



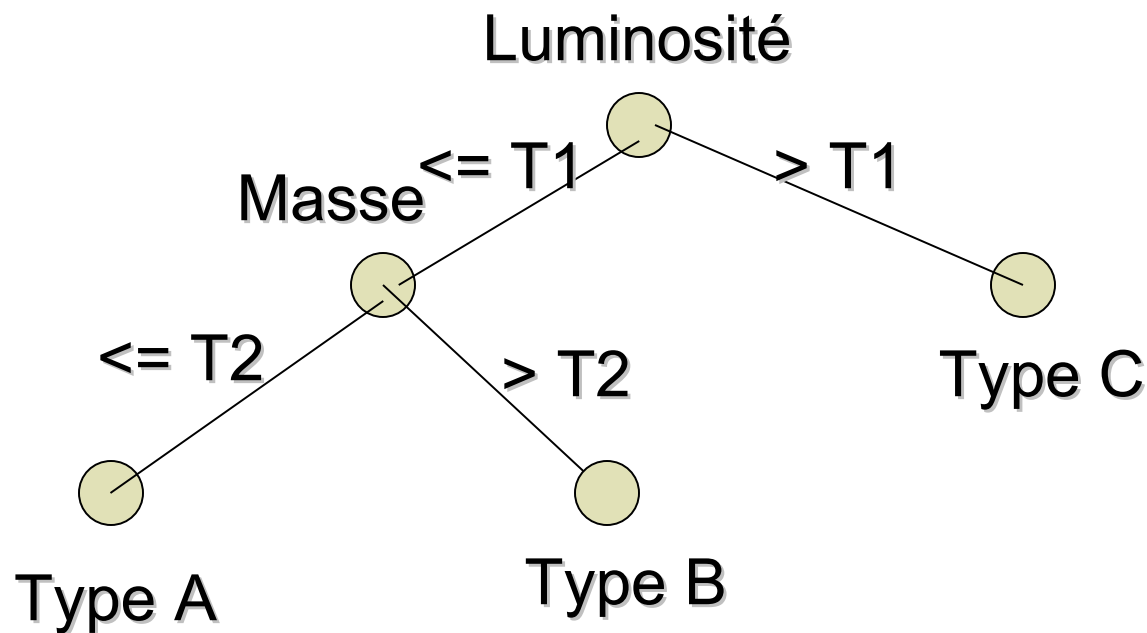
# Arbres de décision

# Sommaire

- Introduction et Définition
- Mécanisme
  - Critères de division
  - Espace d'hypothèses
- Apprendre des arbres de décision, c'est aussi :
  - Éviter l' « overfitting » grâce à l'élagage
  - Attributs manquants et numériques

# Illustration

Exemple : Apprendre à classifier des étoiles.

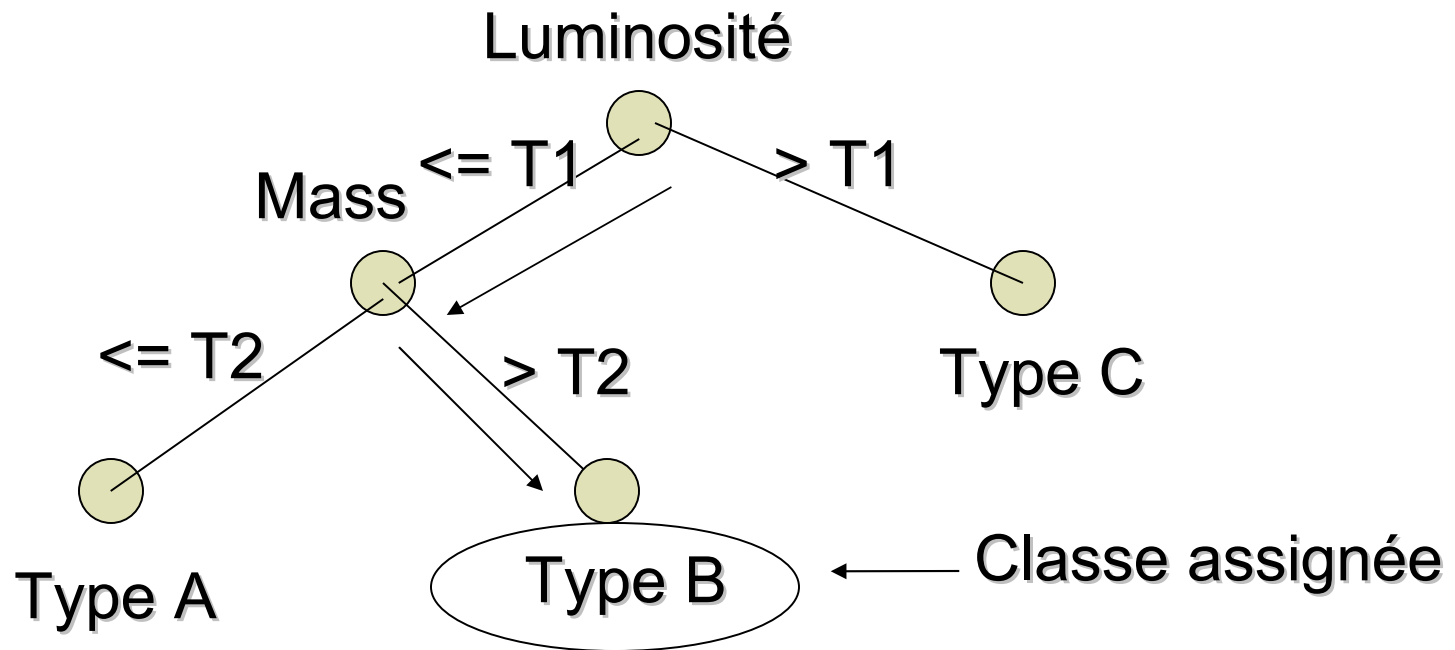


# Définition

- ✓ Un algorithme “arbre de décision” estime un concept cible par une représentation d’arbre, où chaque nœud interne correspond à un attribut, et chaque nœud terminal (ou feuille) correspond à une classe.
- ✓ Il y a deux types de nœuds :
  - ✓ Nœud interne : se déploie en différentes branches selon les différentes valeurs que l’attribue peut prendre. Exemple : luminosité  $\leq T1$  or luminosité  $> T1$ .
  - ✓ Nœud terminal : décide la classe assignée à l’exemple.

# Classifier des exemples

$X = (\text{Luminosité} \leq T1, \text{Masse} > T2)$

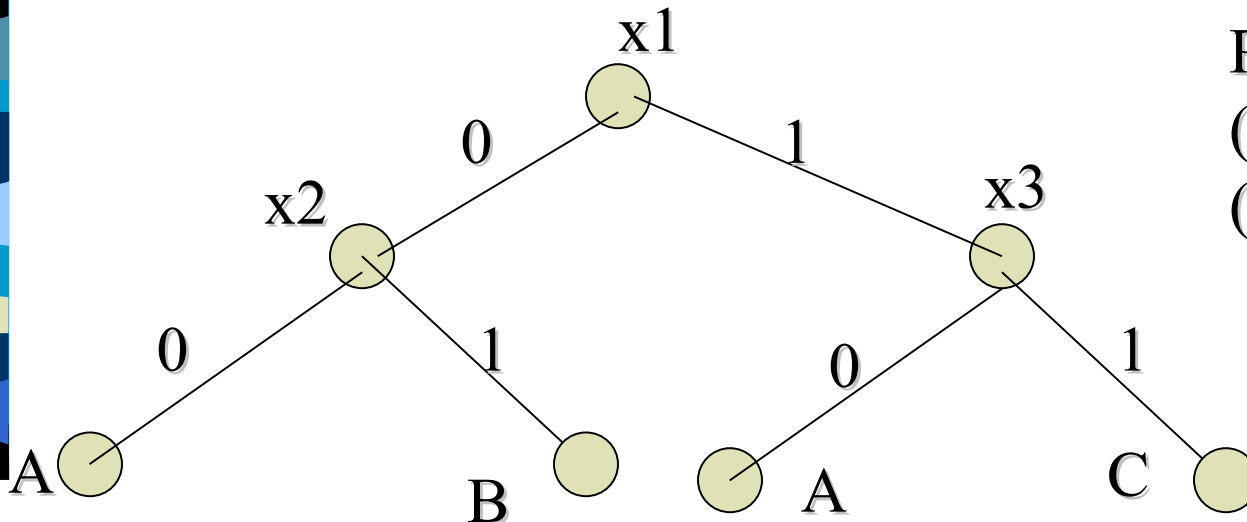


# Représentation

Les arbres de décision adoptent une représentation DNF (Disjunctive Normal Form).


