Présentation du domaine de recherche ; Algorithme CART

Pierre Connault, encadré par Pascal Massart

21 février 2008

Résumé

La "fatigue" est une des principales causes de rupture des matériaux utilisés dans les constructions humaines. On entend par ce terme la modification des propriétés des matériaux consécutives à l'application des cycles d'efforts, cycles dont la répétition peut entraîner la rupture des pièces constituées avec ces matériaux. La rupture des pièces consécutive à la fatigue peut conduire à des dommages importants ou la perte de vies humaines. Ceci est d'autant plus dangereux que la rupture apparaît sans prévenir, et peut se produire sous des contraintes très inférieures aux limites de résistance du matériau dont elle est constituée.

La fatigue prend toute son importance pour l'ingénieur responsable de l'exploitation. L'étude de la fatigue est difficile pour plusieurs raisons :

- c'est un phénomène aléatoire complexe,
- ce phénomène est influencé par de très nombreux facteurs du processus de production : fonte, affinage, refonte, formage puis usinage,
- ces facteurs peuvent être emprunts d'une forte variabilité.

A partir d'une base de données regroupant les résultats d'essais, on cherche à déterminer les facteurs influents en fatigue et à prévoir la fatigue du matériau. Nous présentons ici l'algorithme CART (Classification and Regression Tree), qui construit un arbre de régression et apport des éléments de résolution de notre problème.

1 Position du problème

1.1 Notations

Conservons l'exemple de la fatigue d'un matériau : on a n essais durant lesquels on a à chaque fois mesuré les valeurs des variables $X_1, X_2, \ldots, X_m$ et la durée de vie Y du matériau testé pendant l'essai.

On dispose d'une base de données regroupant ces nombreuses variables. On veut expliquer Y en fonction des autres $X_1, X_2, \ldots, X_m$. Y sera appelée la variable de sortie, les X_i seront appelées les variables d'entrée. Par exemple X_1 peut représenter une certaine quantité de produit ajouté pendant la fabrication d'un matériau, X_2 la température de fabrication, et on dispose de nombreux autres facteurs $X_3, X_4, X_5 \ldots$; Y, la durée de vie du matériau durant l'essai, est partiellement fonction de ces quantités.

En d'autres termes, soit Y et $X_1, X_2, \ldots, X_m$ des variables aléatoires. On notera souvent $X = (X_1, X_2, \ldots, X_m)$. La *base de données*, appelée aussi *base*, est la donnée de n $(m+1)$-uplets $(X_{j,1}, X_{j,2}, \ldots, X_{j,m}, Y_j)$ indépendants et de même loi que (X, Y). Les variables $\{X_{j,i}; Y_i | 1 \leq i \leq n,\ 1 \leq j \leq m\}$ sont appelées les *observations*. Elles se présentent en pratique comme des nombres réels, on n'a pas accès à leur loi, mais la notion de variable aléatoire est commode pour les appréhender théoriquement.

Il existe une fonction $\overline{f}$ mesurable telle que

$$E[Y|X_1, X_2, \ldots, X_n] = \overline{f}(X_1, X_2, \ldots, X_m).$$

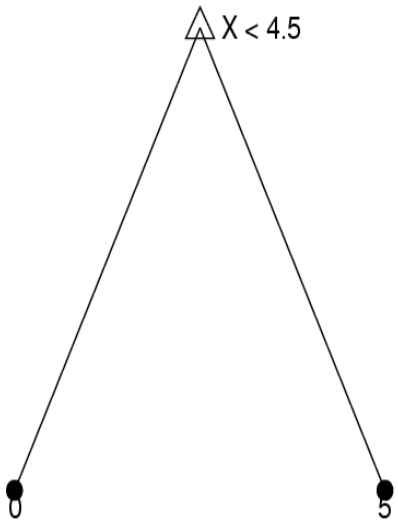
En ce sens la donnée des variables d'entrée permet théoriquement de prédire la sortie Y. Si toutes les variables aléatoires considérées appartiennent à l'espace $\mathcal{L}^2(\Omega)$, on sait aussi que l'espérance conditionnelle $E[Y|X_1, X_2, \ldots, X_m]$ est la projection orthogonale de Y sur le sous-espace $\mathcal{L}^2(\mathcal{F}(X_1, X_2, \ldots, X_m))$, où $\mathcal{F}(X_1, X_2, \ldots, X_m)$ désigne la tribu engendrée par les variables $X_1, X_2, \ldots, X_m$. On a donc :

$$\overline{f} = \texttt{argmin}_{f \in \mathcal{M}(\mathbb{R}^m, \mathbb{R})} E[(Y - \overline{f}(X))^2].$$

