Synthèse de documents scientifiques

Lucas Dufour

05 avril 2021

Contexte

Notre projet d'initiation à la recherche se porte sur l'analyse d'image. Le but est de constituer un réseau de neurones afin de reconnaître dans un premier temps quels champignons appartiennent au groupe des bolets et dans un second temps quels sont ceux qui sont comestibles. Pour satisfaire cet objectif, nous devons classifier les champignons grâce à des critères de sélection. Pour ce faire, nous pouvons utiliser des algorithmes de classification. Le but de ces recherches est une étude des différents algorithmes de classification dans l'optique de les exploiter dans la reconnaissance et la classification de champignons.

État de l'art et présentation des algorithmes

Après une rapide recherche sur les algorithmes de classification, nous pouvons en citer quelques-uns qui semblent être très utilisés dans le milieu : l'algorithme C4.5 qui utilise des arbres de décision, NaiveBayes, SVM (*Support Vector Machine*), *Apriori algorithm*, *AdaBoost*, etc [1]. L'algorithme C4.5 est implémenté en 1997 et puise ses fondations dans CLS et ID3. L'algorithme C5.0 est une amélioration du C4.5, il est intéressant de l'étudier. Ces algorithmes de classification se basent sur différentes méthodes sur lesquelles je reviendrai (très) brièvement.

ID3, C4.5 et C5.0 (arbres de décision)

"Cette méthode utilise des arbres de décision pour construire le modèle de classification [2, p. 18]". Le *dataset*[1] est sous divisé de telle sorte qu'un arbre est formé. Les feuilles de cet arbre représentent les décisions finales de l'algorithme. À chaque branche est associée un choix. Les algorithmes utilisant des arbres de décision peuvent utiliser des valeurs continues[2] ou des valeurs discrètes.[3] Les algorithmes ID3 et maintenant C4.5 et C5.0 utilisent cette méthode.

C4.5 construit un arbre de décision en se basant sur des calculs d'entropie. En cherchant quels attributs maximisent l'entropie pour un *dataset* donné, on peut obtenir les attributs les plus pertinents et ainsi dresser l'arbre de décision. Dans les années 1990, C4.5 semble moins efficace avec des valeurs continues qu'avec des valeurs discrètes. Il a été modifié et amélioré en utilisant des méthodes de discrétisation par l'entropie. Les modifications apportées permettent une diminution significative du taux d'erreur ainsi qu'une diminution de la taille des arbres [3]. C4.5 sort sous sa 8ème version en prenant en compte les modifications de l'article [3] (les résultats sont donnés par le créateur de l'algorithme). C5.0 améliore encore l'algorithme : il est plus efficace et moins coûteux en mémoire.

[1] Le terme *dataset* est conservé dans sa version anglophone pour des raisons de simplification du propos. Comprendre ici jeu de données

[2] Valeurs numériques (18.0 | 18.4 | 26.1 | 32)

[3] Des catégories (ex. l'attribut Couleur peut prendre une des valeurs suivantes : Vert | Rouge | Marron | Blanc)

Support Vector Machine (SVM)

Support Vector Machine (SVM) analyse les données fournies et en extrait des patterns. Il est très intéressant puisqu'il ne requiert qu'un petit nombre d'exemples pour son entrainement (une douzaine peut permettre de donner une première base de travail). L'algorithme sépare les données en classes grâce à des hyperplans, l'objectif étant de maximiser la marge[4] entre deux classes. Les méthodes de minimisation de la marge sont utiles puisqu'elles reposent sur des mathématiques simples (essentiellement du calcul de distance) et sont donc facilement généralisables. [1, p. 10].

K-nearest neighbor (KNN)

La classification *K-nearest neighbor* (KNN) améliore le principe du classificateur de Rote qui classifie les données en se basant sur l'exacte similitude avec un élément du *training set*[5] [1, p. 22]. *KNN* recherche dans le *training set* les k voisins le plus proche de l'échantillon testé. La proximité d'un voisin revient à calculer une distance euclidienne entre les échantillons du *training set* et la donnée d'entrée. Le taux de succès de la classification est fortement corrélé à la valeur de k choisie [4].

