Impact de la taille de l'ensemble d'apprentissage : une étude empirique

Christophe Salperwyck[*,**], Vincent Lemaire[*]

[*]Orange Labs
2, Avenue Pierre Marzin 22300 Lannion
prenom.nom@orange-ftgroup.com
[**] LIFL (UMR CNRS 8022) - Université de Lille 3
Domaine Universitaire du Pont de Bois
59653 Villeneuve d'Ascq Cedex

Résumé. Les techniques d'apprentissage ont montré leurs capacités à traiter des volumétries importantes de données. La plupart des approches d'apprentissage statistique utilisent des bases d'apprentissage de taille finie et produisent des modèles statiques. Cependant dans certaines situations : apprentissage actif ou incrémental, la tâche d'apprentissage commence avec très peu de données. Il est nécessaire dans ces cas de s'intéresser à des algorithmes capables de générer des modèles avec peu de données. Les classifieurs de la littérature sont souvent classés à l'aide de critères tels que leurs performances en classification, leurs capacités à trier les données (ranking)... Mais cette taxonomie des classifieurs peut singulièrement évoluer si l'on s'intéresse à leurs capacités en présence d'une faible quantité d'exemples. A notre connaissance seulement quelques études ont été réalisées sur cette problématique. Cette étude cherche à étudier un plus large panel à la fois d'algorithmes d'apprentissage (9 types d'algorithmes différents) mais aussi de jeux de données (17 bases de l'UCI).

1 Introduction

Les techniques d'apprentissage ont montré leurs capacités à traiter des volumétries importantes de données et ce sur des problèmes réels (Guyon et al., 2009; Féraud et al., 2010). Néanmoins le plus gros effort a été réalisé pour des analyses de données homogènes et stationnaires. La plupart des approches d'apprentissage statistique (machine learning) utilisent des bases d'apprentissage de taille finie et produisent des modèles statiques. Cette étude se positionne dans le cadre de l'apprentissage supervisé.

Cependant dans certaines situations, la tâche d'apprentissage commence avec très peu de données. Il est nécessaire dans ces cas là de s'intéresser à des algorithmes capables de générer des modèles avec peu de données et si possible avec une faible variance. L'apprentissage actif et l'apprentissage incrémental sont les deux principales catégories d'apprentissage pour lesquelles un algorithme capable d'apprendre avec peu de données est nécessaire.

L'**apprentissage actif** (Settles, 2010) est utilisé dans les cas où de nombreuses données sont disponibles mais leurs étiquetages coûtent cher. Dans ce cas on cherche à sélectionner le

plus petit nombre de données permettant de réaliser le meilleur apprentissage. On se retrouve donc avec peu de données que l'on espère être très pertinentes. Un algorithme capable de réaliser de bonnes prédictions avec peu de données est donc nécessaire afin d'avoir une procédure d'étiquetage la moins coûteuse possible.

L'**apprentissage incrémental** (Michalski et al., 1986) commence par nature par apprendre avec peu de données car théoriquement il doit apprendre dès qu'on lui fournit les premiers exemples. Le modèle est raffiné au fur et à mesure de l'arrivée des nouveaux exemples. La qualité du modèle fourni au départ dépend donc là aussi de la capacité de l'algorithme à apprendre avec peu d'exemples. L'apprentissage incrémental existe depuis de nombreuses années mais est récemment revenu au goût du jour avec les flux de données. En effet de nombreuses applications génèrent des flux de données : réseaux de capteurs, millions de logs d'accès à des pages Web... Les données arrivent rapidement et ne sont visibles qu'une fois, il est donc nécessaire d'apprendre celles-ci au fur et à mesure de leurs arrivées. L'apprentissage incrémental apparaît dès lors comme une solution naturelle à ces problèmes de flux.

De leur coté **les classifieurs** de la littérature tels que les arbres de décisions, réseaux de neurones, support vector machine... sont souvent classés à l'aide de critères tels que leurs performances en classification, leurs capacités à trier les données (ranking)... Mais cette taxonomie des classifieurs peut singulièrement évoluer si l'on s'intéresse à leurs capacités en présence d'une faible quantité d'exemples. On peut citer par exemple les arbres d'Hoeffding (Domingos et Hulten, 2000) qui sont très largement utilisés comme méthode d'apprentissage incrémental sur les flux. La construction de l'arbre est incrémentale et les feuilles se transforment en nœud au fur et à mesure de l'arrivée des exemples. Or il apparait (Gama et al., 2003) qu'il est intéressant de mettre des classifieurs dans les feuilles de l'arbre avant que celles-ci ne soient transformées. Un classifieur pertinent avec peu de données permet d'avoir des modèles locaux de qualité dans les feuilles.

A notre connaissance seulement quelques études ont été réalisées sur cette problématique des performances des classifieurs de la littérature versus la taille de l'ensemble d'apprentissage. Forman et Cohen ont étudié trois algorithmes (Bayésien naïf, SVM et régression logistique) sur des jeux de petites tailles et déséquilibrés. Dans (Lim et al., 2000), les auteurs s'intéressent à la vitesse d'apprentissage en temps absolu contrairement à cette étude qui s'intéresse à la qualité d'apprentissage par rapport aux nombres d'exemples appris.

L'étude présentée dans cet article cherche à étudier un plus large panel à la fois d'algorithmes d'apprentissage (9 types d'algorithmes différents) mais aussi de jeux de données (17 bases de l'UCI). Dans un premier temps (section 2) les classifieurs qui seront utilisés dans cette étude ainsi que les valeurs de leurs paramètres testés seront présentés. Le protocole expérimental sera présenté section 3 : les jeux de données utilisés, le découpage des jeux de données en ensemble d'apprentissage et de test, les métriques d'évaluation de nos résultats. La section 4 présentera les résultats obtenus et les analysera par rapport à la typologie des différents classifieurs étudiés. Dans une dernière partie, nous conclurons et nous proposerons une suite possible à cette étude.

2 Vitesse d'apprentissage et Typologie des algorithmes testés

2.1 Vitesse d'apprentissage

Il est important de noter que cette étude ne se concentre pas sur les bornes ou temps de convergence d'un classifieur étant donné un ensemble d'apprentissage comportant n exemples. Ceci correspondrait à déterminer en quelque sorte le temps CPU qu'il faudrait à ce classifieur pour apprendre les n exemples. L'analyse de la vitesse de convergence ou la complexité de l'algorithme d'apprentissage n'est pas le but de cette étude.

On entend par vitesse d'apprentissage dans cette étude, la capacité d'un classifieur à obtenir une solution "intéressante" à un problème de classification à l'aide d'un minimum d'exemple présent en apprentissage. Un exemple de comparaison est donné dans la figure 1 : cette figure présente les résultats que pourraient obtenir deux classifieurs sur un même problème de classification. L'axe des abscisses indique le nombre d'exemples (n) utilisé pour entraîner le classifieur et l'axe des ordonnées l'AUC obtenue (AUC=$f(n)$).

