Apprentissage multi-tâches par *boosting* d'arbres de décision

Jean-Baptiste Faddoul[1], Boris Chidlovskii[2], Fabien Torre[1], Rémi Gilleron[1]

[1] Inria Lille - Nord Europe
40, avenue Halley, Park Plaza, 59650 Villeneuve d'Ascq, France
jean.faddoul@gmail.com

[2] Xerox Research Center Europe
6 chemin Maupertuis, 38240 Meylan, France
chidlovskii@xrce.xerox.com

Résumé : Nous nous intéressons à l'apprentissage multi-tâches dans le cas où il n'y a pas de correspondance entre les labels des différentes tâches. Apprendre simultanément plusieurs tâches, en exploitant ce qui les lie, permet d'accroître les performances prédictives sur chaque tâche. Nous proposons d'enrichir les apprenants faibles de l'environnement *Adaboost Multi-Tâches* avec des algorithmes d'apprentissage d'arbres de décision multi-tâches. Nous adaptons tout d'abord l'apprentissage d'arbres de décision pour le cas multi-tâches. Dans ce contexte, nous prouvons une intéressante caractéristique de la mesure du gain d'information. Nous utilisons cette propriété pour définir un critère original pour apprendre des arbres de décision multi-tâches. Ce critère guide la construction de l'arbre à partir d'exemples des différentes tâches, tout en modélisant les différents degrés de liaison entre les tâches. Nous modifions `MT-Adaboost` en vue d'utiliser les arbres de décision multi-tâches comme hypothèses faibles. Nous validons expérimentalement l'intérêt de notre nouvelle méthode, nous rapportons les résultats obtenus sur trois jeux de données incluant *les mails Enron* et *une collection de détection de spams*.

Mots-clés : Apprentissage multi-tâche, arbres de décision, gain d'information, *boosting*, algorithme AdaBoost.

1 Introduction

L'apprentissage multi-tâches (Caruana, 1997) vise à améliorer les performances sur différentes tâches, en apprenant un modèle représentant l'information commune partagée par les tâches. Classiquement, les techniques existantes supposent que les tâches partagent soit les mêmes exemples soit les mêmes labels : ces hypothèses ont été mises en œuvre en classification (Xue *et al.*, 2007) et en *feature learning* (Argyriou *et al.*, 2006).

Cependant, dans beaucoup de contextes réels, ces hypothèses ne sont pas vérifiées, c'est le cas par exemple pour le rangement automatique de pages web dans un annuaire hiérarchique comme DMOZ ou Yahoo (Liu *et al.*, 2005). Quand on construit un classifieur dédié à l'organisation Yahoo, il peut être profitable de connaître les choix faits dans la hiérarchie DMOZ. Les deux tâches sont clairement liées, mais leurs labels ne sont pas les mêmes puisque dossiers et sous-dossiers ne sont pas les mêmes dans les deux hiérarchies. De plus, toutes deux évoluent avec le temps : de nouvelles catégories sont ajoutées et d'autres disparaissent par manque d'intérêt. Un autre exemple, qui motive notre travail, est celui de la classification automatique de mails personnels dans différents dossiers (Mantrach & Renders, 2012). Comme dans le cas des annuaires du web, les dossiers sont différents d'un utilisateur à l'autre mais il y a sans doute avantage à apprendre les deux tâches simultanément.

L'apprentissage multi-tâches sans correspondance entre labels a été considéré dans (Quadrianto

et al., 2010), où le problème a été reformulé comme un apprentissage pour chaque tâche d'un estimateur à maximum d'entropie tout en maximisant l'information mutuelle entre les ensembles de labels. Cette approche suppose une même relation entre les tâches dans la totalité de l'espace des exemples. Cette hypothèse d'une relation globale apparaît souvent trop forte. Les relations entre tâches peuvent prendre différentes formes avec différentes intensités selon les régions de l'espace des exemples. Par suite, il est impératif d'avoir des apprenants qui prennent en compte ces différentes formes, détectent les différentes intensités et s'adaptent en conséquence.

Dans un article précédent (Faddoul *et al.*, 2010), nous proposions une méthode qui permet de relâcher les hypothèses que nous venons d'évoquer sur le partage d'exemples et de labels entre tâches, ainsi que l'hypothèse de relation globale. Pous cela, nous avions conçu un algorithme d'apprentissage multi-tâches nommé `MT-Adaboost` qui étend à l'apprentissage multi-tâches l'algorithme de *boosting* Adaboost (Freund & Schapire, 1996; Schapire & Singer, 1999). Le *boosting* est une technique pour produire puis combiner des classifieurs faibles et ainsi améliorer l'efficacité prédictive du classifieur appris. Comme hypothèses faibles, nous définissions et utilisions des *stumps multi-tâches*, lesquels sont des arbres dans lesquels chaque nœud est un *stump* dédié à une unique tâche et, effectivement, nous montrions que cette forme d'hypothèses alliée au *boosting* permet de capturer des relations locales entre les tâches. Cependant, cette méthode présente certaines limitations. L'algorithme qui recherche un *stump* multi-tâche est glouton : il opère niveau par niveau, dirigé par une heuristique qui à chaque fois choisit le meilleur *stump* pour la tâche la plus facile. Dans cet enchaînement, à chaque étape un classifieur dédié à une tâche est appris, et cet apprentissage est influencé par les choix effectués précédemment sur le chemin pour d'autres tâches. Malheureusement, cette conception séquentielle et la méthode gloutonne mise en œuvre échouent parfois à capturer les relations entre tâches. De plus, les *stumps* multi-tâches sont des classifieurs binaires et leur extension au cas multi-classes ne va pas de soi.

