DÉPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE
UNIVERSITÉ DE MONTRÉAL

IFT3150 : PROJET D'INFORMATIQUE

Algorithme du gradient stochastique pour l'estimation de modèles de choix discrets

Alexandre BRILHANTE

Supervisé par
Prof. Fabian BASTIN

8 janvier 2018

Table des matières

Liste des tableaux

Table des figures

1 Introduction

1.1 Description

Le succès actuel des méthodes de réseaux neuronaux, en particulier de réseaux profonds, ont eu comme effet secondaire de populariser l'algorithme du gradient stochastique, connu également en programmation stochastique comme la méthode d'approximation stochastique parmi la communauté de programmation non-linéaire. Bien qu'à ce jour, de nombreuses questions demeurent ouvertes, des progrès significatifs ont été obtenus dans l'étude de la convergence de la méthode ainsi que dans la recherche de stratégies d'accélération de la méthode.

Cette méthode d'optimisation demeure toutefois marginale dans le cadre de l'estimation de modèles de choix discrets, or le problème d'estimation par maximum de vraisemblance d'un modèle logit multinomial est mathématiquement très similaire au problème d'entraînement d'un réseau neuronal présentant des fonctions d'activation sigmoïdes. Pourtant, certaines extensions du modèle logit, en particulier le modèle mixed logit, demeure à ce jour numériquement coûteux à estimer comme chaque observation requiert d'approximer l'espérance de la probabilité de choix, typiquement par intégration Monte Carlo.

Le but de ce projet est d'adapter l'algorithme du gradient stochastique pour l'estimation de modèles mixed logit par maximum de vraisemblance, et de vérifier numériquement son efficacité tant en termes de qualité d'estimation que de vitesse de convergence. Dans un premier temps, la taille d'approximation sera fixée pour chaque individu, permettant de comparer la méthode avec une approche directe, ou méthode batch. Par après, diverses extensions seront envisagées pour prendre en compte les faiblesses de l'algorithme de gradient stochastique que du modèle mixed logit. Tout d'abord, il est connu que le choix de la longueur de pas à chaque itération, aussi appelé taux d'apprentissage dans la communauté d'intelligence artificielle, influence significativement le succès ainsi que la rapidité de la méthode de gradient stochastique. En tenant compte des résultats théoriques récents, nous essayerons d'améliorer et d'automatiser la longueur de pas. Dans un second temps, nous intégrerons une approximation de second ordre, basée sur l'identité de l'information de Fisher, pour accélérer la méthode. Enfin, une des difficultés majeures des modèles mixed logit est la qualité d'approximation de l'espérance de chaque observation, qui non seulement affecte la précision des résultats d'estimation, mais est également à l'origine d'un biais dans le cadre de l'estimation par maximum de vraisemblance, en raison de l'opérateur logarithmique. Nous tâcherons de définir une approche de gradient stochastique à deux niveaux, variant à chaque itération non seulement l'ensemble d'observations considérées (souvent réduite à un singleton), mais également l'échantillonnage Monte Carlo utilisé pour approximer l'espérance de probabilité de choix.

1.2 Survol

Commençons par une illustration simple d'un choix discret :

Où allons-nous dîner ?

Plusieurs paramètres possibles peuvent influencer ce choix :
- Budget ;
- Temps disponible ;
- Distance ;
- Goûts personnels ;
- etc.

Un agent économique, une firme ou un individu, fait face à un choix, ou une série de choix, au cours du temps, parmi un ensemble fini d'alternatives. Nous voulons observer le processus comportemental menant au choix de l'agent. Ce dernier veut maximiser son utilité. On suppose donc que nous pouvons capturer les préférences individuelles au moyen d'utilité associée à chaque alternative connue par l'agent mais pas par le chercheur.

L'utilité se décompose en une partie observée V et une partie non-observée ϵ, aléatoire, et ce de manière additive. Nous pouvons donc seulement calculer des probabilités de choix.

Les alternatives forment l'ensemble de choix. En voici les propriétés :
- L'ensemble de choix doit être exhaustif et donc représenter toutes les alternatives possibles ;
- Les alternatives doivent être mutuellement exclusives choisir une alternative fait en sorte que toutes les autres sont rejetées ;
- L'ensemble de choix doit contenir un nombre fini d'alternatives.

