Vers une classification non supervisée adaptée pour obtenir des arbres de décision simplifiés

Olivier Parisot, Yoanne Didry, Pierrick Bruneau, Thomas Tamisier

Département Informatique, Systèmes et Collaboration (ISC)
Centre de Recherche Public - Gabriel Lippmann, Belvaux, Luxembourg
parisot@lippmann.lu

Résumé. L'induction d'arbre de décision est une technique puissante et populaire pour extraire de la connaissance. Néanmoins, les arbres de décision obtenus depuis des données issues du monde réel peuvent être très complexes et donc difficiles à exploiter. Dans ce cadre, cet article présente une solution originale pour adapter le résultat d'une classification non supervisée quelconque afin d'obtenir des arbres de décision simplifiés pour chaque cluster.

1 Introduction

Utilisée à l'origine comme un outil d'aide à la décision, les arbres de décision sont très populaires en fouille et en analyse visuelle des données. Ce succès s'explique notamment par le fait qu'ils utilisent un formalisme transparent et simple à comprendre (Murthy, 1998).

En pratique, la génération automatique d'arbres de décision depuis des données est possible grâce à l'induction d'arbre de décision, une technique bien connue (Quinlan, 1986). Malheureusement, les arbres de décision générés depuis des données issues du monde réel peuvent être très grands et difficiles à exploiter. De nombreuses approches de simplification ont donc été proposées. La plus connue, l'élagage (*pruning*) (Breslow et Aha, 1997), consiste à supprimer les parties de l'arbre qui ont un faible pouvoir explicatif. D'autre part, il existe des approches qui travaillent directement sur les données, en utilisant des méthodes de *preprocessing* (Parisot et al., 2013a). Enfin, la classification non supervisée (ou *clustering*) est un outil particulièrement utile pour la fouille de données, et à priori, cette technique peut donc être également exploitée pour obtenir des arbres de décision plus simples : en conséquence, une étude récente a proposé une nouvelle approche de classification non supervisée pour obtenir des arbres de décisions plus simples (Parisot et al., 2013b).

Dans cet article, nous présentons une méthode permettant d'adpater une classification non supervisée existante afin de simplifier l'arbre de décision obtenu à partir de chaque cluster.

2 Contribution

La méthode proposée (Figure 1) a pour but d'adapter une classification non supervisée pouvant être obtenue au moyen de toute technique existante non hiérarchique (k-means, EM,

etc.) (Gan et al., 2007). Plus précisément, l'adaptation d'une classification non supervisée composée de clusters $C_1, \ldots, C_n$ revient à effectuer une succession de déplacements d'éléments entre clusters, de C_i vers $C_j (i \neq j)$, sans création ni suppression de clusters. De manière à s'assurer durant l'adaptation que la classification obtenue n'est pas trop éloignée de la classification initiale, l'indice de Jaccard (Jaccard, 1908) est utilisé pour comparer leur similarité (classifications similaires si indice proche de 1, différentes si indice proche de 0).

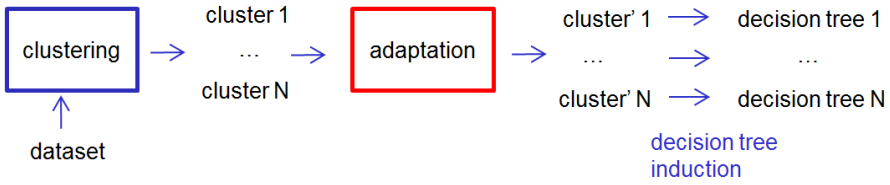

FIG. 1 – *Méthode d'adaptation d'une classification non supervisée.*

La phase d'adaptation se matérialise sous la forme d'un algorithme (Algorithme 11) dont les paramètres d'entrée sont les suivants : un jeu de données comprenant un attribut *classe*, une classification non supervisée initiale, ainsi qu'un indice Jaccard minimum sous lequel il ne faut pas descendre. Le résultat obtenu est une classification non supervisée adaptée dans laquelle chaque cluster sert à produire un arbre de décision simplifié.