Pour une classe donnée, chaque branche déployée de la racine vers une feuille de la dite classe, est une conjonction de valeurs d'attributs. Les différentes branches se terminant vers cette classe forment une disjonction.

Pour une classe A:  
 $(\sim X1 \ \& \ \sim x2)$  OR  
 $(X1 \ \& \ \sim x3)$



arbres de décision

# Arbres de décision : problèmes traités

- 
- Attributs numériques et nominaux.
  - La fonction ciblée prend un nombre discret de valeurs.
  - Une représentation DNF est effective pour représenter le concept cible.
  - Les données peuvent contenir des erreurs.
  - Certains exemples peuvent avoir des attributs manquants.

# Approche « diviser & conquérir »

- Nœuds de décision internes
  - Univariés : utilisent 1 seul attribut,  $x_i$ 
    - Numérique  $x_i$  : split binaire :  $x_i > w_m$
    - Discret  $x_i$  : split en  $n$ , pour les  $n$  valeurs possibles
  - Multivariés : utilisent tous les attributs,  $\mathbf{x}$
- Feuilles
  - Classification : Labels de classes, ou des proportions
  - Regression : Numérique; moyenne  $r$  average, ou local fit
- L'apprentissage est glouton ; trouver la meilleure division récursivement (Breiman et al, 1984; Quinlan, 1986, 1993)



# Arbres de décision

- Introduction et Définition
- Mécanisme
  - Critères de division
  - Espace d'hypothèses
- Apprendre des arbres de décision, c'est aussi :
  - Éviter l' « overfitting » grâce à l'élagage
  - Attributs manquants et numériques

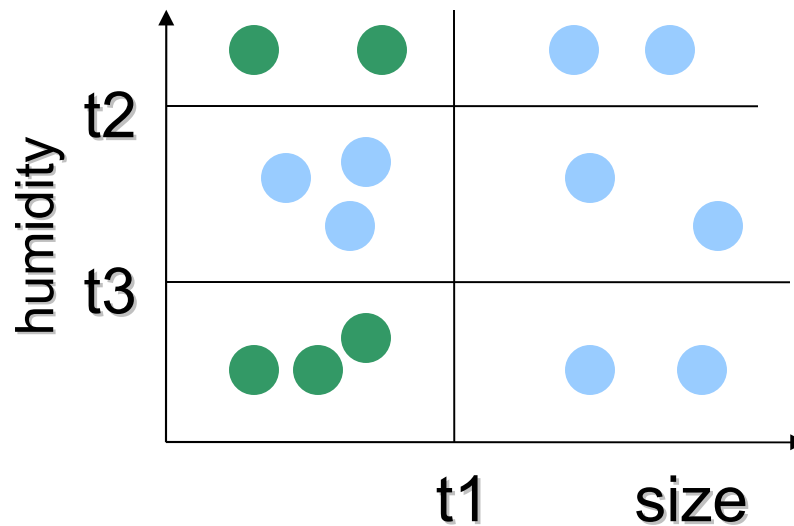
# Mécanisme

Il y a différentes approches de construction d'arbres à partir de données. Nous nous focaliserons sur une approche descendante (top-down) gloutonne (greedy) :

Idée de base :

1. Choisir le meilleur attribut  $a^*$  à placer comme racine de l'arbre.
2. Partager l'ensemble d'apprentissage  $D$  en sous-ensembles  $\{D_1, D_2, \dots, D_k\}$  où chaque  $D_i$  contient des exemples ayant la même valeur pour  $a^*$
3. Récursivement, appliquer l'algorithme sur chaque nouveau sous-ensemble jusqu'à ce que les exemples aient la même classe.