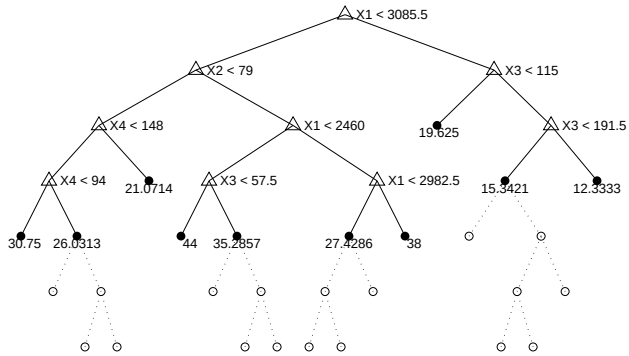
C'est $\overline{f}$ que l'on cherche ; cependant, en statistiques, la distribution de (X, Y) est inaccessible : on ne connaît que les observations. Ne pouvant calculer $\overline{f}$ à partir d'un nombre fini d'observations, on se propose alors d'*estimer* $\overline{f}$, c'est à dire de construire à partir des observations une fonction $\widehat{f}$ assez proche de $\overline{f}$ en un certain sens. Ici, ce n'est d'ailleurs pas tant l'estimation de $\overline{f}$ que l'on recherche que la construction explicite, à partir des données, d'une fonction $\widehat{f}$ ayant de bonnes capacités prédictives, au sens où $E[(Y - \widehat{f}(X))^2]$ n'est pas trop élevé. Ne pouvant en pratique considérer toutes les fonctions mesurables, on est amené à rechercher $\widehat{f}$ sous une certaine forme que l'on se fixe. L'algorithme CART recherche $\widehat{f}$ sous forme d'un arbre.

1.2 Arbres

On rappelle qu'un arbre binaire est un ensemble de *noeuds* ; de chaque noeud partent 0, 1 ou 2 *branches*, vers le bas ; on appelle *feuille* un noeud dont ne part aucune branche. Considérons par exemple l'arbre ci-après, qui a un seul noeud, et deux branches menant à deux feuilles :



On lit un tel arbre de la façon suivante : si $X < 4.5$ alors on emprunte la branche de gauche, qui mène à une feuille ; à cette feuille est attribuée la valeur 0, on obtient donc $Y = 0$. Si au contraire $X \geq 4.5$ alors on emprunte la branche de droite ; on lit : $Y = 5$. L'arbre établit donc un lien fonctionnel entre Y et X, que l'on pourrait d'ailleurs expliciter au moyen de fonctions indicatrices. On peut proposer des arbres plus complexes, le principe de lecture est le même :



Supposons par exemple qu'avec cet arbre on veuille calculer Y à partir de $X_1 = 4000$, $X_2 = 400$, $X_3 = 50$ et $X_4 = 18$. La réponse à la première question "$X_1 < 3085.5\,?$" est `non`, on prend la branche de droite. Celle-ci nous mène à la question suivante, "$X_3 < 115\,?$", qui admet pour réponse `oui`. On emprunte alors la branche de gauche, qui mène à une feuille à laquelle est affectée la valeur 19.625. La valeur calculée par la fonction est donc $Y = 19.625$.

A chaque fois, la valeur de Y est lue au niveau de la feuille : elle est obtenue en fonction des variables d'entrées, en partant de la racine et en suivant le chemin découvert au fur et à mesure des questions posées au niveau des noeuds jusqu'à la feuille correspondante. L'intérêt d'une telle représentation réside dans la facilité de lecture qu'elle procure, et en particulier dans la *hiérarchisation* de l'effet des variables aléatoires d'entrée et de la sélection de leur influence sur la variable aléatoire Y : la première question devrait être la plus importante, la deuxième l'être un peu moins, etc.

