

OK3: Méthode d’arbres à sortie noyau pour la prédiction de sorties structurées et l’apprentissage de noyau

Pierre Geurts^{1,2}, Louis Wehenkel², Florence d’Alché-Buc¹

¹ IBISC FRE CNRS 2873 & Epigenomics Project, GENOPOLE,
523, Places des Terrasses, 91 Evry, France
{pgeurts,dalche}@ibisc.univ-evry.fr

² Département d’Électricité, Électronique et Informatique, Université de Liège,
Sart Tilman, B28, 4000, Liège, Belgique
{p.geurts,l.wehenkel}@ulg.ac.be

Résumé : Dans cet article, nous proposons une extension des méthodes d’arbres pour la prédiction de sorties structurées. Cette extension est basée sur l’utilisation d’un noyau sur la sortie de ces méthodes qui leur permet de construire un arbre à la seule condition qu’un noyau puisse être défini sur l’espace de sortie. Cet algorithme, appelé OK3 (pour “output kernel trees”), généralise les arbres de classification et de régression ainsi que les méthodes d’ensemble d’arbres. Il hérite de plusieurs caractéristiques de ces méthodes telles que l’interprétabilité, la robustesse aux variables non pertinentes et la résistance à l’échelle sur le nombre d’entrées. Lorsqu’on dispose uniquement d’une matrice de Gram sur les sorties de l’échantillon d’apprentissage, il apprend le noyau de sortie comme une fonction des entrées. Nous montrons que cet algorithme donne de bons résultats sur un problème de complétion d’image et sur un problème d’inférence de réseau biologique.

1 Introduction

Étendre l’apprentissage statistique à des espaces de sortie structurés apparaît maintenant comme un nouveau défi théorique et pratique. Dans de nombreux domaines d’application tels que la biologie computationnelle, le traitement des langages naturels ou plus généralement la reconnaissance de motifs, on rencontre des données structurées sous la forme de séquences, d’arbres ou de graphes et l’exploitation de ces données par les méthodes d’apprentissage requièrent de nouveaux outils. Une manière élégante pour traiter ce type de données est d’utiliser des méthodes à base de noyau qui permettent d’encapsuler la connaissance sur la structure de l’espace d’intérêt dans la définition d’un noyau. Étant donné le succès de ces méthodes pour traiter des espaces d’entrée structurés, plusieurs méthodes à noyau originales ont été proposées récemment pour prendre

en compte une structure sur l'espace de sortie (Weston *et al.*, 2002; Cortes *et al.*, 2005; Tsochantaridis *et al.*, 2005; Taskar *et al.*, 2005; Weston *et al.*, 2005).

Dans cet article, nous partons d'une famille de modèles différente, les modèles à base d'arbre, et nous étendons ces modèles de manière à traiter un espace de sortie muni d'un noyau, les entrées restant encodées par un vecteur d'attributs. Les méthodes d'arbres ont largement prouvé leur utilité quand l'interprétabilité est importante ou quand les variables doivent être sélectionnées. Lorsqu'elles sont de plus utilisées conjointement avec une méthode d'ensemble, elles présentent de bonnes performances en termes de compromis biais/variance et sont considérées comme un outil général et puissant pour autant que la représentation attributs des entrées soit appropriée. Dans ce contexte, nous proposons une extension noyau directe des arbres de régression à sorties multiples. Cette extension est basée sur le fait que la mesure de score utilisée pour évaluer et sélectionner les questions candidates lors de la construction d'un arbre nécessitent uniquement le calcul de produits scalaires qui peuvent donc être remplacés par des noyaux. Pour autant qu'un noyau ait été défini sur l'espace de sortie, il devient donc possible de construire un arbre qui projette des sous-régions de l'espace d'entrée définies par des hyper-plans parallèles aux axes vers des hyper-sphères définies dans l'espace caractéristique correspondant au noyau de sortie. L'algorithme peut également être utilisé à partir d'une matrice de Gram uniquement pour apprendre une approximation du noyau sous-jacent comme une fonction des entrées. Cette nouvelle extension des arbres, que nous appelons OK3 (pour "output kernel trees"), peut aussi bénéficier des méthodes d'ensemble, la linéarité de l'opérateur d'agrégation préservant l'interprétation noyau.

L'article est organisé comme suit : La section 2 présente la "kernelisation" des méthodes d'arbre et ses principales propriétés. La section 3 décrit et discute les expériences réalisées sur deux problèmes : un problème de complétion de motif (Weston *et al.*, 2002) et un problème d'inférence de graphe (Vert & Yamanishi, 2004). La section 4 donne quelques perspectives.

2 Arbres à sortie noyau

Le problème général d'apprentissage supervisé peut être formulé comme suit : à partir d'un échantillon d'apprentissage $LS = \{(x_i, y_i) | i = 1, \dots, N_{LS}\}$ avec $x_i \in \mathcal{X}$ et $y_i \in \mathcal{Y}$, trouver une fonction $f : \mathcal{X} \rightarrow \mathcal{Y}$ qui minimise l'espérance d'une fonction de perte ℓ sur la distribution conjointe des paires entrées/sortie :

$$E_{x,y} \{\ell(f(x), y)\}. \quad (1)$$

2.1 Arbres de régression standards

Les arbres de régression (Breiman *et al.*, 1984) propose une solution à ce problème lorsque l'espace de sortie est l'axe réel, $\mathcal{Y} = \mathbb{R}$, et la fonction de perte ℓ est l'erreur quadratique :

$$\ell(f(x), y) = (f(x) - y)^2. \quad (2)$$

L'idée générale de cette méthode est de séparer récursivement l'échantillon d'apprentissage par des questions binaires basées sur les variables d'entrée, en essayant à chaque

nouvelle séparation de réduire autant que possible la variance empirique dans les sous-échantillons gauche et droit correspondant à la question. L'éclatement d'un noeud est arrêté lorsque la sortie est constante dans ce noeud ou un critère d'arrêt est satisfait (par exemple, la taille du sous-échantillon local est inférieure à une taille donnée ou la séparation est jugée non significative selon un test statistique donné).