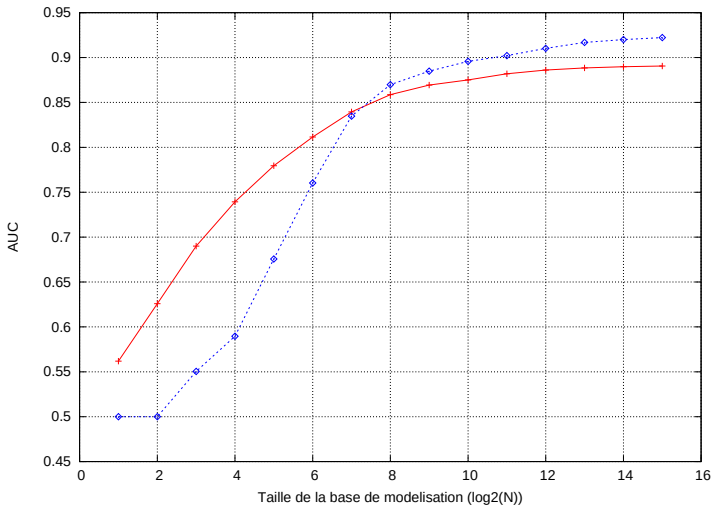


FIG. 1 – *Illustration de la vitesse d'apprentissage de deux classifieurs (AUC en test)*

Cette figure illustre deux comportements différents : l'algorithme "rouge" (+) atteint une meilleure performance dans un premier temps mais ensuite lorsque n devient grand l'algorithme "bleu" (○) est finalement plus optimal. Si l'on souhaite avoir toujours le meilleur modèle à disposition, il est évident que l'on doit prendre en compte les deux modèles et que le modèle rouge (+) devra être utilisé en premier puis le modèle bleu (○) par la suite.

Ceci illustre parfaitement le but de cette étude : dans le cas de l'apprentissage incrémental ou de l'apprentissage actif et pour des ensembles d'apprentissage de petites tailles existe-t-il des types de classifieurs plus adaptés que d'autres ?

2.2 Typologie des algorithmes testés

Il existe diverses familles de modèles de classification basées sur l'apprentissage de paramètres. Ces familles peuvent être :

- les classifieurs linéaires pour lesquels il s'agit d'apprentissage de surface séparatrice linéaire. Ce type de classifieur est basé sur une combinaison linéaire des variables explicatives du phénomène tel que $y = \sum_j w_j X_j$ où X_j désigne l'une des variables explicatives et w_j son poids.
- les classifieurs non linéaires tel que les perceptrons multicouches, les k plus proches voisins ou les arbres de décision.

Sachant que :

- ces familles peuvent se recouvrir partiellement ;
- le placement n'est pas toujours aisé (un arbre de décision peut être vu comme une surface de décision linéaire par morceaux) ;
- un classifieur linéaire peut traiter des problèmes de classification non linéaire sous peine d'une projection préalable (l'"astuce" du noyau pour les séparateurs à vaste marge (SVM))
- des sous découpes sont présentes. Par exemple les classifieurs linéaires peuvent être rangés en 2 grandes sous familles de méthodes pour estimer les paramètres du vecteur w d'un classifieur linéaire :
  - modèle génératif : il s'agit d'abord de modéliser les probabilités conditionnelles $P(X|C)$. L'estimation de $P(C|X)$ viendra ensuite sous la condition de connaître $P(C)$. On citera à titre d'exemple : l'analyse discriminante linéaire (LDA - (Fisher, 1936)) et le classifieur Bayésien naïf (Langley et al., 1992).
  - modèle discriminant : il s'agit d'abord de maximiser la qualité de la classification sur un jeu de test en estimant directement $P(C|X)$ ou un score apparenté à l'étiquette des données de modélisation. On citera à titre d'exemple : la régression linéaire, la régression logistique, le perceptron et les machines à vecteurs de support.

On peut encore en trouver d'autres sous appellations comme :

- les classifieurs probabilistes qui estiment les probabilités conditionnelles des classes ($P(C|X)$) étant donné un vecteur d'entrée constitué par les combinaisons des variables explicatives du phénomène. Ce n'est pas le cas de nombreux classifieurs comme les scores retournés par les séparateurs à vaste marge (qui sont pourtant souvent utilisés comme mesures de confiance pour la classification de nouveaux exemples).
- les classifieurs paramétriques qui présupposent que les variables explicatives suivent une loi de probabilité définie au préalable (a priori). On citera par exemple le classifieur naïf de Bayes dans le cas où on présuppose que les $P(X_j)$ suivent des lois gaussiennes (Gama et al., 2005).

L'étude proposée par cet article ne peut couvrir l'ensemble des familles des classifieurs. Elle s'est néanmoins attachée à paver au mieux cette espace. Le tableau 1 résume les types de classifieurs qui ont été testés. Ces différents classifieurs et leurs paramètres sont présentés dans la sous-section suivante.

	Classifieur linéaire	Classifieur non linéaire
Modèle génératif	Naïve Bayes, Bayésien naïf sélectif	Réseaux Bayésien
Modèle discriminant	Régression logistique	Arbre de décision, Forêt d'arbres de décision

TAB. 1 – *Typologie versus classifieurs testés.*

2.3 Classifieurs testés

Pour tester l'ensemble des classifieurs cités dans la section précédente (table 1) deux logiciels accessibles au 'public' [1] ont été utilisés : WEKA (Witten et Frank, 2005) (version 3.7.1) et Khiops (Boullé, 2004) (version 7.0). Pour chacun de ces logiciels se sont les valeurs par défaut qui ont été utilisées et dont on présente les résultats en section 4. A titre d'exemples les paramètres par défaut pour la régression logistique sont fixés à -1 (∞) pour le nombre d'itération maximale et à 10^8 pour le ridge.

- Pour le logiciel Weka :
  - Bayes
    - Non supervisé (John et Langley, 1995) : Bayésien naïf standard. L'estimateur numérique est basé sur l'analyse des données d'apprentissage.
    - Supervisé : même chose que précédemment sauf qu'une discrétisation supervisée (MDL) est utilisée pour convertir les attributs numériques en attributs nominaux.
    - BayesNet (Bouckaert, 2004) : réseau Bayésien utilisant divers algorithmes de recherche et de mesures de qualité. Utilisé avec les paramètres par défaut : "SimpleEstimator" et algorithme de recherche "K2 search".
  - Régression :
    - Régression logistique : régression logistique polynomial avec un estimateur ridge qui est basé sur (Cessie et Houwelingen, 1992). Paramètres par défaut sauf pour le nombre d'itération maximale : 100.
  - Arbres :
    - ADTree (Freund et Mason, 1999) : arbre de décision avec plusieurs sous arbres combinés avec du boosting.
    - J48 (Quinlan, 1993) : arbre de décision C4.5 proposé par Quinlan.
    - SimpleCart (Breiman et al., 1984) : arbre binaire basé sur le coefficient de Gini.
    - RandomForest (Breiman, 2001) : forêt d'arbres par sélection aléatoire d'un sous ensemble des attributs ; 10 arbres par défaut dans la forêt ; 40 arbres ont aussi été testés.
  - Vote :
    - VFI (Demiröz et Güvenir, 1997) : classification par vote sur les intervalles des attributs. Les intervalles sont construits autour des classes pour chaque attribut (en fait une sorte de discrétisation). Le compte des classes est enregistré pour chaque intervalle pour chaque attribut et la classification se fait par vote. Le paramètre par défaut utilise l'option "weight feature intervals by confidence". Nous avons aussi essayé sans.

