De l'importance de la fonction de poids dans le noyau des sous-arbres

Florian Ingels, Romain Azaïs

De l'importance de la fonction de poids dans le noyau des sous-arbres

Florian Ingels[1] & Romain Azaïs[2]

Laboratoire Reproduction et Développement des Plantes, Univ Lyon, ENS de Lyon, UCB Lyon 1, CNRS, INRA, Inria, F-69342, Lyon, France
[1] *florian.ingels@inria.fr*
[2] *romain.azais@inria.fr*

Résumé. Les méthodes à noyaux sont une des approches permettant l'apprentissage à partir de données arborescentes. Parmi elles, nous nous intéressons au noyau des sous-arbres, qui présente l'avantage combinatoire de pouvoir énumérer l'ensemble des objets utilisés pour estimer la similitude entre deux arbres. Nous introduisons le concept de réduction DAG, puis de recompression DAG, qui mènent à un algorithme efficace pour calculer le noyau. La possibilité d'aborder une base de données sous un format fortement compressé nous permet d'introduire une nouvelle fonction de poids, qui parvient à capturer de l'information dans des données où les poids proposés dans la littérature n'y arrivent pas.

Mots-clés. Données arborescentes, SVM, méthodes à noyaux, noyau des sous-arbres

Abstract. Kernel methods are one of the main techniques used for learning on tree structured data. Among them, we interest ourselves to the subtree kernel, which has the combinatorial advantage of being able to enumerate all objects used in the estimation of the similarity between trees. We introduce concepts of DAG reduction and DAG recompression, which lead us to a performant algorithm for computing the kernel. The ability of dealing with a highly compressed database allows us to introduce a new weight function, which manages to capture information in data where weights in the literature do not.

Keywords. Tree data, SVM, kernel methods, subtree kernel

1 Introduction

Les données arborescentes apparaissent naturellement dans de nombreux domaines, notamment en biologie (plantes, réseaux sanguins, arbres phylogénétiques) mais également en informatique (hiérarchie arborescente d'une page HTML). Formellement, un arbre est un graphe connecté sans cycle. On dit d'un arbre T qu'il est enraciné s'il existe un sommet unique $R(T)$, appelé la racine, qui n'a pas de parent, et tous les autres sommets autres que la racine admettent un unique parent. Les feuilles sont les sommets sans enfants. Un

sous-arbre (*subtree*) de T enraciné en ν est l'arbre composé de ν et de tous ses descendants dans T. On distingue également les arbres ordonnés (où l'ordre des enfants d'un noeud à de l'importance) des arbres non-ordonnés, et les arbres avec étiquette sur les noeuds de ceux qui n'en ont pas. Ces distinctions permettent de rendre compte de la diversité des structures arborescentes; par exemple, l'introduction d'étiquettes permet de prendre en considération une géométrie induite sur l'arbre.

L'étude de telles données recèle de nombreuses difficultés, notamment par leur caractère profondément non-euclidien. Les outils statistiques habituels ne s'appliquent pas, et il est également particulièrement difficile d'obtenir des représentations efficaces d'une base de données. De nombreuses méthodes ont été développées pour comparer de telles structures entre elles, parmi lesquelles se trouvent le calcul de la distance d'édition (Zhang, 1989) ou bien encore les méthodes à noyaux (Schölkopf et al., 2004).

Ces dernières sont elles-mêmes très vastes, mais la plupart reposent sur l'idée de mesurer la proximité entre deux arbres arbres en comparant les sous-structures en commun. On les appelle noyaux de convolution (voir Da San Martino pour un survol des différentes techniques). Parmi elles, le noyau des sous-arbres (*subtree kernel*) auquel nous nous intéressons a été introduit par Vishwanathan et Smola (2000), et utilise comme sous-structure de comparaison les sous-arbres comme définis au-dessus,

$$K(T_1, T_2) = \sum_{\tau \in S(T_1) \cap S(T_2)} \omega_\tau N_\tau(T_1) N_\tau(T_2) \tag{1}$$

où $S(T)$ désigne l'ensemble des sous-arbres de T, $N_\tau(T)$ le nombre d'occurrences du sous-arbre τ dans l'arbre T, et ω_τ est un poids attribué au sous-arbre τ.

Une fois calculés, les noyaux ont plusieurs applications, dont notamment les machines à vecteur de support (SVM pour *Support Vector Machines*) qui sont un outil classique de classification (Cristianini et Taylor, 2000). On a à disposition une base de données $(x_i, y_i)_{i=1\dots n} \in \mathcal{X} \times \mathcal{Y}$, avec $\mathcal{X}$ l'espace des arbres et $\mathcal{Y}$ l'espace des classes (typiquement, $\mathcal{Y} = \{-1, 1\}$). Ces données sont utilisées pour entraîner l'algorithme de classification, qui est ensuite employé pour prédire la classe (inconnue) de nouvelles données $(x_j)_{j=n+1\dots m} \in \mathcal{X}$. Les SVM exploitent des calculs de produit scalaire entre les données. Le recours au noyau permet notamment de simuler un produit scalaire dans un espace hilbertien, sans toutefois avoir besoin d'expliciter cet espace ni la transformation qui permet d'y envoyer les données. Cette technique est connue sous le nom d'astuce du noyau, et repose sur le théorème de Mercer (1909).