1.3 Choix discrets

Pour tenir compte des attributs et des caractéristiques non-observées, nous décomposons les utilités comme suit :

$$U_{ij} = V_{ij} + \epsilon_{ij}$$

où

$$V_{ij} = V_{ij}(\beta_j, x_{ij}) = \beta_j^T x_{ij} = \sum_{k=1}^{K_j} \beta_j^k x_{ij}^k$$

et
- i : individu ;
- j : alternatives ;
- ϵ_{ij} : facteurs non-observés suivant une distribution de Gumbel ;
- x_{ij} : vecteur contenant tous les attributs ;
- K_j : nombre d'attributs ;

— β : vecteur des paramètres à estimer.

Notre fonction de choix est donc :

$$U_{ij} = V_{ij} + \epsilon_{ij}$$

Assumons que les ϵ_{ij} soient indépendants et identiquement distribués suivant une distribution de Gumbel avec un facteur d'échelle μ, nous avons :

$$U_{ij} = V_{ij}$$

Dès lors, la probabilité de choix de l'individu i choisisse parmi les alternatives $A_j \in A(i)$ peut être exprimée comme :

$$P_{ij}(\beta) = \frac{e^{\mu V_{ij}(\beta)}}{\sum_{m=1}^{|A(i)|} e^{\mu V_{im}(\beta)}}$$

1.3.1 Logit

Définissons maintenant le modèle logit comme suit :

$$L_{ij}(\beta) = P_{ij}(\beta) = \frac{e^{\mu V_{ij}(\beta)}}{\sum_{m=1}^{|A(i)|} e^{\mu V_{im}(\beta)}}$$

Ce modèle possède les propriétés suivantes :

— Peut représenter les variations de goût liées aux caractéristiques individuelles ;
— Substitution proportionnelle à travers les alternatives ;
— Ne peut capturer la dynamique de choix répétées.

1.3.2 Mixed Logit

Nous supposons que β_k peut être écrit comme :

$$\beta_k = h(\gamma, \theta)$$

où γ est un vecteur de tirs aléatoires d'une loi uniforme sur $(0, 1)$.

Pour un individu aléatoire i, β_i a une densité multivariée f_θ qui dépend d'un vecteur de paramètres θ puisque $f(\beta|\theta) = f(\gamma|\theta)$. L'individu i sélectionne toujours l'alternative j ayant la plus grande utilité U_{ij}.

Dès lors, notre modèle mixed logit sera défini comme :

$$L_{ij}(\gamma, \theta) = \frac{e^{V_{ij}(\beta(i), x_{ij})}}{\sum_{m=1}^{|A(i)|} e^{V_{im}(\beta(i), x_{im})}}$$

Ce modèle résout certaines limitations du modèle logit vu plus tôt :

— Permet une variation aléatoire de goûts ;
— Ne limite pas les effets de substitution où une augmentation de la probabilité d'une alternative implique nécessairement une diminution dans la probabilité pour les autres alternatives ;
— Permet la corrélation de facteurs non-observés au fil du temps.

1.3.3 Estimation

Logit :

$$\max_{\beta} LL(\beta) = \max_{\beta} \frac{1}{I} \sum_{i=1}^{I} \ln P_{ij}(\beta)$$

Mixed Logit :

$$\max_{\theta} LL(\theta) = \max_{\theta} \frac{1}{I} \sum_{i=1}^{I} \ln P_{ij}(\theta)$$

$$SP_{ij_i}(\theta) = \frac{1}{R} \sum_{r=1}^{R} L_{ij}(\gamma_r, \theta)$$

$$\max_{\theta} SLL^R(\theta) = \max_{\theta} \frac{1}{I} \sum_{i=1}^{I} \ln SP^R_{ij_i}(\theta)$$

Dans les deux modèles, nous désirons calculer la log-vraisemblance. Dans le cas du mixed logit, nous travaillons avec une approximation construite sur des tirs de γ et donc par intégration Monte Carlo. SP représente la probabilité simulée alors que SLL correspond à la log-vraisemblance simulée.

2 Algorithmes classiques

2.1 Newton

On cherche à construire une bonne approximation d'un zéro de la fonction d'une variable réelle $f(x)$ en considérant son développement de Taylor au premier ordre : partant d'un point x_0 que l'on choisit de préférence proche du zéro à trouver, on approche la fonction au premier ordre, soit à peu près égale à sa tangente en ce point.