Plus synthétiquement, un arbre n'est ici qu'une fonction de $(X_1, X_2, \ldots, X_m)$ constante par morceaux sur des rectangles du type

$$\{a_1 \leq X_1 \leq b_1; a_2 \leq X_2 \leq b_2; \ldots; a_m \leq X_m \leq b_m\}.$$

On notera $\mathcal{T}$ l'ensemble des arbres binaires.

Revenons à CART. C'est un algorithme visant à rechercher un arbre $\widehat{T} \in \mathcal{T}$, le meilleur qui soit en terme de prédiction. A l'issue de l'algorithme, on obtient donc un arbre prédisant la variable de sortie Y inscrite au niveau des feuilles en fonction des variables d'entrées X_i, ce qui réalise notre objectif.

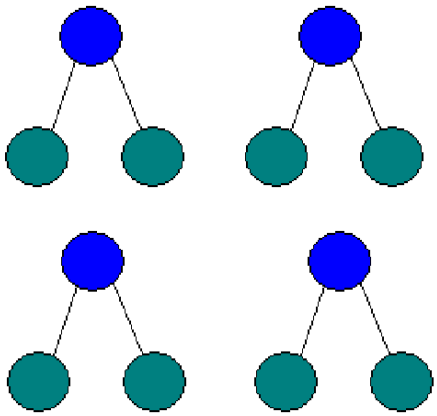
2 Algorithme

2.1 Construction d'un arbre maximal

CART commence par construire un gros arbre $T_{\texttt{Max}}$. Le point crucial pour cette construction est de déterminer quelle question placer au niveau de chaque noeud. De plus CART ne connaît à l'avance ni le nombre de noeuds ni leur emplacement.

CART commence par le premier noeud, ou racine, celui qui sera le plus haut dans l'arbre. Pour déterminer quelle question placer sur ce noeud, il pose *toutes les questions binaires possibles* à partir des variables d'entrée : par exemple, "$X_1 < 37$?" ; "$X_2 > 41$?". Le nombre de questions est élevé mais fini, car le nombre de données est fini.

Chaque question produit donc deux sous-ensembles, qui partitionnent la base de données : celui pour lequel la réponse à la question est `oui`, et celui pour lequel la réponse est `non`. Ils correspondent à deux noeuds. Pour chaque question, CART calcule la partition associée : il obtient ainsi différentes partitions de la base de données ; il lui faut choisir l'une d'elles.



Cart pose toutes les questions possibles.

De chaque partition associée à une question, il évalue pour cela l'hétérogénéité par rapport à la variable de sortie Y dans les deux noeuds inférieurs obtenus. Si t est un noeud, correspondant donc à une partie de l'ensemble $\{1, \ldots, n\}$, on note :

$$R(t) = \sum_{e \in t} (Y_e - \mu_t)^2$$

e désignant un élément de t (c'est-à-dire un numéro de ligne de la base de données appartenant à t), Y_e la valeur de Y lue dans la base de données, et

$$\mu_t = \frac{\sum_{e \in t} Y_e}{|t|}$$

la moyenne des valeurs de $(Y_e)_{e \in t}$ dans le noeud t.

Autrement dit, $R(t)$ est la variance empirique des éléments affectés au noeud t, multipliée par le cardinal de t.

Pour quantifier l'hétérogénéité d'une subdivision, CART calcule le risque de chaque noeud et effectue la somme :

$$\texttt{Hétérogénéité de la Subdivision} = R(t_g) + R(t_d).$$

CART identifie ensuite la partition d'hétérogénéité minimale dans les deux noeuds : c'est cette partition qui classe le mieux les données, puisque celles-ci sont regroupées de la façon la plus homogène possible dans chaque noeud. C'est donc elle que CART retient en définitive, ainsi que la question qui lui était associée.

On peut remarquer qu'en subdivisant, on diminue effectivement l'hétérogénéité : si t est la racine et t_g et t_d les noeuds associés à une question, on a $R(t) \geq R(t_g) + R(t_d)$.