## Illustration



- Classe P : vénéneux
- Classe N : non-vénéneux

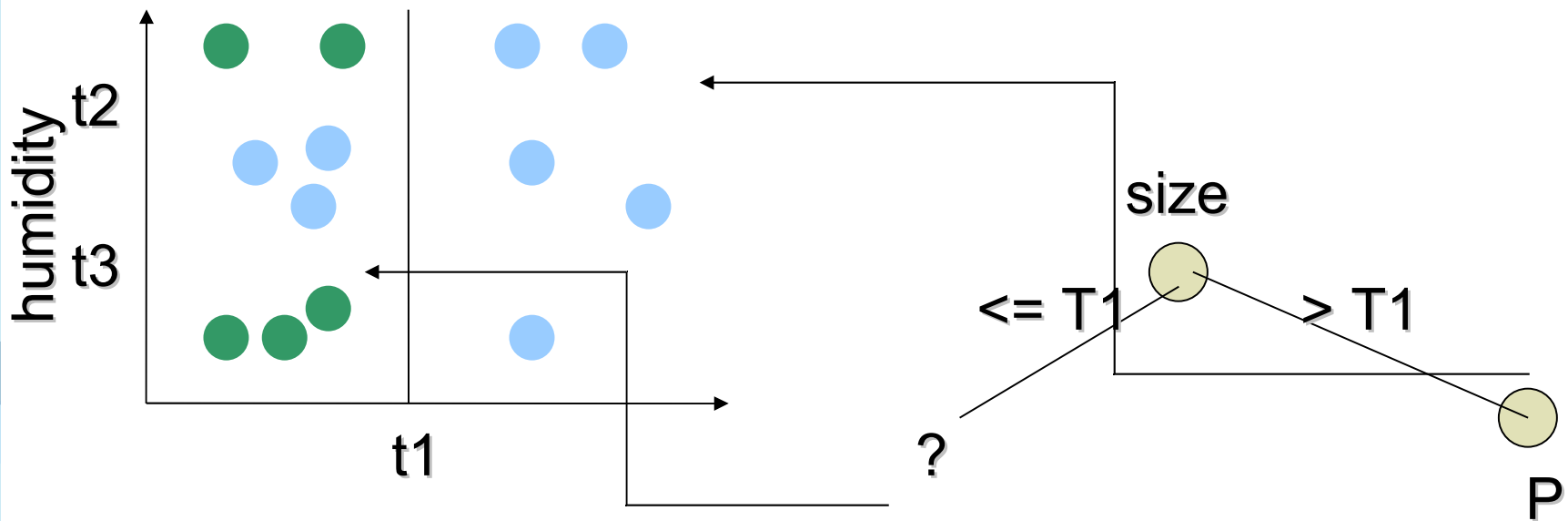
Attributs : size et humidity

Size a deux valeurs :  $>t_1$  ou  $\leq t_1$

Humidity a trois valeurs :  $>t_2$ , ( $>t_3$  et  $\leq t_2$ ),  $\leq t_3$

# Illustration

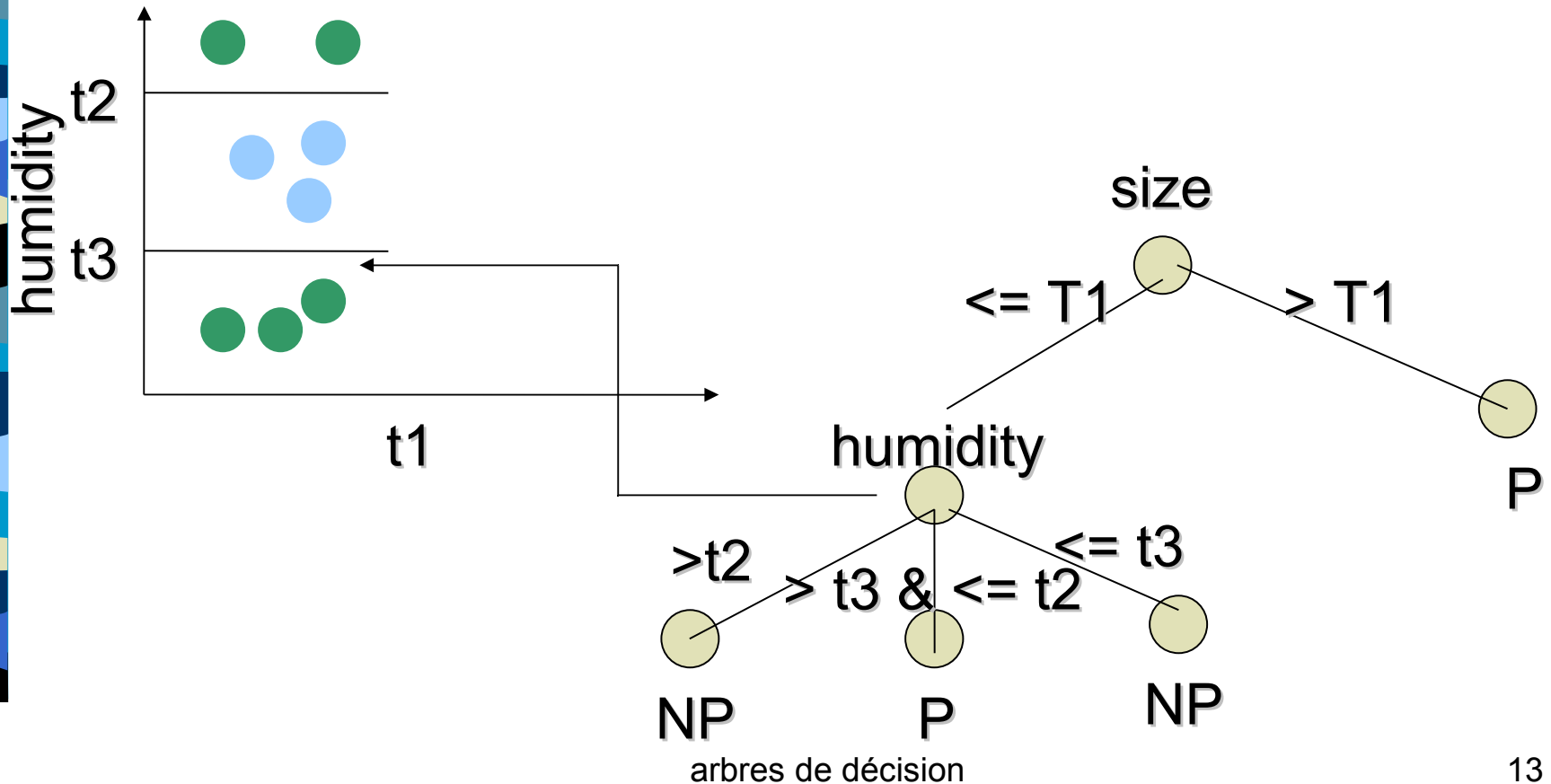
Supposons que nous choissions **size** comme le meilleur attribut :



- Classe P : vénéneux
- Classe N : non-vénéneux

# Illustration

Supposons que nous choisissons **humidity** comme le prochain meilleur attribut :



## Mécanisme formel

- Créer une racine pour l'arbre
- Si tous les exemples sont de la même classe ou si le nombre d'exemples est en dessous d'un seuil, retourner cette classe
- Si plus d'attributs disponibles, retourner la classe majoritaire
- Soit  $a^*$  le meilleur attribut
- Pour chaque valeur possible  $v$  de  $a^*$ 
  - Ajouter une branche de  $a^*$  étiquetée " $a^* = v$ "
  - Soit  $S_v$  le sous-ensemble d'exemples où l'attribut  $a^*=v$
  - Récursivement, appliquer l'algorithme à  $S_v$

## Critères de division

Quel attribut est le meilleur pour discriminer les données ?

### Quelques définitions de la théorie de l'information

Une mesure d'incertitude ou d'entropie associée à une variable aléatoire  $X$  est définie par :

$H(X) = - \sum p_i \log p_i$ , le logarithme étant en base 2.

## Entropie 1

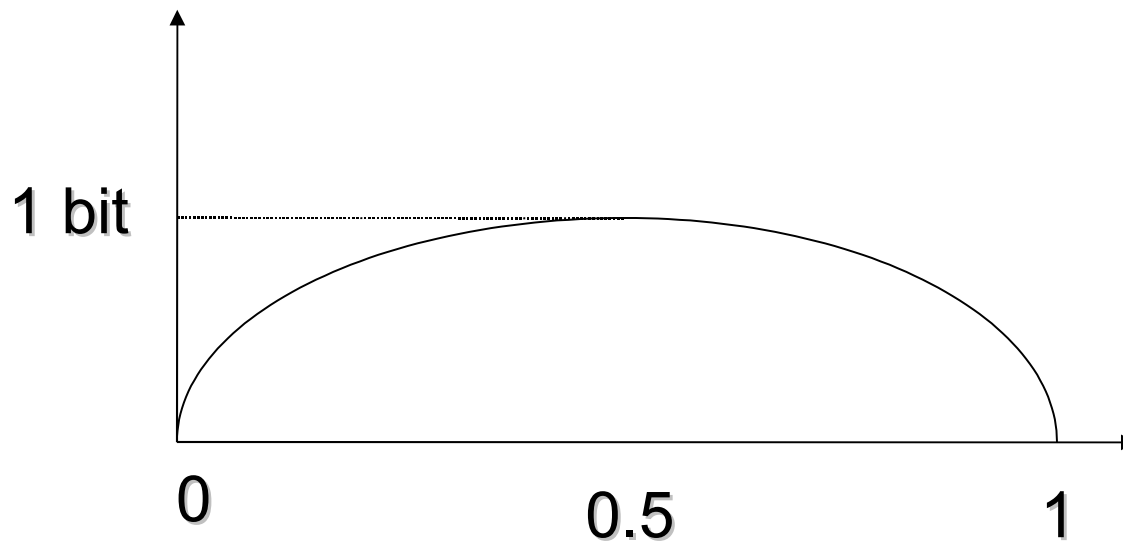
Il y a 2 événements complets possibles **A** et **B**  
(Exemple : lancer une pièce biaisée)

- ✓  $P(A) = 1/256, P(B) = 255/256$   
 $H(X) = 0.0369 \text{ bit}$
- ✓  $P(A) = 1/2, P(B) = 1/2$   
 $H(X) = 1 \text{ bit}$
- ✓  $P(A) = 7/16, P(B) = 9/16$   
 $H(X) = 0.989 \text{ bit}$



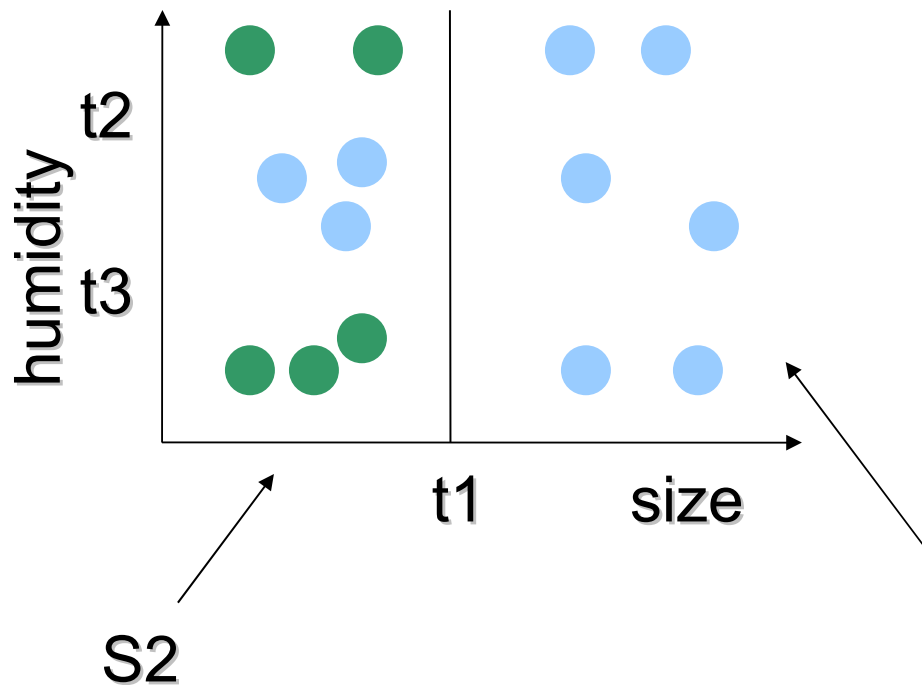
## Entropie 2

L'entropie est une fonction concave comme illustrée



# Division basée sur l'entropie

Notre exemple précédent :



Size divise l'ensemble d'apprentissage en deux :

