Classification Supervisée (par Apprentissage)

Introduction

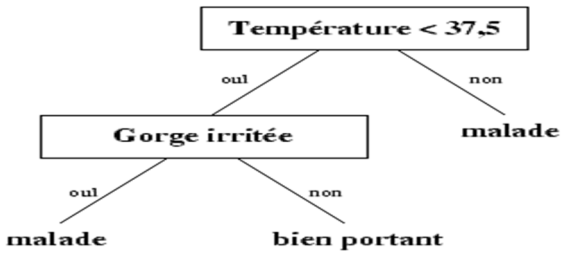
Une première approche possible pour classifier des individus est l'approche basée sur les systèmes experts. En effet, la connaissance d'un, ou de plusieurs, expert(s) peut être décrite sous la forme d'un ensemble de règles. Cet ensemble forme un système expert qui est utilisé pour classifier de nouveaux cas. Cette approche, largement utilisée en 80', dépend fortement de la capacité à extraire et à formaliser les connaissances de l'expert. Nous considérons ici une autre approche pour laquelle la procédure de classification sera extraite automatiquement à partir d'un ensemble d'exemples, sachant qu'un exemple consiste en la description d'un cas avec la classification correspondante.

En effet, un système d'apprentissage extrait, à partir de cet ensemble d'exemples (base d'apprentissage), une procédure de classification. Il s'agit ainsi d'induire une procédure de classification générale à partir d'un ensemble d'exemples. Le problème est ainsi inductif, puisque il s'agit d'extraire une règle générale à partir d'exemples observées. La procédure générée devrait classifier correctement les exemples de l'échantillon mais surtout avoir un bon pouvoir prédictif pour classifier correctement de nouvelles données. En particulier, les méthodes issues de l'intelligence artificielle sont des méthodes non paramétriques. On distingue les méthodes symboliques (la procédure de classification produite peut être écrite sous forme de règles) des méthodes non symboliques ou adaptatives (la procédure de classification produite est de type « boîte noire »). Parmi les méthodes symboliques, les plus utilisées sont basées sur les arbres de décision. Pour les méthodes adaptatives, on distingue deux grandes classes : les réseaux de neurones et les algorithmes génétiques.

Apprentissage automatique : les arbres de décision

Il est souvent essentiel de produire des procédures de classification compréhensibles par l'utilisateur (*e.g.* l'aide au diagnostic médical où le médecin doit pouvoir interpréter les raisons du diagnostic). Les arbres de décision répondent à cette contrainte car ils représentent graphiquement un ensemble de règles et sont facilement interprétables.

Exemple : *La population est un ensemble de patients. Il y a 2 classes:* `malade` *et* `bien portant`*. Les descriptions sont faites avec les 2 attributs:* `Température` *qui est un attribut à valeurs décimales et* `gorge irritée` *qui est un attribut logique.*



Les nœuds internes sont appelés *nœuds de décision*. Un tel nœud est étiqueté par un *test* qui peut être appliqué à toute description d'un individu de la population. En général, chaque test examine la valeur d'un seul attribut de l'espace des descriptions. Les réponses possibles au test correspondent aux labels des arcs issus de ce nœud. Dans le cas de nœuds de décision binaires, les labels des arcs sont facultatifs et, par convention, l'arc gauche correspond à une réponse positive au test. Les *feuilles* sont étiquetées par une classe (dite *classe par défaut*).

Un arbre de décision est la représentation graphique d'une procédure de classification. En effet, à toute description complète est associée une seule feuille de l'arbre. Cette association est définie en commençant à la racine de l'arbre et en descendant dans l'arbre selon les réponses aux tests qui étiquettent les nœuds internes. La classe associée est alors la classe par défaut associée à la feuille qui correspond à la description. La procédure de classification obtenue a une traduction immédiate en termes de règles de décision. Les systèmes de règles obtenus sont particuliers car l'ordre dans lequel on examine les attributs est fixé et les règles de décision sont mutuellement exclusives.

Pour l'arbre de décision de l'exemple, un patient ayant une température de 39 et ayant la gorge non irritée sera classé comme malade par cet arbre. La traduction de ceci en règles de décision est:

- *SI* `Température<37,5` *ET* `gorge irritée` *ALORS* `malade`
- *SI* `Température<37,5` *ET NON(*`gorge irritée`*) ALORS* `bien portant`
- *SI NON(*`Température<37,5`*) ALORS* `malade`

Nous étudions ici différents algorithmes d'apprentissage par arbres de décision. Ces algorithmes prenant en entrée un échantillon *S* et construisent un arbre de décision. Étant donnés un échantillon *S*, un ensemble de classes {1,...,*c*} et un arbre de décision *t*, à chaque position *p* de *t* correspond un sous-ensemble de l'échantillon qui est l'ensemble des exemples qui satisfont les tests de la racine jusqu'à cette position. Par conséquent, on peut définir, pour toute position *p* de *t*, les quantités suivantes :

N(*p*) est le cardinal de l'ensemble des exemples associé à *p*,
N(*k*/*p*) est le cardinal de l'ensemble des exemples associé à *p* qui sont de classe *k*,
P(*k*/*p*) = *N*(*k*/*p*)/*N*(*p*) la proportion d'éléments de classe *k* à la position *p*.