Dans le présent article, nous proposons une nouvelle technique pour l'apprentissage multi-tâches qui affronte les limitations que nous venons d'évoquer. Tout d'abord, nous proposons en lieu et place des *stumps*, des hypothèses appelées *Multi-Task Decision Tree* (`MT-DT`). Pour les apprendre, nous revisitons le bien connu algorithme *C4.5* et l'adaptons au cadre multi-tâches. Les arbres de décision sont par nature des classifieurs multi-classes, propriété qui sera héritée par nos `MT-DT`s. Une contribution importante de notre travail est la démonstration que l'apprentissage des `MT-DT`s peut bénéficier d'une nouvelle mesure de gain d'information dédiée aux données multi-tâches. Au contraire des *stumps* multi-tâches, la mesure et le critère associé que nous proposons dans cet article pour sélectionner les tests placés aux nœuds des arbres de décision impliquent bien simultanément les données en provenance des différentes tâches.

Nous poursuivons en plongeant les `MT-DT`s dans le cadre du boosting et modifions notre algorithme `MT-Adaboost` pour prendre en compte des tâches non binaires, c'est-à-dire multi-classes. Nous nous inspirons ici des travaux de Schapire & Singer (1999), lesquels ont analysé Adaboost avec des apprenants faibles et des variations qui permettent de traiter les étiquetages multi-classes. Notre modification de `MT-Adaboost` peut être vu comme une adaptation de leur algorithme *Adaboost.M1*.

Dans la section suivante, nous formalisons l'apprentissage multi-tâches, introduisons les arbres de décision multi-tâches, prouvons une propriété de la mesure du gain d'information dans le cas de l'apprentissage multi-tâches et enfin dérivons de ce résultat un nouveau critère pour l'apprentissage des `MT-DT`s. Dans la Section 3 , nous présentons notre cadre pour l'apprentissage multi-tâches par *boosting* avec les `MT-DT`s comme hypothèses faibles. Finalement, nous décrivons les résultats expérimentaux à la Section 4 et concluons Section 5.

2 Apprentissage multi-tâches

2.1 Notations et définitions

Classiquement, une tâche de classification supervisée T est définie sur un espace d'exemples $\mathcal{X}$ et un espace de labels $\mathcal{Y}$. Soient D une distribution sur $(\mathcal{X}, \mathcal{Y})$ et $f : \mathcal{X} \rightarrow Y$ une fonction cible. Étant donné un ensemble d'exemples destiné à l'apprentissage $S = \{(x_i, y_i) \mid x_i \in \mathcal{X},\ y_i = f(x_i),\ 1 \leq i \leq m\}$, l'objectif est de découvrir une hypothèse h qui minimise une mesure d'erreur définie sur D par erreur$(h) = Pr_{<x,y>\sim D}[h(x) \neq y]$.

Nous considérons maintenant N tâches de classification $T_1, \ldots, T_N$ sur le même espace d'exemples $\mathcal{X}$ et des ensembles de labels $Y_1, \ldots, Y_N$, où les ensembles Y_i sont corrélés mais distincts : $Y_i \cap Y_j = \emptyset$ pour $i \neq j$. Nous souhaitons résoudre les N tâches de classification, simultanément pour bénéficier d'un gain en prédiction. Nous supposons une distribution D^N sur $\mathcal{X} \times \{1, \ldots, N\}$ et que, pour tout $j \in \{1, \ldots, N\}$, la projection de la distribution sur la j^{e} composante correspond à la distribution originale pour la tâche T_j. Un algorithme d'apprentissage multi-tâche prendra en entrée un ensemble d'exemples $S = \{< x_i, y_i, j >|\ x_i \in \mathcal{X},\ y_i = f_j(x_i) \in Y_j,\ j \in \{1, \ldots, N\},\ 1 \leq i \leq m\}$. Notons qu'une même instance x peut apparaître plusieurs fois dans S, pour différentes tâches, avec les labels correspondants. Le but est de trouver une hypothèse $h : \mathcal{X} \rightarrow Y_1 \times \ldots \times Y_N$ qui minimise erreur$(h) = Pr_{<x,y,j>\sim D^N}[h_j(x) \neq y]$, où $h_j(x)$ est la j^{e} composante de $h(x)$ pour $j \in \{1, \ldots, N\}$.

2.2 Arbres de décision pour les problèmes multi-tâches

L'apprentissage d'arbres de décision est un sujet largement étudié dans le domaine des statistiques, comme dans celui de l'apprentissage automatique. Il s'agit d'utiliser un arbre de décision comme modèle prédictif qui associe des valeurs cibles aux observations issues d'un espace d'instances. Dans le cas de la classification, chaque nœud interne de l'arbre est un test portant sur un attribut du problème, une branche s'interprète comme la conjonction des tests des nœuds qu'elle regroupe et se termine par une feuille qui contient le label de la classe prédite.

