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

Alexandre Brilhante

IFT3150: Projet d'informatique
Département d'informatique et de recherche opérationnelle
Université de Montréal

alexandre.brilhante@umontreal.ca

6 décembre 2017

Plan

Illustration

Où allons-nous dîner?

Plusieurs paramètres possibles peuvent influencer ce choix:

- Budget
- Temps disponible
- Distance
- Goûts personnels
- etc.

Contexte

Un agent économique 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.

L'agent veut maximiser son utilité. On suppose donc que nous pouvons capturer les préférences individuelles au moyen d'utilité, connue par l'agent mais pas par le chercheur, associée à chaque alternative.

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

Ensemble de choix

Les alternatives forment l'ensemble de choix.

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.

Fonction de choix

Pour tenir compte des attributs et 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 de paramètres à estimer

Probabilité de choix

Fonction de choix:

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

Assumons que les ϵ_{ij} soient indépendants et identiquement distribués, nous avons donc:

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

Dès lors, la probabilité de choix est:

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

Logit

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

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

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 $(0, 1)$.

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

- 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.

Estimation

Logit:

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

Mixed Logit:

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

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

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

Nous travaillons avec une approximation construite sur des tirs de γ.

Newton multidimensionnel

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.

Formellement, on part d'un point x_0 appartenant à l'ensemble de définition de la fonction et on construit par récurrence la suite:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Newton multidimensionnel: limites

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.

Régions de confiance

Nous considérons le problème $f(x)$ et le sous-problème $q(x)$:

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

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

où $f(x) \in C^1$ et $\nabla f(x)$ est lipschitzienne sur $\Re^n$ et $q(s)$ est sous la contrainte $\|s\| \leq \Delta$

Les algorithmes d'optimisation itératifs travaillent typiquement à chaque itération avec un problème d'optimisation bien plus simple.

- Dans le cas des méthodes de recherche linéaire, le sous-problème est plus simple car il est unidimensionnel.
- En régions de confiance, le sous-problème reste en dimension n mais la fonction objectif est simplifiée et l'espace de recherche restreint.

Régions de confiance

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.

Étapes:

- Choisir un pas s_k pour réduire le modèle de $f(x_k + s)$
- Accepter $x_{k+1} = x_k + s_k$ si la réduction prédite par le modèle est vérifiée en $f(x_k + s_k)$
- Autrement, poser $x_{k+1} = x_k$ et raffiner le modèle

Régions de confiance: calcul du pas

Gradient conjugué tronqué: on minimise le modèle quadratique en choisissant chaque direction de recherche de sorte qu'elle soit conjuguée à la direction de recherche précédente par rapport au Hessien 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.

Jeu de données

- Jeu de données sur les transports en Australie
- 210 individus
- 4 alternatives
- 7 variables
- β_5 sera simulé 1000 fois pour le modèle mixed logit

Résultats partiels - Logit et Mixed Logit

Logit:

Algorithme	β_1	β_2	β_3	β_4	β_5	β_6	k
Newton	5.21	3.87	3.16	0.01	-0.09	-0.01	7
GC	5.21	3.87	3.16	0.01	-0.09	-0.01	5
GCT	5.21	3.87	3.16	0.01	-0.09	-0.01	10

Mixed Logit:

Algorithme	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	k
Newton	20.03	12.92	11.85	-0.01	-0.33	0.00	-0.031	7
GC	20.03	12.92	11.85	-0.01	-0.33	0.00	-0.031	5
GCT	20.03	12.92	11.85	-0.01	-0.33	0.00	-0.031	12

Approximations du Hessien

BHHH: correspond à la moyenne des produits des gradients individuels évalués à un même point.

BFGS: évite de construire explicitement la matrice hessienne et construit à la place une approximation de l'inverse de la dérivée seconde de la fonction à minimiser, en analysant les différents gradients successifs.

SR1: généralisation de la méthode de la sécante, estime la matrice hessienne avec les gradients évalués à deux points.

Résultats partiels - Mixed Logit

Mixed Logit avec approximation du Hessien:

	Hessien	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	k
CG	BHHH	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100
	BFGS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100
	SR1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100
CGT	BHHH	0.01	0.08	0.01	0.03	-0.03	0.00	0.00	100
	BFGS	7.36	5.23	4.46	0.01	-0.14	0.01	-0.02	58
	SR1	7.36	5.23	4.46	0.01	-0.14	0.01	-0.02	47

Algorithme du 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 approchée par le gradient d'une seule composante de la somme, soit un seul individu, sélectionné au hasard, dans notre jeu de données.

La DGS peut être beaucoup plus rapide que la descente de gradient classique parce que le nombre de mises à jour est beaucoup plus grand. C'est particulièrement vrai pour des jeux de données volumineux.

Pseudocode

Input: initial vector of parameters ω and learning rate η
  while an approximate minimum is not obtained **do**
    Randomly shuffle examples in the training set
    for $i = 1, 2, 3, ..., n$ **do**
      $\omega = \omega - \eta \nabla f_i(\omega)$
    end for
  end while

Remarques

- Le pas de gradient, ou taux d'apprentissage, peut être fixe, adaptif ou déterminé par un échéancier.
- Une approximation de second ordre peut être intégrée pour accélérer la méthode.

Ce qu'il reste à faire

- Finaliser le code de l'algorithme du gradient stochastique en Julia.
  - Automatiser le taux d'apprentissage.
  - Introduire une approximation de second ordre.
- Estimer un jeu de données plus grand.
- Finaliser le rapport.

Réferences

Fabian Bastin (2004)

Trust-Region Algorithms for Nonlinear Programming and Mixed Logit Models

Presses Universitaires de Namur

Kenneth Train (2012)

Discrete Choice Methods with Simulation

Cambridge University Press

Questions? Merci!