Plus précisément, une mesure de score est définie pour évaluer et sélectionner les questions qui s'écrit comme suit :

$$\text{Score}_R(T, S) = \text{var}\{y|S\} - \frac{N_l}{N} \text{var}\{y|S_l\} - \frac{N_r}{N} \text{var}\{y|S_r\}, \quad (3)$$

où T est la question à évaluer, S est l'échantillon d'apprentissage local de taille N au noeud à éclater, S_l et S_r sont ses successeurs gauche et droit de tailles respectives N_l et N_r , et $\text{var}\{y|S\}$ désigne la variance empirique de la sortie y dans l'échantillon S calculée par :

$$\text{var}\{y|S\} = \frac{1}{N} \sum_{i=1}^N (y_i - \frac{1}{N} \sum_{i=1}^N y_i)^2. \quad (4)$$

Une fois l'arbre construit, chaque feuille L est étiquetée par une prédiction \hat{y}_L calculée par :

$$\hat{y}_L = \frac{1}{N_L} \sum_{i=1}^{N_L} y_i, \quad (5)$$

où N_L est le nombre d'instances de l'échantillon d'apprentissage qui atteignent cette feuille. Ces prédictions minimisent l'erreur quadratique moyenne lorsque l'arbre est utilisé pour prédire la sortie dans l'échantillon d'apprentissage.

Cet algorithme peut être étendu de manière directe au cas de la prédiction d'une sortie multiple, c'est-à-dire $\mathcal{Y} = \mathbb{R}^n$, avec la fonction de perte $\ell(f(x), y) = \|f(x) - y\|^2$, où $\|\cdot\|$ est la norme Euclidienne dans \mathbb{R}^n . Dans ce cas, la variance calculée par (3) et les prédictions aux feuilles sont respectivement remplacées par :

$$\text{var}\{y|S\} = \frac{1}{N} \sum_{i=1}^N \|y_i - \frac{1}{N} \sum_{i=1}^N y_i\|^2 \quad (6)$$

$$\hat{y}_L = \frac{1}{N_L} \sum_{i=1}^{N_L} y_i. \quad (7)$$

Cette dernière expression est le centre de masse dans la feuille et la première expression est la moyenne dans S du carré de la distance Euclidienne du vecteur de sortie au centre de masse.

2.2 Un noyau sur la sortie

En remarquant que la variance (6) peut s'exprimer seulement en termes de produits scalaires entre vecteurs de sortie, on peut appliquer directement l'astuce du noyau à la mesure de score et ainsi obtenir une manière directe de traiter des espaces ou ensembles

de sortie plus complexes. Plus précisément, définissons une fonction $\phi : \mathcal{Y} \rightarrow \mathcal{H}$ qui projette chaque sortie y vers un vecteur dans un espace de Hilbert \mathcal{H} , aussi appelé espace caractéristique. En utilisant la variance (6) calculée dans cet espace pour construire un arbre, on résout en fait le problème d'apprentissage supervisé défini plus haut avec une fonction de perte égale à :

$$\ell(f(x), y) = \|\phi(f(x)) - \phi(y)\|^2. \quad (8)$$

Dans ce cas, la variance $\text{var}\{\phi(y)|S\}$ peut s'écrire :

$$\frac{1}{N} \sum_{i=1}^N \langle \phi(y_i), \phi(y_i) \rangle - \frac{1}{N^2} \sum_{i,j=1}^N \langle \phi(y_i), \phi(y_j) \rangle \quad (9)$$

où $\|\cdot\|$ et $\langle \cdot, \cdot \rangle$ désignent respectivement la norme et le produit scalaire dans \mathcal{H} .

Définissons maintenant $k : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ une fonction noyau semi-définie positive sur l'espace de sortie, qui induit donc une fonction ϕ de \mathcal{Y} vers un espace de Hilbert \mathcal{H} (de dimension potentiellement infinie) tel que :

$$k(y, y') = \langle \phi(y), \phi(y') \rangle. \quad (10)$$

La variance empirique sur le vecteur de sortie dans \mathcal{H} devient :

$$\text{var}\{\phi(y)|S\} = \frac{1}{N} \sum_{i=1}^N k(y_i, y_i) - \frac{1}{N^2} \sum_{i,j=1}^N k(y_i, y_j), \quad (11)$$

qui s'insère directement dans le calcul de la mesure de score et étend les méthodes d'arbres à une classe très large d'espaces de sortie complexes pour lesquels des noyaux ont déjà été définis. Nous appellerons cette formulation noyau de l'algorithme d'induction d'arbre OK3, pour "Output Kernel Trees".

Les arbres de régression à sorties multiples ainsi que les arbres de classification sont des cas particuliers de OK3. La première méthode consiste à utiliser un noyau linéaire entre les vecteurs de sortie alors qu'en utilisant un noyau de Dirac, $k(y, y') = 1(y = y')$, avec une sortie qualitative, on obtient des arbres identiques aux arbres de classification standards avec une mesure de score basée sur l'entropie gini.

En termes de noyau, la fonction de perte que la méthode OK3 essaye de minimiser peut s'écrire :

$$\ell(f(x), y) = k(f(x), f(x)) + k(y, y) - 2k(f(x), y), \quad (12)$$

ce qui montre que choisir un noyau revient en fait à choisir une fonction de perte.

2.3 Prédiction

Dans l'espace caractéristique de sortie, la prédiction (7) associée à une feuille devient :

$$\hat{\phi}_L = \frac{1}{N_L} \sum_{i=1}^{N_L} \phi(y_i). \quad (13)$$

Généralement, nous sommes cependant intéressés par une prédiction dans l'espace original \mathcal{Y} . Une valeur de prédiction intéressante est la sortie dans \mathcal{Y} dont le vecteur caractéristique est $\hat{\phi}_L$, c'est-à-dire la pré-image $\hat{y}_L = \phi^{-1}(\hat{\phi}_L)$. En toute généralité, nous ne pouvons cependant pas faire l'hypothèse que nous avons accès à l'espace caractéristique et de plus, il peut ne pas exister dans \mathcal{Y} de point \hat{y}_L tel que $\phi(\hat{y}_L) = \hat{\phi}_L$. Dans ces cas, une approximation de la pré-image peut être obtenue à partir du noyau seulement par :

$$\begin{aligned}\hat{y}_L &= \arg \min_{y' \in \mathcal{Y}} \left\| \phi(y') - \frac{1}{N_L} \sum_{i=1}^{N_L} \phi(y_i) \right\|^2 \\ &= \arg \min_{y' \in \mathcal{Y}} \frac{1}{N_L} \sum_{i=1}^{N_L} \left\| \phi(y') - \phi(y_i) \right\|^2 \\ &= \arg \min_{y' \in \mathcal{Y}} k(y', y') - \frac{2}{N_L} \sum_{i=1}^{N_L} k(y_i, y'),\end{aligned}\tag{14}$$

qui est la sortie dont la projection dans l'espace caractéristique est la plus proche du centre de masse dans la feuille. En pratique, le calcul de l'arg min dans (14) est possible seulement pour des choix particuliers de \mathcal{Y} et de noyau k . Par exemple, avec un noyau linéaire dans \mathbb{R}^n , le calcul de (14) donne lieu à la prédiction (7).