Une fois ceci effectué pour le premier noeud, on peut recommencer avec les deux nouveaux noeuds obtenus : chacun correspond en effet à un nouvel ensemble de données. CART obtient ainsi par récurrence un arbre du même type que ceux décrits dans la section précédente. Notons que le processus termine effectivement (l'arbre est fini), car le nombre de questions possibles est fini. Reste à affecter une valeur à chaque feuille ; à chaque feuille t on affecte la moyenne empirique μ_t de ses données.

A ce stade, CART a donc obtenu un arbre maximal, noté $T_{\texttt{Max}}$. Qui plus est, $T_{\texttt{Max}}$ est un candidat intéressant comme un "bon" estimateur : en effet, on a à chaque étape essayé de trier les données au mieux ; on est donc autorisé à penser que $T_{\texttt{Max}}$ prédit bien Y. C'est en fait plus compliqué.

Avant de poursuivre, une remarque sur la notation employée : tel qu'on construit l'arbre $T_{\texttt{Max}}$, on constate qu'à un noeud t correspond une partie de l'ensemble $\{1, \ldots, n\}$: t est naturellement associé aux indices des éléments qu'il contient. Par exemple, la racine est associée à l'ensemble $\{1, \ldots, n\}$ tout entier ; la question posée au niveau de la racine est de la forme "$X_j < a$?" et donne deux noeuds, le noeud de gauche associé à la partie $\{i | X_{i,j} < a\}$ et celui de droite associé à $\{i | X_{i,j} \geq a\}$.

On identifiera systématiquement le noeud t et la partie de l'ensemble $\{1, \ldots, n\}$ associée.

2.2 Le compromis biais-variance

La fonction de cette courte section est de permettre, sur un exemple, de comprendre le dilemme biais-variance. Simulons 400 fois le tirage d'une gaussienne $Z = \mathcal{N}(0, 1)$. Notons $a = (a_1; a_2; a_3; \ldots; a_{400})$ le résultat. On suppose connu le fait que Z est absolument continue par rapport à la mesure de Lebesgue, mais on feint d'ignorer sa loi[(1)]. Uniquement à partir des données, on cherche à estimer la densité de Z par un histogramme. Dans la définition de notre histogramme, le seul paramètre que nous pouvons choisir est le nombre de classes k. Nous allons discuter la valeur de k choisie.

- Imaginons pour commencer que $k = 1$. L'histogramme est alors une fonction indicatrice d'intervalle. La densité obtenue a plus l'air de celle d'une variable aléatoire de loi uniforme que d'une gaussienne, elle est en fait trop simple par rapport à la densité de Z. Il faudrait choisir k plus grand, de façon à mieux exploiter l'information apportée par la répartition des nombres $(a_1; a_2; a_3; \ldots; a_{400})$. L'histogramme obtenu est donc un mauvais prédicteur.
- A l'inverse, prenons une valeur de k très élevée par rapport au nombre de données, par exemple $k = 800$. L'histogramme obtenu présente de nombreuses petites bosses ; on n'a pas non plus de gaussienne clairement visible, et la densité obtenue est plus complexe qu'une gaussienne. Ici, l'histogramme exploite toute l'information apportée par les observations, et plus encore, puisqu'il s'attache aux fluctuations des données : les bosses observées loin de la moyenne 0 sont tout à fait contingentes et dues au caractère aléatoire de Z. L'histogramme obtenu *reflète* donc le vecteur a, mais ne parvient

(1) On ignore la loi de Z dans un "vrai" problème.

pas à se détacher de celui-ci pour estimer la vraie loi. Il est donc un mauvais prédicteur.

Il faut donc trouver une valeur de k raisonnable, qui évite les deux écueils précédemment constatés. Il faut :

- D'une part, prendre k assez grand pour avoir un modèle assez complexe ; on appelle biais le manque de précision du modèle.
- D'autre part, prendre k assez petit pour se détacher des fluctuations ; on appelle variance l'erreur due aux fluctuations des données.

La conciliation de ces deux exigences contraires est appelé le compromis biais/variance.

Par analogie avec la situation précédente, on comprend donc que dans le cas des arbres également, il existe un compromis à établir entre précision du modèle et fluctuation des données.

