# Arbres de décision

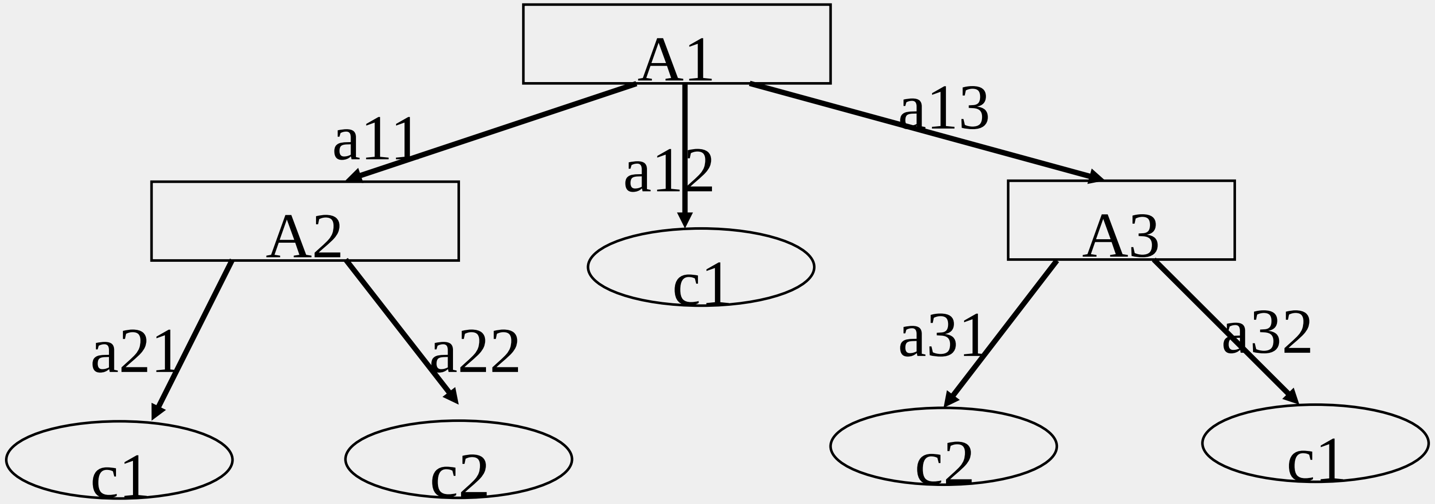
# Arbres de décision (classification)

- Algorithme d’apprentissage supervisé :
  - ✓ Problèmes de classification (binaire ou multivaluée)
  - ✓ Les attributs peuvent être discrets (binaires ou multi-valués) ou continus.

- Arbres de classification inventés 2 fois :
  - ✓ Par des statisticiens: CART (Breiman et al. 1982
  - ✓ Par la communauté d’IA : ID3, C4.5 (Quinlan et al. 1984)

# Arbres de décision (classification)

- Un arbre de décision est un arbre où :
  - Chaque nœud intermédiaires teste un attribut
  - Chaque branche correspond à une valeur d'attribut
  - Chaque feuille est étiquetée par une classe



# Arbres de décision (classification)

**Premier exemple : le diagnostic médical**

| Patient | Rythme cardiaque | Pression artérielle | Classe |
|---|---|---|---|
| 1 | irrégulier | normale | Malade |
| 2 | régulier | normale | En forme |
| 3 | irrégulier | anormale | Malade |
| 4 | irrégulier | normale | Malade |
| 5 | régulier | normale | En forme |
| 6 | régulier | anormale | Malade |
| 7 | régulier | normale | En forme |
| 8 | régulier | normale | En forme |