1. Toutes les expériences de cette étude sont reproductibles.

– Pour le logiciel Khiops [2] : outils développé par Orange Labs. Il implémente, pour la classification supervisée, un Bayésien naïf et Bayésien naïf sélectif.
Le Bayésien naïf et le Bayésien naïf sélectif (Boullé, 2006b) ont été testé avec différents prétraitements sur les attributs numériques et nominaux. Les deux prétraitements testés sur les attributs nominaux sont :
– Basic Grouping : un groupe par valeur observée
– Discrétisation MODL (Boullé, 2006a).
Sur les attributs numériques trois prétraitements différents ont été testés :
– Equal Frequency
– Equal Width
– Méthode de groupement MODL (Boullé, 2005).
Les deux méthodes non supervisées Equal Frequency et Equal Width sont utilisées avec un nombre fixé à 10 intervalles. Si le nombre de valeurs observées est en dessous de 10, le nombre d'intervalles est réduit à ce nombre de valeurs observées.

3 Protocole expérimental

A travers la mise en place du protocole expérimental, on cherche à étudier le comportement des algorithmes en fonction de la taille de l'ensemble d'apprentissage. Le premier point consiste à utiliser des jeux de données à la fois variés et reconnus par la communauté de la fouille de données. Le second concerne le découpage des jeux de données en un ensemble d'apprentissage et de test pertinent par rapport à la problématique. Le troisième s'intéresse à la méthode d'évaluation mettant en avant les algorithmes qui apprennent bien avec peu de données.

3.1 Jeux de données

Afin que les résultats de cette étude puissent être généralisés des jeux de données variés du domaine de la classification ont été utilisés. Dans ce but, un large panel de jeux de données a été sélectionné à partir du dépôt de l'UCI (Blake et Merz, 1998). Des jeux avec seulement des variables catégorielles, ou seulement des variables continues, ou un mix des deux ont été utilisés. La taille de ceux-ci va d'une centaine à plusieurs dizaine de milliers d'exemples. La table 2 présente les caractéristiques en termes de nombre d'exemples, nombre d'attributs numériques et nominaux, précision du classifieur prédisant la classe majoritaire.

Cette étude ne s'intéresse qu'à des problèmes de classification binaire. Pour des problèmes à plus de deux classes le protocole expérimental devrait être différent [3].

3.2 Découpage des jeux de données en ensemble d'apprentissage et de test

Afin de générer nos jeux de données de petites tailles tout en gardant un protocole semblable à ceux utilisées dans les publications du domaine, nous avons choisi une 10-validation

2. www.khiops.com

3. En effet, la question se pose si l'on doit comparer, par exemple, l'apprentissage de peu d'exemples sur un problème binaire versus un problème à 20 classes.

	Jeu de données	#Var	#Var Cont.	#Var Cat.	# Exemples (n)	Précision maj.
1	Adult	15	7	8	48842	0.7607
2	Australian	14	6	8	690	0.5550
3	Breast	10	10	0	699	0.6552
4	Bupa	6	6	0	345	0.5797
5	Crx	15	6	9	690	0.5550
6	German	24	24	0	1000	0.7
7	Heart	13	10	3	270	0.5555
8	Hepatitis	19	6	13	155	0.7935
9	Horsecolic	27	7	20	368	0.6304
10	Hypothyroid	25	7	18	3163	0.9522
11	Ionosphere	34	34	0	351	0.6410
12	Mushroom	22	0	22	8416	0.5332
13	Pima	8	8	0	768	0.6510
14	SickEuthyroid	25	7	18	3163	0.9073
15	Sonar	60	60	0	208	0.5336
16	Spam	57	57	0	4307	0.6473
17	Tictactoe	9	0	9	958	0.6534

TAB. 2 – *Caractéristiques des jeux de données.*

croisée. Sur l'ensemble d'apprentissage (90%) de cette validation croisée, les exemples ont été tirés aléatoirement parmi les tailles suivantes :

$$S = \{S_1, S_2, ..., S_{max}\} = \left\{2, 2^2, 2^3, 2^4, \ldots, 2^{(\lfloor log_2(0.9n-1)\rfloor)}\right\}$$

où n est la taille maximale du jeu de données [4]. Le tirage a été réalisé 10 fois pour chaque jeu de données afin d'obtenir une moyenne et une variance sur le résultat. La performance des algorithmes est évaluée sur les 10% restant (des n exemples, voir table 2) de la validation croisée. La figure 3.2 présente ce protocole.

3.3 Le critère d'évaluation : ALC

Les expérimentations menées dans cet article donnent une courbe d'AUC (Fawcett, 2004) sur l'ensemble de test versus le nombre d'exemples utilisés pour réaliser l'apprentissage. Chaque courbe est constituée de $|S|$ points pour tous les algorithmes d'apprentissage utilisés.

Les performances en prédiction ont été évaluées avec l'ALC (Area under the Learning Curve - (Guyon et al., 2010)). L'AUC (Area Under the ROC Curve) est calculée pour tous les points $(2, 4, 8...)$ définis dans le jeu de données. Le score obtenu correspond à l'ALC normalisé calculé de la manière suivante :

$$score = \frac{(ALC - Arand)}{(Amax - Arand)}$$

où $Amax$ est la meilleure aire atteignable (soit 1) et $Arand$ est l'aire d'une solution basée sur des prédictions aléatoires (soit 0.5).

4. .9n correspond au fait que l'on réalise une 10-validation croisée

Impact de la taille de l'ensemble d'apprentissage

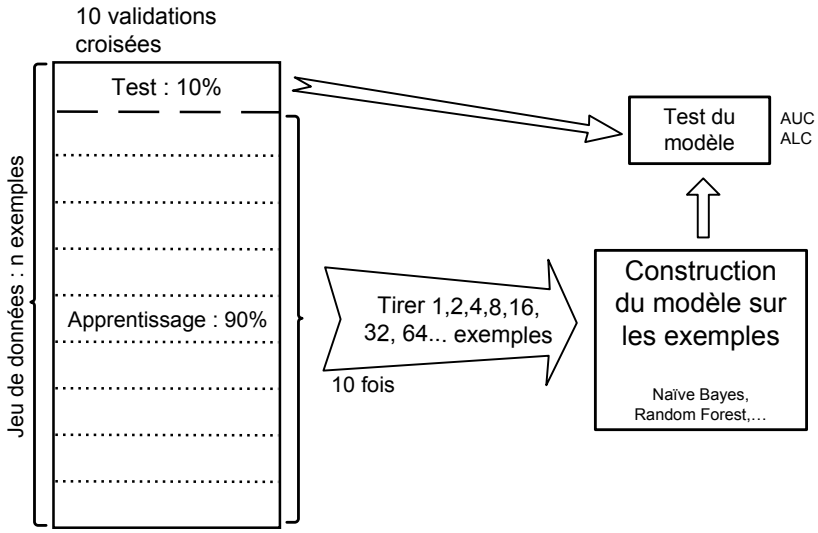


FIG. 2 – *Construction des jeux de données*

Ce critère possède de bonnes propriétés par rapport à l'objectif de cette étude. En effet, on cherche à évaluer la capacité d'un algorithme à apprendre avec une faible quantité de données. Ce critère met particulièrement en avant les plus petits jeux de données. L'abscisse de notre courbe d'ALC est logarithmique ce qui signifie que l'AUC obtenue pour 2 exemples contribue autant à la valeur de l'ALC que l'AUC obtenue pour 1024 exemples.