Naive Bayes

Naive Bayes permet d'assigner un objet décrit par quelques vecteurs à une classe à laquelle appartient un ensemble d'objets similaires. C'est un algorithme facile à comprendre mais il ne sera pas le meilleur classificateur[6]. [1, p. 25]. Les résultats de l'article [5] semblent également aller dans ce sens. Certaines configurations se portent bien à l'utilisation de cet algorithme. C'est par exemple le cas lorsque les caractéristiques sont indépendantes ou lorsqu'elles sont fonctionnellement dépendantes [6].

Comparaison des algorithmes de classification

Les différents résultats observés dans l'article [7] nous montrent que le choix du classificateur et des paramètres associés n'est pas chose facile. On y apprend également que le choix du classificateur et ses performances sont corrélés au *dataset*. Dans le but de déterminer la comestibilité des champignons grâce à un algorithme, l'étude [5] propose de mettre en concurrence trois algorithmes de classification afin de déterminer quel est le meilleur. La comparaison de C4.5, Naive Bayes et SVM se base sur la fiabilité des résultats et la vitesse d'exécution des algorithmes susnommés. L'exécution se fait - bien évidemment - sur le même *dataset*, composé de 8416 éléments catégorisés en 22 attributs et menant à la décision suivante : {EDIBLE | POISONOUS}.

Toujours selon A. Wibowo [5], C4.5 est l'algorithme le plus efficace sur le *dataset* donné [Tab 1], il permet en plus d'obtenir les champs les plus pertinents. Dans notre contexte, c'est un atout de taille puisque cela nous aide à recentrer l'analyse d'image sur les éléments les plus discriminants.

[4] Marge : distance minimale entre deux points d'hyperplans différents

[5] Le terme *training set* est conservé dans sa version anglophone pour des raisons de simplification du propos. Il s'agit des données utilisées pour entrainer l'algorithme.

[6] Ici, algorithme de classification

Tab. 1 : Comparaison des performances des algorithmes de classification sur un même *dataset* [5]

	C4.5	Naive Bayes	SVM
Fiabilité (%)	100	95.8887	100
Temps d'exécution (ms)	180	340	320

En matière de fiabilité, on remarque que *SVM* est aussi pertinent que C4.5 mais il reste plus lent. Cette observation corrobore ce qui est annoncé dans l'article [1, p. 12]. En effet, il y est dit que c'est un algorithme assez simple à comprendre mais qu'il n'est pas très efficace.[7] Il est cependant possible de l'améliorer en découpant le problème initial en plusieurs problèmes faciles à résoudre.

La pertinence de C4.5 dans le contexte d'utilisation du projet nous pousse à étudier plus en profondeur cet algorithme. Il est alors nécessaire de comparer C4.5 et une version plus récente : C5.0. Cette nouvelle version est plus efficace, plus précise et moins gourmande en mémoire. C4.5 et C5.0 peuvent tous les deux produire des arbres de décision mais aussi des ensembles de règles. C4.5 n'est cependant pas très efficace (lent et coûteux en mémoire) pour produire ces ensembles de règles. C5.0 permet de solutionner les problèmes de sur-apprentissage[8] et anticipe quels attributs sont les plus pertinents. Cela permet de rendre les arbres produits par C5.0 plus petits. [2]

Vers des solutions hybrides

D'autres articles présentent des solutions hybrides en associant plusieurs algorithmes de classification. Dans [8], il est question de créer un algorithme hybride *decision tree-SVM* (DT-SVM), associant alors C5.0 et *SVM* pour la détection d'intrusion système. On augmente ainsi les performances de l'algorithme en exploitant au maximum les points forts de l'un pour combler les faiblesses de l'autre. L'idée est d'améliorer les performances de SVM grâce à C5.0. La création de solutions hybrides demande une réflexion sur la topologie de l'hybridation, elle peut être en série ou en parallèle [9, p. 6, Fig.4]. Ici le *dataset* original est classifié avec un algorithme C5.0 pour recréer un nouveau *dataset* qui sera utilisé comme entrée de *SVM* ; il s'agit d'une topologie en série. Les données sont comparées à une exécution de *SVM* standard, sans surcouche de C5.0. Les résultats montrent que cet hybride donne de meilleures performances générales. Les améliorations sont encore plus probantes sur des petits échantillons.