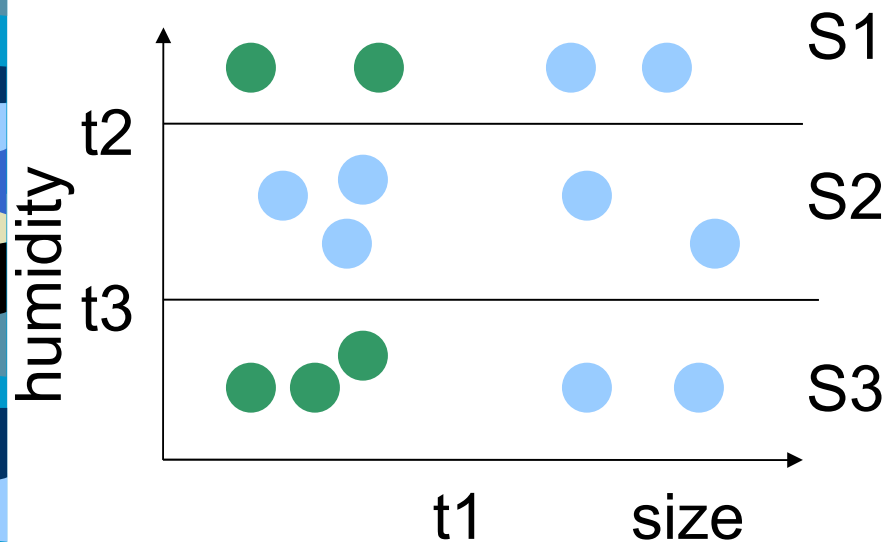
$$S1 = \{ 6P, 0NP \}$$

$$S2 = \{ 3P, 5NP \}$$

$$H(S1) = 0$$

$$H(S2) = -(3/8)\log_2(3/8) - (5/8)\log_2(5/8)$$

## Division basée sur l'entropie



humidity divise l'ensemble d'apprentissage en trois :

$$S_1 = \{ 2P, 2NP \}$$

$$S_2 = \{ 5P, 0NP \}$$

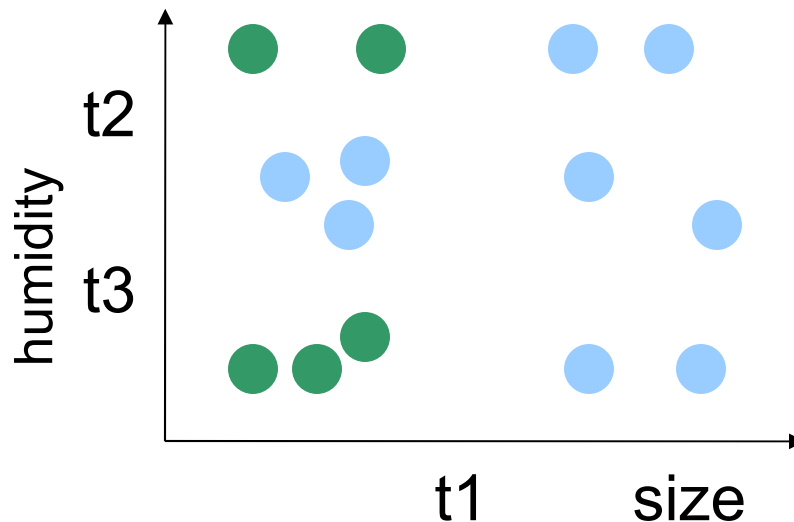
$$S_3 = \{ 2P, 3NP \}$$

$$H(S_1) = 1$$

$$H(S_2) = 0$$

$$H(S_3) = -(2/5)\log_2(2/5) - (3/5)\log_2(3/5)$$

# Gain d'Information



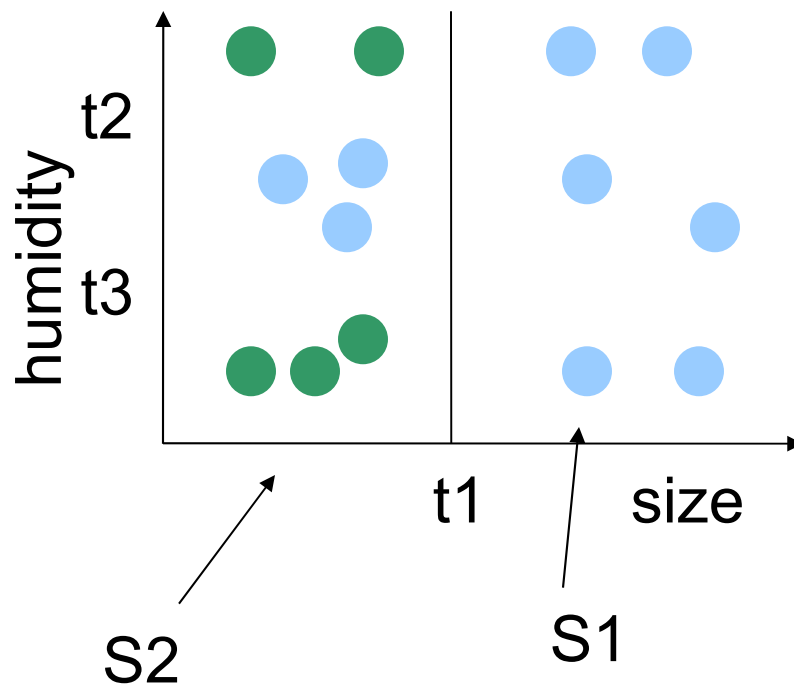
Le gain d'information IG associé à l'attribut A : IG (A)

$$IG(A) = H(S) - \sum_v (S_v/S) H(S_v)$$

$H(S)$  est l'entropie de tous les exemples

$H(S_v)$  est l'entropie d'un sous-ensemble après partition de  $S$ , selon les différentes valeurs possibles de l'attribut A.

# Gain d'Information



$$H(S_1) = 0$$

$$H(S_2) = -(3/8)\log_2(3/8) - (5/8)\log_2(5/8)$$

$$H(S) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14)$$

$$|S_1|/|S| = 6/14$$

$$|S_2|/|S| = 8/14$$

# Arbres de décision

- Introduction et Définition
- Mécanisme
  - Critères de division
  - Espace d'hypothèses
- Apprendre des arbres de décision, c'est aussi :
  - Éviter l' « overfitting » grâce à l'élagage
  - Attributs manquants et numériques

## Espace d'hypothèses

- ✓ Rechercher à travers l'espace des hypothèses de tous les arbres de décision possible.
- ✓ On ne garde qu'une seule hypothèse à la fois. Au lieu d'en avoir plusieurs.
- ✓ Pas de retour arrière. On choisit la meilleure alternative et on développe l'arbre.
- ✓ On préférera des arbres compacts à des arbres plus larges.
- ✓ On préférera des arbres où les attributs à entropie faible sont placés en haut.

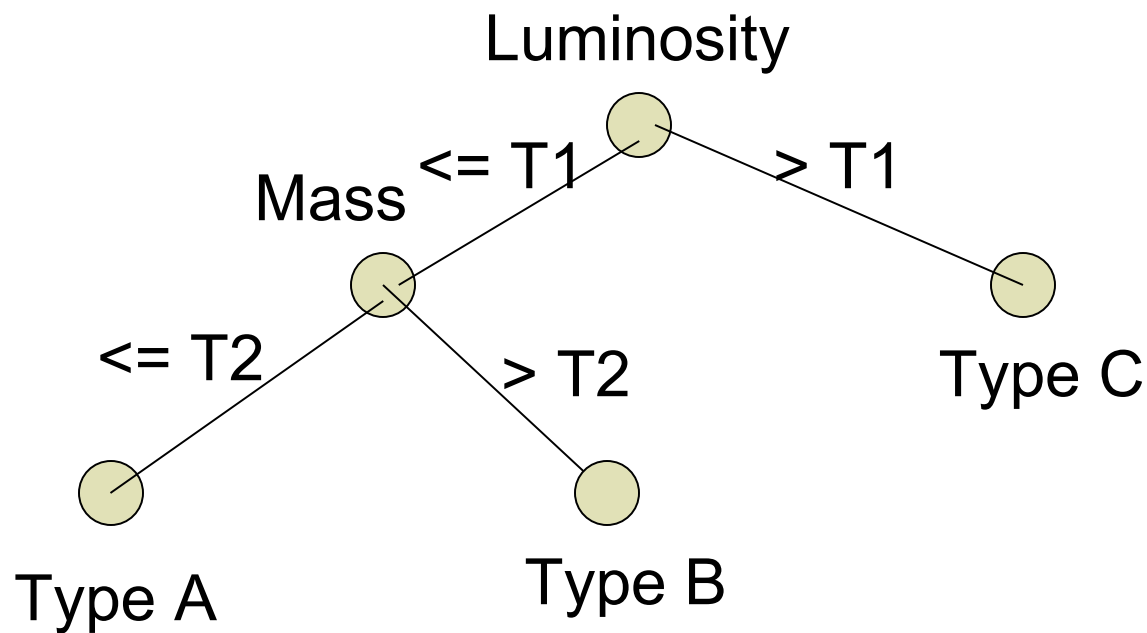
# Arbres de décision

- Introduction et Définition
- Mécanisme
  - Critères de division
  - Espace d'hypothèses
- Apprendre des arbres de décision, c'est aussi :
  - Éviter l' « overfitting » grâce à l'élagage
  - Attributs manquants et numériques



# Un exemple d'arbre de décision

Exemple : Apprendre à classer des étoiles



## Taille des hypothèses ?

Une approche descendante gloutonne dénote une préférence pour de petites hypothèses.