Dans la famille des algorithmes de type C4.5 (Quinlan, 1993), l'apprentissage d'un tel arbre de décision utilise la notion de *gain d'information* (IG), issue de la théorie de l'information. À la racine de l'arbre, l'algorithme choisit le test qui apporte le plus grand IG sur l'ensemble d'apprentissage. Un tel test partitionne les exemples disponibles pour l'apprentissage en deux sous-ensembles sur lesquels l'entropie est la plus faible. L'algorithme applique ensuite cette heuristique récursivement sur chaque sous-ensemble, jusqu'à obtenir un nœud ne contenant que des exemples partageant le même label qui devient alors une feuille concluant sur cette classe (nous ne discutons pas ici des techniques d'élagage qui peuvent être utilisées pour obtenir un arbre plus concis et plus général).

Le gain d'information au sujet d'une variable aléatoire Y que l'on obtient en observant une variable aléatoire X prendre la valeur $X = x$ peut être mesurée par la divergence $D_{KL}(p(Y|X)||p(Y))$ entre l'apriori $p(Y)$ et $p(Y|X)$, la distribution a posteriori de Y étant donné X. Le gain d'information permet d'estimer une amélioration moyenne et définit une séquence de tests à privilégier. Ainsi les algorithmes d'apprentissage d'arbres de décision utilise récursivement ce principe en sélectionnant le test avec le gain d'information le plus important et en partitionnant les exemples en fonction de ce test.

Dans cet article, nous adaptons l'apprentissage d'arbre de décision fondé sur une mesure IG au cadre multi-tâches. Une première différence manifeste entre le mono-tâche et le multi-tâches est que les arbres de décision habituels ne peuvent convenir en présence de plusieurs tâches : dans un tel arbre les nœuds internes sont des tests qui guident le processus de décision, jusqu'à une feuille qui

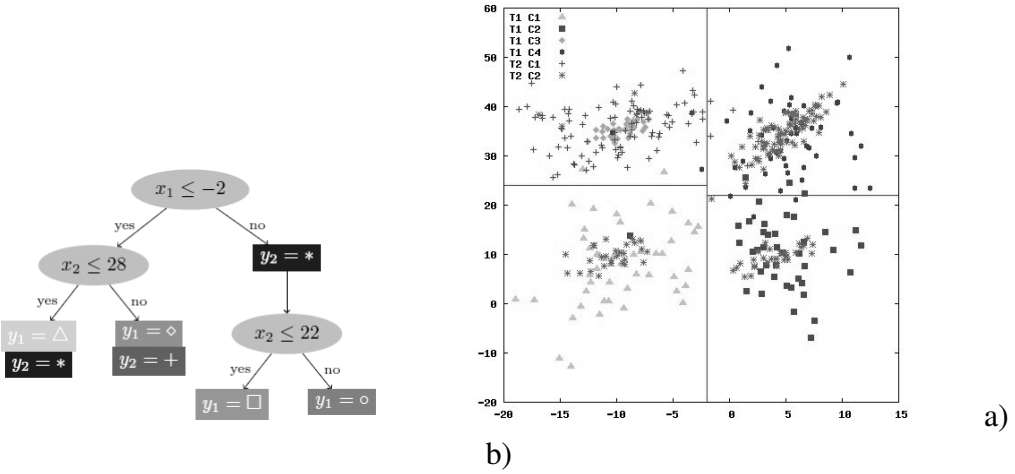

FIGURE 1 – a) Exemple de MT−DT, b) Deux tâches, mélanges de gaussiennes.

correspond à la décision finale.

En présence de plusieurs tâches, il faut envisager qu'un test oriente plusieurs tâches à la fois et que les décisions pour chaque tâche puissent être prises à des moments différents, en particulier dans des nœuds internes de l'arbre et plus seulement aux feuilles. La structure des *multi-task decision tree* (MT−DT) que nous proposons permet cela : une décision finale pour une tâche peut être prise dans un nœud interne de l'arbre et pas nécessairement dans une feuille. Cela se produit quand un test à un nœud fournit assez d'informations pour classer les exemples pour une certaine tâche T, mais que le sous-arbre sous ce nœud est nécessaire pour classer ces mêmes exemples pour les autres tâches.

La Figure 1.a donne l'exemple d'un MT−DT dédié à deux tâches, T_1 et T_2, produits par un mélange de gaussiennes 2D (voir la Figure 1.b). T_2 a deux labels et T_1 quatre, deux d'entre eux ($\square$ et $\diamond$) sont liés avec le label $+$ de T_2 et les deux autres ($\triangle$ et $\circ$) sont corrélés au second label ($*$) de T_1. Sur cet exemple, le MT−DT présente un nœud à décision précoce pour la tâche T_2 car savoir que la valeur de l'attribut x_1 est plus grande que -2 est suffisante pour prédire la classe $*$ pour T_2.

