TP cours 3 : Optimisation sur base de krigeage

Ecole-chercheur Mexico, La Rochelle

N. Durrande - V. Picheny

L'objectif du TP consiste à trouver les paramètres du simulateurs numérique du volcan qui minimisent l'erreur de prédiction par rapport aux données observées par satellite. On s'intéresse donc à un problème de calibration que l'on va traiter comme un problème d'optimisation.

Ce TP comprend deux parties : pour commencer il faudra obtenir un bon modèle de krigeage qui approche la fonction `compute_wls` (deuxième partie du précédent TP - qu'on approfondira ici). La seconde partie consiste à utiliser l'algorithme EGO du package `DiceOptim` pour trouver les paramètres qui minimisent cette erreur.

1 Modélisation par krigeage

Q1. Récupérer le meilleur plan d'expériences de 100 points en dimension 5 obtenu lors de la séance de lundi. Si vous n'êtes pas arrivé à un résultat convainquant vous pouvez utiliser le plan d'expériences et les observations fournies dans le fichier `XY_volcano.Rdata` (à charger avec la fonction `load`).

Q2. Tester différentes covariances et tendances en estimant les paramètres par maximum de vraisemblance. Les valeurs des paramètres estimés peuvent-elle nous renseigner sur la fonction que l'on approche ? Proposer une interprétation.

Q3. Tester la qualité de prédiction de la moyenne de krigeage de différents modèles en calculant le critère

$$Q_2 = 1 - \frac{\sum_{i=1}^{n}(y_i - m(x_i))^2}{\sum_{i=1}^{n}(y_i - mean(y_i)^2}$$

sur des résidus obtenus par leave-one-out (cf fonction `leaveOneOut.km`). Vous avez peut-être constaté précédement qu'ajouter des termes de tendance augmente toujours la vraisemblance du modèle mais est-ce que cela augmente forcément le Q_2 ?

Q4. Normaliser les résidus obtenus par leave-one-out et utiliser le code qui vous est fourni pour comparer leur distribution à une loi $\mathcal{N}(0,1)$. Le résultat vous parrait-il satisfaisant ? Pour compléter, utiliser la fonction `plot(model)`. Choisir le modèle qui vous parrait le meilleur, c'est celui-là que l'on utilisera par la suite pour l'optimisation.

2 Optimisation Globale avec EGO

Q5. Charger la package `DiceOptim` et utiliser la fonction `EGO.nstep` pour effectuer `nsteps=20` itérations de l'algorithme EGO. Cette fonction prend en entrée le modèle de krigeage ainsi que la fonction à optimiser `compute_wls`. Par la suite, on notera `res` l'objet qui est retourné par la fonction `EGO.nsteps`.

Q6. Observer les différents éléments contenus dans `res`. Tracer l'évolution des valeurs de la fonction objectif aux points visités par EGO en fonction de l'itération. Quelle est la plus petite valeur observée et quels sont les paramètres associés ? Y a-t-il une amélioration par rapport à la meilleure valeur observée sur le plan d'expérience ? Comment observe-t-on le compromis exploration / intensification ?

Q7. Représenter à l'aide d'un graphique `pairs` la distribution dans l'espace des paramètres testés par l'optimiseur. Identifier des zones d'exploration et d'intensification.

Q9. Utiliser la fonction `sectionview` de `DiceView` pour représenter une vue en coupe du modèle (`res$lastmodel`) centrée sur l'optimum.

Q10. Reprendre les questions 5 à 9 en augmentant la valeur de `nsteps`. De manière alternative, on peut essayer de relancer `EGO.nstep` à partir du dernier modèle (`res$lastmodel`), où d'un nouveau modèle bâti à partir du plan d'expériences enrichi des 20 points.

Q11 (bonus). Reprendre les dernières questions sur l'analyse de sensibilité à l'aide de krigeage du TP précédent. Les résultats sont-il cohérents avec le comportement observé de l'algorithme EGO ?

Il y a 2 solutions : soit faire l'AS sur la moyenne du krigeage, soit faire l'AS sur des trajectoires. Dans le premier cas, on a besoin d'encapsuler la sortie de la fonction `predict(m)`. Dans le deuxième cas, on peut directement utiliser la fonction `sobolGP` du package `sensitivity`. Se reporter au script d'aide. Comparer les deux résultats obtenus et les temps de calcul respectifs.