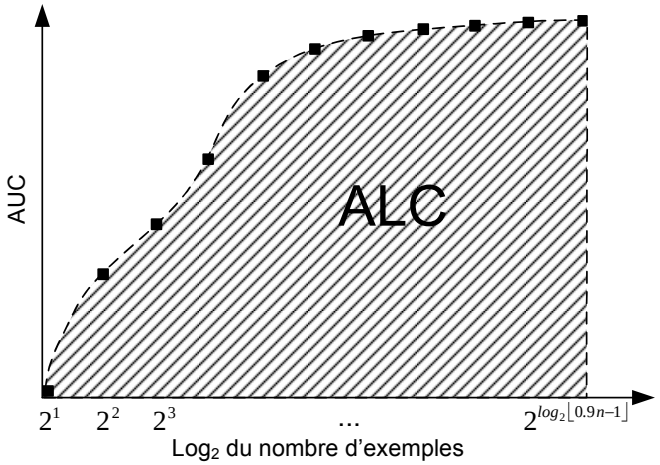


FIG. 3 – *Calcul de l'ALC : aire sous la courbe d'AUC*

4 Résultats

Cette section présente les résultats de l'étude. Les notations utilisées sont tout d'abord expliquées, ensuite les différentes tables et courbes de résultats sont présentées pour finalement les analyser et en proposer une interprétation.

4.1 Notations

Afin de pouvoir obtenir des tableaux de résultats complets et concis, nous avons dû utiliser des abréviations quant aux noms des algorithmes utilisés. L'environnement d'exécution constitue la première abréviation. Les algorithmes utilisés provenant de Weka sont préfixés par "W" et ceux provenant de Khiops n'ont pas de préfixes. Les algorithmes testés constituent la seconde abréviation :

- Weka
  - W-ADT : ADTree
  - W-BN : BayesNet - Réseau Bayésien
  - W-Log100 : Régression logistique
  - W-NB-NS : Bayésien naïf non supervisé
  - W-NB-S : Bayésien naïf supervisé
  - W-RF10/40 : Random Forest avec 10/40 arbres
  - W-VFI / W-VFIn : VFI avec/sans l'option "weight feature intervals by confidence"
  - W-SCart : SimpleCart
- Khiops Le format est ALGO-VAR_CONT-VAR_CAT
  - ALGO - NB : Bayésien naïf ou SNB : Bayésien naïf sélectif
  - VAR_CONT pour les variables continues - EF : EqualFrequency / EW : EqualWidth / M : MODL
  - VAR_CAT pour les variables catégorielles - BG : Basic Grouping / M : MODL

4.2 Tableaux et courbes présentés

Tableaux

Les performances en ALC obtenues par les différents algorithmes sur les différents jeux de données sont présentés dans la table 3. La dernière ligne donne la moyenne de l'ALC sur tous les jeux de tests par algorithme. Afin d'avoir une vue plus synthétique, la table 5 donne le rang moyen de chaque algorithme ainsi que son ALC moyenne et son AUC finale (moyenne des AUC du plus grand ensemble d'apprentissage : $2^{(\lfloor log_2(0.9n-1) \rfloor)}$).

Etant donné que cette étude se focalise particulièrement sur l'apprentissage des jeux de données de petites tailles, il est intéressant regarder le début des courbes d'AUC. Dans cette optique, des tableaux identiques aux tableaux 3 sont proposés dans les tableaux 4 mais ici l'ALC est calculée entre 2 et 2^6 exemples en apprentissage. Ce tableau permet d'observer plus particulièrement les algorithmes sur le début de la phase d'apprentissage. De façon identique, on retrouve pour les rangs par algorithme la table 6 qui se limite aussi à 2^6 exemples.

Courbes

Nous avons choisi quatre courbes mettant en avant différents types d'algorithmes. L'idée est de comparer le comportement de cet algorithme par rapport aux autres algorithmes sur un jeu

	W-ADT	W-BN	W-Log100	NB-EF-BG	NB-EF-M	NB-EW-BG	W-NB-NS	W-NB-S	W-RF10	W-RF40	W-VFI	W-VFI-N
Adult	58.05	**64.28**	54.19	62.31	58.98	62.64	59.88	63.70	54.73	58.44	58.02	63.80
Australian	59.40	71.55	55.17	65.75	64.07	64.11	56.43	**72.15**	59.88	63.23	59.03	69.06
Breast	74.97	89.12	87.27	**92.31**	**92.31**	92.18	88.26	89.04	88.25	89.89	88.40	92.30
Bupa	20.17	2.02	**24.06**	14.81	14.81	13.20	12.52	1.98	21.74	23.99	9.05	9.32
Crx	60.16	70.12	54.60	63.91	62.79	62.32	56.30	**70.63**	59.02	62.47	57.33	66.57
German	25.98	13.10	26.69	27.61	27.61	26.13	30.53	13.12	27.29	**31.69**	24.84	26.89
Heart	47.19	62.19	51.25	61.31	58.26	60.21	54.56	62.84	53.64	57.15	49.75	**64.22**
Hepatitis	39.92	60.38	37.27	49.64	46.22	47.78	37.92	**63.16**	46.93	50.36	46.71	58.88
Horsecolic	52.73	59.81	48.31	51.87	44.01	52.06	40.61	61.58	57.34	**62.27**	54.20	58.31
Hypothyroid	59.21	50.78	59.27	65.87	**66.75**	64.06	52.81	48.77	63.05	66.13	52.94	60.41
Ionosphere	57.42	59.18	43.17	65.91	65.91	64.68	56.58	59.06	65.86	**70.29**	52.06	61.54
Mushroom	82.20	**92.23**	87.20	88.99	86.26	88.99	92.07	92.07	88.76	89.97	89.25	91.01
Pima	37.70	30.76	41.99	41.50	41.50	40.16	38.35	30.84	40.68	**44.43**	26.77	30.06
SickEuthyroid	60.46	52.57	57.49	59.44	59.29	59.50	50.28	49.25	61.87	**66.85**	45.41	47.93
Sonar	32.64	36.54	36.04	38.10	38.10	37.32	31.72	36.51	41.80	**48.32**	23.12	34.37
Spam	66.53	72.86	68.86	79.93	79.93	77.29	71.07	72.76	75.92	**80.08**	67.72	78.29
Tictactoe	32.40	28.40	**53.24**	26.91	20.52	26.91	28.73	28.73	37.77	41.83	26.00	27.61
Moyenne ALC	51.01	53.88	52.12	56.25	54.55	55.27	50.51	53.89	55.56	**59.26**	48.86	55.33

