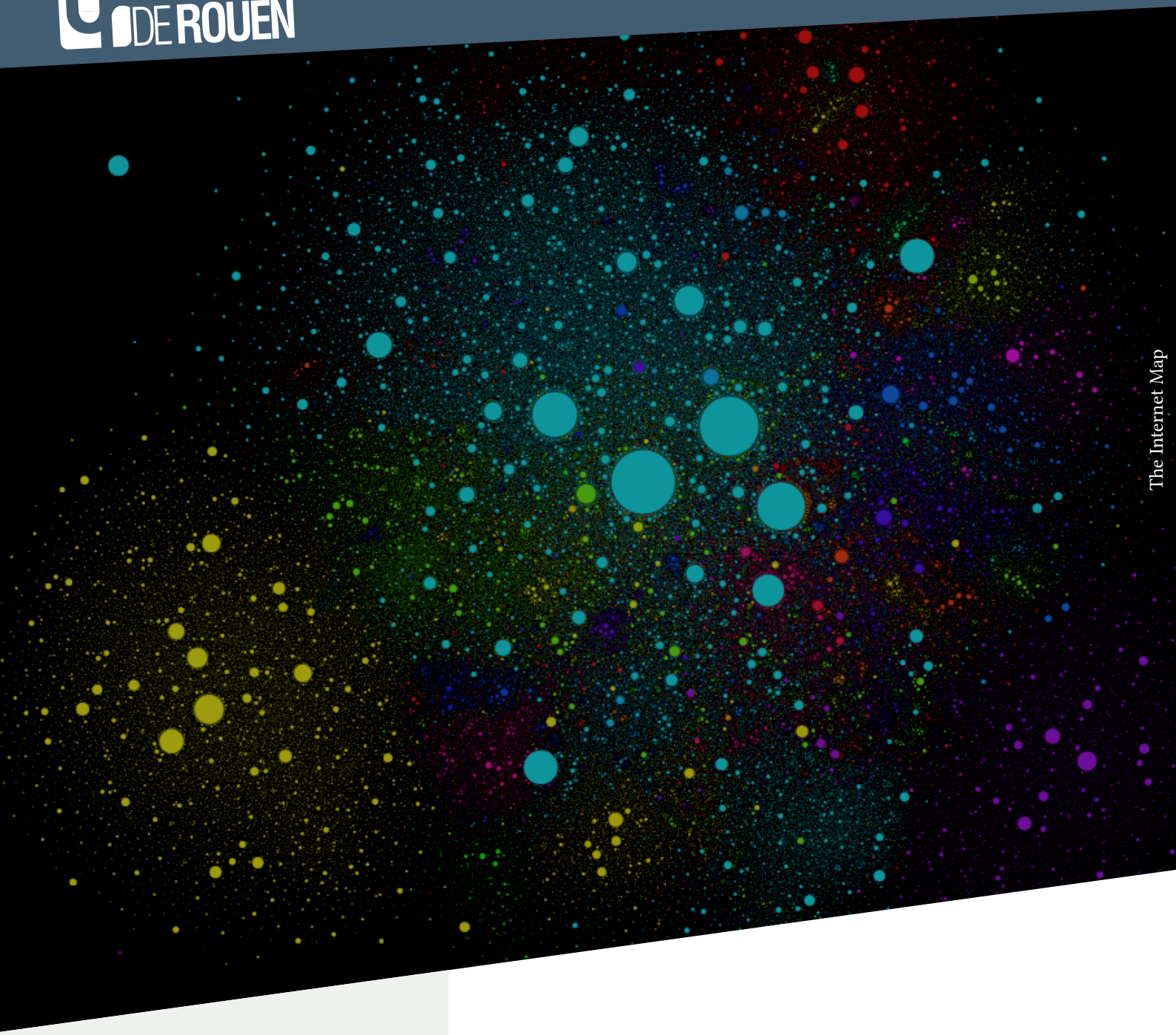
UNIVERSITÉ DE ROUEN



The Internet Map

Fusion d'Information

Année 2014-2015

Compte-rendu de TP

Méthodes de construction d'ensembles de classifieurs

Thomas Robert

1 | Bagging

1.1 Principe

Le principe du *bagging* est inspiré du *bootstrap*. Dans les deux cas, cette méthode consiste à tirer B jeux d'apprentissage par tirages aléatoires avec remise sur le jeu de données initial. Le résultat final consiste à « mélanger » les B résultats unitaires obtenus pour chaque jeu d'apprentissage afin d'obtenir un résultat plus stable et plus performant en généralisation.