Le problème du calcul de la pré-image est un problème commun à toutes les méthodes qui utilisent un noyau en sortie (cf. section 2.8). Plusieurs techniques ont été proposées pour approcher (14) (Scholkopf & Smola, 2002; Ralaivola & d'Alché-Buc, 2003). L'approximation que nous proposons s'inspire de l'approximation utilisée dans (Weston *et al.*, 2002) qui remplace la minimisation sur \mathcal{Y} par une minimisation uniquement sur les sorties qui apparaissent dans l'échantillon d'apprentissage. En prenant en compte la structure d'arbre, nous avons de plus choisi de restreindre cette minimisation aux sorties qui apparaissent dans la feuille atteinte par l'exemple de test, ce qui réduit encore d'avantage les temps de calcul.

Il est cependant important de noter que le calcul des pré-images n'est pas requis lors de la construction de l'arbre. Il est également possible de calculer l'erreur de prédiction sur un cas de test (x, y) sans calculer de pré-image, en utilisant (12) et en exploitant (13) et la formulation noyau du produit scalaire pour exprimer $k(f(x), f(x))$ à partir seulement de valeurs de noyau $k(y_i, y_j)$ entre éléments de l'ensemble d'apprentissage. Il est donc possible par exemple de mener une validation croisée (dans le contexte de l'élagage ou pour ajuster certains méta-paramètres de la méthode) sans avoir à calculer de pré-images.

2.4 Ensembles d'arbre à sortie noyau

Bien qu'utiles pour des questions d'interprétabilité, les arbres utilisés seuls sont généralement peu compétitifs avec d'autres méthodes en termes de précision, essentiellement à cause d'une variance importante. Dans le contexte de la classification et de la régression, les méthodes d'ensemble ont été proposées pour réduire la variance et donc améliorer la précision. En général, ces méthodes construisent un ensemble d'arbres

et ensuite fournissent une prédiction en combinant les prédictions des arbres de l'ensemble.

Parmi ces méthodes, celles qui sont basées sur le calcul de scores pour construire les arbres de l'ensemble et qui (dans le contexte de la régression) moyenne simplement les prédictions, peuvent être étendues avec des noyaux.

La prédiction d'un ensemble d'arbres \mathcal{T} , qui est une moyenne de somme telle (13), peut s'écrire comme une somme pondérée de vecteurs de l'espace caractéristique tirés de l'échantillon d'apprentissage :

$$\hat{\phi}_{\mathcal{T}}(x) = \sum_{i=1}^{N_{LS}} k_{\mathcal{T}}(x_i, x) \phi(y_i). \quad (15)$$

En effet, pour un arbre isolé t , (13) peut être réécrit comme :

$$\hat{\phi}_t(x) = \sum_{i=1}^{N_{LS}} k_t(x_i, x) \phi(y_i), \quad (16)$$

où $k_t(x, x') = N_L^{-1}$ si x et x' atteignent la même feuille L de t et 0 sinon ; la prédiction moyenne d'un ensemble d'arbres $\mathcal{T} = \{t_1, \dots, t_M\}$ est donnée par (15) avec

$$k_{\mathcal{T}}(x, x') = M^{-1} \sum_{i=1}^M k_{t_i}(x, x'). \quad (17)$$

Notons que $k_{\mathcal{T}}$ est positif et normalisé sur l'échantillon d'apprentissage ($\sum_i k_{\mathcal{T}}(x_i, x) = 1$). Il est strictement positif seulement pour des paires d'entrées qui atteignent la même feuille dans au moins un des arbres de l'ensemble. En fait, $k_{\mathcal{T}}$ est un noyau semi-défini positif construit sur l'espace d'entrées de manière supervisée lors de la construction de l'arbre (Geurts *et al.*, 2006).

La prédiction d'un ensemble est donc la pré-image approchée calculée de la manière suivante :

$$\begin{aligned} \hat{y}_{\mathcal{T}}(x) &= \arg \min_{y' \in \mathcal{Y}} \left\| \phi(y') - \sum_{i=1}^{N_{LS}} k_{\mathcal{T}}(x_i, x) \phi(y_i) \right\|^2 \\ &= \arg \min_{y' \in \mathcal{Y}} k(y', y') - 2 \sum_{i=1}^{N_{LS}} k_{\mathcal{T}}(x_i, x) k(y_i, y'). \end{aligned} \quad (18)$$

Comme pour un arbre isolé, nous simplifierons (18) en remplaçant la minimisation sur \mathcal{Y} par une minimisation sur les y_i de l'échantillon d'apprentissage tels que $k_{\mathcal{T}}(x_i, x)$ est non nul.

Dans nos expérimentations, nous comparerons OK3 avec des arbres isolés et avec des ensembles d'arbres obtenus avec la randomisation du bagging et des extra-trees. Le bagging construit chaque arbre à partir d'une échantillon bootstrap tiré de l'échantillon d'apprentissage original en optimisant localement la mesure de score lors du choix des questions de l'arbre (Breiman, 1996). La méthode extra-trees construit chaque arbre à

partir de l'échantillon d'apprentissage original en randomisant le choix des questions à chaque noeud. Le lecteur intéressé peut se reporter à (Geurts *et al.*, 2006) pour une description exacte de cet algorithme et sa comparaison avec d'autres méthodes d'ensemble.

2.5 Sélection et classement des attributs

Une caractéristique intéressante des méthodes d'arbre est qu'elles peuvent être exploitées pour classer les attributs selon leur importance pour prédire les valeurs de sortie. Le calcul de ce classement est particulièrement intéressant dans le contexte des méthodes d'ensemble qui autrement ne sont pas interprétables.