Au delà de la différence structurelle, la difficulté principale pour passer du mono-tâche au multi-tâches est d'adapter au mieux la mesure du gain d'information. Dans la section suivante, nous montrons comment réaliser cette adaptation tout en la fondant théoriquement. Ce critère amélioré, associé à une technique de *boosting* nous permettront d'obtenir un gain significatif lors de nos expériences.

2.3 Gain d'information et multi-tâches

Comme nous l'avons indiqué, l'apprentissage d'arbres de décision est basé un critère entropique, et plus particulièrement sur la mesure de dépendance mutuelle entre un ensemble de labels Y et un attribut observé X. Le gain d'information, dénoté $IG(Y;X)$, peut être exprimé comme suit :

$$IG(Y;X) = H(Y) - H(Y|X), \tag{1}$$

où $H(Y) = -\sum_{y \in Y} p(y) \log p(y)$ est l'entropie marginale de l'ensemble de labels Y et $H(Y|X) = \sum_x p(x) H(Y|X=x)$ est l'entropie conditionnelle de Y sachant X.

Supposons maintenant que nous avons N tâches et leurs ensembles de labels respectifs $Y_1, \ldots, Y_N$. Pour apprendre des MT−DTs, l'idée est de traiter toutes les tâches ensemble en utilisant la concaténation des ensembles de labels, notée $\oplus_{j=1}^{N} Y_j$. La *tâche globalisée* prend en entrée un échantillon $S = \{< x_i, y_i > |\ x_i \in \mathcal{X},\ y_i = f(x_i) \in \oplus_{j=1}^{N} Y_j,\ 1 \leq i \leq m\}$.

Pour l'apprentissage, il est alors possible d'utiliser le *gain d'information joint* :

$$IG_1 = IG(\oplus_{j=1}^{N} Y_j; X)$$

Une alternative à IG_1 est de considérer la somme non pondérée des gains d'information sur les tâches individuelles :

$$IG_2 = \sum_{j=1}^{T} IG(Y_j; X)$$

Cependant, nous avons observé expérimentalement que IG_2 échoue à faire mieux que IG_1. Et en effet, théoriquement cette fois, nous allons prouver que IG_1 est équivalent à la somme pondérée des gains d'information sur les tâches individuelles et inférer un critère IG supérieur à IG_1. Cette nouvelle mesure, notée IG_3, prend le maximum des valeurs parmi les IGs individuels.

$$IG_3 = max\{IG(Y_j; X), j = 1, \ldots, N\}$$

Nous rappelons tout d'abord, dans le lemme suivant, une caractéristique de l'entropie (Gray, 2011). Ce lemme établit une relation entre l'entropie d'un ensemble complet de valeurs et les entropies de ses sous-ensembles disjoints.

Lemme 1
Pour $q_{kj} \geq 0$, tel que $\sum_{k=1}^{n} \sum_{j=1}^{m} q_{kj} = 1, p_k = \sum_{j=1}^{m} q_{kj}, \forall k = 1, \ldots, n$, nous avons :

$$H(q_{11}, \ldots, q_{1m}, q_{21}, \ldots, q_{2m}, \ldots, q_{n1}, \ldots, q_{nm}) = \tag{2}$$

$$H(p_1, \ldots, p_n) + \sum p_k H\left(\frac{q_{k1}}{p_k}, \ldots, \frac{q_{km}}{p_k}\right), p_k > 0, \forall k. \tag{3}$$

Ce lemme va nous permettre de prouver le théorème suivant qui fait le lien entre le gain d'information joint sur l'ensemble des tâches $IG(\oplus_{j=1}^{N} Y_j; X)$ et les gains individuels calculés sur chaque tâche $IG(Y_j; X), j = 1, \ldots, N$.

Théorème 1
Étant données N tâches dotées des ensembles de classes à prédire $Y_1, \ldots, Y_N$, soit p_j la proportion d'exemples de la tâche j dans le jeu de données complet $p_j = \frac{|S_j|}{\sum_{i=1}^{N} |S_i|}, j = 1, \ldots, N$, $\sum_{j=1}^{N} p_j = 1$. Nous avons :

$$IG(\oplus_{j=1}^{N} Y_j; X) = \sum_{j=1}^{N} p_j IG(Y_j; X) \leq max(IG(Y_1; X), \ldots, IG(Y_N; X)). \tag{4}$$

Preuve. D'abord, nous utilisons le Lemme 1 pour développer le terme d'entropie $H(\oplus_{j=1}^{N} Y_j)$ dans la formule (1) du gain d'information :

$$H(\oplus_{j=1}^{N} Y_j) = H(p_1, \ldots, p_N) + \sum_{j=1}^{N} p_j H(Y_j), \tag{5}$$

où $\sum_{j=1}^{N} p_j = 1$. Ensuite, nous développons le terme d'entropie conditionnelle dans l'équation (1), comme suit :

$$
\begin{aligned}
H(\oplus_{j=1}^{N} Y_j|X) &= \sum_x p(x) H(\oplus_{j=1}^{N} Y_j|X=x) && (6)\\
&= \sum_x p(x) \left(H(p_1,\ldots,p_N) + \sum_{j=1}^{N} p_j H(Y_j|X=x) \right) && (7)\\
&= H(p_1,\ldots,p_N) + \sum_{j=1}^{N} p_j \sum_x p(x) H(Y_j|X=x) && (8)\\
&= H(p_1,\ldots,p_N) + \sum_{j=1}^{N} p_j H(Y_j|X). && (9)
\end{aligned}
$$