Le « mélange » dépend de ce qu'on évalue. Dans le cas du *bootstrap* on évalue une valeur, par exemple l'écart-type, le « mélange » consiste donc dans ce cas à faire la moyenne des écarts-type.

Selon le même pricipe, par *bagging*, on apprend un classifieur, le « mélange » pourra par exemple être un vote majoritaire (mieux vaut dans ce cas choisir B astucieusement pour éviter des égalités...).

Notons également que le *bootstrap* permet également de calculer une sorte d'intervalle de confiance sur le résultat que l'on obtient. Il suffit de calculer l'écart-type de l'ensemble des B valeurs trouvées. Pour le *bagging*, un principe similaire permet de calculer l'erreur en test (données non tirées dans l'apprentissage, nommées *out-of-bag*) pour chaque tirage, et l'écart-type de cette erreur. On peut ainsi avoir une idée des performances moyenne de chaque classifieur pris unitairement.

1.2 Résultats

Pima La méthode de *bagging* a été essayée sur le jeu de données *pima* (jeu à 2 classes) avec 10 classifieurs de type seuil (ou souche binaire).

Afin d'observer une amélioration des performances du mélange par rapport à un classifieur seul, il a fallu diminuer la taille du jeu d'apprentissage afin de forcer les classifieurs à être suffisamment diversifiés. Ainsi, seul 10% du jeu de données a été utilisé en apprentissage.

Les résultats finaux, obtenu par moyennage des résultats sur 20 découpages aléatoires du jeu, sont les suivants :

- Erreur *out-of-bag* de chaque classifieur du mélange pris seul : $33.6 \pm 7.6\%$
- Taux d'erreur en test du mélange en *bagging* : 28.1%
- Taux d'erreur en test du classifieur seul : 29.2%

Satimage La méthode a également été essayée sur le jeu *satimage* (jeu à 6 classes) avec 30 classifieurs de type *1-ppv*.

Cette fois, 30% du jeu de données a été conservé pour l'apprentissage. Les résultats (moyennage du 10 découpages aléatoires) sont :

- Erreur *out-of-bag* de chaque classifieur du mélange pris seul : $11.9 \pm 1.0\%$
- Taux d'erreur en test du mélange en *bagging* : 11.2%
- Taux d'erreur en test du classifieur seul : 11.3%

Conclusion On constate que les résultats permettent d'améliorer légèrement les performances par rapport au classifieur seul, sans que la différence soit énorme. Cela peut tout simplement être du aux données qui ne se prêtent pas bien a la méthode avec le classifieur testé.

2 | Boosting & AdaBoost

2.1 Principe

Le *boosting* consiste à construire un ensemble de classifieurs de façon séquentielle de façon à ce qu'un classifieur classe mieux les points ayant le plus mal été classifiés par les classifieurs précédents. Chaque classifieurs est également pondéré par ses performances.

2.2 Résultats

On essaye cette méthode sur les jeux *pima* et *satimage*. Dans les deux cas on utilise une souche binaire comme classifieur de base, car elle permet de pondérer l'importance de chaque point vis-à-vis de l'erreur de classification à minimiser.

On affichera pour chaque jeu de données l'évolution des performances du mélange au fur et à mesure de l'ajout de classifieurs dans l'ensemble. La performance du classifieur seul est donc visible pour un "ensemble" de 1 classifieur.

Pima On crée un ensemble de 10 classifieurs en utilisant 70% du jeu de données pour l'apprentissage. On obtient les résultats (en moyenne sur 10 découpages du jeu de données) présentés sur la figure 1.

On constate que l'utilisation d'un mélange de classifieurs permet d'améliorer les performances, passant de 26 à 21 % d'erreur en test.

Satimage On a dans ce cas un jeu de données à 6 classes, on apprend donc 6 mélanges de 30 classifieurs, selon la méthode *1 vs all*. Chaque mélange décide donc soit de voter pour la classe pour laquelle il a été appris, soit de rejeter (vote *all*). La décision globale consiste à rejeter en cas de conflit entre deux mélanges ou si tous les mélanges rejettent le point.