Dans le contexte de la méthodes OK3, nous proposons de calculer l'importance d'un attribut en sommant la réduction totale de variance apportée par chaque question (dans un arbre ou dans un ensemble d'arbres) où apparaît cet attribut. Cette réduction s'écrit $N.\text{Score}(S, T)$ pour un test T en un noeud correspondant à un sous-échantillon S . Un attribut qui n'apparaît pas dans un arbre recevra donc une importance nulle et les attributs qui apparaissent au sommet de l'arbre vont généralement obtenir des hauts scores. Étant donné que la variance peut être calculée uniquement sur base de valeurs du noyau, ce calcul ne fait à nouveau pas référence aux espaces de sortie \mathcal{Y} ou \mathcal{H} .

2.6 Apprentissage supervisé d'un noyau

Supposons maintenant que les sorties des exemples de l'échantillon d'apprentissage ne soient pas disponibles mais que les valeurs du noyau entre ces sorties hypothétiques soient données sous la forme d'une matrice de Gram K de dimension $N_{LS} \times N_{LS}$ avec $K_{i,j} = k(y_i, y_j)$. Étant donné que la construction d'un arbre requière uniquement les valeurs du noyau, nous pouvons toujours construire un arbre (ou un ensemble d'arbres) à partir de cette matrice. De plus, à partir de cette matrice uniquement, nous pouvons également faire une prédiction sur la valeur de noyau entre les sorties (inconnues) qui correspondraient à deux points de l'espace d'entrée.

En effet, considérons d'abord un arbre isolé et deux vecteurs d'entrée x_1 et x_2 atteignant les feuilles L_1 et L_2 respectivement. Si on avait accès aux sorties dans l'échantillon d'apprentissage, ces feuilles correspondraient respectivement à deux sous-échantillons de sorties $\{y_1^1, \dots, y_{N_{L_1}}^1\}$ et $\{y_1^2, \dots, y_{N_{L_2}}^2\}$. Si, de plus, on connaissait la projection ϕ , on pourrait calculer à partir de cette information les prédictions dans \mathcal{H} pour x_1 et x_2 ainsi que leur produit scalaire :

$$\hat{k}(x_1, x_2) = \frac{1}{N_{L_1} N_{L_2}} \sum_{i=1}^{N_{L_1}} \sum_{j=1}^{N_{L_2}} \langle \phi(y_i^1), \phi(y_j^2) \rangle. \quad (19)$$

Étant donné que cette dernière expression fait uniquement référence à des produits scalaires, elle peut être directement obtenues à partir de valeur de la matrice de Gram :

$$\hat{k}(x_1, x_2) = \frac{1}{N_{L_1} N_{L_2}} \sum_{i=1}^{N_{L_1}} \sum_{j=1}^{N_{L_2}} k(y_i^1, y_j^2). \quad (20)$$

Pour un ensemble d'arbres \mathcal{T} , cette prédiction s'écrit comme un produit de convolution de la matrice de Gram K par le noyau correspondant à l'ensemble d'arbres $k_{\mathcal{T}}$:

$$\hat{k}_{\mathcal{T}}(x_1, x_2) = \sum_{i=1}^{N_{LS}} \sum_{j=1}^{N_{LS}} k_{\mathcal{T}}(x_i, x_1) k_{\mathcal{T}}(x_j, x_2) K_{i,j}. \quad (21)$$

Évidemment, ce noyau approché sera semi-défini positif pour autant que la matrice de Gram K le soit. Ainsi, OK3 fournit un moyen de généraliser un noyau arbitraire comme une fonction de certaines variables d'entrée. Cette possibilité est particulièrement intéressante par exemple quand l'information sur la sortie est donnée directement sous la forme d'une similarité entre objets décrits dans un espace d'entrée et on désire généraliser cette mesure de similarité. Notre deuxième application montrera que le problème d'inférence de graphe supervisée peut être formulé de cette manière. Notre méthode permet également de résoudre le problème de complétion de noyau défini par (Tsuda *et al.*, 2003). Par rapport à l'approche transductive adoptée dans (Tsuda *et al.*, 2003), notre méthode construit un modèle sous la forme d'une fonction $\hat{k}_{\mathcal{T}}$ mais elle n'exploite pas les entrées des objets de test lors de l'apprentissage. Bien que ce ne soit pas illustré ici, mentionnons également comme autre application potentielle la possibilité de calculer une approximation d'un noyau dont l'évaluation serait particulièrement coûteuse en temps.

2.7 Questions d'implémentation

Dans le cas de la formulation (6), il est possible de calculer de manière incrémentale le score pour toutes les questions correspondant à une variable numérique et donc la détermination de la meilleure question parmi N est d'ordre $O(N)$ pour un échantillon de taille N . Avec un noyau de sortie, cette mise à jour incrémentale du score est toujours possible mais elle requière maintenant de l'ordre de $O(N)$ opérations, ce qui donne une complexité générale quadratique de la procédure de recherche de la meilleure question en un noeud. Cet accroissement de la complexité est inévitable dans le cas général. Cependant, s'il existe un espace caractéristique de dimension faible d correspondant au noyau, il peut être plus avantageux d'utiliser la méthode d'arbres de régression à sortie multiple qui est elle de complexité $O(N.d)$. Les deux approches produiront dans ce cas exactement les mêmes arbres.

Dans le cas général, il y a également un coût additionnel lié au calcul d'une prédiction par rapport aux arbres standards. Le calcul de la pré-image, ainsi que le calcul d'une prédiction sur le noyau, est d'ordre $O(N_L^2)$ pour un arbre isolé et quadratique dans le support de $k_{\mathcal{T}}(x, \cdot)$ pour un ensemble. Dans le cas d'un arbre isolé, cependant, il est possible de pré-calculer les prédictions à toutes les feuilles de l'arbre pour éviter l'effet quadratique lors du test. Étant donné que les méthodes d'ensemble sont généralement utilisées avec des arbres non élagués, on peut s'attendre également à ce qu'en pratique, la taille du support $k_{\mathcal{T}}(x, \cdot)$ soit quasiment indépendante de la taille de l'échantillon et donc que la complexité du calcul d'une prédiction reste bien en deçà de sa pire valeur $O(N_{LS}^2)$.