Nous combinons maintenant les termes d'entropie (5) et d'entropie conditionnelle (9) pour évaluer la gain d'information joint $IG(\oplus_{j=1}^{N} Y_j; X)$. Nous obtenons :

$$
\begin{aligned}
IG(\oplus_{j=1}^{N} Y_j; X) &= H(\oplus_{j=1}^{N} Y_j) - H(\oplus_{j=1}^{N} Y_j|X) && (10)\\
&= \sum_{j=1}^{N} p_j IG(Y_j; X) && (11)\\
&\leq \sum_{j=1}^{N} p_j max(IG(Y_1; X), \ldots, IG(Y_N; X)) && (12)\\
&= max(IG(Y_1; X), \ldots, IG(Y_N; X)). && (13)
\end{aligned}
$$

Ce qui termine la preuve du théorème.

Le théorème 1 indique que le critère IG_3 pour l'apprentissage d'arbres de décision dans le cas multi-tâches est supérieur à la version jointe IG_1. Cela suggère que maximiser le gain d'information sur les tâches individuelles permettra d'apprendre de meilleures règles de décision que d'utiliser un critère portant sur l'ensemble des tâches.

La Figure 2.3 compare les critères IG_1 et IG_3 sur plusieurs jeux de données produits aléatoirement. Deux jeux de labels sont produits en utilisant différentes combinaisons des lois Uniforme, Normale (avec $\mu = 0$ et $\sigma = 1$), de Poisson ($\lambda = 1$) et Bêta (avec $\alpha = \beta = 0.5$). Le nombre de labels dans chaque ensemble varie de 2 à 20. Les valeurs des attributs sont systématiquement obtenues par un tirage uniforme. Comme le montre la figure 2.3, IG_3 amène de 2% à 18% de gain d'information supplémentaires que IG_1, avec un gain minimal dans le cas de deux lois uniformes.

Le pseudo-code permettant l'apprentissage de `MT-DTs` est présenté à l'Algorithme 1. Celui-ci appelle la fonction `split` qui fournit le test maximisant une mesure de gain quelconque sur un échantillon multi-tâches S, ainsi que la partition produite par ce test sur S. À chaque étape, la méthode ajoute un nœud-décision pour les tâches n'ayant plus de représentant dans la population courante, ou n'ayant que des représentants avec le même label. Ensuite, un nouveau test est choisi pour les exemples restants, etc. Dans la partie expérimentale, nous comparerons les trois candidats évoqués pour la mesure IG, à savoir : la jointe IG_1, la non-pondérée IG_2 et la maximale IG_3.

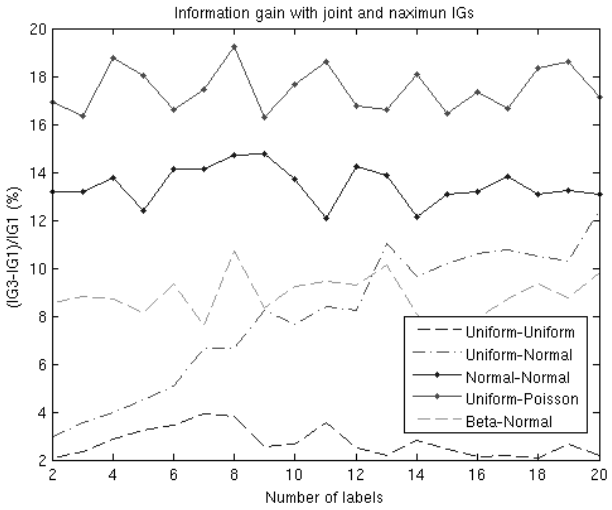

FIGURE 2 – Gain d'information pour des données aléatoires.

Algorithme 1 Apprentissage des `MT-DTs`.

ENTRÉE : $S = \cup_{j=1}^{N} \{e_i =< x_i, y_i, j > |\ x_i \in \mathcal{X};\ y_i \in Y_j\}$
ENTRÉE : IG, la mesure de gain d'information à utiliser
`res = []` { *stockera les nœuds choisis et éventuels nœuds-décision* }
pour $j = 1$ à N **faire**
  si les exemples de la tâche j ont tous le même label **ou** $S_j = \emptyset$ **alors**
    Ajouter à `res` un nœud-décision (j, y) { y *est soit le label commun de* S_j *ou le label majoritaire dans le nœud père si* $S_j = \emptyset$ }
    $S = S \setminus S_j$
  fin si
fin pour
Appel à `split(`S`)`
Trouver `bestnode`, le test qui maximise IG
Calculer $[S^1, \ldots, S^V]$, la partition de S selon `bestnode`
Ajouter `bestnode` à `res`
Appeler récursivement sur $[S^1, \ldots, S^V]$ pour obtenir les enfants de `res`
renvoyer `res`

3 *Boosting* multi-tâches

Dans la section précédente, nous avons défini les MT−DTs et proposé un algorithme d'apprentissage de ces MT−DTs utilisant une mesure de gain d'information particulière. Pour palier certains défauts connus des arbres de décisions comme le sur-apprentissage, nous travaillons maintenant à embarquer notre algorithme pour les MT−DTs comme apprenant faible dans une méthode de *boosting*.