Pour apprendre cet ensemble *(1 vs all)* d'ensembles *(AdaBoost)*, on utilise 50 % du jeu de données. On obtient les résultats présentés en figure 2.

On constate que l'ajout de classifieurs dans les mélanges AdaBoost améliore légèrement le taux de rejet, passant de 7.2 % pour 1 classifieur à 6.5 % pour 30 classifieurs, mais surtout, on diminue énormément le taux de rejet. En effet, le taux de rejet pour 1 seul classifieur est très important. Cela est très probablement dû au fait que tous les mélanges AdaBoost du mélange *1 vs all* rejettent l'observation. L'ajout de davantage de classifieurs dans les mélanges AdaBoost permet de prendre de meilleures décisions, diminuant le taux de rejet.

Conclusion On constate que la méthode d'AdaBoost permet d'améliorer les performances d'un classifieur de base par la construction d'un ensemble de classifieurs de façon a ce qu'un

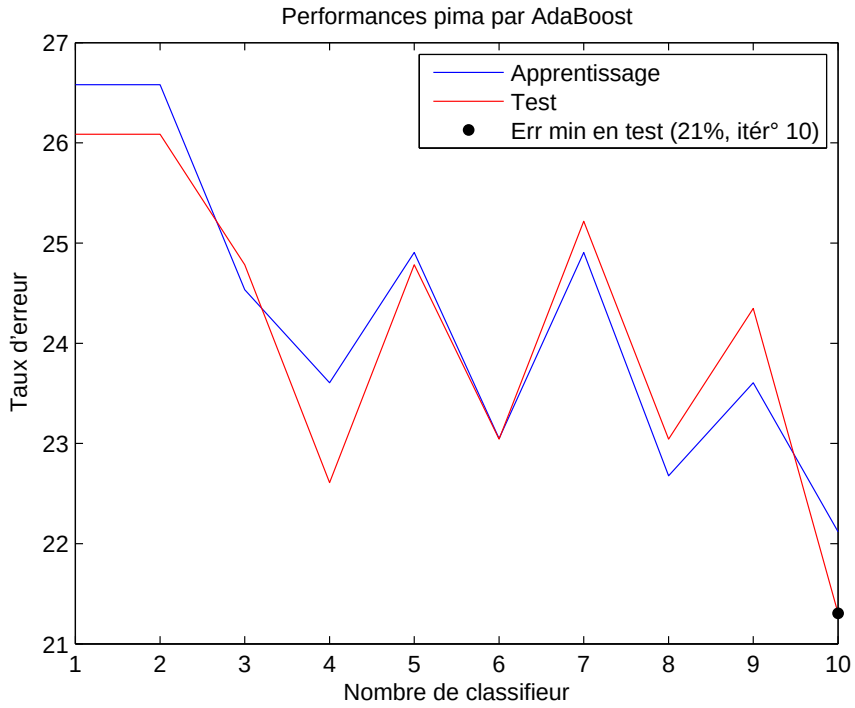


FIGURE 1 – Résultats moyens par *AdaBoost* sur le jeu de données *pima*

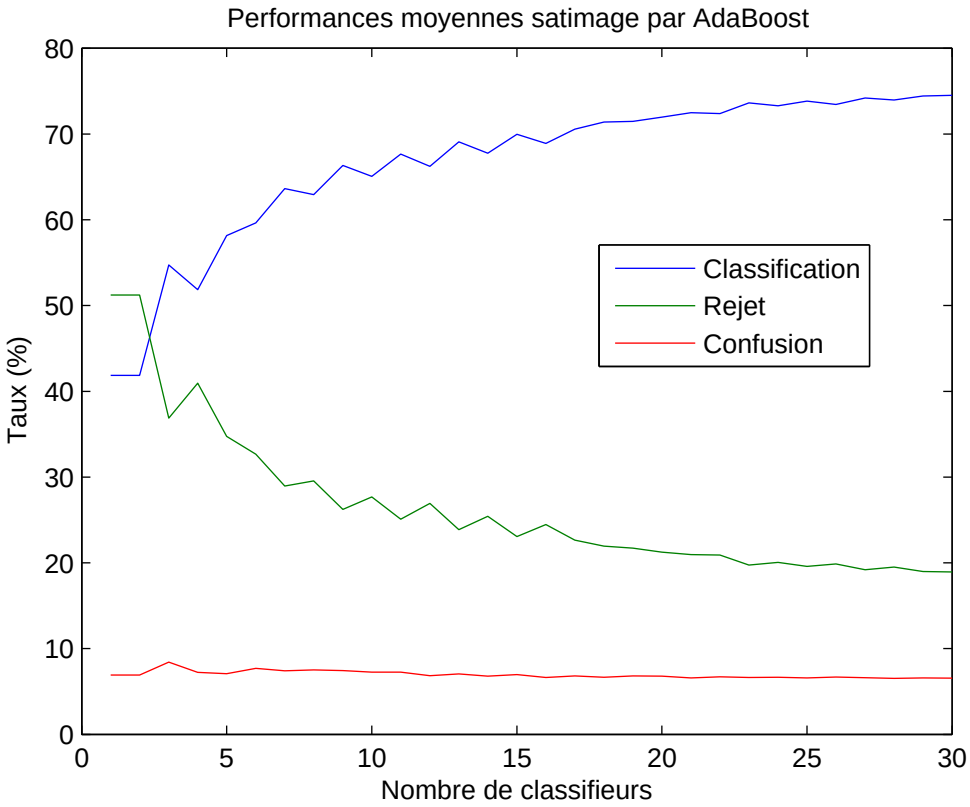


FIGURE 2 – Résultats moyens par *AdaBoost* sur le jeu de données *satimage*

classifieur apprenne en fonction des performances des classifieurs déjà présent dans l'ensemble, permettant d'améliorer les performances.

3 | Random Subspace Method

3.1 Principe

La méthode des *Random Subspace* consiste à construire un ensemble de classifieur où chaque classifieur n'utilise qu'un sous-ensemble des dimensions du jeu de données initial.

Utiliser un sous-espace réduit de l'espace initial permet de ne pas prendre en compte l'intégralité des dimensions, ce qui permet par exemple de moins souffrir de la malédiction de la dimensionalité *(curse of dimensionality)*.

Par ailleurs, cette méthode permet de forcer les classifieurs du mélange à présenter de la diversité, élément clé des méthodes d'ensembles. Dans ce sens, cette méthode ressemble un peu à la méthode de *bagging*, où l'utilisation d'une partie du jeu de données seulement permet de créer des classifieurs différents.

