Arbre de décision

Maxime CHAMBREUIL
maxime.chambreuil@insa-rouen.fr

Table des matières

Dans ce TP , on se restreindra à des problèmes de classification à 2 classes.

1 Explication

1.1 Initialisation

On a d'abord créé 2 séries de 150 points respectant la loi normale avec des moyennes différentes. Chacune de ses séries constitue une classe, on a donc un vecteur d'appartenance avec des 0 ou des 1. On a ensuite séparé ses 2 séries pour obtenir 3 ensembles : des points d'apprentissage, de validation et de test.

1.2 Les fonctions

1.2.1 Treeclass

T=treeclass(X,yapp,'splitmin',S,'method','classification','splitcriterion','gdi', 'prune','off')

En entrée :
X : Matrice des points d'apprentissage
yapp : Vecteur d'appartenance des points d'apprentissage
splitmin , S : permet de spécifier le nombre minimal de points dans une classe impure
classification : spécifie que l'on veut un arbre de classification
splitcriterion, gdi : permet de spécifier le critère de séparation, ici Gini
prune, off : pour spécifier si on veut élaguer l'arbre ('on') ou pas

En sortie :
T : Arbre de décision complet

Cette fonction nous permet de générer notre arbre de décision à l'aide des données d'apprentissage.

1.2.2 Treetesting

y = treetesting (T , xtest)

En entrée :
T : Arbre de décision
xtest : Matrice des points de test

En sortie :
y : Vecteur d'appartenance des points de test (1 ou 2)

Cette fonction permet d'obtenir la classe de nos points de test, en les soumettant à notre arbre de décision. Pour exploiter les résultats, il ne faut pas oublier de retirer 1 à notre vecteur des classes.

1.2.3 Treepruning

T2 = treepruning (T , 'level' , 1)

En entrée :
T : Arbre de décision complet level, 1 : pour élaguer le premier niveau

En sortie :
T : Arbre de décision élagué

Cette fonction permet d'afficher l'arbre de décision avec les questions posées à chaque noeud.

1.2.4 Treedisp

treedisp (T)

En entrée :
T : Arbre de décision

Cette fonction permet d'afficher l'arbre de décision avec les questions posées à chaque noeud.

1.2.5 optimisationArret

[Nopt , Topt] = optimisationArret (X , yapp , xval , yval , N)

En entrée :
X : Matrice des points d'apprrentissage
yapp : Vecteur des classes d'appartenance pour les points d'apprentissage
xval : Matrice des points de validation
yval : Vecteur des classes d'appartenance pour les points de validation (0 ou 1)
N : nombre maximal de points dans une classe impure

En sortie :
Nopt : Nombre maximal de points dans un noeud Topt : Arbre de décision optimal, obtenu pour Nopt

Cette fonction diminue le nombre de maximal de points dans un noeud (N) et calcule à chaque fois l'abre de décision. Avec les données de validation, on s'arrête dés que l'erreur commence à augmenter pour éviter le surapprentissage ("Early stopping"). On retourne alors l'arbre de décision optimal avec le nombre de points maximal dans un noeud qui a permis de l'obtenir.

1.3 Déroulement de l'algorithme

On a commencé par générer notre arbre complet pour tracer la frontière de décision. puis nous avons élagué notre arbre. Ensuite, nous avons généré un arbre optimal avec la fonction "optimisationArret". A chaque fois nous avons calculé le nombre de points bien classé. A l'aide de la fonction "ginput", nous avons rajouté un point dans la classe 0 en spécifiant qu'il était dans la classe 1, pour tester la stabilité de l'algorithme.