Adapté de Adaboost.M1 (Schapire & Singer, 1999), nous proposons l'algorithme *Multi-Task Adaboost* (MT−Adaboost) présenté à l'Algorithme 2. T est le nombre d'étapes de *boosting*, init est une procédure fixant les valeurs de la distribution initiale D_1 sur S, WL est un apprenant faible qui renvoie un MT−DT à partir d'un échantillon S et d'une distribution D sur S. Le résultat est un classifieur multi-tâche H, fonction de $\mathcal{X}$ dans $Y_1 \times \ldots \times Y_N$.

Comme un algorithme de *boosting* mono-tâche, MT−Adaboost appelle plusieurs fois l'apprenant faible WL. À chaque appel t, l'algorithme fournit la distribution actuelle D_t et les exemples d'apprentissage S à WL, lequel découvre et renvoie $h_t : \mathcal{X} \rightarrow Y_1 \times \ldots \times Y_N$ qui minimise l'erreur mesurée sur S par rapport à D_t. La distribution D_{t+1} est ensuite calculée en fonction de D_t et de h_t : les exemples correctement classés par h_t ont leurs poids multipliés par $0 \leq \beta_t \leq 1$ (donc diminués), et les poids des exemples en erreur sont eux laissés inchangés. Finalement, ces poids sont normalisés à l'aide de Z_t.

Le classifieur final H, pour une tâche donnée j, est un vote pondéré des prédictions des classifieurs faibles pour cette tâche. Pour être précis, pour une paire exemple-tâche (x, j), le classifieur H proposera le label $y \in Y_j$ qui maximise la somme des poids des hypothèses faibles prédisant y pour la tâche j. Le poids associé à l'hypothèse h_t est fixé à $ln(1/\beta_t)$, si bien que le poids sera d'autant plus grand que l'erreur de h_t sera faible. MT−Adaboost bénéficie des mêmes propriétés théoriques que Adaboost.M1 : si les hypothèses sont réellement *faibles* (erreur strictement plus petite que $1/2$), alors l'erreur empirique du classifieur final H décroît vers zéro exponentiellement vite, avec T le nombre d'étapes de *boosting*.

4 Expérimentations

Dans cette section, nous présentons les expérimentations que nous avons menées sur trois jeux de données. Ces jeux de données sont décrits, ainsi que la méthodologie d'évaluation mise en œuvre. Puis nous comparons en mono-tâche les performances prédictives d'un apprentissage d'arbres de décision par C4.5 avec nos MT−DTs appris en utilisant les critères IG_1, IG_2 et IG_3. Enfin, nous discutons les résultats expérimentaux obtenus par les arbres *boostés* construits par notre algorithme MT−Adaboost.

4.1 Jeux de données

4.1.1 Données artificielles

Le premier jeu de données a été produit aléatoirement. Il contient deux tâches binaires avec comme labels $\mathcal{Y}_1 = \{y_{11}, y_{12}\}, \mathcal{Y}_2 = \{y_{21}, y_{22}\}$ et partageant 100 attributs numériques. Chaque tâche est un mélange de deux gaussiennes multivariées, une par label, et les labels des deux tâches sont reliés deux à deux : pour $i \in \{1, 2\}$, les gaussiennes de y_{1i} et de y_{2i} utilisent des paramètres tirés de la même distribution. Enfin, pour chaque tâche, on se donne 100 exemples, 50 de chaque label.

Algorithme 2 `MT-Adaboost`.

ENTRÉE : $S = \cup_{j=1}^{N}\{e_i =< x_i, y_i, j > |\ x_i \in \mathcal{X};\ y_i \in Y_j\}$

1: $D_1 = \texttt{init}(S)$ { *initialisation de la distribution* }
2: **pour** $t = 1$ à T **faire**
3: $h^t = \texttt{WL}(S, D_t)$ { *apprentissage faible et obtention d'un* `MT-DT` }
4: Calculer l'erreur de h^t : $\epsilon_t = \sum_{j=1}^{N} \sum_{i:h_j^t(x_i)\neq y_i} D_j(x_i)$.
5: **si** $\epsilon_t > 1/2$ **alors**
6: $T = t - 1$ et sortir de la boucle
7: **fin si**
8: $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
9: **si** $h_j^t(x_i) == y_i$ **alors** { *Mise à jour de la distribution* }
10: $D_{t+1}(e_i) = \frac{D_t(e_i)\times\beta_t}{Z_t}$
11: **sinon**
12: $D_{t+1}(e_i) = \frac{D_t(e_i)}{Z_t}$
13: **fin si** { Z_t *est un facteur tels que* D_{t+1} *soit une distribution* }
14: **fin pour**
15: **renvoyer** H le classifieur défini par :

$$H_j(x) = \arg\max_{y\in Y_j}(\sum_{i=1}^{i=T}(\ln 1/\beta_t)),\ 1 \leq j \leq N$$

4.1.2 Enron

Le jeu de données « Enron » (`http://www.cs.cmu.edu/~enron/`) contient tous les mails envoyés et reçus pendant plusieurs années par 150 personnes occupant des postes à responsabilité dans la société Enron. Les annotations de ce corpus ont été réalisées par deux sources distinctes.