2.8 Liens avec d'autres travaux

OK3 est lié à plusieurs travaux dans le monde des arbres. A notre connaissance, les arbres de régression multiple datent des travaux de (Segal, 1992) qui a proposé de remplacer la distance euclidienne dans (4) par une distance de Mahalanobis. L'algorithme qui en découle est strictement équivalent à OK3 avec un noyau $k(y_i, y_j) = y_i V^{-1} y_j^T$ où V est une matrice de covariance et il se réduit aux arbres à sorties multiples lorsque V est la matrice identité. Notre méthode est également proche des arbres de clustering prédictif (PCT) proposés par (Blockeel *et al.*, 1998) et appliqués entre autres à la classification hiérarchique (Todorovski *et al.*, 2002) et le classement (Blockeel *et al.*, 2002). PCT généralise les arbres de régression en remplaçant la notion de variance par la forme générale $\frac{1}{N} \sum_{i=1}^N d(y_i, p)^2$ où d est une mesure de distance arbitraire et le prototype p est défini comme le point qui minimise cette dernière expression selon p . Quand d est la distance Euclidienne dans l'espace caractéristique de sortie, la méthode PCT produit les mêmes arbres que la méthode OK3. La différence principale est que PCT requière le calcul explicite d'un prototype p .

À côté des méthodes d'arbres, plusieurs travaux se sont focalisés récemment sur le problème de la prédiction d'une sortie complexe avec des méthodes à base de noyaux (Weston *et al.*, 2002; Tsochantaridis *et al.*, 2005; Weston *et al.*, 2005; Taskar *et al.*, 2005; Cortes *et al.*, 2005).

(Weston *et al.*, 2002) définissent un noyau sur la sortie et cherchent à trouver la prédiction qui minimise (1) avec la fonction de perte (8) associée au noyau. Leur méthode, appelée KDE ("Kernel Dependency Estimation"), consiste d'abord à appliquer une ACP non-linéaire sur le noyau de sortie de manière à réduire la dimension de l'espace caractéristique et ensuite à appliquer une "kernel ridge regression" sur chacune des directions de sortie. Dans (Cortes *et al.*, 2005), une nouvelle formulation de KDE est proposée qui ne requière plus la réduction de dimension et qui, comme OK3, travaille implicitement dans l'espace caractéristique complet. Par rapport à notre méthode, l'utilisation de la "kernel ridge regression" bénéficie de l'utilisation d'un noyau sur l'espace d'entrée qui peut potentiellement améliorer la précision, au détriment cependant de l'interprétabilité et des temps de calcul.

L'approche adoptée dans (Tsochantaridis *et al.*, 2005; Weston *et al.*, 2005; Taskar *et al.*, 2005) consiste à exploiter un noyau défini directement sur les paires entrée-sortie avec des méthodes à vastes marges. Contrairement à OK3 ou KDE, ce noyau donne lieu à une représentation combinée des entrées et des sorties dans un même espace caractéristique, ce qui peut s'avérer avantageux lorsque les espaces d'entrée et de sortie sont interdépendants. Ces méthodes sont donc capables de traiter des problèmes plus complexes au prix cependant d'une augmentation des temps de calcul, partiellement due aux calculs des pré-images lors de l'apprentissage.

Notre méthode peut également être rapprochée de la méthode des k-moyennes à noyau (Dhillon *et al.*, 2004). Les deux méthodes minimisent la même fonction de perte et sont basées sur la même astuce, c'est-à-dire le calcul de la distance moyenne au centre de masse à partir d'un noyau. La différence principale est évidemment que la méthode des k-moyennes à noyau minimise ce critère de manière non supervisée alors que notre algorithme le fait de manière supervisée à partir des variables d'entrée.

3 Expérimentations

Notre algorithme étant, à strictement parlé, une généralisation des arbres de classification et de régression, nous savons déjà qu'il fonctionne plutôt bien dans ces cas (particulièrement en combinaison avec des ensembles). Dans cette section, nous nous focaliserons donc sur des problèmes d'espaces de sortie structurés et d'apprentissage de noyaux non linéaires.

3.1 Reconstruction d'image

Pour donner une première illustration de la méthode, nous reproduisons ici l'expérience de reconstruction d'image de (Weston *et al.*, 2002). Le problème est de prédire la moitié inférieure d'une image représentant un chiffre manuscrit à partir de sa moitié supérieure. La base de données que nous avons utilisée constitue un sous-ensemble de 1000 images de 16×16 pixels extraites de la base de données USPS¹. Nous avons pris les 100 premiers exemples de chaque chiffre dans l'ordre original de la base de données. Les 8 premières lignes (128 variables) sont utilisées comme entrées et les 8 dernières lignes constituent la sortie. Comme noyau de sortie, nous avons utilisé un noyau basé sur une fonction à base radiale (RBF) $k(y, y') = \exp(-\|y - y'\|^2 / 2\sigma^2)$ avec $\sigma = 7.0711$ ². Comme (Weston *et al.*, 2002), nous comparons les différents algorithmes avec une validation croisée en 5 parties³.

Trois familles de méthodes sont comparées : k -NN, kernel dependency estimation (KDE, (Weston *et al.*, 2002)), et OK3⁴. Pour le k -NN, nous recherchons les plus proches voisins en utilisant un noyau RBF sur les entrées et la prédiction est calculée comme la sortie de l'échantillon d'apprentissage la plus proche du centre de masse calculé sur les k voisins. Pour KDE, nous comparons un noyau linéaire et un noyau RBF sur les entrées. Les paramètres de ces deux méthodes sont sélectionnés par une nouvelle validation croisée en 5 parties à l'intérieure de la précédente validation croisée⁵. Pour OK3, nous comparons des arbres isolés (non élagués⁶), le bagging, et la méthode extra-trees. Nous avons utilisé des ensembles de 100 arbres et les valeurs de paramètres par défaut de la méthode extra-trees (Geurts *et al.*, 2006).

Pour chaque méthode, nous avons calculé l'erreur moyenne correspondant à la fonction de perte du noyau RBF, c'est-à-dire $\ell(y, y') = 2(1 - \exp(-\|y - y'\|^2 / 2\sigma^2))$, qui

¹La base de données "ZIP code" issue de <http://www-stat-class.stanford.edu/~tibbs/ElemStatLearn/data.html>

²Cette valeur a été choisie de manière à ce que nos erreurs soient du même ordre de grandeur que les erreurs reportées dans (Weston *et al.*, 2002). Elle est potentiellement différente de la valeur exacte utilisée dans cette dernière publication.