Est-ce la bonne chose à faire ?

**Occam's Razor : Préférer l'hypothèse la plus simple au regard des données**

William of Occam (1320) : one should not increase, beyond what is necessary, the number of entities required to explain anything. **Principle of parsimony.**

Gros débat en philosophie des sciences.

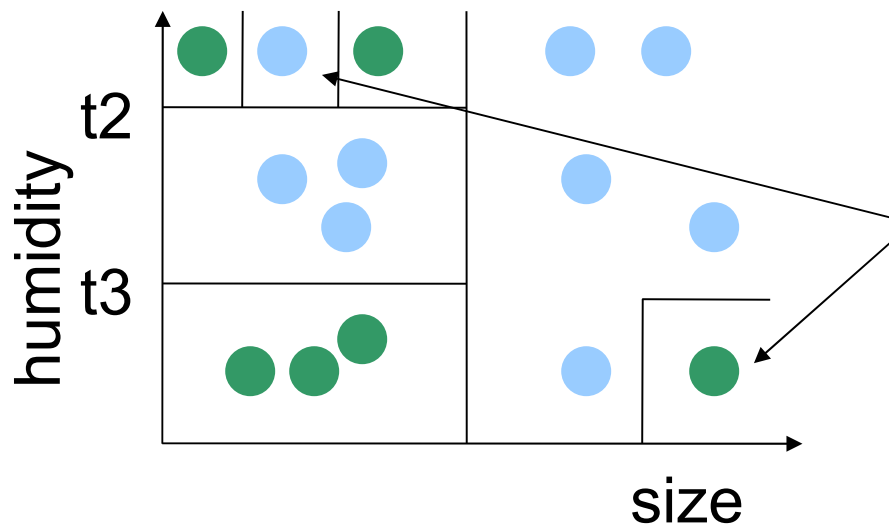
## Questions pratiques lors de la construction d'arbres

Quand l'on construit un arbre de décision, les questions pratiques que l'on peut énumérer sont les suivantes :

- ✓ Quelle devrait être la profondeur de l'arbre ?
- ✓ Comment considérer les attributs continus ?
- ✓ Qu'est ce qu'un bon critère de division ?
- ✓ Que se passe t-il quand il manque des valeurs d'attributs ?
- ✓ Comment améliore t-on l'efficacité computationnelle ?

## Profondeur de l'arbre ?

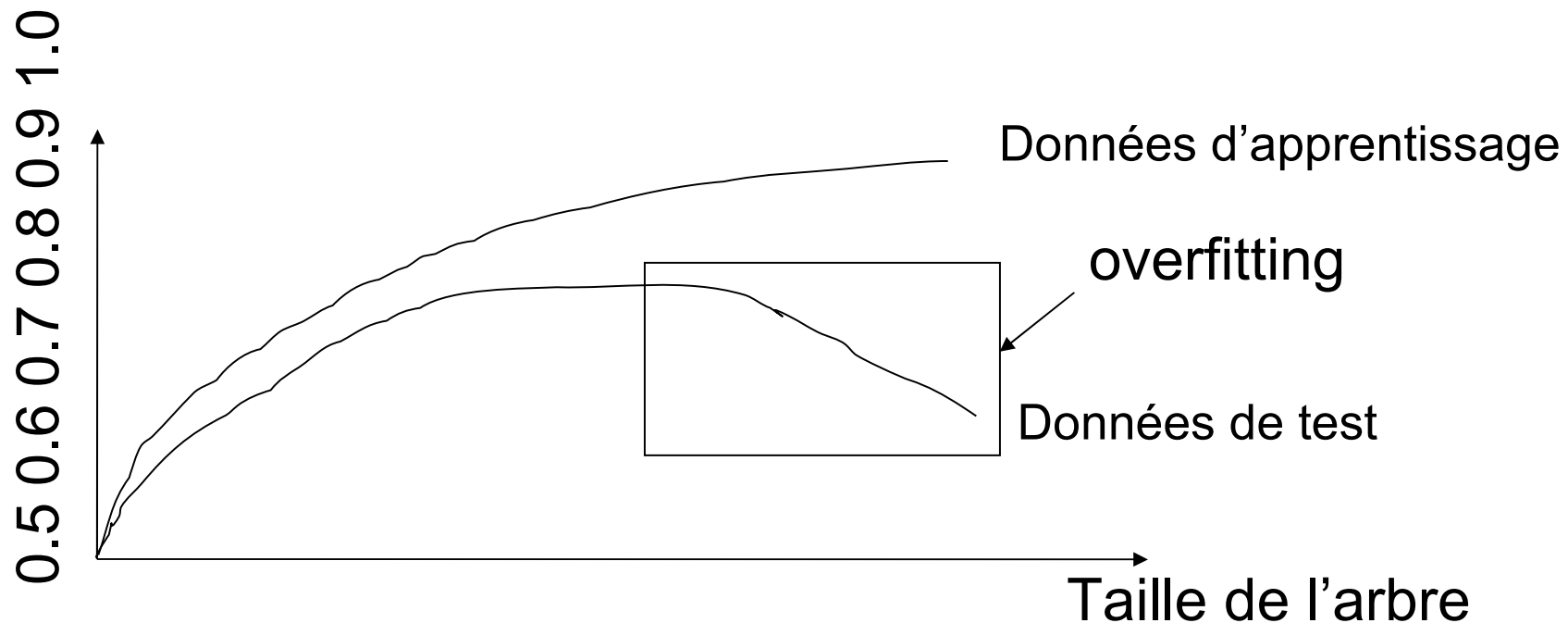
Un arbre « colle » trop aux données quand on le laisse croître au point de capturer des “aberrations”. Cela détériore sa capacité prédictive sur de nouveaux exemples.



Possiblement juste du bruit, mais l'arbre se développe pour capturer ces exemples

## « Overfitting » des données : Définition

Soit un espace d'hypothèses  $H$ . Une hypothèse  $h$  dans  $H$  « overfits » un ensemble de données  $D$ , s'il y a une autre hypothèse  $h'$  de  $H$  où  $h$  a un meilleur taux de classification (accuracy) que  $h'$  sur  $D$  mais moins bon que celui de  $h'$  sur  $D'$ .



## Les causes du « Overfitting » des données

- Erreurs ou bruit

Exemples étant labellisés incorrectement par une classe  
Valeurs d'attributs incorrectes.

- Patrons coïncidents

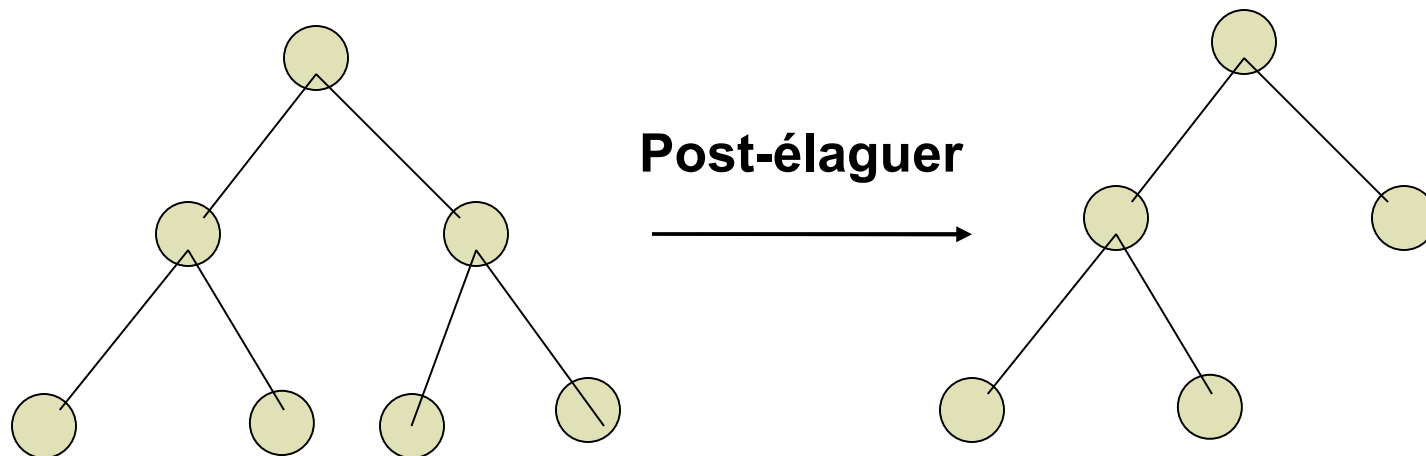
Des exemples dévient d'un patron en raison du trop faible nombre d'exemples.