Soit $s_k = x - x_k$. Le modèle quadratique peut s'écrire comme :

$$m(s_k) = f(x_k) + g_k^T s_k + \frac{1}{2} s_k^T H_k s_k$$

où

$$g_k = \nabla f(x_k)$$

et

$$H_k = \nabla^2 f(x_k)$$

Si H_k est définie positive, $m_(s_k)$ est strictement convexe et son minimum s'obtient en annulant la dérivée de $m_(s_k)$ par rapport à s_k, soit :

$$0 = g_k + H_k s_k$$

ce qui mène à :

$$s_k = -H_k^{-1} g_k$$

Pour des raisons de précisions numériques nous ne calculons pas H_k^{-1}, cependant on obtient s_k en résolvant :

$$H_k s_k = -g_k$$

La convergence n'est assurée que si le point de départ est suffisamment proche de la solution, et est alors quadratique. On peut obtenir la convergence globale en utilisant la direction de Newton comme direction de descente avec une méthode de recherche linéaire, en s'assurant que a_k puisse prendre la valeur 1 afin d'assurer une convergence quadratique au cours des dernières itérations.

Notons que si la hessienne n'est pas définie positive, le modèle n'est pas convexe, et annuler le gradient, pour autant qu'on puisse le faire, ne garantit pas de trouver un minimum local.

2.2 Région de confiance

La méthode de région de confiance a été proposée par Conn, Gould et Toint (2000). Nous considérons le problème $f(x)$ et le sous-problème $q(s)$:

$$\min_{x \in \Re^n} f(x)$$

$$\min_s q(s) = g^T s + \frac{1}{2} s^T H s$$

où $f(x) \in C^2$ et $q(s)$ est sous la contrainte :

$$\|s\| \leq \Delta$$

Le sous-problème est défini en prenant :

$$g = \nabla f(x_k)$$

$$H = \nabla^2 f(x_k)$$

$$M = R^T R \text{ préconditionneur}$$

Les algorithmes d'optimisation itératifs travaillent à chaque itération avec un problème d'optimisation bien plus simple. En région de confiance, le sous-problème reste en dimension n mais la fonction objectif est simplifiée et l'espace de recherche restreint. Chaque sous-problème de région de confiance doit être résolu approximativement et aussi efficacement que possible. Afin d'être capable de garantir la convergence globale de la méthode, nous voudrions au minimum que la solution approximative atteigne le même niveau de réduction de la valeur du modèle que ne le ferait un pas dans la direction de plus forte pente contraint par la région de confiance.

Algorithme basique de région de confiance :

Input vecteur initial x_0, rayon initial Δ_0, constantes η_1, η_2, γ_1 et γ_2 qui satisfont : $0 < \eta_1 \leq \eta_2 < 1$ et $0 < \gamma_1 \leq \gamma_2 < 1$

1: $f(x_0)$
2: $k = 0$
3: choisir $||\cdot||_k$ et définir un modèle m_k dans $B_k = \{x \in \Re^n | |x - x_k| | k \leq \Delta_k\}$
4: calculer un pas s_k qui réduit suffisamment m_k tel que $x_k + s_k \in B_k$
5: $f(x_k + s_k)$
6: $\rho = \frac{f(x_k) - f(x_k + s_k)}{m_k(0) - m_k(s_k)}$
7: **if** $\rho \geq \eta_1$ **then**
8: $\quad x_{k+1} = x_k + s_k$
9: **else**
10: $\quad x_{k+1} = x_k$
11: **end if**
12: **if** $\rho \geq \eta_2$ **then**
13: $\quad \Delta_{k+1} = [\Delta_k, \infty]$

end if
if $\rho \in [\eta_1, \eta_2]$ **then**
    $\Delta_{k+1} = [\gamma_2 \Delta_k, \Delta_k]$
end if
if $\rho < \eta_1$ **then**
    $\Delta_{k+1} = [\gamma_1 \Delta_k, \gamma_2 \Delta_k]$
end if
$k = k + 1$
Output s

Présentons maintenant le calcul du pas : le gradient conjugué tronqué, aussi connu sous le nom de la méthode de Steihaug-Toint. Ce dernier consiste à minimiser le modèle quadratique en choisissant chaque direction de recherche de sorte qu'elle soit conjuguée à la direction de recherche précédente par rapport à la hessienne jusqu'à ce l'on traverse la frontière de la région de confiance ou que l'on rencontre une direction à courbure négative auquel cas on la suit jusqu'à la frontière de la région de confiance. Sans la troncature, rien ne garantit que la méthode converge.