	W-J48	NB-M-BG	NB-M-M	NB-EW-M	SNB-EF-BG	SNB-EF-M	SNB-EW-BG	SNB-EW-M	SNB-M-BG	SNB-M-M	W-SCart
Adult	41.68	64.06	56.63	59.25	55.25	55.13	55.73	55.52	57.92	56.02	40.97
Australian	51.60	65.60	63.22	62.32	53.20	54.13	52.04	53.04	54.85	57.18	42.70
Breast	66.84	76.24	76.24	92.18	79.29	79.29	79.40	79.40	76.30	76.30	57.85
Bupa	10.54	1.91	1.91	13.20	13.58	13.58	13.58	12.00	12.00	1.91	8.53
Crx	51.68	63.73	62.90	61.01	52.92	54.38	51.99	53.15	54.50	57.31	44.03
German	14.39	14.14	14.14	26.13	22.27	22.27	22.11	22.11	14.15	14.15	10.06
Heart	29.21	57.34	45.95	56.90	40.91	40.75	38.92	38.76	42.81	41.28	22.88
Hepatitis	21.05	46.83	40.49	44.66	30.99	32.43	31.32	32.50	30.52	31.34	10.62
Horsecolic	40.18	55.08	49.01	44.06	45.12	40.67	45.36	40.90	47.42	43.50	2.04
Hypothyroid	43.79	54.46	52.09	65.06	56.56	56.62	60.77	61.03	53.04	52.56	39.54
Ionosphere	42.89	53.04	53.04	64.68	55.15	55.15	55.48	55.48	53.42	53.42	36.27
Mushroom	77.71	88.99	86.26	86.26	82.56	81.77	82.56	81.77	82.56	81.77	74.62
Pima	25.34	29.71	29.71	40.16	37.65	37.65	37.56	37.56	29.89	29.89	20.08
SickEuthyroid	47.80	52.14	45.32	59.64	56.16	55.95	55.57	55.68	46.20	45.49	45.21
Sonar	19.40	31.97	31.97	37.32	25.26	25.26	25.21	25.21	27.12	27.12	16.20
Spam	52.67	66.63	66.63	77.29	65.93	65.93	60.53	60.53	66.33	66.33	46.11
Tictactoe	21.79	26.91	20.52	20.52	25.50	20.29	25.50	20.29	25.50	20.29	25.47
Moyenne ALC	38.74	49.93	46.83	53.57	46.96	46.54	46.59	46.17	44.97	44.46	31.95

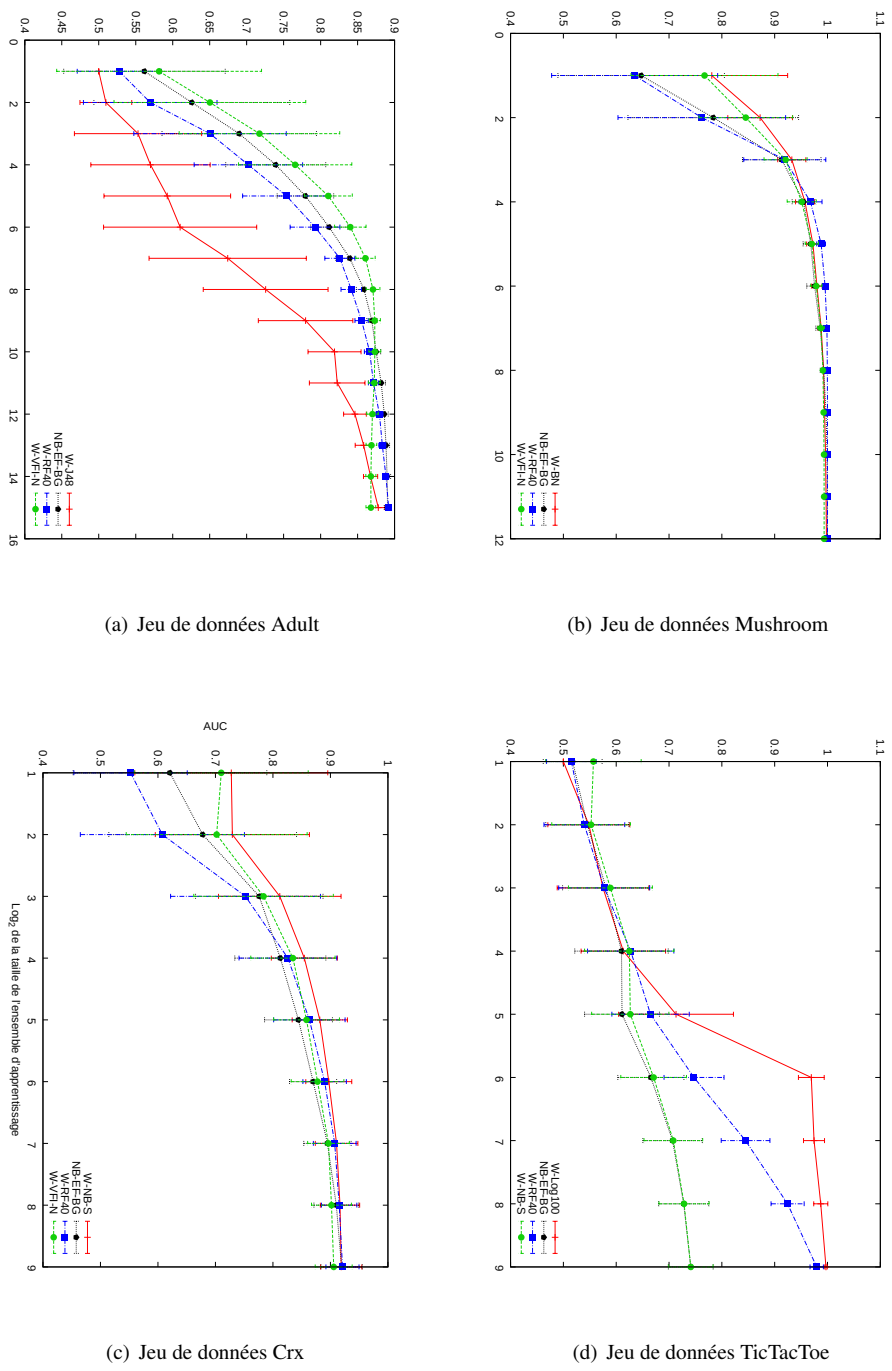
TAB. 3 – *ALC par algorithme et jeu de données*

	W-ADT	W-BN	W-Log100	NB-EF-BG	NB-EF-M	NB-EW-BG	W-NB-NS	W-NB-S	W-RF10	W-RF40	W-VFI	W-VFI-N
Adult	28.64	39.37	30.01	40.85	30.41	40.73	29.73	38.23	30.58	33.46	37.73	**46.18**
Australian	45.86	65.32	45.53	57.02	53.94	55.99	44.12	**66.17**	48.43	52.13	51.11	62.37
Breast	61.52	83.70	82.08	88.89	88.89	**88.92**	82.76	83.63	82.88	85.20	84.21	88.86
Bupa	14.19	2.02	**17.75**	10.48	10.48	9.76	8.26	1.96	14.90	16.28	8.30	8.66
Crx	46.73	62.86	44.37	54.27	52.00	53.40	43.71	**63.59**	46.86	50.82	48.45	58.91
German	17.49	4.54	15.51	17.20	17.20	16.92	18.51	4.59	18.64	**21.74**	15.29	18.44
Heart	42.48	59.44	47.95	58.07	54.24	57.68	49.47	60.14	50.22	53.59	46.64	**61.78**
Hepatitis	36.16	57.85	35.04	46.30	41.94	45.26	32.16	**60.91**	43.53	46.90	43.34	55.85
Horsecolic	44.41	54.46	41.07	44.70	33.88	45.11	29.08	**57.08**	49.18	54.92	49.17	52.52
Hypothyroid	26.39	9.91	33.30	38.32	**38.81**	37.85	16.01	7.44	35.21	37.81	35.21	34.46
Ionosphere	45.63	47.00	35.35	58.20	58.20	56.82	44.75	46.89	55.36	**60.61**	37.83	50.35
Mushroom	61.75	**84.61**	73.47	77.27	71.14	77.27	84.41	84.41	75.56	78.09	77.66	82.42
Pima	28.26	19.00	30.58	31.50	31.50	31.17	26.76	19.12	32.82	**35.95**	26.51	31.45
SickEuthyroid	31.74	17.08	32.90	37.90	36.64	39.65	20.62	12.17	35.28	**42.33**	33.59	34.56
Sonar	26.88	31.71	33.79	33.12	33.12	33.15	26.72	31.65	36.10	**42.42**	22.51	33.67
Spam	44.03	54.31	57.14	68.25	68.25	67.19	55.39	54.33	60.64	66.77	55.83	**68.26**
Tictactoe	17.09	19.82	**27.38**	17.54	10.61	17.54	20.27	20.27	19.21	21.63	16.04	18.71
Moyenne ALC	36.43	41.94	40.19	45.88	43.01	45.55	37.22	41.92	43.26	47.10	40.55	**47.50**