L' « overfitting » peut causer de graves détériorations des performances

# Solutions

Deux grandes classes de solutions :

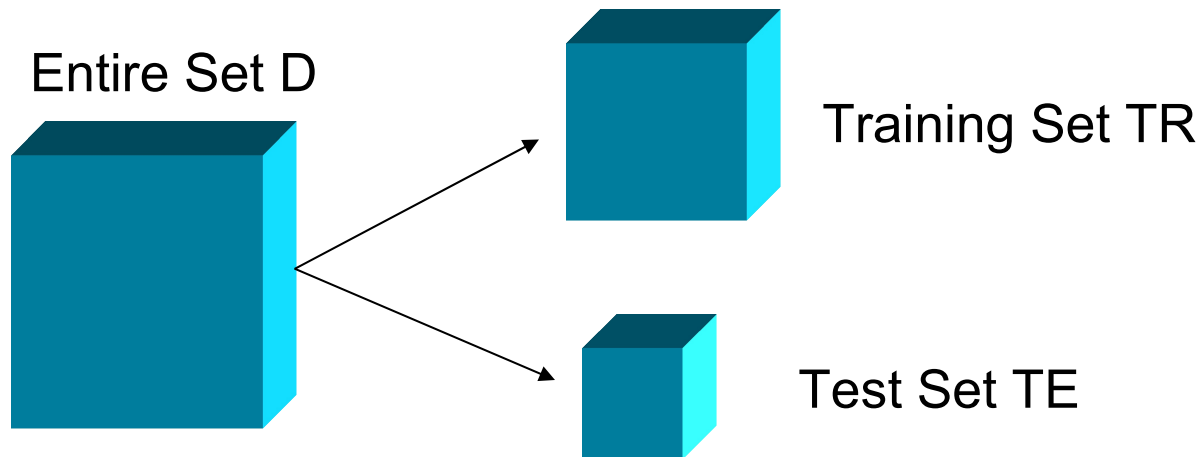
- Arrêter la croissance de l'arbre tôt. En pratique, cette solution est plutôt difficile à implémenter en raison de la difficulté à identifier le point d'arrêt.
- Déployer l'arbre jusqu'à ce que l'algorithme s'arrête. Post-élaguer l'arbre. C'est une méthode plus populaire.



# Valider l'arbre obtenu

## 1. Approche via ensembles d'apprentissage et de test

- ✓ Diviser D en TR et TE
- ✓ Construire un arbre de décision avec TR
- ✓ Tester les arbres élagués sur TE pour décider du meilleur.



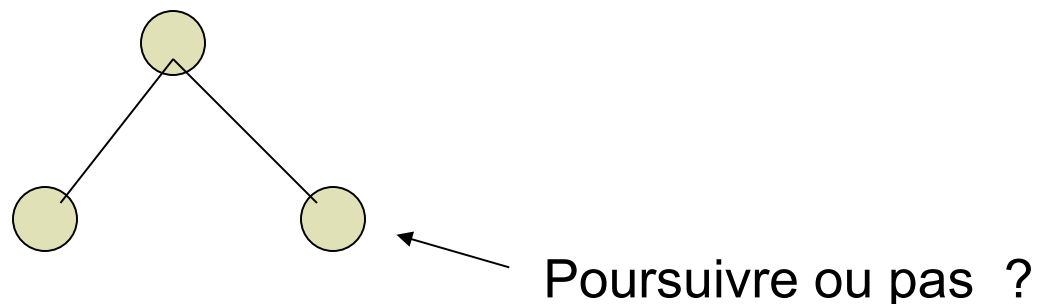
arbres de décision



# Valider l'arbre obtenu

## 2. Utiliser un test statistique

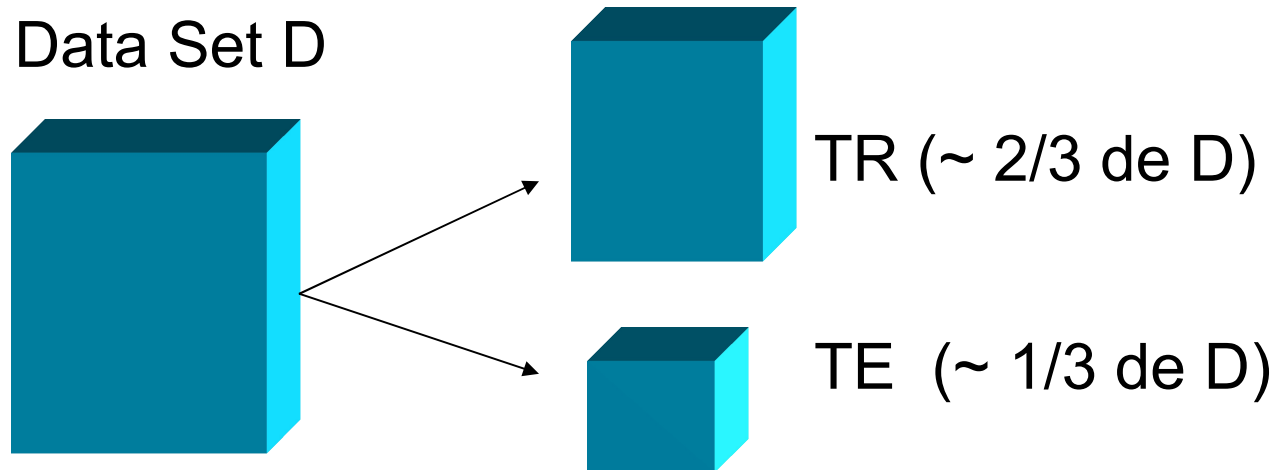
- ✓ Utiliser tout l'ensemble  $D$  pour l'apprentissage
- ✓ Utiliser un test statistique pour décider si vous devez poursuivre ou non à un nœud (e.g., chi squared).



## Valider l'arbre obtenu

3. Utiliser une grandeur pour capturer la taille de l'arbre et le nombre d'erreurs commises par l'arbre.
  - ✓ Utiliser tout l'ensemble D pour apprendre
  - ✓ Utiliser cette grandeur pour savoir quand arrêter l'expansion de l'arbre
  - ✓ La méthode est connue comme : « *minimum description length principle* ».

# Apprentissage et Validation



Deux approches :

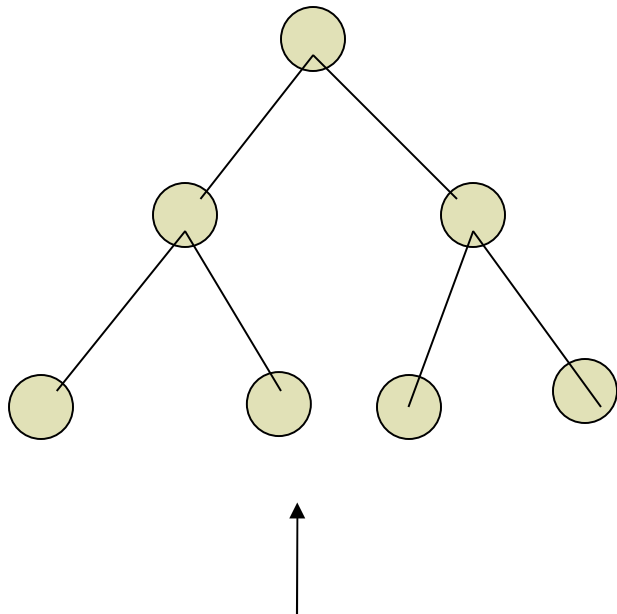
- C. Reduced Error Pruning
- D. Post-élagage de règles