3.2 Résultats

Pima La méthode a été essayée sur le jeu de données *pima* (jeu à 2 classes et 8 dimensions) avec 10 classifieurs de type seuil (ou souche binaire), et 3 dimensions dans chaque sous-espace. Le mélange a été appris en utilisant 50 % du jeu de données en apprentissage, et les résultats sont obtenus par moyenne des résultats sur 20 tirage du découpage du jeu entre apprentissage et test.

On obtient les résultats suivants :

- Erreur en apprentissage de chaque classifieur seul : $29.2 \pm 3.2\%$
- Taux d'erreur en test du mélange *RSM* : 29.2%
- Taux d'erreur en test du classifieur seul : 26.0%

Satimage La méthode a également été essayée sur le jeu *satimage* (jeu à 6 classes et 36 dimensions) avec 30 classifieurs de type *1-ppv* et 20 dimensions dans les sous-espaces.

Cette fois, 30% du jeu de données a été conservé pour l'apprentissage. Les résultats (moyennage du 10 découpages aléatoires) sont :

- Erreur en apprentissage de chaque classifieur seul : 0% puisqu'il s'agit d'un *1-ppv*
- Taux d'erreur en test du mélange *RSM* : 10.3%
- Taux d'erreur en test du classifieur seul : 11.5%

Conclusion On constate que les performances du RSM sont variable, puisque la méthode dégrade les performances sur le jeu *pima* alors qu'elle les améliore sur le jeu *satimage*.

Concernant le jeu *pima*, cela peut être du au fait que peu de dimensions permettent de découper le jeu de données avec de bonnes performances. L'utilisation de sous-espaces crée des classifieurs ayant de mauvaises performances, et réduisant les performances du mélange.

On voit en revanche que la méthode améliore les performances sur le jeu de données *satimage*, où le classifieur de base, le *1-ppv*, est sensible aux phénomènes de dimensionnalité.