	W-J48	NB-M-BG	NB-M-M	NB-EW-M	SNB-EF-BG	SNB-EF-M	SNB-EW-BG	SNB-EW-M	SNB-M-BG	SNB-M-M	W-SCart
Adult	11.21	30.34	40.53	19.72	23.43	20.65	23.48	21.34	24.63	17.83	4.35
Australian	38.85	52.99	57.47	52.79	36.58	37.57	36.15	37.43	39.30	42.05	24.83
Breast	54.86	88.92	63.19	63.19	67.84	67.84	68.03	68.03	63.14	63.14	40.97
Bupa	6.78	9.76	1.31	1.31	8.87	8.87	8.14	8.14	1.30	1.30	4.23
Crx	39.50	51.15	54.52	52.17	36.12	37.66	36.06	37.56	38.72	42.01	26.97
German	8.43	16.92	3.82	3.82	9.83	9.83	10.95	10.95	3.81	3.81	2.88
Heart	25.93	53.66	53.81	39.91	34.19	33.94	32.90	32.73	36.44	34.35	17.85
Hepatitis	18.19	41.07	43.88	35.83	26.66	27.93	26.98	28.10	25.99	26.20	8.54
Horsecolic	31.30	34.56	48.65	39.77	34.61	27.70	35.41	28.48	37.81	30.81	1.67
Hypothyroid	11.47	38.19	18.51	14.28	19.32	19.50	29.47	29.63	14.87	14.49	7.11
Ionosphere	31.15	56.82	39.30	39.30	41.67	41.67	42.30	42.30	38.53	38.53	21.60
Mushroom	52.99	71.14	77.27	71.14	62.30	60.64	62.30	60.64	62.30	60.64	46.68
Pima	18.38	31.17	15.81	15.81	24.77	24.77	26.04	26.04	15.86	15.86	9.92
SickEuthyroid	16.31	38.65	22.80	9.66	25.84	25.53	27.55	27.60	9.53	8.90	10.63
Sonar	15.39	33.15	26.04	26.04	18.60	18.60	19.24	19.24	21.24	21.24	11.52
Spam	32.00	67.19	41.92	41.92	38.96	38.96	35.31	35.31	40.13	40.13	21.05
Tictactoe	8.92	10.61	17.54	10.61	13.53	9.83	13.53	9.83	13.53	9.83	5.50
Moyenne ALC	24.80	42.72	36.85	31.60	30.77	30.09	31.40	30.79	28.65	27.71	15.66

TAB. 4 – *ALC par algorithme et jeu de données mais avec l'aire entre* 2 *et* $2^6 = 64$ *exemples en apprentissage*

Impact de la taille de l'ensemble d'apprentissage



(a) Jeu de données Adult

(b) Jeu de données Mushroom

(c) Jeu de données Crx

(d) Jeu de données TicTacToe

FIG. 4 – *AUC en fonction du log_2 du nombre d'exemples appris*

Algorithme	Rang moyen	ALC moyen	AUC finale moyenne
W-RF40	**3.65**	**59.26**	**91.30**
NB-EF-BG	4.53	56.25	88.04
NB-EW-BG	6.47	55.27	86.82
W-VFI-N	6.82	55.33	83.13
NB-EF-M	7.06	54.55	88.04
W-RF10	7.29	55.56	89.70
W-BN	8.18	53.88	87.36
W-NB-S	8.53	53.89	87.35
NB-EW-M	9.00	53.57	86.80
W-ADT	10.88	51.01	88.66
W-Log100	10.88	52.12	88.55
W-NB-NS	11.00	50.51	87.09
NB-M-BG	11.94	49.93	86.81
W-VFI	13.65	48.86	82.04
SNB-EF-BG	14.88	46.96	88.15
SNB-EW-BG	15.47	46.59	87.06
NB-M-M	15.94	46.83	86.84
SNB-EF-M	16.06	46.54	88.20
SNB-EW-M	16.53	46.17	86.99
SNB-M-BG	16.53	44.97	87.07
SNB-M-M	17.82	44.46	87.19
W-J48	20.65	38.74	82.84
W-SCart	22.24	31.95	81.70

TAB. 5 – *Rang moyen. ALC moyen et AUC finale moyenne*

de données qui ne leur est pas favorable. L'un des meilleurs exemples est la figure 4(d) qui montre un jeu de données qui fonctionne très bien avec une régression logistique mais qui est très mauvais avec un Bayésien naïf.

4.3 Analyse

Analyse globale

Cette étude a été réalisée sans optimisation des paramètres des algorithmes mais principalement en utilisant les paramètres par défaut de ceux-ci. C4.5 (J48) et CART (SimpleCart) sont des références en apprentissage mais cette étude montre qu'ils ne sont pas adaptés pour bien travailler avec des petits jeux de données. En effet ils se retrouvent derniers que ce soit au niveau du rang moyen ou de l'ALC moyenne.

Au contraire, le Bayésien naïf est connu pour bien fonctionner sur de petits jeux de données (Domingos et Pazzani, 1997). Les figures 4(a) et 4(b) confirment ce résultat. Très rapidement les classifieurs générés sont capables de donner de bons résultats, on observe assez souvent qu'il faut entre 2^5 et 2^8 exemples en apprentissage pour être très proche de l'AUC maximale.

Les algorithmes basés sur les arbres ne fonctionnent pas bien directement (C4.5, Cart) mais une fois combinés avec des techniques de bagging/boosting ils sont parmi les meilleurs de cette étude : RandomForest. La figure 4 montre le bon comportement général des forêts d'arbres sur 4 jeux de données. Il faut noter que les sous-figures 4(c) et 4(d) mettent en avant des classifieurs (W-NB-S et W-Log100) qui sont particulièrement bien adaptés à ces deux jeux de données (Crx et TicTacToe). Même dans ces cas, RandomForest affiche de bons résultats ;

Impact de la taille de l'ensemble d'apprentissage

Algorithme	Rang moyen	ALC moyen	AUC finale moyenne
W-RF40	**4.35**	47.10	**86.21**
W-VFI-N	4.59	**47.50**	82.43
NB-EF-BG	4.65	45.88	83.45
NB-EW-BG	5.41	45.55	82.34
W-RF10	6.88	43.26	84.09
NB-EF-M	7.47	43.01	83.29
NB-EW-M	8.29	42.72	82.12
W-BN	8.88	41.94	81.58
W-NB-S	8.88	41.92	81.28
W-Log100	10.00	40.19	79.99
W-VFI	10.24	40.55	77.91
W-NB-NS	11.53	37.22	82.60
W-ADT	11.76	36.43	83.02
NB-M-BG	11.88	36.85	80.92
SNB-EW-BG	15.65	31.40	81.93
SNB-EF-BG	15.88	30.77	82.44
NB-M-M	16.18	31.60	80.29
SNB-EW-M	16.76	30.79	81.76
SNB-EF-M	17.12	30.09	82.20
SNB-M-BG	17.76	28.65	80.18
SNB-M-M	19.41	27.71	80.12
W-J48	19.94	24.80	74.14

TAB. 6 – *Rang moyen, ALC moyen et AUC finale moyenne mais avec l'aire entre* 2 *et* $2^6 = 64$ *exemples en apprentissage*

de manière générale il se situe mieux que les autres classifieurs sur la grande majorité des jeux de données.

