MODÉLISATION STOCHASTIQUE - DATA731

Travaux Pratiques / Modèles statistiques récursifs

Objectifs

On s'intéresse dans cet ensemble d'exercices à la construction de modèles récurrents dans un objectif d'anticiper/synthétiser/prédire une valeur ou un événement à partir d'une série d'observations.

Exercice 1 : Étude bibliographique et rappels des pré-requis

L'intérêt modèles récurrents est immédiat pour les acteurs/analystes des marchés financiers, mais pas seulement : les applications sont très nombreuses et le premier exercice demandé consiste à identifier les domaines majeurs où la modélisation et l'analyse prédictive sont une approche systématique. Ensuite, il faut analyser les variantes de modélisations couramment utilisées. Enfin, il est conseillé d'actualiser ses connaissances sur les modèles autoregréssifs et l'estimation des paramètres de ces modèles en utilisant les équations de Yule-Walker.

Exercice 2 : [rétro-TP-AR] Analyse et synthèse par modèle AR

Environnements de programmation : Matlab (licence gratuite via le portail USMB), Python et/ou R.

L'objectif de cet exercice décliné sous la forme de TP par : (a) pédagogie inversée, du code vers les équations codées et (b) l'apprentissage par transfert et re-interprétation de code [1], est la prise en main effective des outils essentiels à l'identification de modèles autoregréssifs (AR). Pour cela, des fichiers de démonstration vous sont fournis ('EstimationModeleAR' et 'AnalyseEtSyntheseParoleParModeleAR' en '.py' et '.m' avec des niveaux de complétude différents).

Partie I Exécuter 'EstimationModeleAR' et donner des explications détaillées sur les déroulements des programmes correspondants. Il est conseillé de :

- faire plusieurs simulations (les réalisations des processus sont différentes à chaque nouvelle exécution),
- modifier certains paramètres et étudier l'influence des modifications sur les résultats,

afin de déduire des conclusions qui mettent en évidence le niveau de maîtrise atteint. Ensuite, complétez la partie manquante pour que le code python reproduise le même résultat que le code matlab. Pour ceux qui ont une préférence pour l'environnement de programmation R, il faut plutôt traduire intégralement le code matlab en code R.

Partie II Même exercice sur le fichier 'AnalyseEtSyntheseParoleParModeleAR'. Étudier entre autres l'impact de la longueur de trame d'analyse sur la qualité du modèle. Commenter en particulier les performances du modèle AR en synthèse de parole sachant qu'il s'agit d'un modèle 'simple' n'impliquant que des dépendances linéaires entre variables.

Attendus : Rapport d'analyse et d'interprétation des procédures de modélisation et des formules mathématiques impliquées dans les programmes '.m' fournis + Code opérationnel simple en Python ou en R. J'insiste sur 'code simple', terme à différencier de 'code optimisé'.

Exercice 3 : [direct-TP-AR] Identification de modèle AR

Cet exercice permet de vérifier par la pratique, les concepts acquis dans l'exercice précédent. Il peut cependant se traiter indépendamment de l'exercice 2. Les fichiers de programmes et données n'étant pas fournis, l'environnement de programmation est libre.

1°) Créer 3 séries temporelles y_1, y_2, y_3 par simulation stochastique selon les équations :

$$y_1[k] + a_1 y_1[k-1] + a_2 y_1[k-2] = z_1[k]$$

$$y_2[k] + b_1 y_2[k-1] + b_2 y_2[k-2] = z_2[k]$$

$$y_3[k] + c_1 y_3[k-1] + c_2 y_3[k-2] = z_3[k]$$

où z_1, z_2, z_3 représentent des réalisations d'un bruit blanc Gaussien centré de variance unité et :

$$[a_1, a_2] = [-0.0707, 0.2500]; \qquad [b_1, b_2] = [-1.6674, 0.9025]; \qquad [c_1, c_2] = [1.7820, 0.8100].$$

1. Déchiffrer et traduire simplement dans un langage de programmation simple (Python) un code simple fournit dans un autre langage de programmation simple (Matlab).

Visualiser ces séries et, pour chacune d'entre elles, donner la famille de modèle stochastique qui lui est associé. Ces séries, sont-elles stationnaires ?

2°) Calculer et visualiser les fonctions d'autocorrélations et les spectres (densités spectrales de puissance) de y_1, y_2, y_3. Indiquer laquelle de ces 2 fonctions permet aisément de mettre en évidence les différences entre ces 3 processus.

3°) Créer une série temporelle y constituée par la somme des trois processus synthétisés en 1°). Visualiser cette série et indiquer sa nature. Calculer et visualiser la fonction d'autocorrélation et la densité spectrale de puissance de y.

4°) On décide de modéliser y par un processus autorégressif d'ordre 2. Estimer les coefficients de ce modèle et comparer les autocorrélations/densités spectrales de y et du modèle estimé.

5°) Mêmes questions qu'au 4°) mais avec cette fois-ci des modèles autorégressifs d'ordres 3, 4, 5, 6, 8, 12, 20, 30 respectivement associés à y. Discuter les résultats obtenus.

6°) On définit une série s par juxtaposition des 2 séries y_1, y_2, y_3 définies dans l'exercice 1. Estimer les paramètres de modèles autorégressifs d'ordres 3 et 4 pouvant être associés à s.

Exercice 4 : Modèles AutoRégressifs Non-linéaires (NAR) avec entrées eXogènes (NARX) (pour aller plus loin)

Les processus réels tels que la parole (étudiée dans l'Exercice 2) sont très complexes, intégrant des variables dont les relations ne sont pas forcément linéaires. On peut donc espérer améliorer les performances d'identification de modèles en ajoutant des composantes non-linéaires dans les fonctionnelles AR. Ce sera l'un des objectifs du projet de fin de module DATA731. On propose dans cet exercice, une première étape de familiarisation aux modèles non-linéaires dits NAR et NARX intégrant à la fois des fonctions neuronales et des linéarités AR.

1°) Cette familiarisation nécessite d'abord de traiter beaucoup d'exemples concernant les modèles simples AR, par exemple en suivant les nombreux tutoriels disponibles en ligne, notamment (Python et Matlab respectivement) :

- `https://machinelearningmastery.com/autoregression-models-time-series-forecasting-python/`
- `https://www.mathworks.com/help/ident/ref/forecast.html`

Appliquez ces traitements sur des données de votre choix et mettez en évidence les relations régissant les interactions entre les variables associées à ces données.

2°) Concernant la prise en compte de non-linéarités, cette familiarisation est plutôt facile sur Matlab (environnement tout-intégré, ou presque) via les exemples suivants :

- (NAR) `https://www.mathworks.com/help/deeplearning/ref/narnet.html`
- (NARX) `https://www.mathworks.com/help/deeplearning/ref/narxnet.html`

L'utilisation de Python peut nécessiter l'installation de bibliothèques tierces telles que : `http://pyneurgen.sourceforge.net/tutorial_nn.html` et la prise en main des modèles récurrents associés (dont les NARX) est disponible à : `http://pyneurgen.sourceforge.net/recurrent.html`

Même question qu'en 1°). On veillera à bien différencier les données utilisées en phase d'*apprentissage* (estimation des paramètres du modèle) de celles utilisées pour les *tests* (prédiction à partir du modèle appris).