La première est le département de la justice des États-Unis (`http://www.usdoj.gov/enron/`) qui a publié une liste de mails pertinents dans le cadre d'un procès contre deux dirigeants de Enron. Associés à d'autres mails annotés manuellement comme *non-pertinent*, ces messages constituent une première tâche binaire *Pertinent versus Non-pertinent*, avec un total de 372 exemples étiquetés.

Les annotations du second corpus proviennent d'étudiants de la *Berkeley University* qui ont étiqueté les messages selon les thèmes abordés, en moyenne 250 mails par thème. Nous utilisons pour nos expériences cinq thèmes : *Business*, *Legal*, *Influence*, *Arrangement* et *Personal*.

Comme les deux tâches ne disposent que de petits échantillons et qu'elles nous semblent liées (par exemple, on peut estimer qu'un mail *personnel* ne sera pas *pertinent* pour la justice), l'apprentissage simultané des deux tâches pourrait être avantageux pour chacune. Il faut noter que les deux corpus sont disjoints : aucun mail n'a annoté à la fois par la justice américaine dans le cadre du procès contre Enron et par les étudiants de Berkeley. Nous utilisons comme attributs des caractéristiques textuelles et d'autres dites *sociales* extraites du réseau social sous-jacent à l'entreprise, voir (Hovelynck & Chidlovskii, 2010) pour plus de détails.

4.1.3 Spam Filtering

Ce jeu de données a servi de matériel à un concours organisé dans le cadre de ECML-PKDD en 2006. Il regroupe les boites aux lettres de quinze utilisateurs. Chaque boite contient 400 mails, étique-

Tâches	Apprentissage (Test)	C4.5	IG_1	IG_2	IG_3
Tâche 1	80 (20)	83.10 ± 1.32	83.74 ± 3.43	83.74 ± 3.43	**85.67±1.21**
Tâche 2	80 (20)	84.24 ± 2.21	83.65 ± 3.77	82.43 ± 2.32	**84.98±1.53**
Moyennes		83.67	83.70	83.09	**85.23**

TABLE 1 – Précision moyenne sur deux tâches artificielles.

Tâches	Apprentissage (Test)	C4.5	IG_1	IG_2	IG_3
Pertinent Vs. Non-Pertinent	299 (74)	80.32 ± 1.87	80.59 ± 2.23	80.01 ± 3.11	**81.81±1.16**
5 thèmes	265 (66)	43.12 ± 1.03	43.65 ± 1.77	44.12 ± 0.42	**48.11±0.023**
Moyennes		61.72	62.12	62.066	**64.96**

TABLE 2 – Précision moyenne sur les tâches Enron.

tés en *spam* ou *non spam*. Chaque mail est représenté par un codage classique de type *sac de mots*. Nous considérons chaque utilisateur comme une tâche. Nous estimons que les tâches ainsi définies sont reliées tout en étant distinctes car il s'agit toujours de classer des messages en *souhaitable* ou *indésirable* mais que chaque utilisateur peut avoir un avis différent des autres concernant la classification d'un message précis.

4.2 Résultats des arbres de décision multi-tâches

Dans cette section, nous décrivons les expériences menées sur l'apprentissage de `MT-DTs` en utilisant les mesures IG_1, IG_2 et IG_3 (décrites Section 2.3). Nous comparons également nos `MT-DTs` à des arbres de décision mono-tâches appris par C4.5. Dans chaque expérience, nous utilisons une validation croisée cinq fois (un apprentissage se fait sur quatre cinquièmes des données, le dernier cinquième servant de test). Chaque méthode est exécutée trois fois et nous présentons les valeurs moyennes obtenues.

Les résultats sur les données artificielles sont présentées à la Table 1. Nous observons que l'apprentissage de `MT-DTs` avec IG_3 apporte une amélioration significative par rapport à C4.5, alors que IG_1 et IG_2 produisent un comportement proche de celui de C4.5, un peu meilleur sur la tâche 1, un peu moins bon sur la tâche 2.

Les résultats pour le jeu de données « Enron » sont présentés Table 2. À nouveau, concernant les précisions obtenues, nous notons une supériorité de IG_3 sur les autres mesures de gain d'information. Cependant, apprendre les tâches simultanément n'apporte pas le même bénéfice à chaque tâche, certaines profitant plus que d'autres de l'apprentissage multi-tâches. Les résultats sur les trois jeux de données utilisés, et en particulier sur les données EMCL'06 (voir la Table 3), montrent que les tâches les plus difficiles (tâches qui ne permettent que de faibles précisions lorsqu'elles sont apprises individuellement) autorisent une marge de progression plus importante. Autrement dit, le transfert d'informations entre tâches n'est pas nécessairement équilibré, les tâches les plus faciles servant de support aux plus difficiles.

