TP cours 2 : Modélisation par krigeage

Ecole-chercheur Mexico, La Rochelle

N. Durrande - V. Picheny

Cette session est dédiée à la construction d'un modèle de krigeage pour approximer la fonction `compute_wls` qui correspond à l'erreur du simulateur numérique. Ce metamodèle sera utilisé par la suite pour l'étape de calibration du simulateur.

Le script (très) incomplet `lab2_script.R` vous est fourni pour vous aider à traiter certaines questions.

1 Construction d'un modèle 1D

Q1. Un premier jeu de données jouet vous est fourni dans le fichier `toy_data.Rdata` : 10 observations Y qui dépendent d'un seul paramètre X. Le charger `load(toy_data.Rdata)` et représenter graphiquement `Y` contre `X`. Que pouvez-vous intuiter de la régularité de la fonction f ?

Q2. Quelques lignes de codes vous sont données pour générer des trajectoires non conditionelles à partir du package DiceKriging. Ce code construit d'abord un modèle de krigeage `m` à l'aide de la fonction `km`, génère des trajectoires (fonction `simulate`) sur une grille (`x`). Modifier la valeur des paramètres `covtype`, `coef.var` et `coef.cov` afin de comprendre l'influence de ces paramètres (il n'est pas interdit de consulter la documentation !). Attention aux intervalles de variations : `coef.var` joue sur la variance des trajectoires et `coef.cov` sur leur régularité (il correspond à une distance sur x). Trouver un jeu de paramètre qui vous semble adapté au jeu de données.

Q3. Simuler et représenter des trajectoires conditionnelles (même script mais option `cond` de la fonction `simulate`). Tracer sur le même graphique les observations (`points(X,Y)`) ; qu'observez-vous ? Tracer la moyenne et des intervalles de confiance à l'aide de la fonction `sectionview` du package `DiceView`.

Q4. Changer les paramètres d'appel de la fonction `km` afin d'estimer les paramètres de covariance par maximum de vraisemblance (il suffit d'enlever certains arguments, cf. documentation). Vous pouvez ensuite utiliser la commande `print(m)` pour afficher des détails sur le modèle. Les paramètres trouvés sont-ils proche de ceux que vous aviez intuités ?

Q5. La fonction `leaveOneOut.km` est bien pratique pour calculer les prédictions du modèle aux points du plan d'expérience en utilisant tous les points du plan sauf celui où l'on effectue la prédiction. Utiliser ce vecteur de prédiction pour calculer les résidus standardisés (erreur de prédiction divisée par l'écart type de prédiction). Les comparer avec une distribution normale centrée réduite. Le résultat vous parait-il satisfaisant ? Vérifier la cohérence de vos résultats avec la commande `plot(m)` qui permet d'automatiser cette tâche.

Q6. Utiliser la fonction `predict` pour prédire la valeur de f au point $x = 4$. Pouvez-vous expliquer (et critiquer) le résultat obtenu ? Pour compléter, visualiser le modèle avec `sectionview` et l'argument `xlim=c(-0.2, 4)`. Créer un nouveau modèle en estimant les paramètres de tendance (krigeage universel) et jouer sur `formula` (premier argument) de la fonction `km` pour obtenir un modèle plus satisfaisant.

2 Cas test du volcan

Q7. Récupérer le meilleur plan d'expérience de 100 points en dimension 5 obtenu lors de la séance d'hier. Si vous n'êtes pas arrivé à un résultat convainquant vous pouvez utiliser la correction présente (`XY_volcano.Rdata`) sur dropbox.

Q8. Tester différentes covariances et tendances en estimant les paramètres par maximum de vraisemblance.

Q9. Valider le modèle (moyenne et intervalles de confiance) en utilisant `plot(model)`. Effectuer des représentations graphiques à l'aide de la commande `sectionview(m, center = rep(0.5, 5))`.

Q10 (bonus). Utiliser la fonction `sobolGP` du package `sensitivity` pour effectuer une analyse de sensibilité sur le metamodèle.