³Pour faciliter la reproduction de nos résultats, les 5 parties sont stratifiées selon la classe et choisie dans l'ordre original des 1000 images.

⁴Pour k -NN et KDE, nous avons utilisé l'implémentation matlab de ces méthodes dans Spider (<http://www.kyb.tuebingen.mpg.de/bs/people/spider/>). Pour OK3, nous avons utilisé notre propre implémentation en C.

⁵ k est sélectionné dans $\{1, 5, 10, 15, 20\}$, σ du noyau RBF dans $\{22.3607, 7.0711, 2.2361, 0.7071, 0.2236, 0.0707, 0.0224\}$, et le paramètre de régularisation de KDE dans $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1\}$.

⁶L'élagage n'apporte pas d'amélioration dans ce cas.

TAB. 1 – Comparaison de différentes méthodes sur le problème de reconstruction d'images (erreur RBF)

Méthode	$N_{LS} = 200$	$N_{LS} = 800$
Base	$1,0945 \pm 0,0125$	$1,0853 \pm 0,0227$
Borne inf.	$0,4701 \pm 0,0064$	$0,3584 \pm 0,0118$
k -NN	$0,8587 \pm 0,0172$	$0,7501 \pm 0,0316$
KDE linéaire	$0,8630 \pm 0,0030$	$0,7990 \pm 0,0215$
KDE RBF	$0,7892 \pm 0,0066$	$0,6778 \pm 0,0264$
OK3+arbre isolé	$1,0399 \pm 0,0150$	$0,9013 \pm 0,0232$
OK3+Bagging	$0,8643 \pm 0,0096$	$0,7337 \pm 0,0263$
OK3+Extra-trees	$0,8169 \pm 0,0109$	$0,6949 \pm 0,0261$

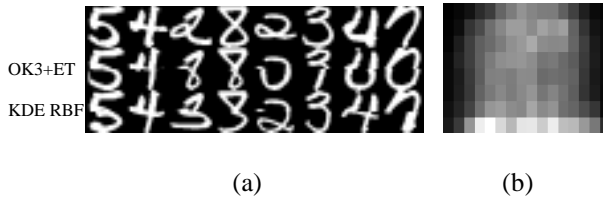


FIG. 1 – (a) Exemples de prédictions et (b) importance des variables sur le problème de reconstruction d'images

est l'erreur que KDE et OK3 tentent explicitement de minimiser. Les résultats sont rassemblés dans la table 1. La première (resp. deuxième) colonne correspond au cas où on a utilisé des sous-échantillons de taille 200 (resp. 800) pour l'apprentissage et de taille 800 (resp. 200) pour le test. La première ligne correspond à l'erreur de base obtenue par un modèle qui produit en sortie pour toute image de test la partie inférieure de l'image d'apprentissage qui est la plus proche du centre de masse de l'échantillon d'apprentissage complet dans l'espace caractéristique. Toutes les méthodes comparées renvoient comme prédiction une sortie présente dans l'échantillon d'apprentissage, ce qui impose une borne inférieure sur l'erreur pour un échantillon de test donné. Cette borne inférieure est atteinte quand la sortie prédite est choisie comme la partie inférieure de toutes les images d'apprentissage qui est la plus proche (au sens du noyau RBF) de la partie inférieure de l'image de test. Cette borne est donnée à la deuxième ligne de la table.

A partir de cette table, nous pouvons observer que pour les deux tailles d'échantillon, toutes les méthodes sont meilleures que l'erreur de base. Elles restent cependant sous-optimales par rapport à la borne inférieure. Parmi les méthodes d'arbres, un arbre seul est à peine meilleur que la prédiction de base. Le bagging et la méthode extra-trees améliore très significativement la précision avec un avantage clair à cette dernière méthode. Le meilleur résultat est néanmoins obtenu par KDE avec un noyau RBF, c'est-à-dire en utilisant le même noyau sur l'espace d'entrée et l'espace de sortie. Il est cependant intéressant de remarquer que la méthode OK3 avec des extra-trees est capable

d'atteindre des résultats presque aussi bons alors que cette méthode doit apprendre son noyau à partir de l'échantillon. La figure 1(a) donne quelques exemples de prédictions données par KDE+RBF et OK3+Extra-trees pour $N_{LS} = 800$.

Pour illustrer la possibilité de classer les entrées, la figure 1(b) représente l'importance de chaque pixel d'entrée obtenue par la méthode extra-trees à partir de l'échantillon complet de 1000 images. Les pixels inutiles (d'importance nulle) sont représentés en noir et les pixels les plus importants en blanc. Cette figure montre que l'information est concentrée dans la partie inférieure des images d'entrée, c'est-à-dire précisément à l'endroit où elles doivent s'aligner avec les images de sortie.

3.2 Inférence de graphe supervisée

Définition du problème.

Soit $G = (V, E)$ un graphe avec des sommets V et des arcs $E \subset V \times V$. Nous supposons que chaque sommet est décrit par des variables dans un espace d'entrée \mathcal{X} , et notons $x(v)$ cette information pour un sommet v . Soit $V' \subset V$ et $E' = \{(v, v') \in E | v, v' \in V'\}$; $G' = (V', E')$ est donc un sous-graphe de G . Nous supposons que le graphe est non dirigé, c'est-à-dire $(v, v') \in E \Rightarrow (v', v) \in E$. Le but de l'inférence de graphe supervisée est de déterminer à partir de la connaissance de G' une fonction $e(x(v), x(v')) : V \times V \rightarrow \{0, 1\}$, idéalement telle que $e(x(v), x(v')) = 1 \Leftrightarrow (v, v') \in E$. À notre connaissance, cette tâche a été introduite pour la première fois dans (Vert & Yamanishi, 2004) pour l'inférence de réseaux biologiques mais elle peut être instanciées dans de nombreux domaines d'application.

Pour résoudre ce problème avec OK3, nous devons d'abord définir un noyau de sortie $k(v, v')$ entre sommets du graphe tel que deux sommets adjacents correspondent à une valeur de k importante et des sommets non adjacents à des valeurs faibles. On peut alors appliquer la méthode OK3 sur le sous-graphe connu, en utilisant l'ensemble de sommets V' décrits par leurs entrées et la matrice de Gram $k(v, v')$. Elle produira un modèle permettant de prédire, à partir de l'équation (21), le noyau entre deux sommets quelconques comme une fonction de leurs entrées. En seuillant les valeurs du noyau ainsi appris, on pourra inférer les arcs entre de nouveaux sommets non vus lors de l'apprentissage.