Algorithm 1 Algorithme d'adaptation de clusters

Require: $0 < jaccardIndexLimit < 1 \wedge nbPassesLimit \geq 0$
Ensure:
  $nbPasses \leftarrow 0, j \leftarrow 1$
  while $j < jaccardIndexLimit \wedge nbPasses < nbPassesLimit$ **do**
    for all item à deplacer **do**
      cherche un cluster C' cible pour item
      if C' existe **then**
        déplace item dans C'
        $j \leftarrow$calcul indice de Jaccard
      end if
    end for
    $nbPasses \leftarrow nbPasses + 1$
  end while

Le principe de l'algorithme est le suivant : des déplacements entre clusters sont effectués tant que l'indice Jaccard minimum spécifié n'est pas atteint. De plus, il se peut qu'au bout d'un certain nombre d'itérations, l'algorithme ne trouve plus de déplacement possible à effectuer, et par conséquent l'indice Jaccard minimum spécifié ne peut jamais être atteint : un nombre de passes maximum est donc également prévu.

À priori, les éléments susceptibles d'être déplacés d'un cluster à un autre sont à chercher parmi ceux qui rendent les arbres de décisions complexes, c'est-à-dire : les éléments mal classifiés/expliqués par l'arbre de décision, et les éléments classifiés par des branches ayant un

faible pouvoir de classification. À posteriori, il faut ensuite s'assurer que le fait de retirer chacun de ces éléments du cluster d'origine a un effet réellement positif sur la taille de l'arbre de décision généré depuis le cluster d'origine. Nous calculons donc l'arbre de décision DT avant retrait, puis l'arbre de décision DT' après retrait, et nous comparons les tailles de DT et de DT' : si la taille de DT' est inférieure à celle de DT, alors l'élément peut être retiré.

Une fois qu'un élément a été sélectionné pour être déplacé dans un autre cluster, nous proposons de choisir le cluster pour lequel l'arbre de décision est le mieux impacté par l'ajout de l'élément. En d'autres termes, nous calculons pour chaque cluster l'arbre de décision DT avant ajout, puis l'arbre de décision DT' après ajout, et nous comparons la taille de DT et DT'. Si pour tout cluster, la taille de DT' est toujours supérieure à la taille de DT, alors il n'y pas de cluster cible : il n'y a donc pas de déplacement. Sinon, le cluster cible est le cluster pour lequel le ratio entre taille de DT' et taille de DT est le meilleur : le déplacement est donc effectué.

3 Expériences

Un prototype a été réalisé en Java, en utilisant des fonctionnalités offertes par Weka (Hall et al., 2009). De plus, des tests ont été réalisés sur une sélection de données UCI (Bache et Lichman, 2013) : nous avons comparé les arbres obtenus dans les situations suivantes :

- CAS1 : Génération de l'arbre de décision sur le jeu de données complet.
- CAS2 : Classification non supervisée du jeu de données avec l'algorithme k-means (pour différentes valeurs de k), et génération de l'arbre de décision pour chaque cluster.
- CAS3 : Application de notre méthode d'adaptation sur les clusters obtenus précédemment, et génération de l'arbre de décision pour chacun des clusters.

La génération des arbres de décision a été effectuée en utilisant l'algorithme C4.5, via l'implémentation J48 : l'algorithme a été configuré en désactivant la fonction d'élagage, de manière à obtenir des arbres de décision initiaux très complexes.