2 Calcul du noyau

Vishwanathan et Smola proposent un algorithme pour calculer le noyau en transformant les arbres en chaînes de caractères. Le sens de parcours de l'arbre a donc une importance

cruciale, ce qui pose problème lorsqu'on considère des arbres non-ordonnés : une permutation de l'ordre des enfants d'un noeud mènerait en effet à des résultats différents. Un autre algorithme, inspiré par le noyau des sous-ensembles d'arbres (proposé par Collins et Duffy en 2001 — voir aussi Da San Martino), propose un calcul récursif mais menant à des poids de la forme $\omega_\tau = \lambda^{H(\tau)}$ où $H(\tau)$ représente la hauteur de τ, soit la longueur du plus chemin entre la racine et les feuilles de τ.

Aucun algorithme à notre connaissance ne permet à la fois de gérer le cas des arbres ordonnés ou non, étiquetés ou non, et avec une souplesse suffisante pour apprendre la fonction de poids à partir des données. Nous proposons maintenant de construire un tel algorithme générique.

Le cardinal des sous-arbres d'un arbre est majoré par le nombre de noeuds de cet arbre. En conséquence, les sous-arbres forment une classe combinatoire assez réduite, qui peut intégralement être décrite. La réduction sous forme de graphe acyclique dirigé (DAG pour *Directed Acyclic Graph*) d'un arbre est une structure qui fait naturellement apparaître les sous-arbres, en plus de montrer des propriétés de compression sans perte remarquables (Buneman et al., 2003). Le DAG représente l'arbre de départ en éliminant la redondance des sous-arbres identiques (voir Figure 1 pour un exemple dans le cas non-ordonné).

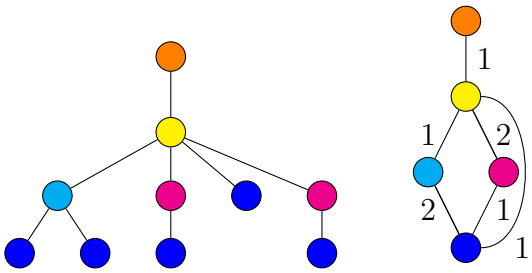


Figure 1: Un arbre non-ordonné et sa réduction DAG. Dans l'arbre, les racines des sous-arbres isomorphes sont colorées à l'identique. Dans le DAG, les sommets représentent les classes d'équivalence et sont colorés suivant la classe de sous-arbres isomorphes qu'ils représentent. Par exemple, le sous-arbre jaune comporte comme enfants un sous-arbre cyan, deux sous-arbres magenta et un sous-arbre bleu.

En fait, il est même possible de conserver toute l'information d'une base de données dans un unique DAG, en plaçant les racines de chaque DAG sous une racine artificielle commune, et en recompressant le résultat (voir Figure 2). Il est par la suite possible de retrouver l'origine de chacun des noeuds fusionnés, en parcourant la descendance de chaque enfant de la racine artificielle.

Le noyau entre deux arbres peut se calculer directement en observant le DAG recompressé. En effet, $K(T_i, T_j)$ ne fait intervenir que les noeuds du DAG dont l'origine contient le couple (i, j). Néanmoins, pour éviter de parcourir le DAG en entier à chaque calcul de noyau, nous introduisons une matrice de correspondance, qui associe au coefficient (i, j) la liste des noeuds en commun entre T_i et T_j, et toutes les informations sur