Inférence d'un réseau d'enzyme.

Nous reproduisons ici l'expérience de (Yamanishi *et al.*, 2005) qui vise à reconstruire un réseau d'enzyme chez la levure. Chaque sommet du réseau correspond à un enzyme et un arc connecte deux enzymes qui catalysent deux réactions successives le long d'une voie métabolique. Le réseau de référence utilisé dans (Yamanishi *et al.*, 2005)⁷ est extrait de la base de données KEGG/PATHWAY. Il est composé de 668 enzymes connectés par 2782 arcs et est considéré comme une sous-réseau fiable du réseau d'enzyme complet de la levure. Chaque enzyme est décrit par trois types de variables : (1) *données d'expression* : 157 variables numériques mesurant l'expression du gène

⁷<http://web.kuicr.kyoto-u.ac.jp/~yoshi/ismb05/>

codant l'enzyme dans différentes expériences de puce à ADN ; (2) *données de localisation* : 23 variables booléennes codant l'absence ou la présence de l'enzyme dans un compartiment intracellulaire particuliers ; (3) *données phylogénétiques* : 145 variables booléennes codant l'absence ou la présence d'une protéine orthologue dans 145 organismes. Pour représenter la structure de graphe, nous avons utilisé un noyau de diffusion (Kondor & Lafferty, 2002), qui correspond à une matrice de Gram $K = \exp(-\beta L)$, où $L = D - A$ est le Laplacien du graphe, D la matrice diagonale qui contient le degré de chaque sommet, et A la matrice d'adjacence. La valeur de β a été fixée à 1. Notre méthode est évaluée par une validation croisée en 10 parties : à chaque exécution, la matrice de Gram est calculée sur 9 parties, OK3 est appliquée sur cette matrice de Gram, et toutes les prédictions de noyaux impliquant au moins un enzyme de la partie restante sont calculées⁸. Un réseau peut alors être retrouvé en connectant les paires d'enzymes correspondant à une valeur de noyau supérieure à un seuil. Pour évaluer la précision de ces prédictions, nous analysons les courbes ROC obtenues en faisant varier ce seuil, le taux de vrai positif étant la proportion d'arcs existants correctement prédits et le taux de faux positif la proportion d'arcs inexistant dans le réseau de référence prédits erronément par la méthode. Les courbes correspondants aux différentes parties de la validation croisée ont été moyennées verticalement et nous avons également calculé les valeurs moyennes des aires en dessous de la courbe ROC (AUC).

Le graphe de gauche de la figure 2 compare OK3 avec différentes méthodes d'arbres en utilisant les 325 attributs en entrée. On observe qu'un arbre seul est clairement battu par le bagging qui est lui-même inférieur aux extra-trees. Une comparaison de la valeur d'AUC de 0,845 obtenue avec les extra-trees avec la meilleure valeur d'AUC de 0,804 obtenue par (Yamanishi *et al.*, 2005) avec les mêmes entrées⁹ nous permet de conclure que nos résultats sont très compétitifs.

Le graphe de droite de la figure 2 reproduit une autre expérience de (Yamanishi *et al.*, 2005), qui vise à évaluer l'effet de l'utilisation de différents sous-ensembles d'attributs. Le profil phylogénétique apparaît comme l'information la plus pertinente, alors que les données de localisation seules semblent ne pas être très informatives. Ce classement des variables est également confirmé par le classement obtenu par la mesure d'importance qui place 5 variables phylogénétiques au sommet alors que les 20 dernières variables sont toutes des variables liées à la localisation. En comparant notre courbe ROC pour les données phylogénétiques avec celle de (Yamanishi *et al.*, 2005), nous observons ici une bien meilleure valeur d'AUC, qui montre que la méthode OK3 exploite plus efficacement ce type de données. Nous pensons qu'une explication de ce phénomène est que les méthodes d'arbres sont naturellement mieux adaptées au traitement des variables discrètes telles que celles encodant l'information phylogénétique.

Discussion.

Les méthodes proposées dans (Yamanishi *et al.*, 2004) et (Vert & Yamanishi, 2004) déterminent une projection des entrées $x(v)$ vers un vecteur $f(x(v))$ de \mathbb{R}^L tel que

⁸Toutes ces conditions d'expérimentation, excepté le découpage exacte pour la validation croisée, sont identiques aux conditions d'expérimentation de (Yamanishi *et al.*, 2005).

⁹La meilleure AUC obtenue par (Yamanishi *et al.*, 2005) est en fait 0,836 mais cette dernière valeur est obtenue à partir d'information chimique non exploitées ici.

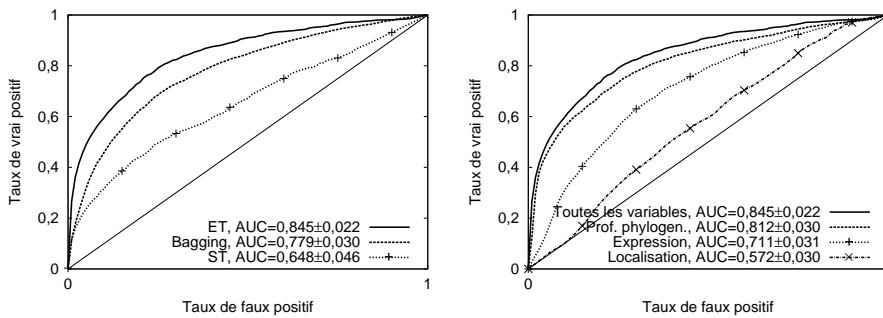


FIG. 2 – À gauche, les courbes ROC d'OK3 avec une arbre isolé (ST), le bagging et les extra-trees (ET), à droite, les courbes ROC utilisant différents jeux d'attributs avec les extra-trees.

les projections $f(x(v))$, $f(x(v'))$ de deux sommets v, v' sont proches lorsque ces deux sommets sont connectés dans G' . Ces projections sont ensuite utilisées pour prédire de nouveaux arcs. Dans (Yamanishi *et al.*, 2004), la projection f est déterminée par les premières composantes obtenues par une analyse de corrélation canonique à noyau entre un noyau sur les entrées et le noyau de diffusion sur le graphe. Dans (Vert & Yamanishi, 2004), la projection f est apprise directement en minimisant une fonctionnelle qui exprime le fait qu'une paire de sommets connectés doit être proche dans l'espace caractéristique.