# Reduced Error Pruning

## Idées clés :

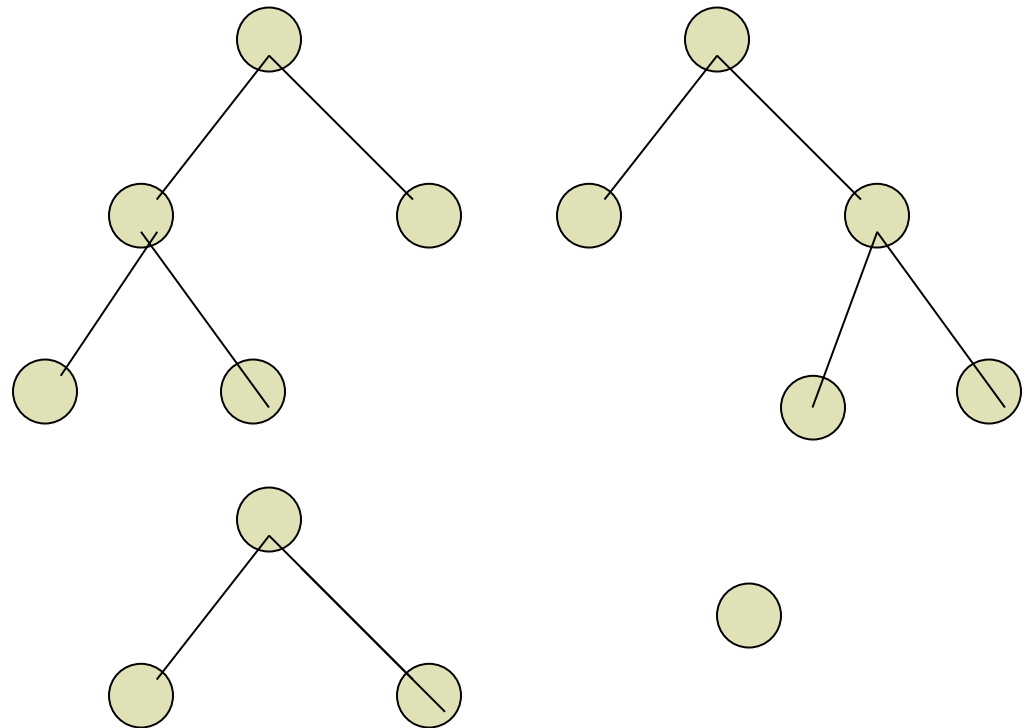
- 1- Considérer tous les nœuds internes de l'arbre.
- 2- Pour chaque nœud, vérifier si en l'enlevant (ainsi que le sous-arbre dont il est racine) et en lui assignant la classe la plus commune, on ne détériore pas la grandeur exactitude («accuracy») sur TE.
- 3- Choisir le nœud  $n^*$  qui donne la meilleure performance et élaguer son sous-arbre.
- 4- Aller à (2) jusqu'à ce qu'aucune amélioration ne soit possible.

# Exemple



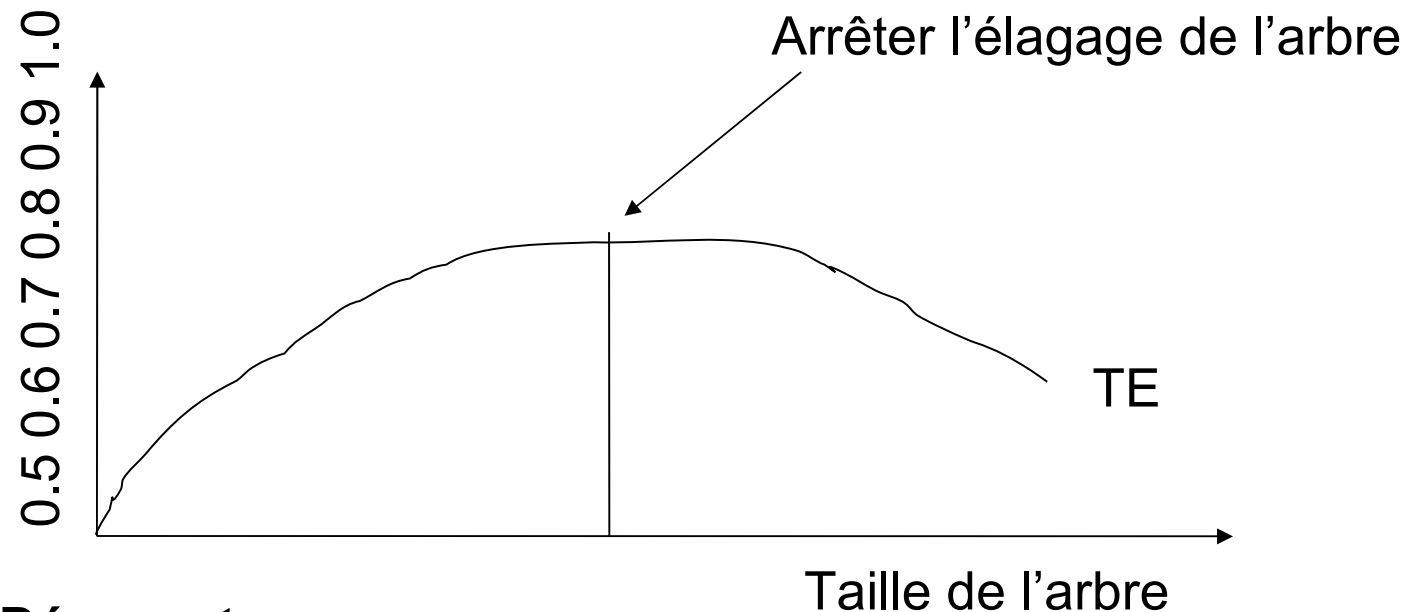
Arbre d'origine

## Arbres possibles après élagage



## Exemple

Le processus continue jusqu'à ce qu'aucune amélioration ne soit observée sur TE :



### Désavantages :

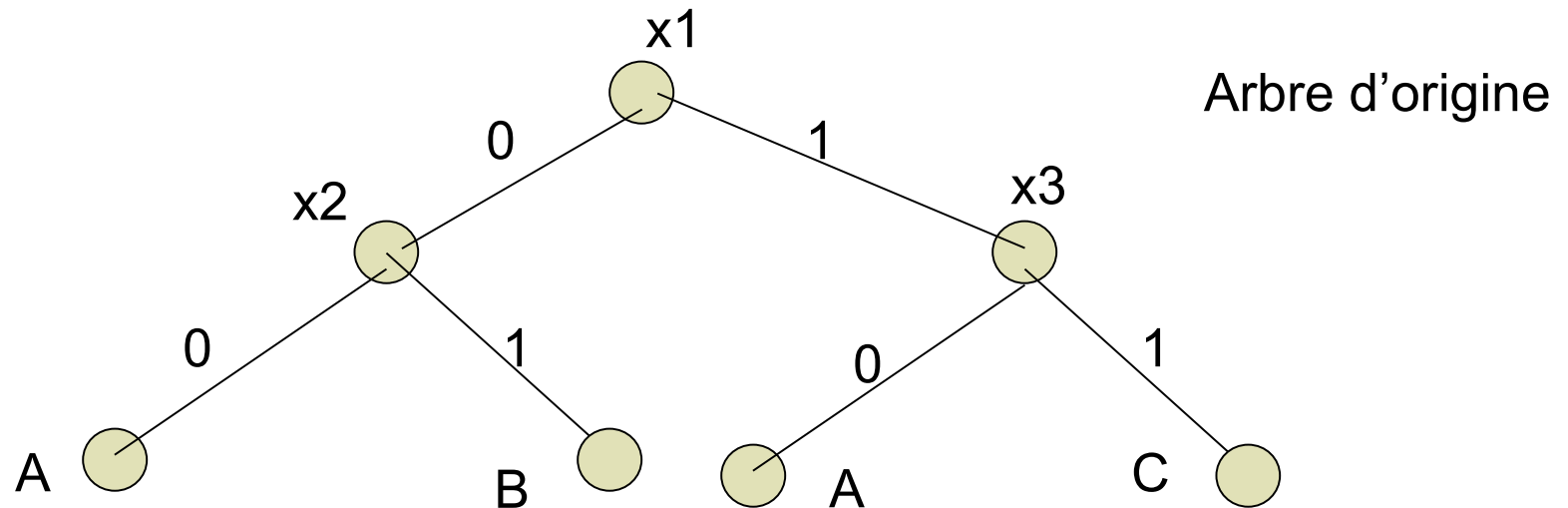
- Si  $D$  est petit, diviser en TR et TE réduira encore plus le nombre d'exemples pour apprendre.

# Post-élagage de règles

## Idées clés :

- Convertir l'arbre en un système de règles.
- Élaguer chaque règle en supprimant les conditions redondantes.
- Ordonner les règles selon leur exactitude (accuracy).