Il est étonnant de voir apparaitre dans le haut du tableau de cette étude un algorithme assez peu connu : VFI. Cet algorithme se comporte particulièrement bien avec le paramètre qui ne pondère pas les intervalles des variables en fonction de la confiance ("weight feature intervals by confidence" : W-VFI-N). Si on s'arrête aux 64 premiers exemples en apprentissage (Tableau 6), il se retrouve premier en ALC et deuxième en termes de rang avec un très faible écart par rapport au premier.

Analyse pour les modèles discriminants

Les algorithmes discriminants sont représentés dans cette étude par les forêts d'arbre (RF et RF40) et la régression logistique (Log100). L'algorithme RandomForest paramétré avec une taille de forêt de 40 arbres est le meilleur de cette étude sur tous les points de vue, premier pour le rang, l'ALC moyenne et l'AUC finale moyenne. La possibilité de cet algorithme à essayer de nombreux arbres avec seulement une partie des variables explicatives lui permet d'être à la fois bon en début d'apprentissage ainsi qu'en fin.

Bien qu'elle se positionne après RandomForest, le Bayésien naïf et VFI, la régression logistique n'est pas si éloignée au niveau de son score d'ALC. Dans certains cas particuliers elle fonctionne même bien mieux que les autres méthodes comme le montre la figure 4(d) sur le jeu de données TicTacToe.

Analyse pour les modèles génératifs

Parmi les modèles génératifs testés dans cette étude, le Bayésien naïf est le plus évalué. On constate que la discrétisation des variables continues et le groupement des variables catégorielles influencent grandement les résultats. Le meilleur choix est constitué par la paire ("EqualFrequency", "Basic Grouping") suivi de ("EqualWidth", "Basic Grouping"). La discrétisation régularisée (Boullé, 2006b) et la méthode de groupement MODL (Boullé, 2005) sont trop robustes pour être capables de générer un modèle avec peu d'exemples. Dans le cadre de cette étude, les méthodes "les plus simples" ont l'avantage de s'exprimer plus rapidement et donc de donner de meilleurs résultats.

Les méthodes régularisées (approche MODL et Bayésien naïf sélectif) sont connues pour avoir un faible biais-variance (Cucker et Smale, 2008). Par conséquent afin de maintenir ce faible biais-variance, leur variance et donc aussi leur AUC va être plus faible en début d'apprentissage. Cette plus grande robustesse les amène à ne pas prédire un résultat très variable et donc à rester "silencieuses". Ces méthodes se retrouvent donc avec des scores faibles dans le cadre de cette étude et se positionnent en dernières positions bien qu'elles soient quand même meilleures que C4.5 et Cart.

Les modèles génératifs générant un classifieur non linéaire (voir table 1) sont représentés dans cette étude par les réseaux Bayésiens. Ils se comportent assez bien mais se situent juste après les RandomForest, VFI et le Bayésien naïf. Sur certains jeux de données : Adult et Mushrooms (figure 4(b)), ils sont les meilleurs classifieurs en termes d'ALC.

Comparaisons avec des analyses existantes

Les résultats de la littérature [5] sont confirmés dans cette étude : les méthodes par vote et ensemble de classifieurs ont de très bons résultats Bauer et Kohavi (1999), les modèles génératifs sont meilleurs que les modèles discriminants quand le nombre d'exemples est faible (Bouchard et Triggs, 2004), les méthodes régularisées sont robustes (Cucker et Smale, 2008).

5 Conclusion et travaux futurs

5.1 Vers un nouveau critère ?

Nous avons vu précédemment les résultats présentés dans deux tableaux : un contenant les résultats de l'aire sous la courbe d'AUC (ALC) s'arrêtant à 2^6 exemples (tableau 6) et un autre donnant l'AUC finale (tableau 5). Le premier tableau indique le comportement de l'algorithme au début de l'apprentissage et le second sa performance en fin d'apprentissage. Les deux axes de la figure 5 reprennent ces deux critères d'évaluation. En moyenne, seul "RandomForest" et "NB-EF-BG" arrivent à être bien positionnés au niveau de ces deux critères. La régression logistique et le boosting d'arbres : ADTree ont de bonnes performances en AUC finale mais leurs ALC sont relativement faibles. Au contraire les algorithmes de type "VFI" ont une bonne ALC mais une mauvaise AUC finale.

5. La plupart des études de la littérature ne sont pas réalisée sur de si petits jeux de données, par conséquence pour les comparer il faut se placer après avoir appris sur au moins $2^7(128)$ à $2^{10}(1024)$ exemples.

Impact de la taille de l'ensemble d'apprentissage

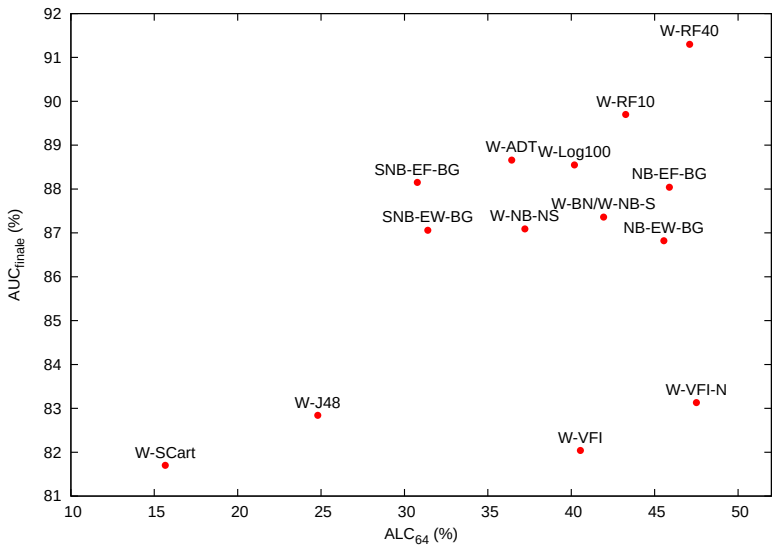


FIG. 5 – *Positionnement des différentes méthodes en termes d'*ALC_{2^6} *et d'*AUC_{finale}

Cette analyse pose la question de faire évoluer le critère d'ALC vers un nouveau critère qui prendrait en compte ces deux aspects. En effet, il serait intéressant d'avoir un critère synthétique nous permettant d'évaluer la qualité d'un algorithme selon ces deux axes. Ce critère pourrait par exemple s'écrire sous la forme suivante :

$$Critère = \frac{ALC_{2^6} + AUC_{finale}}{2}$$

5.2 Combinaison d'algorithmes

Dans cette étude, peu de cas se présentent où un algorithme particulier est très bon sur le début de l'apprentissage et un autre algorithme est très bon sur la fin de l'apprentissage. Cependant si nous étions dans une telle situation, c'est à dire comme celle de la figure 6, il serait intéressant d'utiliser deux algorithmes distincts pour réaliser l'apprentissage : W-VFI-N pour les 2^8 premiers exemples puis une Bayésien naïf sélectif pour la fin de l'apprentissage.