Pour l'arbre de décision de l'exemple, si en plus, on dispose d'un échantillon de 200 patients. On sait que 100 sont malades et 100 sont bien portants, et la répartition entre les deux classes M (pour malade) et S (pour bien portant) est donnée par :

	gorge irritée	*gorge non irritée*
température < 37,5	*(6 S, 37 M)*	*(91 S, 1 M)*
température ≥ 37,5	*(2 S, 21 M)*	*(1 S, 41 M)*

On a alors : N(11)=43 ; N(S/11)=6 ; N(M/11)=37 ; P(S/11)=6/43 et P(M/11)=37/43.

Exemple introductif et préliminaires

Considérons l'exemple suivant pour introduire les algorithmes d'apprentissage par arbres de décision. Une banque dispose des informations suivantes sur un ensemble de clients:

client	*M*	*A*	*R*	*E*	*I*
1	moyen	moyen	village	Oui	oui
2	élevé	moyen	bourg	Non	non
3	faible	âgé	bourg	Non	non
4	faible	moyen	bourg	Oui	oui
5	moyen	jeune	ville	Oui	oui
6	élevé	âgé	ville	Oui	non
7	moyen	âgé	ville	Oui	non
8	faible	moyen	village	Non	non

L'attribut ternaire *M* décrit la moyenne des montants sur le compte client. Le second attribut ternaire *A* donne la tranche d'âge du client. Le troisième attribut ternaire *R* décrit la localité de résidence du client. Le dernier attribut binaire *E* a la valeur `oui` si le client a un niveau d'études supérieures. La classe associée à chacun de ces clients correspond au contenu de la colonne *I*. La classe `oui` correspond à un client qui effectue une consultation de ses comptes bancaires en utilisant Internet. On souhaite trouver un arbre de décision qui soit capable de dire si un client effectue des consultations de ses comptes par Internet en connaissant les valeurs des attributs *M* (montant), *A* (âge), *R* (résidence) et *E* (études) pour ce client.

Il s'agit de construire, à partir de ce tableau, un arbre de décision qui classifie les clients de façon descendante. Lorsqu'un test est choisi, on divise l'ensemble d'apprentissage pour chacune des branches et on réapplique récursivement l'algorithme. Sur notre exemple, on initialise avec l'arbre vide. L'échantillon contient 8 éléments, 3 sont de classe `oui` et 5 de classe `non`. Ainsi, à la racine de l'arbre qui n'est étiqueté par aucun test, l'échantillon peut être caractérisé par le couple (3,5). On se pose alors la question de savoir si ce nœud est terminal (*i.e.* est-il nécessaire de rechercher un test qui discrimine de façon intéressante l'échantillon). Par exemple, on attribuerait une feuille si nous étions dans le cas (0,8), c'est-à-dire si aucun client n'utilise Internet. Pour notre cas supposons que nous devions choisir un test, nous aurions quatre choix possibles.

$$(3,5) \rightarrow M\ (1,2)\ (2,1)\ (0,2)$$

$$(3,5) \rightarrow A\ (1,0)\ (2,2)\ (0,3)$$

$$(3,5) \rightarrow R\ (1,1)\ (1,2)\ (1,2)$$

$$(3,5) \rightarrow E\ (3,2)\ (0,3)$$

Ceci illustre les choix possibles en racine où les branches du test *M* sont labellisés dans l'ordre par `faible`, `moyen` et `élevé` ; du test *A* dans l'ordre par `jeune`, `moyen` et `âgé` ; du test *R* dans l'ordre par `village`, `bourg` et `ville` ; et du test *E* dans l'ordre par `oui` et `non`.

