

TD 4 — Arbres de décision : impureté, profondeur, élagage

Séance 4

Christophe Ambroise

2026-09-04

Objectifs

Comprendre pourquoi le taux d'erreur ne peut pas servir à faire croître un arbre, puis construire un arbre sur Iris : mesurer l'effet de la profondeur, parcourir le chemin d'élagage de Breiman, choisir la pénalité α par validation croisée, et lire — avec prudence — l'importance des variables.

Les exercices 2 à 4 portent sur Iris : 150 fleurs, quatre mesures en centimètres, trois espèces équilibrées. L'exercice 1 se traite **sans ordinateur**, sur un nœud fictif à deux classes.

Partie théorique

Exercice 1 — Trois mesures d'impureté, et une découpe invisible

Pour un nœud contenant une proportion p d'observations de la classe 1, on compare trois mesures d'impureté :

$$\text{erreur : } \min(p, 1 - p), \quad \text{Gini : } 2p(1 - p), \quad \text{entropie : } -p \log_2 p - (1 - p) \log_2 (1 - p).$$

Le gain d'une découpe qui envoie n_g observations à gauche et n_d à droite est

$$\Delta i = i(t) - \frac{n_g}{n} i(t_g) - \frac{n_d}{n} i(t_d).$$

Considérons un nœud de **400 observations, dont 100 de classe 1**, et une découpe qui produit :

- à gauche : 250 observations, dont 50 de classe 1 ;
- à droite : 150 observations, dont 50 de classe 1.

1. Vérifier que les deux fils sont **strictement moins impurs** que le père au sens de Gini. Calculer les trois impuretés du père.
2. Calculer Δi pour les trois mesures. Que constate-t-on pour le taux d'erreur ?
3. Expliquer géométriquement : que vaut Δi lorsque les deux fils tombent **du même côté** de $p = 1/2$ et que i est **linéaire** sur cet intervalle ?
4. En déduire pourquoi CART fait croître l'arbre avec Gini ou l'entropie, mais l'élague avec le taux d'erreur.

Partie empirique

Exercice 2 — L'effet de la profondeur

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import StratifiedKFold, cross_validate

iris = pd.read_csv("../data/iris.csv")
X, y = iris.drop(columns="species"), iris["species"]
plis = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)

for p in range(1, 9):
    arbre = DecisionTreeClassifier(max_depth=p, random_state=0)
    s = cross_validate(arbre, X, y, cv=plis, return_train_score=True)
    print(p, 1 - s["train_score"].mean(), 1 - s["test_score"].mean())
```

```
library(rpart)
iris_df <- read.csv("../data/iris.csv")
iris_df$species <- factor(iris_df$species)

# rpart limite la profondeur par `maxdepth` ; cp = 0 désactive l'élagage.
ctl <- rpart.control(maxdepth = 3, cp = 0, minsplit = 2, minbucket = 1, xval = 0)
arbre <- rpart(species ~ ., data = iris_df, method = "class", control = ctl)
```

1. Tracer l'erreur d'apprentissage et l'erreur de validation croisée en fonction de la profondeur maximale, de 1 à 8.
2. À quelle profondeur l'erreur d'apprentissage atteint-elle zéro ? Combien l'arbre a-t-il alors de feuilles, pour 150 observations ?
3. Que vaut l'erreur en profondeur 1 ? Interpréter ce chiffre à partir de la structure d'Iris.
4. L'erreur de validation croisée remonte-t-elle après son minimum ? Comparer avec la courbe en U de la séance 1, et expliquer l'écart.
5. `max_depth` contraint toutes les branches de la même façon. Pourquoi est-ce une objection sérieuse ?

Exercice 3 — Élagage par coût-complexité

Le cœur de CART :

$$R_\alpha(T) = R(T) + \alpha |T|,$$

où $R(T)$ est le taux d'erreur d'apprentissage et $|T|$ le nombre de feuilles.

```
# La suite emboîtée et ses alpha de coupure
complet = DecisionTreeClassifier(random_state=0).fit(X, y)
chemin = complet.cost_complexity_pruning_path(X, y)
alphas = chemin.ccp_alphas[:-1] # le dernier réduit l'arbre à sa racine

# Un arbre élagué pour un alpha donné
arbre = DecisionTreeClassifier(random_state=0, ccp_alpha=alphas[3]).fit(X, y)
arbre.get_n_leaves()
```

```
# rpart élague nativement : `cptable` EST le chemin, avec sa validation croisée.
complet <- rpart(species ~ ., data = iris_df, method = "class",
                 control = rpart.control(cp = 0, minsplit = 2, xval = 10))
cp <- as.data.frame(complet$cptable) # colonnes CP, nsplit, rel error, xerror, xstd
petit <- prune(complet, cp = 0.02)
```

1. Combien d'arbres la suite emboîtée contient-elle? Vérifier que le nombre de feuilles **décroît** quand α croît, et que l'erreur d'apprentissage **croît**.
2. Estimer l'erreur de chaque arbre du chemin par validation croisée à dix plis. Où se trouve le minimum? Combien de feuilles a l'arbre retenu?
3. Regarder la colonne des erreurs validées. Que remarque-t-on sur les quatre premiers α ? Que fait `argmin` dans cette situation, et est-ce souhaitable?
4. **Règle du 1 écart-type** (Breiman) : retenir le **plus petit** arbre dont l'erreur validée reste sous $\min(\text{err}) + \text{écart-type}$. L'appliquer. Combien de feuilles? Que coûte-t-elle en erreur?
5. Afficher l'arbre retenu et énoncer chaque chemin racine-feuille en français.

Exercice 4 — Instabilité et importance des variables

```
# Instabilité : refaire un arbre sur des sous-échantillons à 90 %
rng = np.random.default_rng(0)
for _ in range(20):
    idx = rng.choice(len(X), int(0.9 * len(X)), replace=False)
    t = DecisionTreeClassifier(random_state=0).fit(X.to_numpy()[idx], y.to_numpy()[idx])
    print(NOMS[t.tree_.feature[0]], round(t.tree_.threshold[0], 2), t.get_n_leaves())

# Importance
pd.Series(arbre.feature_importances_, index=X.columns).sort_values(ascending=False)

importance <- petit$variable.importance
```

1. Ajuster vingt arbres sur des sous-échantillons de 90 %. La **première découpe** change-t-elle? Et le **nombre de feuilles**?
2. Le CM annonce que CART est instable. Sur Iris, cette instabilité se voit-elle là où on l'attendait?
3. Afficher `feature_importances_` de l'arbre à trois feuilles. Combien de variables ont une importance strictement positive?
4. Estimer maintenant, par validation croisée, l'erreur d'un arbre construit sur **chaque variable prise seule**. Confronter au tableau des importances. Qu'en conclure sur la lecture d'une importance nulle?
5. Le script R donne des importances non nulles pour les quatre variables. Chercher pourquoi.

Remarque finale

! À retenir

Gini et l'entropie font croître l'arbre, le taux d'erreur l'élague : deux rôles, deux mesures. La pénalité α transforme un continuum en une suite finie d'arbres emboîtés, dont la validation croisée désigne un plateau plutôt qu'un optimum — et la règle du 1 écart-type y retient le plus petit. L'importance des variables, enfin, décrit l'arbre obtenu, pas le phénomène observé.

Les deux scripts exécutables correspondants sont `python/td04_arbres.py` et `r/td04_arbres.R`. Ce dernier utilise `rpart`, qui implémente l'élagage de Breiman avec sa propre validation croisée interne.