Gradient conjugué tronqué :
Input vecteur initial x_0
$g_0 = \nabla f(x_0)$
$v_0 = M^{-1} g_0$
$d_0 = -v_0$
for $k = 1, 2, 3, \ldots, n$ jusqu'à convergence **do**
    $\kappa_k = d_k \times H d_k$
    if $\kappa_k \leq 0$ **then**
        calculer σ_k comme la racine positive de $||s_k + \sigma_k d_k||_M = \Delta$
        $s_{k+1} = s_k + \sigma_k d_k$
        break
    end if
    $\alpha = \frac{g_k \times v_k}{\kappa_k}$
    if $||s_k + \alpha_k d_k||_M \geq 0$ **then**
        calculer σ_k comme la racine positive de $||s_k + \sigma_k d_k||_M = \Delta$
        $s_{k+1} = s_k + \sigma_k d_k$
        break
    end if
    $s_{k+1} = s_k + \alpha_k d_k$
    $g_{k+1} = g_k + \alpha_k H d_k$
    $v_{k+1} = M^{-1} g_{k+1}$
    $\beta = \frac{g_{k+1} \times v_{k+1}}{g_k \times v_k}$
    $d_{k+1} = -v_{k+1} + \beta_k d_k$
end for
Output s

Aussi longtemps que le nombre de conditionnement de M reste borné sur la séquence de sous-problèmes approximativement résolus par l'algorithme de région de confiance alors n'importe quel itéré généré par l'algorithme de gradient conjugué tronqué est suffisant pour assurer la convergence vers un point critique de premier ordre. Ceci découle du fait que le premier itéré généré par l'algorithme du gradient conjugué tronqué est le pas de Cauchy pour le modèle et n'importe quel itéré généré par la suite donne une valeur plus petite pour le modèle.

3 Algorithme du gradient stochastique

La descente de gradient stochastique repose sur l'idée de la descente locale : on modifie itérativement ω pour diminuer $f(\omega)$, jusqu'à ce que ça devienne impossible, c'est-à-dire que nous sommes arrivés à un minimum local et peut-être global avec de la chance. Dans la descente de gradient stochastique, la vraie valeur du gradient de $f(\omega)$ est donc approchée par le gradient d'une seule composante de la somme, soit un seul individu, sélectionné au hasard, dans notre jeu de données. La méthode peut alors être beaucoup plus rapide que la descente de gradient classique, ou méthode de la plus forte pente, parce que le nombre de mises à jour est beaucoup plus grand. C'est particulièrement vrai pour des jeux de données volumineux.

Algorithme du gradient stochastique :

Input vecteur initial ω, j taille de mini-lots, taux d'apprentissage η

while jusqu'à convergence et k $\leq$ nombre d'itérations maximum **do**
    mélanger le jeu de données
    for i = 1, 2, 3, ..., n **do**
        construire un mini-lot avec les individus i à j
        calculer les scores pour chaque individu du mini-lot
        calculer le pas avec le gradient conjugué tronqué à partir de $\nabla f(\omega)$,
        des scores et η
        $\omega = \omega$+pas
        $i = i + j$
    end for
end while

Output ω, k

Précisons que par scores, soient la matrice contenant les gradients de chaque individu, nous utilisons la propriété de l'algorithme de Berndt, Hall, Hall et Hausman (BHHH). Basé sur l'identité de l'information de Fisher, nous accélérons l'algorithme sans avoir à calculer la hessienne exacte ce qui aurait pu être très coûteux en temps de calcul et en mémoire.

Notons aussi que le le vecteur initial ω, la taille des mini-lots j et le taux d'apprentissage η doivent être bien choisis. En effet, η trop grand pourrait faire diverger la méthode, alors que s'il est trop petit la méthode pourrait converger mais de façon très lente. Pour nos résultats, nous avons déterminé ω_0 très près de la solution, 7 et 210 pour j et 0.01 pour η.

4 Résultats et analyse

4.1 Résultats

Présentons notre jeu de données, de Greene (2011), utilisé pour nos estimations :

- Jeu de données sur les transports en Australie ;
- 210 individus ;
- 4 alternatives ;
- 6 paramètres pour le logit, 7 pour le mixed logit ;
- Le 5e paramètre sera simulé 1000 fois pour le modèle mixed logit ;
- 1000 itérations maximum pour chaque algorithme.