Laquelle des quatre possibilités faut-il choisir ? Si on regarde le test sur le type de résidence *R*, on remarque que ce test ne permet une discrimination sur aucune des branches, on peut donc se dire que le choix de ce test ne fait rien gagner, il sera donc à rejeter. Par contre, pour le test sur l'âge *A*, on remarque que sur la première branche, tous les éléments correspondants de l'échantillon sont de classe `oui` et que sur la troisième branche, tous les éléments sont de classe `non`. Ce test peut donc être considéré comme « intéressant ». Ce raisonnement informel doit être automatisé. Pour ce faire, il faut introduire des quantités qui permettent de comparer les différents choix possibles. Ceci revient à définir des fonctions qui permettent de mesurer le degré de mélange des exemples entre les différentes classes. Une telle fonction doit vérifier la propriété suivante : prendre son minimum lorsque tous les exemples sont dans une même classe (le nœud est pur) et son maximum lorsque les exemples sont équirépartis. Par exemple, si on dispose de 8 éléments et de deux classes, une telle fonction devra prendre son minimum pour les couples (0,8) et (8,0) et son maximum pour le couple (4,4). Il existe différentes fonctions qui satisfont ces propriétés, parmi lesquelles figurent la fonction de ***Gini*** et la fonction ***Entropie***. Soit *S* un échantillon, soit *p* une position, en reprenant les notations définies précédemment, ces fonctions sont définies par :

$$Entropie(p) = -\Sigma_{k=1}^{c} P(k/p) \times \log(P(k/p))$$

$$\begin{aligned} Gini(p) &= 1 - \Sigma_{k=1}^{c} P(k/p)^2 \\ &= 2\ \Sigma_{k<k'} P(k/p)P(k'/p) \end{aligned}$$

Considérons le cas de deux classes et soit x la proportion d'éléments de classe 1 en position p. On a donc $Entropie(p) = -x \log x - (1-x) \log (1-x)$. Cette fonction de x prend ses valeurs dans l'intervalle [0,1], a son minimum pour $x=0$ et $x=1$ qui vaut 0 et a son maximum pour $x=1/2$ qui vaut 1. La fonction de Gini est définie par $Gini(p) = 2x(1-x)$. Cette fonction de x prend ses valeurs dans l'intervalle [0,1/2], a son minimum pour $x=0$ et $x=1$ qui vaut 0 et a son maximum pour $x=1/2$ qui vaut 1/2. Ces deux fonctions sont symétriques par rapport à $x=1/2$. Pour notre exemple courant, considérons, par exemple, l'arbre construit à l'aide de l'attribut E, nous avons :

- $Entropie(\varepsilon) = -3/8 \log 3/8 - 5/8 \log 5/8 \simeq 0{,}954$
- $Entropie(1) = -3/5 \log 3/5 - 2/5 \log 2/5 \simeq 0.970$
- $Entropie(2) = -0/3 \log 0/3 - 3/3 \log 3/3 = 0$
- $Gini(\varepsilon) = 2 \times 3/8 \times 5/8 \simeq 0{,}469$
- $Gini(1) = 2 \times 3/5 \times 2/5 = 0.480$
- $Gini(2) = 2 \times 0/3 \times 3/3 = 0$

On dispose ainsi de fonctions permettant de mesurer le degré de mélange des classes pour tout échantillon et donc pour toute position de l'arbre en construction. Appelons i la fonction choisie. Il reste à définir une fonction permettant de choisir le test qui doit étiqueter le nœud courant. Rappelons que, sur notre exemple, à la racine de l'arbre, il nous faut choisir entre les quatre tests correspondants aux quatre attributs disponibles. Dans ce but, on introduit une fonction gain par :

$$Gain(p,t) = i(p) - \sum_{j=1}^{N} P_j \times i(pj)$$

où p désigne une position, $test$ un test d'arité N et P_j est la proportion d'éléments de S à la position p qui vont en position pj (qui satisfont la $j^{ème}$ branche du test $test$). Si on considère comme fonction i la fonction entropie, le terme $i(p)$ représente l'entropie actuelle du nœud p, le deuxième terme de la différence représente l'entropie espérée en introduisant le test $test$ qui est égale à la somme pondérée des entropies des nouveaux nœuds créés. On souhaite obtenir des entropies les plus faibles possibles car, d'après les propriétés de la fonction entropie, si l'entropie est faible, la plupart des éléments se trouvent dans une même classe. On cherche donc à obtenir le gain maximum. Sur notre exemple, nous obtenons :

- $Gain(\varepsilon,M) = Entropie(\varepsilon) - (3/8\ Entropie(1) + 3/8\ Entropie(2) + 2/8\ Entropie(3)) = Entropie(\varepsilon) - 0{,}620$
- $Gain(\varepsilon,A) = Entropie(\varepsilon) - (1/8\ Entropie(1) + 4/8\ Entropie(2) + 3/8\ Entropie(3)) = Entropie(\varepsilon) - 0{,}500$
- $Gain(\varepsilon,R) = Entropie(\varepsilon) - (2/8\ Entropie(1) + 3/8\ Entropie(2) + 3/8\ Entropie(3)) = Entropie(\varepsilon) - 0{,}870$
- $Gain(\varepsilon,E) = Entropie(\varepsilon) - (5/8\ Entropie(1) + 3/8\ Entropie(2)) = Entropie(\varepsilon) - 0{,}607$