Conclusion

En fonction des besoins, il est possible de se tourner vers différents algorithmes. Les algorithmes C4.5 et C5.0 semblent pertinents puisqu'ils sont rapides, peu gourmand en mémoire et permettent de déterminer les facteurs discriminants. De plus, C4.5 a déjà été utilisé dans un contexte de classification de champignon et il obtenait les meilleurs résultats. Toutefois, nous avons appris que SVM pouvait être utile sur un petit ensemble de données. Enfin, nous remarquons qu'il est possible de s'atteler à la conception d'un algorithme sur mesure, un algorithme hybride qui fusionne plusieurs classificateurs entres eux. Cette solution demande toutefois un travail plus long mais offre des résultats encore meilleurs.

[7] En parlant de la vitesse d'exécution

[8] Sur-apprentissage (*overfitting*) est problématique puisque l'algorithme commence à apprendre du bruit et peut rejeter des solutions correctes.

Références bibliographiques

[1] X. Wu *et al.*, « Top 10 Algorithms in Data Mining », *Knowl Inf Syst*, vol. 14, n 1, p. 1-37, janv. 2008, doi : 10.1007/s10115-007-0114-2.

[2] R. Pandya et J. Pandya, « C5. 0 Algorithm to Improved Decision Tree with Feature Selection and Reduced Error Pruning », *IJCA*, vol. 117, n 16, p. 18-21, mai 2015, doi : 10.5120/20639-3318.

[3] J. R. Quinlan, « Improved Use of Continuous Attributes in C4.5 », *Journal of Artificial Intelligence Research*, vol. 4, p. 77-90, mars 1996, doi : 10.1613/jair.279.

[4] G. Guo, H. Wang, D. Bell, Y. Bi, et K. Greer, « KNN Model-Based Approach in Classification », in *On The Move to Meaningful Internet Systems 2003 : CoopIS, DOA, and ODBASE*, vol. 2888, R. Meersman, Z. Tari, et D. C. Schmidt, Éd. Berlin, Heidelberg : Springer Berlin Heidelberg, 2003, p. 986-996.

[5] A. Wibowo, Y. Rahayu, A. Riyanto, et T. Hidayatulloh, « Classification Algorithm for Edible Mushroom Identification », in *2018 International Conference on Information and Communications Technology (ICOIACT)*, mars 2018, p. 250-253, doi : 10.1109/ICOIACT.2018.8350746.

[6] I. Rish, « An Empirical Study of the Naive Bayes Classifier », p. 6, 2001, [En ligne]. Disponible sur : https://www.cc.gatech.edu/~isbell/reading/papers/Rish.pdf.

[7] D. R. Amancio *et al.*, « A Systematic Comparison of Supervised Classifiers », *PLoS ONE*, vol. 9, n 4, p. e94137, avr. 2014, doi : 10.1371/journal.pone.0094137.

[8] V. Golmah, « An Efficient Hybrid Intrusion Detection System Based on C5.0 and SVM », *IJDTA*, vol. 7, n 2, p. 59-70, avr. 2014, doi : 10.14257/ijdta.2014.7.2.06.

[9] M. Woźniak, M. Graña, et E. Corchado, « A Survey of Multiple Classifier Systems as Hybrid Systems », *Information Fusion*, vol. 16, p. 3-17, mars 2014, doi : 10.1016/j.inffus.2013.04.006.