Définissons :

- *GC* : gradient conjugué ;
- *GCT* : gradient conjugué tronqué ;
- *GS* : gradient stochastique ;
- j : taille du mini-lots ;
- k : nombre d'itérations.

Voici donc le tableau 1 représentant les résultats des 4 algorithmes.

Algorithme	β_1	β_2	β_3	β_4	β_5	β_6	k
Newton	5.20744	3.86904	3.16319	0.01329	-0.09613	-0.01550	7
GC	5.20744	3.86904	3.16319	0.01329	-0.09613	-0.01550	5
GCT	5.20744	3.86904	3.16319	0.01329	-0.09613	-0.01550	10
GS, $j = 210$	5.20702	3.86923	3.16245	0.01329	-0.09612	-0.01550	73

TABLE 1 – Résultats : Logit

De même que le tableau 2 pour le modèle mixed logit :

Algorithme	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	k
Newton	20.0264	12.915	11.8469	-0.00981	-0.33295	0.00137	-0.03086	7
GC	20.0264	12.915	11.8469	-0.00981	-0.33295	0.00137	-0.03086	5
GCT	20.0264	12.915	11.8469	-0.00981	-0.33295	0.00137	-0.03086	14
GS, $j = 7$	19.995	13.0128	11.9929	-0.02608	-0.34082	-0.00428	-0.04377	1000
GS, $j = 210$	20.0441	12.9728	11.8984	-0.00992	-0.33371	0.00528	-0.03093	1000

TABLE 2 – Résultats : Mixed Logit

4.2 Analyse

4.2.1 Logit

Analysons maintenant les résultats. Pour le modèle logit, notre fonction f à optimiser est illustrée par la figure 1.

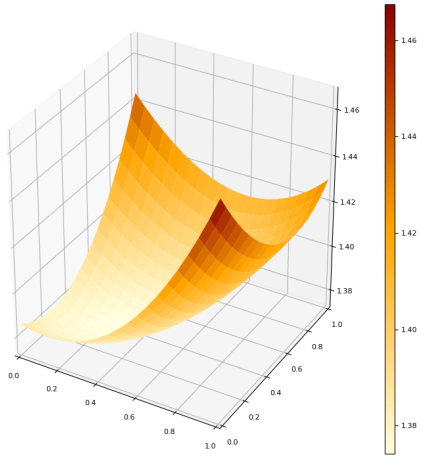

FIGURE 1 – Logit : f

Tel que présenté par le tableau 1, les méthodes classiques convergent assez rapidement. Effectivement, la méthode de Newton et le gradient conjugué tronqué trouvent la solution en 7 et 10 itérations respectivement alors que pour le gradient stochastique c'est plutôt en 73 itérations. La figure 2 présente la descente de gradient effectué par l'algorithme.

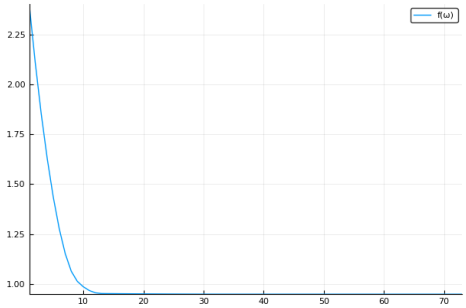

FIGURE 2 – Logit : ∇f

4.2.2 Mixed Logit

Passons maintenant au modèle mixed logit. Sa fonction f est illustré par la figure 3. Rappelons que nous avons simulé le 5e paramètre 1000 fois.

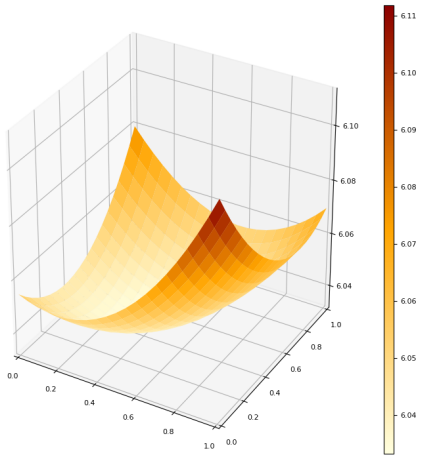

FIGURE 3 – Mixed Logit : f

Encore une fois, les méthodes classiques convergent rapidement. En ce qui concerne le gradient stochastique, que ce soit avec des tailles de 7 ou 210 pour les mini-lots, il se rapproche de la solution de plus en plus à chaque itération. Observons la descente de gradient effectué par l'algorithme avec les figures 4 et 5.