Le gain maximal ou encore l'entropie espérée minimale est obtenue pour le choix du test A. On remarque que le choix du test R est très mauvais, ce qui confirme l'intuition. Après l'introduction de la problématique et de quelques éléments fondamentaux utilisés par les algorithmes d'apprentissage par arbre de décision, nous allons présenter le schéma général des algorithmes, puis présenter deux algorithmes particuliers CART et ID3.

Généralités sur l'apprentissage des arbres de décision

Idée : Diviser récursivement les exemples d'apprentissage par des tests définis à l'aide des attributs jusqu'à ce que les sous-ensembles d'exemples ne contiennent (presque) que des exemples appartenant à une même classe. Dans toutes les méthodes, on trouve les 3 opérateurs suivants :

1. **Décider si un nœud est terminal**, c'est-à-dire décider si un nœud doit être étiqueté comme une feuille. Par exemple : tous les exemples sont dans la même classe, il y a moins d'un certain nombre d'erreurs, ...
2. **Sélectionner un test à associer à un nœud**. Par exemple : aléatoirement, utiliser des critères statistiques, ...

3. **Affecter une classe à une feuille**. On attribue la classe majoritaire sauf dans le cas où l'on utilise des fonctions coût ou risque.

Les méthodes vont différer par les choix effectués pour ces différents opérateurs, c'est-à-dire sur le choix d'un test (par exemple, utilisation du gain et de la fonction entropie) et le critère d'arrêt (quand arrêter la croissance de l'arbre, soit quand décider si un nœud est terminal). Le schéma général est comme suit :

Algorithme d'apprentissage générique
entrée : *langage de description ; échantillon S*
Début
Initialiser à l'arbre vide ; la racine est le noeud courant
Répéter
Décider si le noeud courant est terminal
Si *le noeud est terminal* ***alors***
Affecter une classe
Sinon
Sélectionner un test et créer le sous-arbre
FinSi
Passer au noeud suivant non exploré s'il en existe
Jusqu'à *obtenir un arbre de décision*
Fin

Avec un tel algorithme, on peut calculer un arbre de décision dont l'erreur apparente est faible, voire nulle. Un arbre de décision *parfait* est un arbre de décision tel que tous les exemples de l'ensemble d'apprentissage soient correctement classifiés. Un tel arbre n'existe pas toujours (s'il existe deux exemples tels que à deux descriptions identiques correspondent deux classes différentes). L'objectif est de construire un arbre d'erreur de classification la plus petite possible. En effet, l'erreur apparente est une vision très optimiste de l'erreur réelle, et trouver un arbre de décision d'erreur apparente minimale est un problème NP-complet. Ainsi, l'algorithme présenté précédemment recherche un « bon » arbre d'erreur apparente faible. L'algorithme procède de façon descendante sans jamais remettre en question les choix effectués et on ne peut jamais exclure qu'un autre choix de test conduise en fait à un meilleur arbre. L »arbre construit est d'erreur apparente faible car les feuilles sont étiquetées de telle manière qu'il y ait peu d'erreurs. Mais, il se peut que l'erreur réelle soit importante (*i.e.* l'arbre construit soit bien adapté à l'échantillon mais ait un pouvoir de prédiction faible). Il est possible que l'erreur apparente diminue constamment lors de la construction de l'arbre alors que l'erreur réelle diminue, se stabilise, puis augmente.
L'idéal serait de trouver un critère qui permette d'arrêter la croissance de l'arbre au bon moment. Malheureusement, dans l'état actuel des recherches, un tel critère n'est pas encore disponible. De plus, le risque d'arrêter trop tôt la croissance de l'arbre est plus important que de l'arrêter trop tard. Par conséquent, les méthodes utilisées procèdent souvent en deux phases. La première phase correspond à l'algorithme présenté dans ce paragraphe ; dans une seconde phase, on *élague* l'arbre obtenu pour essayer de faire diminuer l'erreur réelle (élaguer un arbre consiste à en supprimer certains sous-arbres). Les méthodes se distinguent donc les unes des autres par les choix des opérateurs, mais aussi par les méthodes d'élagage utilisées. Les méthodes les plus utilisées, et qui diffèrent par leurs choix (opérateurs + élagage), sont CART et C4.5