2 Observations

2.1 Figures

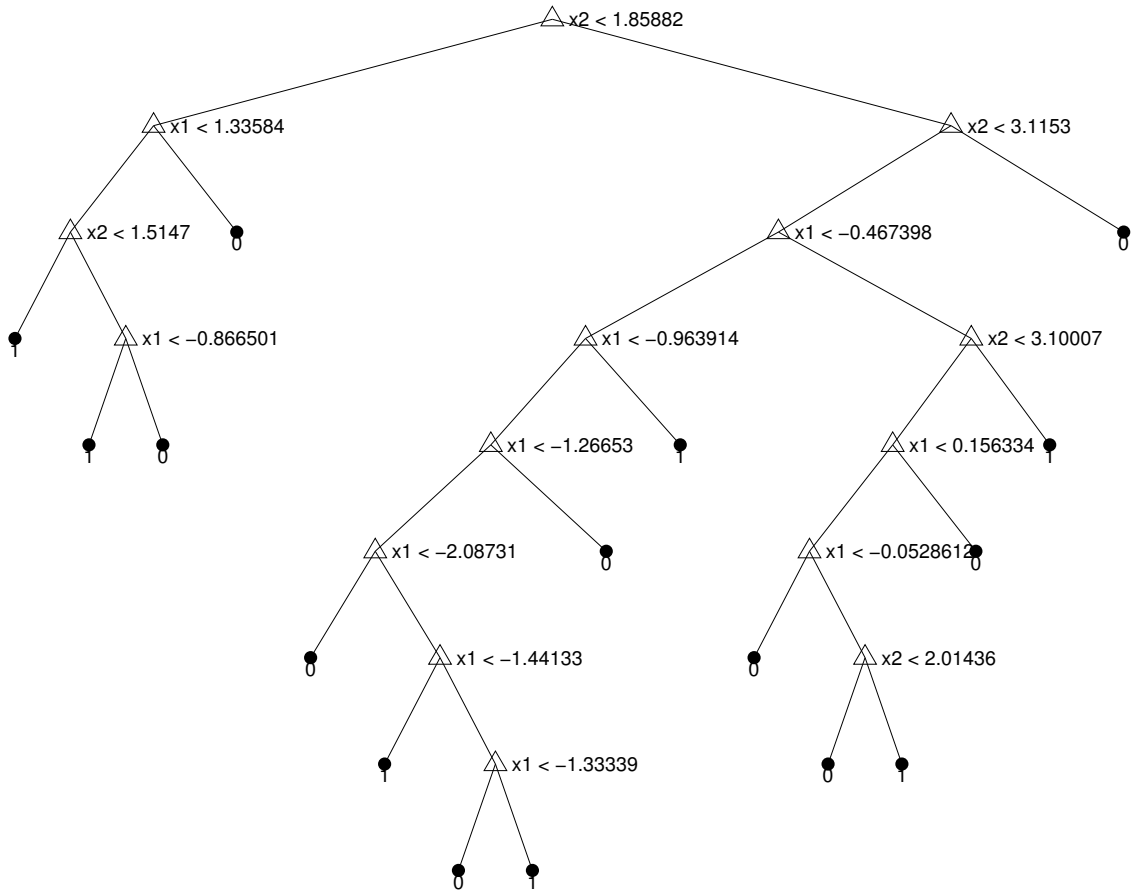

FIG. 1 – Arbre de décision complet

2.1.1 Critère d'arrêt : Nombre de points dans un noeud

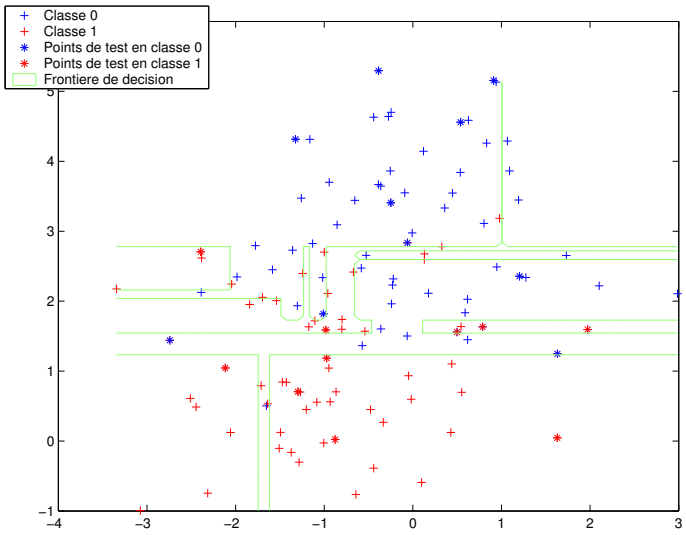

FIG. 2 – Frontière de décision avec un point par noeud

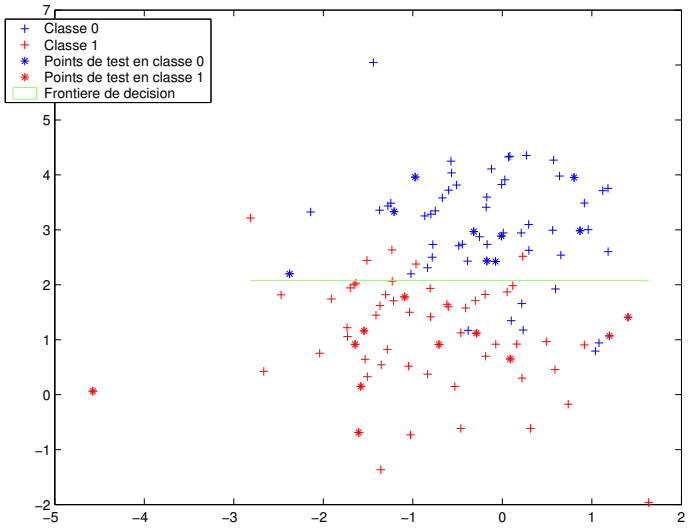

FIG. 3 – Frontière de décision avec 20 points par noeud

2.1.2 Critère d'arrêt : Augmentation de l'erreur en validation

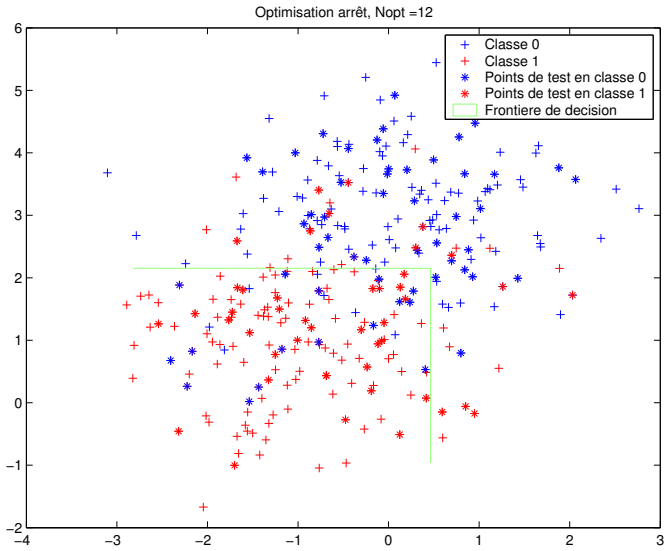

FIG. 4 – Frontière de décision avec les données de validation

2.1.3 Stabilité

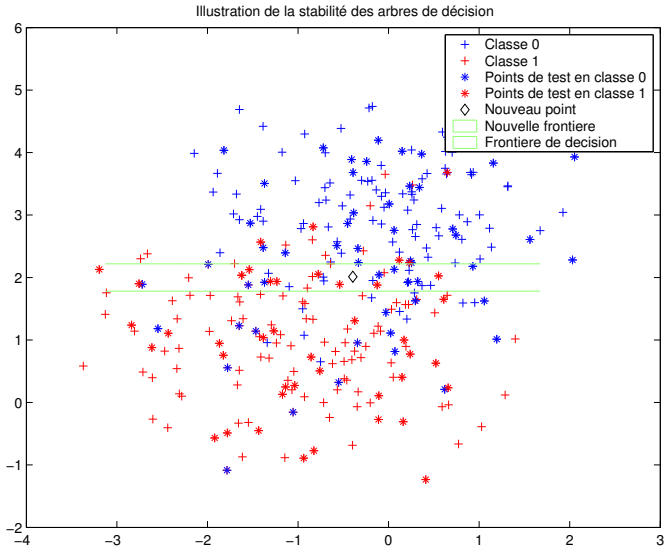

FIG. 5 – Stabilité de l'algorithme

2.2 Pourcentage de bien classé

Méthode	Expérience 1	Exp 2	Exp 3	Exp 4
Arbre avec 1 point par noeud	84	85	78	84
Arbre avec 20 points par noeud	87	84	82	86
Arbre élagué (niveau 1)	85	88	80	87
Arbre élagué (niveau 2)	84	89	81	90
Arbre élagué (niveau 3)	50	85	81	50
Arbre optimisé	84	84	79	85

3 Interprétations

Concernant le critère d'arrêt, plus il y a de points dans un noeud pour la génération de notre arbre, plus la frontière de décision est simple et tend vers une droite non-brisée. Il semble que notre seuil de 20 points soit très proche du nombre de points optimum dans un noeud car les résultats sont similaires à ceux de l'erreur en validation, qui permet d'optimiser ce critère d'arrêt.

On remarque aussi que plus on élague de niveaux, plus le taux de bonne classification augmente puis diminue. On passe donc par un optimum qui ne permet pas d'être atteint avec la méthode d'élagage par niveau.

Enfin on peut noter que l'ajout de notre point (le diamant noir) a fait passer la frontière de décision de 1,9 à 2,1.

4 Conclusion

Ce TP nous a permis de montrer qu'un arbre de décision optimisé permet d'avoir un bon taux de classification, avec un temps de calcul dérisoire. Mais il ne faut pas oublier que nos échantillons n'étaient pas très importants (100 points). Par ailleurs, l'algorithme est assez instable.

Ce TP était intéressant dans la mesure on se rend bien compte du comportement de l'algorithme avec le déplacement de la frontière de décision. Comme l'algorithme est plutôt performant, nous avons pu faire beaucoup de tests et essayer différentes valeurs de paramètres.

Le seul inconvénient que je trouve à ce TP, c'est l'utilisation de la fonction "treedisp" qui est soumis à la disponibilité d'un jeton.