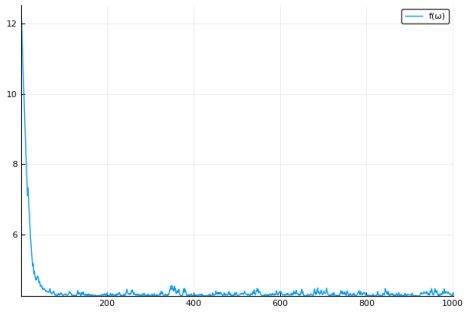

FIGURE 4 – Mixed Logit avec $j = 7$: ∇f

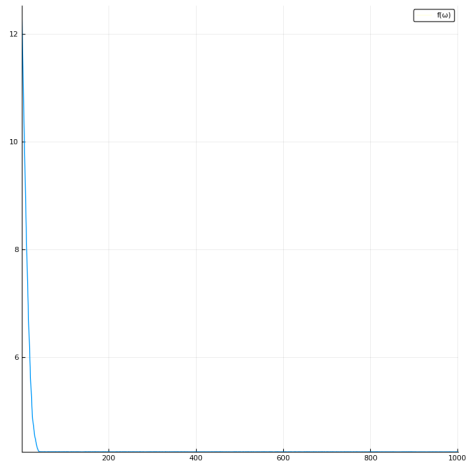

FIGURE 5 – Mixed Logit avec $j = 210 : \nabla f$

Dans le cas où les mini-lots sont de taille 210, la pente est assez forte ce qui peut nous faire penser que l'algorithme convergera plus rapidement sur des gros jeux de données. Dans l'autre cas, la pente, forte aussi, représente bien la propriété stochastique de chaque itération sur le mini-lot. Nous pouvons donc présumer que la taille des mini-lots et le taux d'apprentissage doivent être choisis judicieusement. Effectivement, nos premiers essais avec le gradient conjugué tronqué et BHHH n'ont pas été concluants. Dans le premier cas, en partant d'un point arbitraire assez loin de la solution., la vitesse de convergence de l'algorithme était extrêmement lente. Dans l'autre cas, un taux d'apprentissage n'améliorait pas la vitesse de convergence.

5 Conclusion

Dans ce projet nous avons donc exploré la méthode de gradient stochastique pour l'estimation de modèles de choix discrets. Alors que les méthodes classiques sont plutôt efficaces et convergent rapidement sur un jeu de données restreint, la question du big data se pose : ces méthodes seront-elles aussi efficaces sur des jeux de données immenses et qu'en est-il du gradient stochastique ? Dans ce cas précis, nous pensons que le gradient stochastique, de par ses propriétés de convergence, sera plus efficace que les méthodes classiques. Effectivement, sur des jeux de données contenant des millions, ou même des milliards, d'individus et de variables, le gradient stochastique est étonnamment efficace et son utilisation est très répandue en machine learning et en particulier en deep learning où il est utilisé pour l'entraînement d'un réseau neuronal présentant des fonctions d'activation sigmoïdes.

Par faute de temps, nous n'avons pas pu expérimenter avec d'autres modèles de choix discrets comme le nested logit, où les alternatives similaires sont regroupés par groupe, ou le probit, où chaque paramètre ne peut prendre que deux valeurs. À titre de considérations futurs, comme stratégie d'accélération de la méthode, nous pourrions ajouter un mécanisme de région de confiance au gradient stochastique afin d'en faire une méthode de gradient conjugué tronqué stochastique et se pencher davantage sur la sélection du taux d'apprentissage.

6 Bibliographie

Références

[1] Fabian Bastin, Trust-Region Algorithms for Nonlinear Stochastic Programming and Mixed Logit Models. Presses Universtaires de Namur, Namur, 2004.

[2] Andrew Conn, Nicholas Gould, Philippe Toint, Trust-Region Methods. Society for Industrial and Applied Mathematics, Philadephie, 2000.

[3] William Greene, Econometric Analysis. Pearson, Londres, 2011.

[4] Kenneth Train, Discrete Choice Methods with Simulation. Cambridge University Press, Cambridge, 2009.