- D'une part, un arbre trop peu ramifié montre un biais très important. Dans le cas extrême d'un arbre à une feuille, la valeur de Y est toujours la moyenne des données.
- D'autre part, un arbre trop profond se perd dans les détails de la base, et présente une variance importante. C'est en particulier, on peut le penser, le cas de $T_{\texttt{Max}}$.

Un algorithme d'élagage a pour fonction de résoudre ce problème.

2.3 Elagage

Désormais, si T et T' sont deux arbres, on notera $T \preceq T'$ si T est un sous-arbre de T' : cela signifie que tous les noeuds et branches de T sont des noeuds et branches de T' et que T et T' ont la même racine. Par ailleurs, si $T \preceq T_{\texttt{Max}}$, on associe à une feuille t de T la valeur moyenne μ_t des données qu'elle contient.

2.3.1 Evaluation de l'erreur $R(T)$ d'un arbre

Soit $T \in \mathcal{T}$ un arbre binaire.

On souhaite disposer d'un critère permettant de déterminer si T est un bon prédicteur, c'est-à-dire si la quantité

$$E[(Y - T(X))^2]$$

est petite.

Idéalement, on rechercherait même à minimiser cette quantité sur $\mathcal{T}$ tout entier, mais on va en fait se contenter d'effectuer la minimisation parmi quelques arbres bien choisis.

N'ayant que les données à disposition, en nombre fini, et ne connaissant pas la loi du couple (X, Y), on en est réduit évaluer une estimation de l'erreur théorique $E[(Y - T(X))^2]$; cette erreur empirique, appelée *taux d'erreur de l'arbre* T, est définie ainsi : $R(T) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - T(X_i))^2$.[(2)] D'après la loi des grand nombres, l'erreur empirique tend vers l'erreur théorique lorsque le nombre d'observations n tend vers l'infini, le taux d'erreur est donc consistant.

En regroupant les termes par feuilles, on voit que le taux d'erreur $R(T)$ vaut également $R(T) = \sum_{t \in \widetilde{T}} R(t)$, où $\widetilde{T}$ désigne l'ensemble des feuilles de l'arbre T.

[(2)] En régression, les quantités $(Y_i - T(X_i))^2$ sont appelées résidus : elles mesurent, pour chaque élément de la base, l'écart entre la valeur $T(X_i)$ prédite par le modèle et la valeur "réelle" Y_i mesurée et inscrite dans la base.

Pour obtenir le meilleur arbre possible, on est donc tenté de prendre :

$$\widehat{T} = \texttt{argmin}_{T \preceq T_{\texttt{Max}}} R(T).$$

Ce choix permet de minimiser le biais (la précision est optimale), mais conduit à une variance importante : l'arbre obtenu est en fait $T_{\texttt{Max}}$, que nous avons construit maximal, et qui se ramifie le plus possible. Nous allons considérer des arbres élagués.

Une solution, pour réaliser méthodiquement l'élagage de l'arbre $T_{\texttt{Max}}$, est de modifier $R(T)$. Si λ est un réel strictement positif, on pose

$$R_\lambda(T) = \sum_{t \in \widetilde{T}} R(t) + \lambda . |\widetilde{T}|.$$

(On a noté $\widetilde{T}$ l'ensemble des feuilles de l'arbre T, et, si A est un ensemble, on note $|A|$ son cardinal.) Le deuxième terme dans la définition de $R_\lambda(T)$ est d'autant plus élevé que le nombre de feuilles de l'arbre T considéré est grand ; il pénalise la complexité de T. Le facteur λ est une constante à régler, qui dose le poids accordé à cette pénalité par rapport au premier terme. Pour chaque $\lambda > 0$ on calcule alors le minimum $\widehat{T_\lambda}$ pour le taux d'erreur pénalisé :

$$\widehat{T_\lambda} = \texttt{argmin}_{T \preceq T_{\texttt{Max}}} R_\lambda(T)$$

Son existence, est garantie par le fait qu'il existe un nombre fini de sous-arbres, et concernant l'unicité, le théorème suivant assure que lorsque plusieurs arbres minimisants existent, l'un d'eux est $\preceq$-minimal.