Bien que nous reprenions le problème général et le protocole d'expérimentation de ces travaux, notre approche diffère de ces derniers par le fait qu'elle calcule l'approximation du noyau directement dans l'espace original, c'est-à-dire sans inférer explicitement la projection f dans une espace Euclidien de dimension spécifiée a priori et sans avoir à définir a priori un noyau sur l'espace d'entrée. En général, l'avantage de notre approche est qu'elle est plus directe, puisque totalement automatique, et qu'elle peut également améliorer la précision, comme nous l'avons observé.

4 Conclusion et perspectives

L'utilisation de noyau en sortie des méthodes d'arbre comme nous le proposons dans ce papier ouvre ces méthodes au problème de prédiction de sorties structurées et à l'apprentissage supervisé d'un noyau. Cette extension nous a permis d'attaquer deux problèmes de natures très différentes : un problème de complétion de motif and un problème d'inférence de graphe. La combinaison de la méthode avec des méthodes d'ensemble permet d'atteindre une bonne précision tout en conservant la possibilité de classer et de filtrer les variables d'entrée. L'interprétabilité, la nature non paramétrique et la complexité raisonnable de cette méthode en fait une alternative intéressante aux méthodes à base de noyaux et enrichit le panel des méthodes capables de traiter des sorties structurées.

En termes de perspectives, nous nous focaliserons sur l'application de cette méthode à différentes tâches, ce qui nous obligera également à nous pencher sur la question du choix d'un noyau de sortie approprié pour un problème donné. Bien que nous nous soyons ici limités aux méthodes d'ensemble basées sur la randomisation, la même extension peut s'appliquer aux méthodes d'ensemble basées sur l'idée du boosting et mérite d'être étudiée. D'un point de vue plus général, le problème de la prédiction d'une sortie structurée en est seulement à ses balbutiements en apprentissage et il reste de nombreuses questions ouvertes dans ce domaine telles que par exemple l'impact du noyau de sortie ainsi que du choix de l'espace d'entrée sur la capacité de généralisation. Ces questions méritent certainement d'être explorées en général et en particulier dans le contexte des arbres à sortie noyau.

Remerciements

Les auteurs remercient Y. Yamanishi pour avoir fourni les données du réseau d'enzymes. Pierre Geurts est postdoctorant auprès du CNRS et collaborateur scientifique auprès du FNRS (Belgique). Florence d'Alché-Buc remercie Genopole qui finance ses activités de recherche dans le cadre d'un financement ATIGE.

Références

- BLOCKEEL H., BRUYNNOOGHE M., DZEROSKI S., RAMON J. & STRUYF J. (2002). Hierarchical multi-classification. In *KDD-2002 Workshop Notes : MRDM 2002, Workshop on Multi-Relational Data Mining*, p. 21–35.
- BLOCKEEL H., DE RAEDT L. & RAMON J. (1998). Top-down induction of clustering trees. In *Proceedings of ICML 1998*, p. 55–63.
- BREIMAN L. (1996). Bagging predictors. *Machine Learning*, **24**(2), 123–140.
- BREIMAN L., FRIEDMAN J., OLSEN R. & STONE C. (1984). *Classification and Regression Trees*. Wadsworth International (California).
- CORTES C., MEHRYAR M. & WESTON J. (2005). A general regression technique for learning transductions. In *Proceedings of ICML 2005*, p. 153–160.
- DHILLON I., GUAN Y. & KULLIS B. (2004). *Kernel k-means, Spectral clustering and normalized cuts*. Rapport interne, University of Austin, Texas.
- GEURTS P., ERNST D. & WEHENKEL L. (2006). Extremely randomized trees. *To appear in Machine Learning*.
- KONDOR R. & LAFFERTY J. (2002). Diffusion kernels on graphs and other discrete input. In *Proc. of the 19th International Conference on Machine Learning*, p. 315–322.
- RALAIIVOLA L. & D'ALCHÉ-BUC F. (2003). Dynamical modeling with kernels for nonlinear time series prediction. *Advances in Neural Information Processing Systems*.
- SCHOLKOPF B. & SMOLA A. (2002). *Learning with kernels*. MIT Press.
- SEGAL M. (1992). Tree structured methods for longitudinal data. *Journal of the American Statistical Association*, **87**, 407–418.

- TASKAR B., CHATALBASHEV V., KOLLER D. & GUESTRIN C. (2005). Learning structured prediction models : A large margin approach. In *Proc. of the 22nd International Conference on Machine Learning*, p. 897–904.
- TODOROVSKI L., BLOCKEEL H., & DZEROSKI S. (2002). Ranking with predictive clustering trees. In *Proc. of the 13th European Conference on Machine Learning*, volume LNAI 2430, p. 444–456.
- TSOCHANTARIDIS I., JOACHIMS T., HOFMANN T. & ALTUN Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, **6**, 1453–1484.
- TSUDA K., AKAHO S. & ASAI K. (2003). The em algorithm for kernel matrix completion with auxiliary data. *Journal of machine learning research*, **4**, 67–81.
- VERT J.-P. & YAMANISHI Y. (2004). Supervised graph inference. *Advances in Neural Information Processing Systems*, **17**, 1433–1440.
- WESTON J., CHAPPELLE O., ELISSEEFF A., SCHOELKOPF B. & VAPNIK V. (2002). Kernel dependency estimation. *Advances in Neural Information Processing Systems*, **15**.
- WESTON J., SCHOELKOPF B. & BOUSQUET O. (2005). Joint kernel maps. In J. CABESTANY, A. PRIETO & F. SANDOVAL, Eds., *Proceedings of the 8th International Work-Conference on Artificial Neural Networks*, volume LNCS 3512, p. 176–191.
- YAMANISHI Y., VERT J.-P. & KANEHISA M. (2004). Protein network inference from multiple genomic data : a supervised approach. *Bioinformatics*, **20**, i363–i370.
- YAMANISHI Y., VERT J.-P. & KANEHISA M. (2005). Supervised enzyme network inference from the integration of genomic data and chemical information. *Bioinformatics*, **21**, i468–i477.