TAB. 1 – *Pour 3 clusters. CAS1 : taille de l'arbre et taux d'erreur sur tout le jeu de données. CAS2/3 : moyennes des tailles d'arbres et des taux d'erreur pour chaque cluster.*

Dataset	CAS1 (full)	CAS2 (kmeans)	CAS3 mji=0.95	CAS3 mji=0.9	CAS3 mji=0.85	CAS3 mji=0.8
cmc	457/0.24	57/0.13	49/0.12	37/0.11	37/0.10	31/0.10
vehicle	173/0.06	49/0.04	41/0.04	37/0.05	31/0.05	25/0.05
autos	76/0.04	30/0.06	25/0.06	21/0.09	n/a	n/a
credit-a	120/0.06	40/0.05	27/0.03	27/0.03	25/0.03	20/0.02
spectrometer	149/0.14	51/0.16	47/0.16	43/0.18	41/0.18	41/0.16
landsat	435/0.03	75/0.02	53/0.02	n/a	n/a	n/a
credit-g	334/0.10	108/0.08	98/0.07	91/0.06	80/0.05	74/0.06
mushroom	47/0.33	16/0.26	16/0.25	16/0.23	n/a	n/a
anneal	59/0.009	19/0.003	18/0.001	n/a	n/a	n/a
bank	5120/0.04	1916/0.04	1492/0.03	1347/0.02	1158/0.02	n/a
adult	2442/0.09	391/0.07	308/0.06	260/0.04	231/0.03	188/0.01

Les résultats obtenus, notamment avec des classifications composées de 3 clusters, (Table 1), montrent que notre méthode permet de simplifier les arbres de décision de chaque cluster. Sur l'ensemble des jeux de données testés, les gains les plus spectaculaires concernent les jeux de données *cmc*, *vehicle*, *credit-a*, *adult* : en effet la taille moyenne des arbres de décision peut être réduite jusqu'à 50%. Un bémol toutefois : dans certains cas comme *mushroom*, k-means produit des clusters ayant des arbres de décision déjà bien simplifiés : notre méthode ne permet pas de les simplifier davantage.

4 Conclusion et perspectives

Dans cet article, nous avons présenté une méthode permettant d'utiliser le résultat d'une classification non supervisée pour simplifier efficacement les arbres de décision de chacun des clusters. La méthode procède par déplacements d'éléments entre clusters en réduisant au fur et à mesure la taille des arbres de décision de chacun des clusters. Les futurs travaux concerneront la recherche d'une solution similaire pour les flux de données.

Références

Bache, K. et M. Lichman (2013). UCI machine learning repository.

Breslow, L. A. et D. W. Aha (1997). Simplifying decision trees : A survey. *Knowl. Eng. Rev. 12*(1), 1–40.

Gan, G., C. Ma, et J. Wu (2007). *Data clustering - theory, algorithms, and applications*. SIAM.

Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, et I. H. Witten (2009). The weka data mining software : an update. *SIGKDD Explor. Newsl. 11*(1), 10–18.

Jaccard, P. (1908). *Nouvelles recherches sur la distribution florale*. Bulletin de la Société vaudoise des sciences naturelles. Impr. Réunies.

Murthy, S. K. (1998). Automatic construction of decision trees from data : A multi-disciplinary survey. *Data Min. Knowl. Discov. 2*(4), 345–389.

Parisot, O., P. Bruneau, Y. Didry, et T. Tamisier (2013a). User-driven data preprocessing for decision support. In Y. Luo (Ed.), *Cooperative Design, Visualization, and Engineering*, Volume 8091 of *Lecture Notes in Computer Science*, pp. 81–84. Springer Berlin Heidelberg.

Parisot, O., Y. Didry, T. Tamisier, et B. Otjacques (2013b). Using clustering to improve decision trees visualization. In *Proceedings of the 17th International Conference Information Visualisation (IV)*, London, United Kingdom, pp. 186–191.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning 1*, 81–106.

Summary

Decision trees are simple and powerful tool for knowledge extraction and visual analysis. However, decision trees tend to be complex when they are built from real world data. To solve this issue, clustering can be efficiently used, and this paper proposes an original solution to adapt any clustering results in order to simplify the decision tree obtained from each cluster.