactor`), ...

   (a) Avec la fonction `quadratic` et un point initial pas trop proche de l'optimum `glob_xstar`, observer l'effet de `stepFactor` sur les itérations quand il n'y a pas de recherche en ligne (`linesearch_type <- "none"`). Remarquer qu'en partant du même point avec la recherche en ligne (`linesearch_type <- "armijo"`), la convergence a lieu, au prix de quelques évaluations supplémentaires à chaque itération.

   (b) La fonction Rastrigin est multimodale. Sur la fonction `"rastrigin"` en `d<-2` dimensions, observer des convergences locales, i.e., des points de convergence qui sont des optima locaux mais pas globaux. Pour être sûr de bien observer le phénomène, il faut laisser l'optimiseur converger suffisamment longtemps, typiquement `budget<-10000` et le critère d'arrêt est autre que le budget.

2 Création d'une nouvelle fonction

3 Ajout de restart à une méthode de descente

A Réponses et commentaires supplémentaires

1.(a).
Unimodales: `sphere,quadratic,rosen,L1norm,tunnel`. AN: `sphere,quadratic,L1norm`

sont convexes, pas les autres. `L1norm` est non différentiable.
Multimodales: `ackley,rastrigin,schwefel,michalewicz,quad_wave`

1.(b).
Quand on augmente le conditionnement, la fonction `quadratic` ressemble de plus en plus à une vallée profonde, néanmoins rectiligne, contrairement à `rosen`.

2.(a). Le conditionnement de la fonction quadratique a un role, plus il est grand, plus `stepFactor` doit être petit. Si `stepFactor` est trop grand, la méthode diverge, en d'autres termes elle sort des bornes de $\mathcal{S}$. Vice versa, un petit `stepFactor` permet la convergence, mais à une vitesse inférieure (petits pas). Exemple de valeurs: `xinit<-c(-4.9,-4.9)`, `cond.no<-3`, `stepFactor <- 0.01,0.1,0.9`.

2.(b). Pour trouver plus facilement des points initiaux menant à une convergence locale, générer le point initial au hasard avec `optAlgoParam$xinit <- runif(n = d,min = pbFormulation$LB,max = pbFormulation$UB)`