Construction et élagage d'un arbre de décision

Nicolas Roux, Mariam Bouzid

Université d'Angers

Résumé Ce projet a été réalisé dans le cadre de l'option «Apprentissage artificiel» du Master 1 Informatique de l'Université d'Angers. L'objectif était d'implémenter un arbre de décision dont la construction est basée sur l'entropie de Shannon, de permettre sa validation via un jeu de test et son élagage via la méthode de réduction d'erreur. ...

Keywords: apprentissage artificiel, méthode des arbres de décision

1 Choix d'implémentation

Nous avons choisi le langage Python, spécifiquement la version 3, afin de nous initier à ce langage qui semble très largement utilisé dans divers domaines de la recherche en informatique. Il est portable, permet à la fois de créer rapidement des prototypes, et, grâce à de nombreux modules externes, d'être efficace en terme de temps de calcul.

Bien que l'algorithme demandé soit déjà implémenté sous Weka et que ses sources soient disponibles, nous avons choisi de ne pas utiliser ces sources d'une quelconque façon, estimant que ce n'était pas l'objectif de ce projet. Nous avons donc codé ceci en partant uniquement des bases théoriques enseignées en cours.

Une interface graphique se justifie dans le cas d'une application importante comportant de très nombreuses possibilités et étant destinée à plusieurs types d'utilisateurs, mais ce n'est pas le cas de ce projet, dont nous pensons que l'intérêt se situe dans la compréhension complète et l'implémentation d'un algorithme d'apprentissage artificiel. Les interactions avec l'utilisateur et l'affichage se font donc intégralement dans un terminal.

2 Classes et méthodes

En préambule, nous recommandons de parcourir le code (commenté) des classes pour en avoir une vision précise. L'algorithme des arbres de décision étant par nature récursif, nous avons représenté l'arbre sous forme récursive, chaque nœud possédant un conteneur de ses nœuds enfants directs.

Notre arbre de décision est représenté par une classe, qui contient les données d'apprentissage et de validation, ainsi qu'une référence vers le nœud racine de l'arbre. Il possède une méthode pour construire récursivement ses nœuds, une méthode de validation, une méthode d'élagage, et deux méthodes pour afficher l'arbre et sa taille. Ces méthodes ne sont toutefois globalement que des appels récursifs sur les méthodes du nœud racine.

La classe comportant le gros de l'algorithme est celle des nœuds. Elle implémente les mêmes méthodes que celles énoncées pour la classe d'arbre de décision, toutefois c'est bien ici que se font les calculs (i.e. le choix d'attribut selon le calcul d'entropie énoncé par Shannon).

Un nœud possède un sous-ensemble du jeu d'exemples, qui correspond à ceux parmi les exemples de son parent qui ont comme valeur d'attribut celle attribuée à ce nœud. C'est une représentation

directe de ce qui se passe dans l'algorithme des arbres de décision. Au niveau structure de données, chaque nœud possède un conteneur (liste) de ses nœuds enfants, permettant ainsi l'appel récursif.
Nous avons aussi une classe Exemple, dont le but principal est de stocker chaque exemple en mémoire afin de ne parser qu'une fois le fichier de données.

3 Tests

Notre algorithme ne gère que des valeurs discrètes. Nos tests ont donc été limités aux quelques jeux de données que nous avons trouvés correspondant à cette contrainte et au problème de classification. Les jeux de données sont inclus dans le dossier «data» accompagnant ce rapport.

L'implémentation a été testée sur 6 jeux de données différents. Pour la phase d'élaboration, nous avons utilisé des jeux simples : *weather-nominal*, *coup-de-soleil* et *contact-lenses*, qui ne comportent que quelques exemples. La pertinence de tests sur ces jeux est donc limitée.

Nous avons en revanche trois autres jeux de données plus conséquents : "tic-tac-toe", "nursery" et "mushroom". L'objectif était de mesurer l'impact de la valeur «V» sur l'élagage de l'arbre, cette valeur étant la variation de précision exigée pour l'élagage d'un nœud. Nous avons observé cet impact à travers deux mesures : le nombre de nœuds et le nombre de feuilles de l'arbre.

Mushroom 3000 exemples au total.

V (%)	0	0,1	0,5	1	5	10
Nombre de noeuds	476	2	2	87	476	476
Nombre de feuilles	678	7	7	127	678	678
Précision	0.787359	0.875369	0.875369	0.858239	0.787359	0.787359

Tic-tac-toe 950 exemples.

V (%)	0	0,1	0,5	1	5	10
Nombre de noeuds	71	28	51	35	62	71
Nombre de feuilles	124	54	87	61	107	124
Précision	0.787359	0.875369	0.875369	0.858239	0.787359	0.787359

Nursery 13000 exemples.

V (%)	0	0,1	0,5	1	5	10
Nombre de noeuds	266	246	266	266	266	266
Nombre de feuilles	596	550	596	596	596	596
Précision	0.952314	0.953395	0.952314	0.952314	0.952314	0.952314

Notes Les jeux d'apprentissage / validation ont été déterminés à proportion constante (50/50) mais aléatoirement pour les jeux « tic-tac-toe » et « nursery » à partir du jeu complet, ce qui introduit des variations locales parfois importantes (d'autant plus que les arbres de décision sont sensibles à de tels changements).
Il ne faut donc pas lire trop loin dans les chiffres présentés ici, qui ne sont représentatifs que de tendances.

Interprétations

Nous retenons de ces tests que :

1. Le fait d'évaluer la précision de l'arbre pour chaque nœud testé devient coûteux en temps de calcul, surtout si l'élagage est très léger.
2. Le gain de précision apporté par l'élagage avec la méthode de réduction d'erreur est moindre quand la précision initiale de l'arbre est élevée.
3. Exiger un gain important de précision à chaque nœud évalué réduit considérablement le nombre de nœuds élagués ainsi que la précision de l'arbre final, une fois tous les nœuds évalués.
4. Un grand nombre d'exemples permet une précision accrue de l'arbre
5. Les arbres de décision sont très sensibles aux variations dans le jeu d'exemples

Les points 2 et 4 peuvent être liés aux jeu utilisés pour les tests, ces conclusions sont donc à prendre avec précaution.