4.3 Résultats des combinaisons d'arbres

Dans la section précédente, nous avons montré expérimentalement l'intérêt d'apprendre des tâches simultanément, en particulier en utilisant des mesures de gain d'information comme IG_3. Dans ces dernières expérimentations, nous comparons les performances d'arbres *boostés* pour les problèmes

Tâches	Apprentissage (Test)	C4.5	IG_1	IG_2	IG_3
Utilisateur-1	320 (80)	86.45 ± 1.23	86.19 ± 1.14	86.00 ± 1.88	**87.65±3.42**
Utilisateur-2	320 (80)	85.13 ± 2.16	85.53 ± 2.22	85.07 ± 3.16	**88.93±3.44**
Utilisateur-3	320 (80)	88.03 ± 2.11	88.22 ± 2.56	**88.52±1.33**	88.19 ± 2.51
Moyennes		86.54	86.65	86.53	**88.26**

TABLE 3 – Précision moyenne sur trois boites aux lettres « ECML'06 ».

Tâches	Apprentissage (Test)	Adaboost C4.5	MT−Adaboost IG_1	MT−Adaboost IG_2	MT−Adaboost IG_3
Pertinent Vs. Non-Pertinent	299 (74)	85.10 ± 1.21	84.66 ± 2.15	84.52 ± 1.2	**86.01±1.53**
5 thèmes	265 (66)	51.34 ± 0.43	52.89 ± 0.87	52.17 ± 0.74	**57.11±0.02**
Moyennes		68.22	68.78	68.35	**71.65**

TABLE 4 – Précision moyenne des arbres *boostés* sur les données « Enron ».

multi-tâches : d'une part nos MT−DTs combinés par MT−Adaboost (algorithme 2) et d'autre part des arbres C4.5 *boostés* par Adaboost.M1 (Schapire & Singer, 1999). Chaque algorithme de *boosting* n'a qu'un paramètre, le nombre d'étapes de *boosting* à effectuer que nous avons systématiquement fixé à la valeur 20. Les valeurs moyennes des précisions obtenues sur trois exécutions sont données à la Table 4, uniquement pour le problème « Enron ». Avec les arbres *boostés*, comme dans le cas des *non-boostés*, nous observons un gain : MT−Adaboost associé aux MT−DTs sont significativement meilleurs que Adaboost avec C4.5 et à nouveau les tâches les plus difficiles profitent le plus du transfert.

5 Conclusion

Nous avons proposé une variation sur l'apprentissage d'arbres de décision pour permettre leur utilisation dans le cadre multi-tâches, avec les contributions suivantes. Tout d'abord, nous avons développé la définition et l'apprentissage des MT−DTs pour gérer l'aspect multi-tâches de nos problèmes d'apprentissage. Le critère pour décider des tests à appliquer à chaque étape de l'arbre utilise simultanément les données de plusieurs tâches, ce qui nous permet de capturer des relations entre celles-ci, qu'elles soient globales ou locales. Nous avons prouvé une importante propriété de la mesure du gain d'information dans le contexte multi-tâches. Cela nous a permis de déduire un critère original basé sur le gain d'information et approprié pour l'apprentissage multi-tâches. Nous avons également modifié l'algorithme MT−Adaboost pour prendre en compte les tâches multi-classes et plus seulement binaires. Enfin, nous avons validé nos méthodes à travers une série d'expérimentations sur trois jeux de données multi-tâches.

Références

ARGYRIOU A., EVGENIOU T. & PONTIL M. (2006). Multi-task feature learning. In *NIPS*, p. 41–48.

CARUANA R. (1997). Multitask learning. In *Machine Learning*, volume 28, p. 41–75.

FADDOUL J. B., CHIDLOVSKII B., TORRE F. & GILLERON R. (2010). Boosting multi-task weak learners with applications to textual and social data. In *Proceedings of the Ninth International Conference on Machine Learning and Applications (ICMLA)*, p. 367–372.

FREUND Y. & SCHAPIRE R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning (ICML)*, p. 148–156.

GRAY R. M. (2011). *Entropy and Information Theory, Second Edition*. Springer-Verlag.

HOVELYNCK M. & CHIDLOVSKII B. (2010). Multi-modality in one-class classification. In *Proceedings of the 19th international conference on World wide web (WWW)*, p. 441–450.

LIU T.-Y., YANG Y., WAN H., ZENG H.-J., CHEN Z. & MA W.-Y. (2005). Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explor. Newsl.*, **7**(1), 36–43.

MANTRACH A. & RENDERS J.-M. (2012). Search engine for mailboxes based on cross-modal, topics and communities query expansion. In *Proceedings of European Conference on Information Retrieval (ECIR'2012)*.

QUADRIANTO N., SMOLA A., CAETANO T., VISHWANATHAN S. & PETTERSON J. (2010). Multi-task learning without label correspondences. In *Proceedings of the Twenty-Fourth Annual Conference on Neural Information Processing Systems (NIPS)*, p. 1957–1965.

QUINLAN J. R. (1993). *C4.5 : Programs for Machine Learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc.

SCHAPIRE R. E. & SINGER Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, **37**.

XUE Y., LIAO X., CARIN L. & KRISHNAPURAM B. (2007). Multi-task learning for classification with dirichlet process priors. *J. Mach. Learn. Res.*, **8**, 35–63.