Théorème 1 *(cf. [1])*

Soit λ un réel strictement positif ; alors il existe un arbre $\widehat{T_\lambda}$ tel que :

- $\widehat{T_\lambda} = \texttt{argmin}_{T \preceq T_{\texttt{Max}}} R_\lambda(T)$.
- *Si $T \preceq T_{\texttt{Max}}$ minimise $R_\lambda(T)$, alors $\widehat{T_\lambda} \preceq T$.*

On obtient $(\widehat{T_\lambda})_{\lambda \geq 0}$, qui est une famille de sous-arbres de $T_{\texttt{Max}}$. Cependant, ce théorème ne nous indique pas comment calculer effectivement les différents $\widehat{T_\lambda}$, ni comment choisir la juste valeur de λ.

2.3.2 Choix du sous arbre

L'idée pour la suite va être

1. de calculer effectivement pour chaque λ l'arbre $\widehat{T_\lambda}$ minimum,
2. de déterminer, parmi ce premier choix arbres, lequel est le meilleur prédicteur.

Remarquons que l'on ne choisit la valeur de λ qu'une fois connus les différents minima pour chaque valeur de λ. Dans une certaine mesure, ce sont donc les données elles-même qui nous indiquent quel modèle retenir.

Calcul de $\widehat{T_\lambda}$ Soit T un arbre et t un noeud de T ; on note T_t l'arbre de racine t, inclus dans T, $\preceq$-maximal pour cette propriété et on appelle T_t la *branche* de T issue du noeud t.

On expose dans un premier temps un algorithme de calcul produisant une suite d'arbres emboîtés de complexité décroissante, puis on détermine leur lien avec les arbres $\widehat{T_\lambda}$.

L'idée importante de l'algorithme est la suivante :

On peut calculer le défaut de prédiction $R(T_t) = \sum_{u \in \widetilde{T_t}} R(u)$**, et on a toujours :**

$$R(T_t) \leq R(t).$$

Si l'égalité a lieu, on gagne à couper la branche entière, puisqu'il en résulte un gain de complexité sans perte de précision.

On définit $R_\lambda(T_t) = \sum_{u \in \widetilde{T_t}} R_\lambda(u)$, et on a pour tout λ et tout noeud t :

$$R_\lambda(T_t) \leq R_\lambda(t).$$

On augmente alors continûment le poids des feuilles λ. Pour une certaine valeur, la branche la moins "robuste" finit par céder : ceci correspond à l'égalité $R_{\lambda_c}(T_t) = R_{\lambda_c}(t)$. On retranche alors de T la branche vérifiant l'égalité, obtenant de cette façon un nouvel arbre, et on poursuit l'algorithme tant que T n'est pas réduit à sa racine. On construit ainsi une suite d'arbres $\preceq$-décroissante.

Faire croître λ continûment n'étant pas viable algorithmiquement, on utilise plutôt la valeur exacte de λ_c ; comme λ_c est la plus petite valeur de λ pour laquelle il existe un noeud de l'arbre T tel que $R_\lambda(T_t) = R_\lambda(t)$, c'est-à-dire pour laquelle

$$R(T_t) + \lambda.|\widehat{T_t}| = R(t) + \lambda,$$

on a :

$$\lambda_c = \mathtt{min}_{t \in T} \frac{R(t) - R(T_t)}{|\widetilde{T_t}| - 1}.$$

Mettons en forme l'algorithme :

Etape 1 *La première étape de l'algorithme consiste à émonder* $T_{\mathtt{Max}}$ *en ôtant toutes les branches* T_t *telles que* $R(T_t) = R(t)$*. On pose donc :*

$$T(0) = T_{\mathtt{Max}} - T_{t_1} - T_{t_2} - \ldots - T_{t_k}$$

où les T_{t_i} *sont les branches telle que* $R(T_{t_i}) = R(t_i)$*.*

On pose $k = 0$ *et* $\lambda_0 = 0$*. On note désormais* $T(k)$ *l'arbre courant. A ce stade,* $T(k) = T(0)$*.*

