Enoncé TP optimisation globale
Module 1 majeure science des données / Cours Optimisation Classique 2017

Rodolphe Le Riche

Evaluation du cours : (TP optimisation locale + TP optimisation globale)/2

Ce TP comporte deux étapes: une prise en main de programmes en R pour visualiser des fonctions coûts et tester des optimiseurs globaux; et un problème à construire et étudier.

Phase 1: prise en main des programmes en R.

1. Télécharger les fichiers mis à disposition sur Campus (ICM / Majeures / Majeure Sciences des données / Statistique exploratoire et outils mathématiques / Cours Optimisation Classique)

Les décompresser, et parcourir la présentation des codes ci-dessous, qui est un extrait du fichier readme.txt .

```
-----------------------------------------------------
MAIN PROGRAMS

The 3 files 3Dplots.R, main4tests.R, postproc_tests_optimizers.R :
* 3Dplots.R : make 3D plots of 2D functions and superpose 1 trajectory of
an optimizer.
* main4tests.R : repeat optimizations to study the behavior of a
particular optimizer with a particular objective function.
* postproc_tests_optimizers.R : do summary plots of one or many repeated
optimizations that were previously performed with main4test.R .

* In main4tests.R describe the function you want to optimize (with the
string nameoffun) and the optimizer you want to use (with nameofoptim).
Also choose the number of repetitions of the optimization (no_tests)
In postproc_tests_optimizers.R compare the results of different
optimizations by filling the vector of strings filenames.

-----------------------------------------------------
BASIC USE

* In 3Dplots.R and main4tests.R :
- fill in the name of the objective function (string nameoffun) and
       the name of the optimizer (string nameofoptim). Possible names for
       nameoffun are the functions given in test_functions.R . Possible
       names for nameofoptim are the optimization algorithms R functions
       present in the directory. Examples of both are given in comments.
- fill in other characteristics of the objective function:
       number of dimensions (variable zdim, which must be = 2 in
       3Dplots.R), position of the optimum (variable glob_xstar), noise
       characteristics (glob_noisy, glob_tau)
- fill in parameters of the optimizer: they are all fields of the param
       list, cf. param specific to each optimizer. Typically, one finds
```

```
        lower and upper bounds on the optimization variables (LB and UB),
        and maximum number of calls to the objective function (budget),
        and the rest depends on nameofoptim.
- then execute the file.
- The results of 3Dplots.R are R plots which are also saved as image files
        fileofplot.png and contour.png . You may want to copy these files
        elsewhere or change their names otherwise they will be overwritten
        by the next call to 3Dplots.R .
- The results of main4tests.R are stored in the files
        "nameofoptim"_"nameoffun"_{some characteristic params}.RData with
        plots in "nameofoptim"_"nameoffun"_{some characteristic
        params}.pdf . The data in the .RData file is meant to be reloaded
        later (cf below). The names of the result files are weakly unique,
        it is probably safer to change them (or move them to another
        directory) before using main4tests.R again.

* In postproc_tests_optimizer.R :
- fill in the names of the .RData files you want to process in order to
        compare the performance of the optimizers (string vector
        filenames). Source the file and enjoy.

------------------------------------------------------
LIST OF FILES

3Dplots.R : plots 2D functions in 3D and superimposes optimizer
       trajectories
cmaes.R : CMA-ES optimization algorithm
lbfgs.R : memory light BFGS optimization algorithm
main4tests.R : main program to call optimizer. Can perform many
        optimizations.
NSalgorithm.R : a fixed step ES-(1+1) search algorithm
of_wrapper.R : wraps calls to the objective function in order to store all
        calls thru global variables when necessary
postproc_tests_optimizers.R : creates plots to compare different
        optimizers
readme.txt : some explanations
RSalgorithm.R : a random search algorithm
utility_optim_functions.R : set of utility functions for plotting and
        casting our data structures
test_functions.R : a set of test functions (potentially noisy) with
        control on the location of the optimum
```

2. Ouvrir 3Dplots.R . Ce fichier permet de visualiser des fonctions 2D et des trajectoires d’optimiseurs. Utiliser l’optimiseur local « lbfgs » sur une fonction convexe (la fonction "sphere" ou "quadratic") puis sur une fonction multimodale au choix. Faire varier le point initial. Que se passe-t-il ?

3. Répéter l’expérience précédente avec un des deux optimiseurs globaux fournis (« random_search » ou « normal_search »). Noter que « normal_search » peut se comporter comme un optimiseur local ou global en fonction du choix de sigma.

4. Ouvrir « main4tests.R ». Nous allons maintenant faire des expériences numériques en répétant

des optimisations. Choisir comme optimiseur random_search puis normal_search, et faire des tests sur la fonction « sphere » en faisant varier la taille du pas sigma (cas de normal_search). Utiliser le fichier « postproc_tests_optimizers.R » pour faire des courbes synthétiques. Discuter les résultats.

5. Comparaison d'optimiseurs locaux et globaux sur une fonction multimodale.
Choisir une fonction multimodale en dimensions 5 et comparer les performances des optimiseurs fournis (lbfgs, normal_search, random_search, cmaes). Pour lbfgs, cmaes et normal_search, prendre le même point initial. lbfgs étant un optimiseur déterministe, un seul test est suffisant.

Phase 2: étude d'un problème, le plan d'expériences optimal

1. Programmer une nouvelle fonction objectif (comme celles dans "test_functions.R") correspondant au problème suivant: trouver un points $\boldsymbol{x}$ qui soient le plus éloigné possible de n points $\boldsymbol{p}^j$ déjà fixés. C'est un problème classique, difficile et important de planification d'une nouvelle expérience quand n expériences ont déjà été réalisées et chaque expériences est décrite par d paramètres.
Mathématiquement, le problème est formulé comme

$$\min_{\boldsymbol{x}\in[L,U]^d\subset\Re^d} -\left[\min_{j=1,n}\left\|\boldsymbol{x}-\boldsymbol{p}^j\right\|\right]$$

où $\boldsymbol{p}^j\in\Re^d$, $j=1,n$ sont des points du plan d'expériences donnés
et L, U sont les bornes sur les variables.

Pour se faire une intuition, on considérera d'abord les cas d=1 et 2 et on fera des dessins.

2. Etudier quels optimiseurs semblent bien ou mal fonctionner sur ce problème en fonction de d (la dimension) et n (le nombre de points déjà connus).

3. (Bonus) modifier soit l'optimiseur "normal_search" (dans l'esprit CMA-ES ou en ajoutant des recherches locales) soit "lbfgs" (en ajoutant des redémarrages) pour essayer d'améliorer l'efficacité des optimiseurs. L'évaluation sera plus sur les idées proposées que la performance constatée (car vous avez peu de temps).