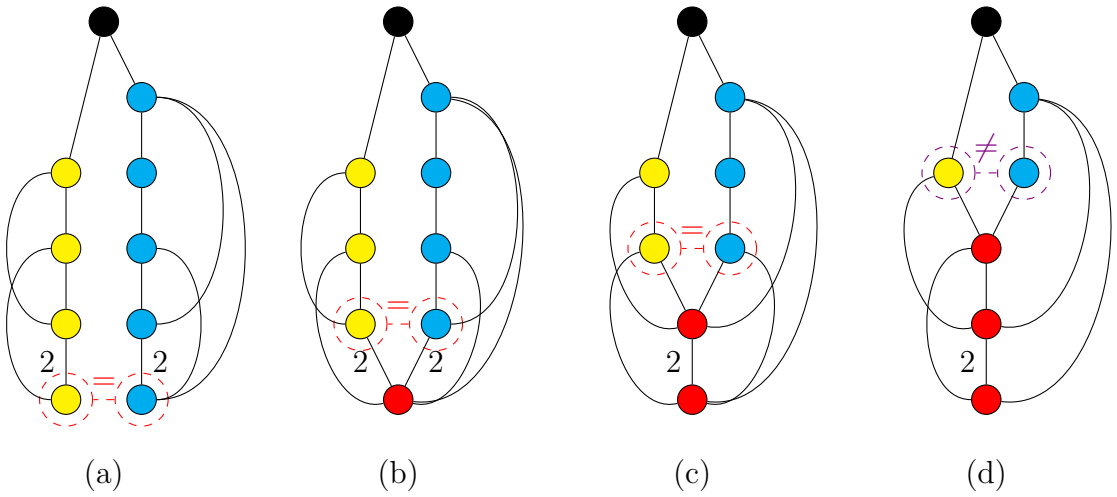


Figure 2: On veut ici recompresser une base de données formée par deux DAG associés à des arbres non-ordonnés, l'un en jaune, et l'autre en bleu. La racine artificielle est colorée en noire. L'algorithme parcourt dans l'ordre croissant les différentes hauteurs de l'arbre, et fusionne à chaque étape les noeuds qui sont isomorphes (i.e. induisant le même sous-arbre). Les noeuds fusionnés sont colorés en rouge. Si à une hauteur donnée, on ne trouve plus aucun noeud à fusionner, on s'arrête.

ces noeuds (notamment $N_\tau(T)$). Un seul parcours suffit pour construire cette matrice. Calculer une unique valeur se fait ensuite à un coût en $O(\min(|S(T_i)|, |S(T_j)|)$ (au lieu de $O(|S(T_i)| + |S(T_j)|)$ pour l'algorithme de Vishwanathan et Smola), ce qui ouvre la voie à des calculs intensifs de noyau, notamment pour en optimiser les paramètres. Notons toutefois que le coût total de la procédure, en incluant les étapes précédentes, est équivalent aux techniques existantes.

3 Apprentissage du poids

Le poids ω_τ, associé à chaque sous-arbre τ, est souvent choisi dans la littérature (Vishwanathan et Smola, 2000) parmi quelques options, dont des poids constants $\omega_\tau = 1$, des Dirac $\omega_\tau = \delta_\tau^{\tau'}$ ou bien encore une décroissance exponentielle $\omega_\tau = \lambda^{H(\tau)}$, avec $\lambda < 1$.

L'étude d'un modèle stochastique d'arbres (non détaillée ici dans un soucis de concision), pour étudier le comportement du noyau des sous-arbres, nous a permis de réaliser que le poids associé aux feuilles devrait toujours être égal à 0. Puisque les feuilles apparaissent dans tous les arbres, les comptabiliser n'apporte pas d'information de classification, mais a tendance à diminuer le contraste du noyau. En extrapolant cette idée pour chaque sous-arbre, nous introduisons une nouvelle fonction poids, appelée *discriminance*, qui a pour objectif d'assigner des poids forts aux sous-arbres qui aident à discriminer les classes,

et des poids faibles aux autres.

Chaque sous-arbre τ se voit attribuer un vecteur ρ_τ de taille K (avec K le nombre de classes), où $\rho_\tau(k), k = 1 \dots K$ est égal à la proportion d'arbres de la classe k qui contiennent τ. Ainsi, ρ_τ appartient à l'hypercube de dimension K. Pour $j = 1 \dots K$, on appelle e_j (resp. $\overline{e}_j$) le vecteur de zéros avec un unique 1 en position j (resp., le vecteur de uns avec un unique 0 en position j). Si $\rho_\tau = e_j$, le sous-arbre τ n'apparaît que dans la classe j et est donc un bon discriminant pour cette classe. Inversement, si $\rho_\tau = \overline{e}_j$, τ apparaît dans toutes les classes sauf la classe j et est donc toujours un bon discriminant pour cette classe. Pour chaque sous-arbre τ, δ_τ mesure la distance entre ρ_τ and son plus proche point d'intérêt e_j ou $\overline{e}_j$,