Etape 2 *On pose :*

$$\lambda_c(k+1) = \mathtt{min}_{t \in T(k)} \frac{R(t) - R(T_t)}{|\widetilde{T_t}| - 1}.$$

et

$$T(k+1) = T(k) - T_{t_1} - \ldots - T_{t_k}$$

où les T_{t_i} *sont les branches de l'arbre* $T(k)$ *vérifiant* $R_{\lambda_c}(T_{t_i}) = R_{\lambda_c}(t_i)$*.*

Etape 3 *On incrémente* k *de* 1*. Si* $T(k)$ *n'est pas réduit à une feuille, on va à l'étape 2. Sinon, l'algorithme termine.*

On obtient donc une suite d'arbres $T(k)$; le théorème suivant relie les arbres $T(k)$ aux arbres $\widehat{T_\lambda}$.

Théorème 2 *(cf [1])*

Pour tout nombre réel strictement positif λ*, on a*

$$\widehat{T_\lambda} = T(\lambda_c(k))$$

avec $\lambda_c(k) \leq \lambda < \lambda_c(k+1)$*.*

Théorème 3 *(cf [1])*
Pour tout réel strictement positif λ, on a

$$\widehat{T_\lambda} = T(\lambda_c(k))$$

avec $\lambda_c(k) \leq \lambda < \lambda_c(k+1)$.

Ainsi les arbres $\widehat{T_\lambda}$ forment une famille $\preceq$-décroissante et les points de décroissance sont les valeurs critiques de λ calculées par l'algorithme.

Choix explicite de la constante λ A ce stade, on connaît une famille d'arbres binaires de régression $\widehat{T_\lambda}$ bons candidats pour la fonction $\overline{f}$ du début de l'article.

Pour évaluer $\widehat{\lambda} = \texttt{argmin}_\lambda E[(Y - T_\lambda(X))^2]$, on est tenté de prendre le critère empirique

$$\widehat{\lambda} = \texttt{argmin}_\lambda R(\widehat{T_\lambda})$$

mais on trouve alors $\widehat{\lambda} = 0$. Le problème est que les données ne sont pas indépendantes des arbres construits, on ne peut donc pas les utiliser pour sélectionner le meilleur arbre.

Il faudrait plutôt pouvoir simuler un nouvel n-échantillon de même loi que (X, Y), mais justement on ignore cette loi. Au lieu de cela, on peut procéder ainsi : d'une part, on effectue *toute la construction décrite précédemment* non pas avec toute la base, mais avec une grosse partie $P_{\texttt{construction}}$ des observations (pour fixer les idées, on peut mettre dans $P_{\texttt{construction}}$ les deux-tiers des éléments de la base). Le tiers restant, $P_{\texttt{validation}}$, est indépendant des arbres construits[(3)]. On peut donc effectuer la sélection du modèle à partir de ce tiers restant :

$$\widehat{\lambda} = \texttt{argmin}_\lambda R_{\texttt{validation}}(\widehat{T_\lambda}),$$

avec $R_{\texttt{validation}}(T) = \sum_{e \in P_{\texttt{validation}}} (Y_e - T(e))^2$.

Cette méthode, donne de bons résultats en pratique si le nombre de données est important ; d'autres méthode de détermination de $\widehat{\lambda}$ sont connues, mais elles obéissent toutes à un critère empirique de "simulation" d'un échantillon indépendant.

Pour finir, on a bien obtenu

$$\hat{f} = \widehat{T_{\widehat{\lambda}}}$$

un estimateur de $\overline{f}$.

En conclusion, soulignons que l'utilisation d'une pénalité portant sur la complexité du modèle pour obtenir des estimateurs robustes, est fréquente en statistiques. CART est à ce point de vue une illustration particulièrement conviviale de part sa lisibilité.

[(3)] (puisque les observations sont iid, et les arbres construits sont en fait $\mathcal{F}((X_i, Y_i)_{i \in P_{\texttt{construction}}})$-mesurables.)

Références

[1] Classification and Regression trees, Breiman, Friedman, Olshen, Stone.

[2] R. Tibshirani. Regression analysis and selection via the Lasso. Journal Royal Statist. Soc. B, 58, pages 267 à 288, 1996.