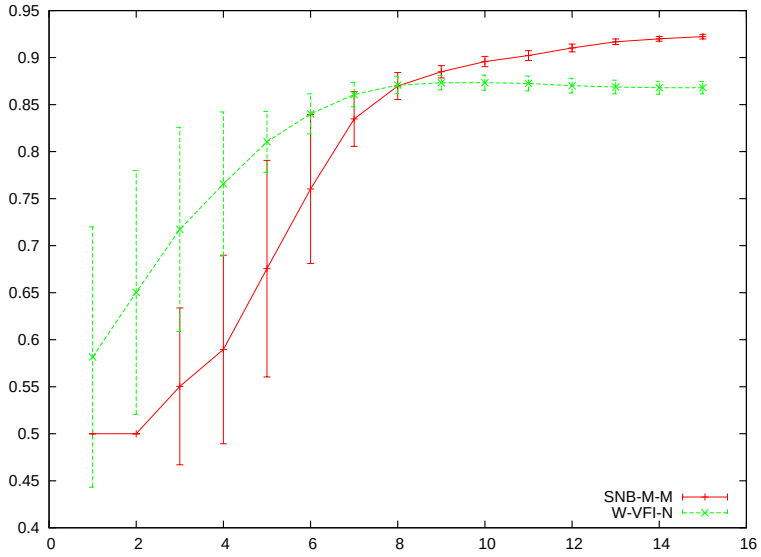


FIG. 6 – *Un jeu de données (Adult) où une combinaison d'algorithmes est souhaitable*

5.3 Recommandations

Il ressort de cette étude que plusieurs algorithmes peuvent être utilisés pour construire un modèle sur un ensemble d'apprentissage de faible taille : le Bayésien naïf qui était attendu pour ces qualités sur de faible quantité de données et les forêts d'arbres. Ces deux algorithmes nécessitent d'être paramétrés pour obtenir leurs meilleurs performances : les couples ("Equal-Frequency", "Basic Grouping") et ("EqualWidth", "Basic Grouping") fonctionnent le mieux pour le Bayésien naïf et une forêt de 40 arbres pour "RandomForest". Si l'on s'intéresse plus particulièrement au tout début de l'apprentissage, une méthode par vote sur des intervalles (VFI) se positionne au même niveau que les meilleures méthodes de cette étude.

Références

Bauer, E. et R. Kohavi (1999). An empirical comparison of voting classification algorithms : Bagging, boosting, and variants. *Machine learning 36*(1), 105–139.

Blake, C. L. et C. J. Merz (1998). UCI Repository of machine learning databases. `http://archive.ics.uci.edu/ml/` visité pour la dernière fois : 15/09/2010.

Bouchard, G. et B. Triggs (2004). The tradeoff between generative and discriminative classifiers. In *IASC International Symposium on Computational Statistics (COMPSTAT)*, pp.

721–728.

Bouckaert, R. (2004). Bayesian Network Classifiers in Weka. `http://weka.sourceforge.net/manuals/weka.bn.pdf`.

Boullé, M. (2004). Khiops : A Statistical Discretization Method of Continuous Attributes. *Machine Learning 55*(1), 53–69.

Boullé, M. (2005). A grouping method for categorical attributes having very large number of values. *Machine Learning and Data Mining in Pattern Recognition*, 228–242.

Boullé, M. (2006a). MODL : A Bayes optimal discretization method for continuous attributes. *Machine Learning 65*(1), 131–165.

Boullé, M. (2006b). Regularization and Averaging of the Selective Naive Bayes classifier. *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, 1680–1688.

Breiman, L. (2001). Random forests. *Machine learning 25*(2), 5–32.

Breiman, L., J. Friedman, R. Olshen, et C. Stone (1984). *Classification and regression trees*. Chapman and Hall/CRC.

Cessie, S. L. et J. V. Houwelingen (1992). Ridge estimators in logistic regression. *Applied Statistics*.

Cucker, F. et S. Smale (2008). Best Choices for Regularization Parameters in Learning Theory : On the Bias-Variance Problem. *Foundations of Computational Mathematics 2*(4), 413–428.

Demiröz, G. et H. Güvenir (1997). Classification by voting feature intervals. *Machine Learning : ECML-97*, 85–92.

Domingos, P. et G. Hulten (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 71–80. ACM New York, NY, USA.

Domingos, P. et M. Pazzani (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine learning 130*, 103–130.

Fawcett, T. (2004). ROC graphs : Notes and practical considerations for researchers. *Machine Learning 31*, 1–38.

Féraud, R., M. Boullé, F. Clérot, F. Fessant, et V. Lemaire (2010). The orange customer analysis platform. In *Industrial Conference on Data Mining (ICDM)*, pp. 584–594.

Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics 7*, 179–188.

Freund, Y. et L. Mason (1999). The alternating decision tree learning algorithm. In *Machine learning*, pp. 124–133.

Gama, J., P. Medas, et P. Rodrigues (2005). Learning Decision Trees from Dynamic Data Streams. In *Proceedings of the ACM Symposium on Applied ComputingLearning Decision Trees from Dynamic Data Streams*.

Gama, J., R. Rocha, et P. Medas (2003). Accurate decision trees for mining high-speed data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 523–528. ACM New York, NY, USA.

Guyon, I., G. Cawley, G. Dror, et V. Lemaire (2010). Results of the Active Learning Challenge. In *JMLR W&CP, Workshop on Active Learning and Experimental Design, collocated with*

AISTATS, Sardinia, Italy, Volume 10, pp. 1–26.

Guyon, I., V. Lemaire, G. Dror, et D. Vogel (2009). Analysis of the kdd cup 2009 : Fast scoring on a large orange customer database. *JMLR : Workshop and Conference Proceedings 7*, 1–22.

John, G. et P. Langley (1995). Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 338–345. Morgan Kaufmann.

Langley, P., W. Iba, et K. Thompson (1992). An analysis of Bayesian classifiers. In *Proceedings of the National Conference on Artificial Intelligence*, Number 415, pp. 223–223.

Lim, T., W. Loh, et Y. Shih (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine learning 40*(3), 203–228.

Michalski, R. S., I. Mozetic, J. Hong, et N. Lavrac (1986). The Multi-Purpose incremental Learning System AQ15 and its Testing Application to Three Medical Domains. *Proceedings of the Fifth National Conference on Artificial Intelligence*, 1041–1045.

Quinlan, J. R. (1993). *C4.5 : programs for machine learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc.

Settles, B. (2010). Active learning literature survey. `http://pages.cs.wisc.edu/~bsettles/pub/settles.activelearning.pdf`.

Witten, I. H. et E. Frank (2005). *Data mining : practical machine learning tools and techniques*. Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, second edition.

Summary

Learning algorithms proved their ability to deal with large amount of data. Most of the statistical approaches use defined size learning sets and produce static models. However in specific situations: active or incremental learning, the learning task starts with only very few data. In that case, looking for algorithms able to produce models with only few examples becomes necessary. The literature's classifiers are generally evaluated with criteria such as: accuracy, ability to order data (ranking)... But this classifiers' taxonomy can really change if the focus is on the ability to learn with just few examples. To our knowledge, just few studies were performed on this problem. This study aims to study a larger panel of both algorithms (9 different kinds) and data sets (17 UCI bases).