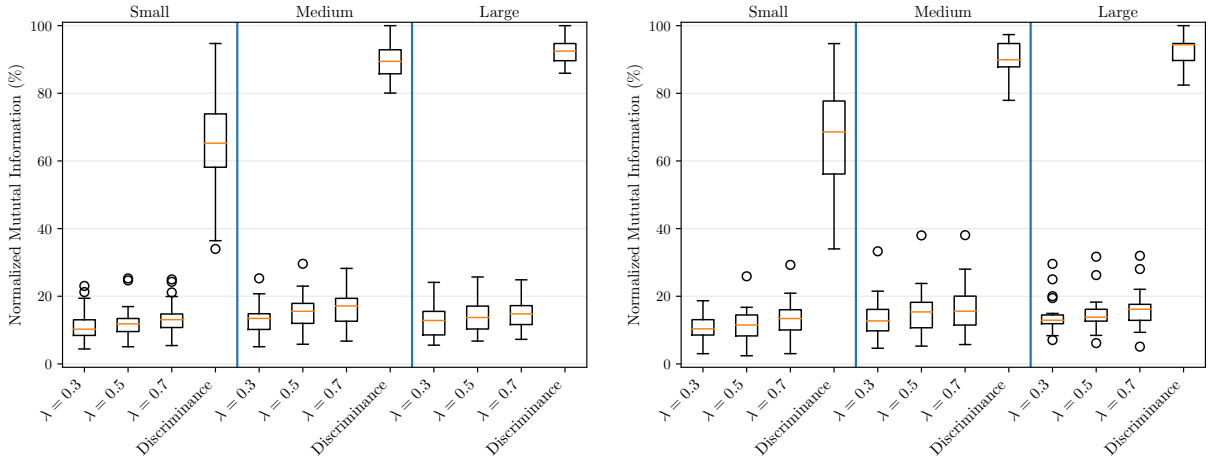
$$\delta_\tau = \min_{j=1}^{K} \min(|\rho_\tau - e_j|, |\rho_\tau - \overline{e}_j|).$$

Quand δ_τ est petit, alors ρ_τ est proche d'un point d'intérêt, et le poids associé devrait être grand. Finalement, la discriminance est définie comme $\omega_\tau = f(1 - \delta_\tau)$, avec $f :]-\infty, 1] \to [0, 1]$ croissante, $f(x) = 0$ si $x \leq 0$ et $f(1) = 1$. Dans la suite, nous choisissons la fonction smoothstep, définie par $f : x \mapsto 3x^2 - 2x^3$.

4 Applications à des données réelles

À partir d'une page HTML, il est possible d'en extraire une structure arborescente, en identifiant chaque balise ouvrante et fermante comme un noeud, dont les enfants sont les balises internes (Buneman et al.). Il est important de noter que durant cette opération, toute information sémantique sur la page est oubliée ; seule l'information topologique est conservée. Cette transcription est faite dans la suite de manière ordonnée puis non-ordonnée.

La base de données que nous introduisons utilise des pages Wikipedia choisies au hasard et converties en arbres par la méthode précédente. La classe d'une page correspond à la langue dans laquelle elle a été rédigée, parmi l'anglais, l'allemand, l'espagnol et le français. Nous avons généré 30 bases de données de la façon suivante. Nous créons une base de test de 120 arbres, et trois bases d'apprentissage : une petite (40 arbres), une moyenne (120 arbres) et une grande (200 arbres). Chaque base comporte le même nombre d'arbres de chaque langue. L'exploitation par les SVM du noyau des sous-arbres pour ce problème de classification de la langue mène aux résultats de la Figure 3. Ainsi, le poids exponentiel introduit dans la littérature ne permet pas de bien classer les données, alors que la discriminance le peut ; ce qui montre que l'information de la langue est présente dans la structure de la page Wikipedia, contrairement à ce qu'on pourrait penser.



(a) Arbres ordonnés

(b) Arbres non-ordonnés

Figure 3: Résultats de classification. Lorsque le poids utilisé est une décroissance exponentielle, la valeur du paramètre λ est indiquée. L'Information Mutuelle Normalisée (NMI) mesure la proximité entre le vecteur des classes prédites et le vecteur des classes attendues, où une valeur de 100% traduit une équivalence parfaite.

Bibliographie

Schölkopf et al. (2004), Kernel methods in computational biology, MIT Press.
Da San Martino, G. (2009), Kernel methods for tree structured data, PhD thesis, alma.
Collins M. et Duffy N. (2001), Convolution kernels for natural language, dans Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS'01), T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.). MIT Press, Cambridge, MA, USA, 625-632.
Mercer J. (1909), Functions of positive and negative type and their connection with the theory of integral equations, *Philos. Trans. Roy. Soc. London*, série A, vol 209, p 415-446.
Buneman P., Grohe M. et Koch C. (2003), Path Queries on Compressed XML, VLDB
Cristianini, N. et Taylor, J. (2000), *Support Machines and Other Kernel-based Learning Methods*, Cambridge University Press.
Smola, A. J. et Vishwanathan, S. (2003), *Fast kernels for string and tree matching*, Advances in neural information processing systems, pp 585-592.
Kawamura et al. (2014), Kernel methods for phenotyping complex plant architecture, Journal of Theoretical Biology.
Zhang, K. et Shasha, D. (1989). Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. SIAM J. Comput.. 18. 1245-1262. 10.1137/0218082.