# Exemple



## Règles :

$\sim x_1 \ \& \ \sim x_2 \rightarrow \text{Class A}$   
 $\sim x_1 \ \& \ x_2 \rightarrow \text{Class B}$   
 $x_1 \ \& \ \sim x_3 \rightarrow \text{Class A}$   
 $x_1 \ \& \ x_3 \rightarrow \text{Class C}$

Règles possibles après élagage  
(basé sur l'ensemble TE) :

$\sim x_1 \rightarrow \text{Class A}$   
 $\sim x_1 \ \& \ x_2 \rightarrow \text{Class B}$   
 $\sim x_3 \rightarrow \text{Class A}$   
 $x_1 \ \& \ x_3 \rightarrow \text{Class C}$



## Avantage du post-élagage de règles

- ❖ Le langage est plus expressif.
- ❖ L'interprétabilité est améliorée.
- ❖ L'élagage est plus flexible.
- ❖ En pratique, cette méthode mène à de bonnes performances.

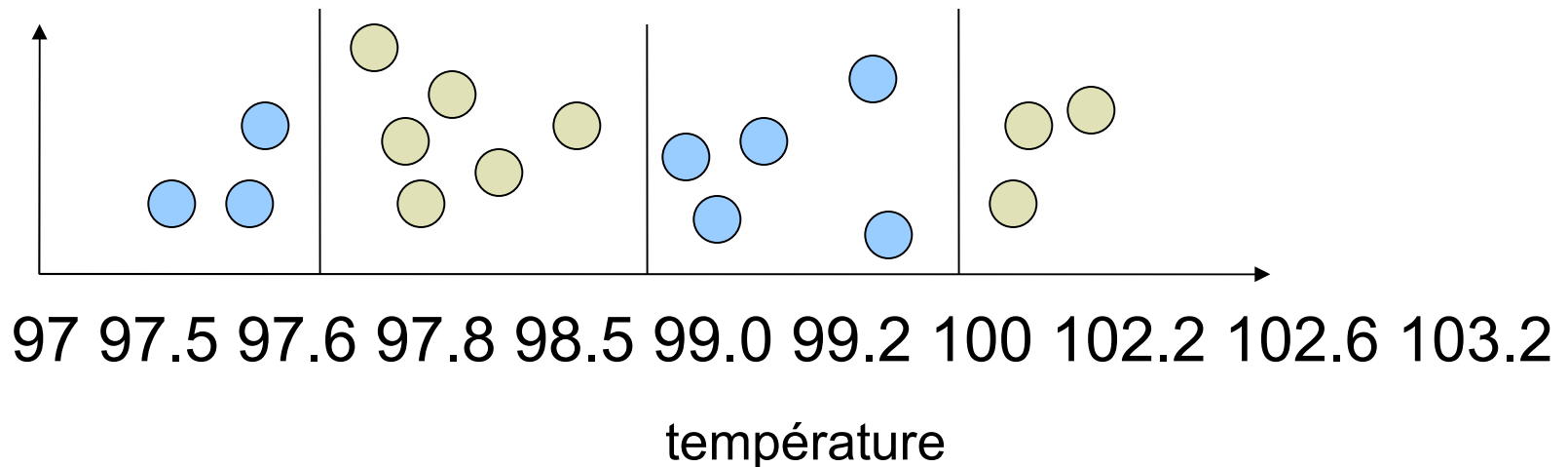
# Arbres de décision

- Introduction et Définition
- Mécanisme
  - Critères de division
  - Espace d'hypothèses
- Apprendre des arbres de décision, c'est aussi :
  - Éviter l' « overfitting » grâce à l'élagage
  - Attributs manquants et numériques

# Discrétiser les attributs continus

Exemple : attribut température.

- 1) Ordonner toutes les valeurs dans l'ensemble d'apprentissage
- 2) Considérer les points de coupure où il y a un changement de classes
- 3) Choisir les points de coupure qui maximisent le gain en information



## Valeurs manquantes d'attributs

### Exemple :

**X** = (luminosité > T1, masse = ?)

Quand on est à un nœud **n** dans l'arbre.

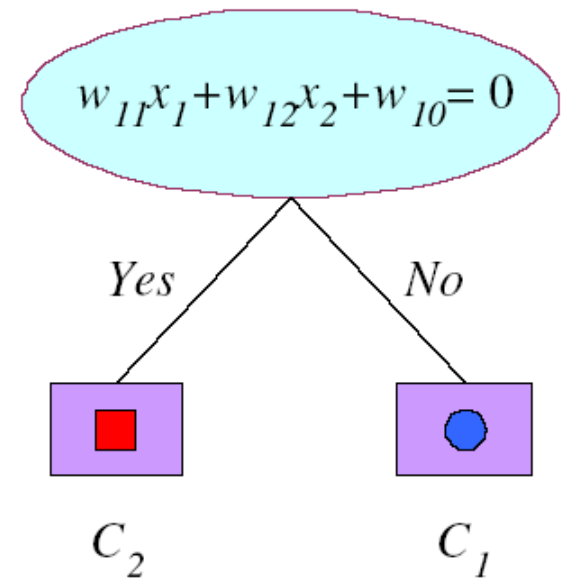
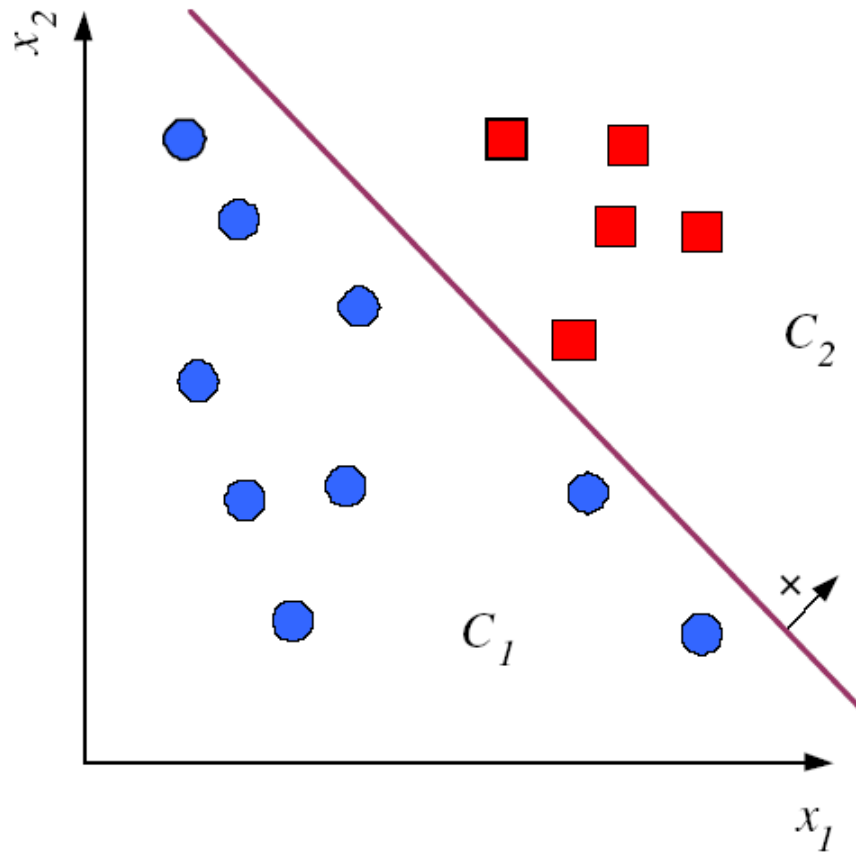
Différentes approches :

- Assigner la valeur la plus fréquente pour cet attribut au nœud **n**.
- Assigner la valeur la plus fréquente pour cet attribut au nœud **n**, parmi les exemples qui ont la même classification que **X**.
- Assigner une probabilité pour chaque valeur de l'attribut, basée sur la fréquence de ces valeurs au nœud **n**. Chaque fraction est alors propager vers le bas de l'arbre.

# Conclusions

- ❑ Produit un arbre de décision à partir d'un ensemble d'exemples décrivant le problème ;
- ❑ Partager récursivement l'ensemble des données en sous-ensembles ;
- ❑ C'est une approche populaire, mature et compréhensible, qui permet d'interpréter facilement les hypothèses produites.
- ❑ L'espace des hypothèses est puissant : toutes les formules DNF possibles.
- ❑ On préférera les arbres compacts à ceux plus larges.
- ❑ « Overfitting » : une question importante dans l'induction d'arbres de décision. Différentes méthodes existent pour éviter l'« overfitting » : reduced-error pruning et post-élagage de règles.
- ❑ Des techniques existent aussi pour tenir compte des attributs à valeurs continues ou manquantes.
- ❑ Efficace pour la classification ;
- ❑ Partition de l'espace des instances (exemples) selon un attribut à la fois est limitatif, mais, ...

# Arbres multivariés



# Références

- Tom Mitchell, Machine Learning, McGraw Hill, 1997.
- Ethem ALPAYDIN, Introduction to Machine Learning, The MIT Press, October 2004.
- Articles divers.