**Base d'apprentissage de 8 données**

# Arbres de décision (classification)

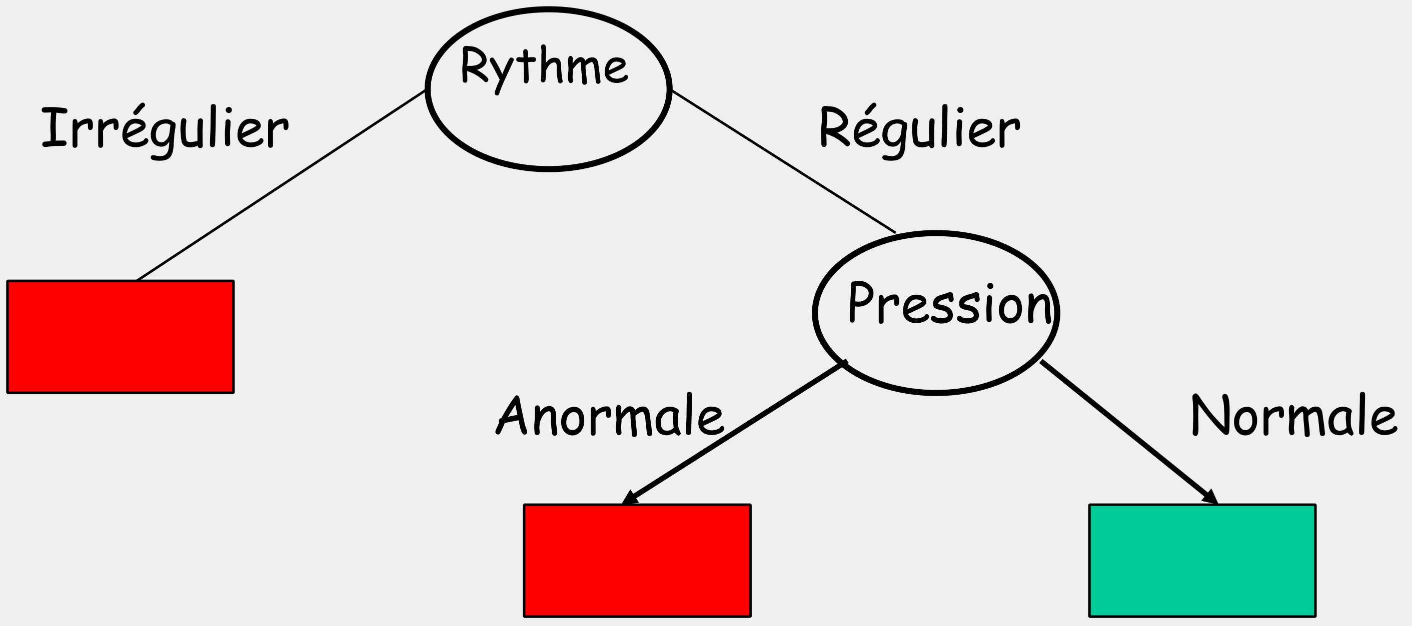
## Objectif

A partir des 8 exemples, apprendre un classifieur capable de fournir une classe à partir de la description du patient

**Remarque :**
Dans cet exemple, les valeurs des attributs sont symboliques (pas d'ordre sur les valeurs)

# Arbres de décision (classification)

Un arbre de décision définit une fonction de classification dans l'espace (l'ensemble) d'entrée



# Arbres de décision (classification)

## Utiliser un arbre de décision en classification

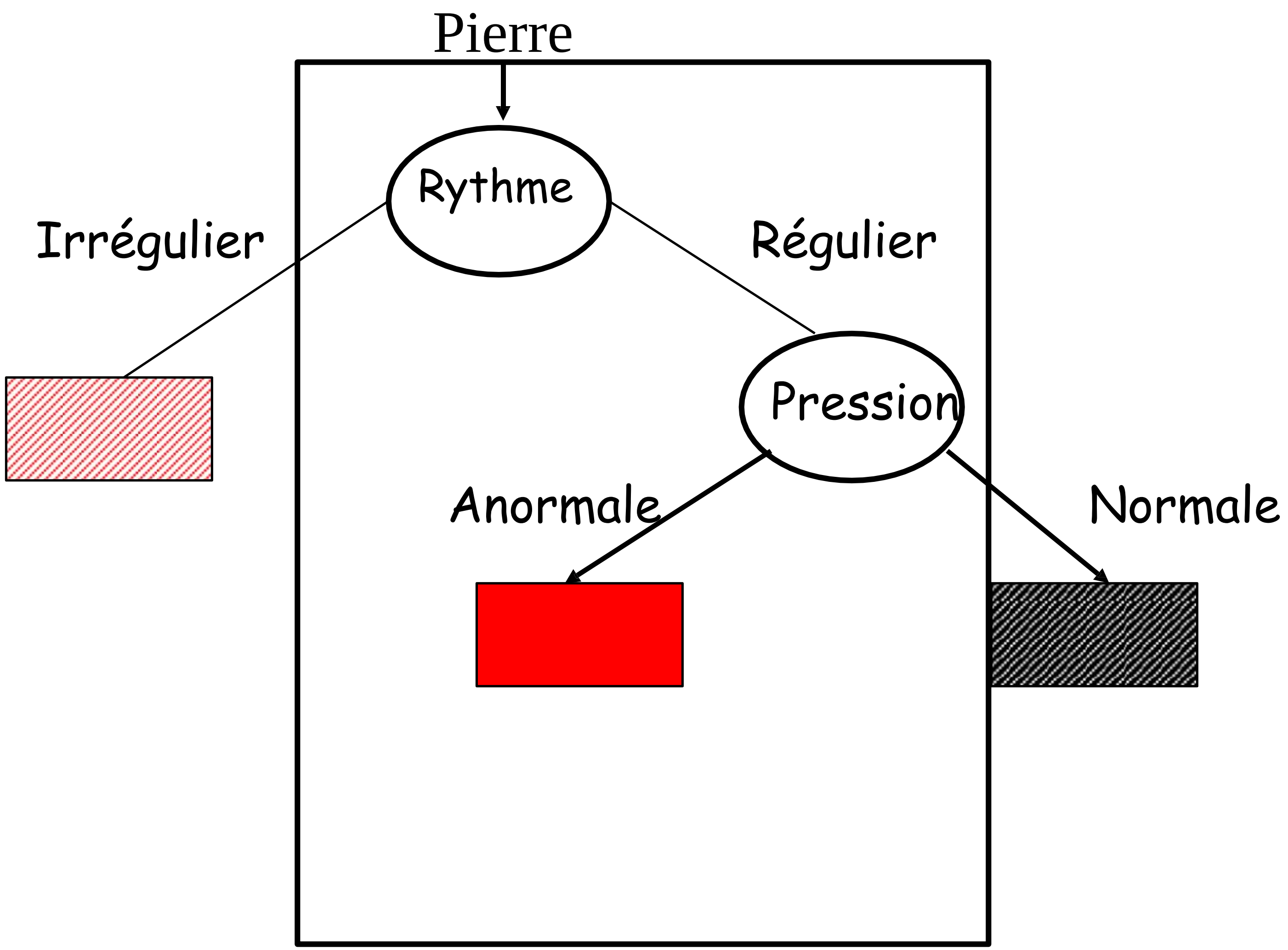
Une fois appris, l'arbre sert à prendre des décision sur de nouvelles données.

Pierre se rend chez le médecin
Il a un rythme cardiaque régulier mais une pression artérielle anormale
Est-il malade ?

Pour donner la réponse : parcourir l'arbre en répondant aux questions.

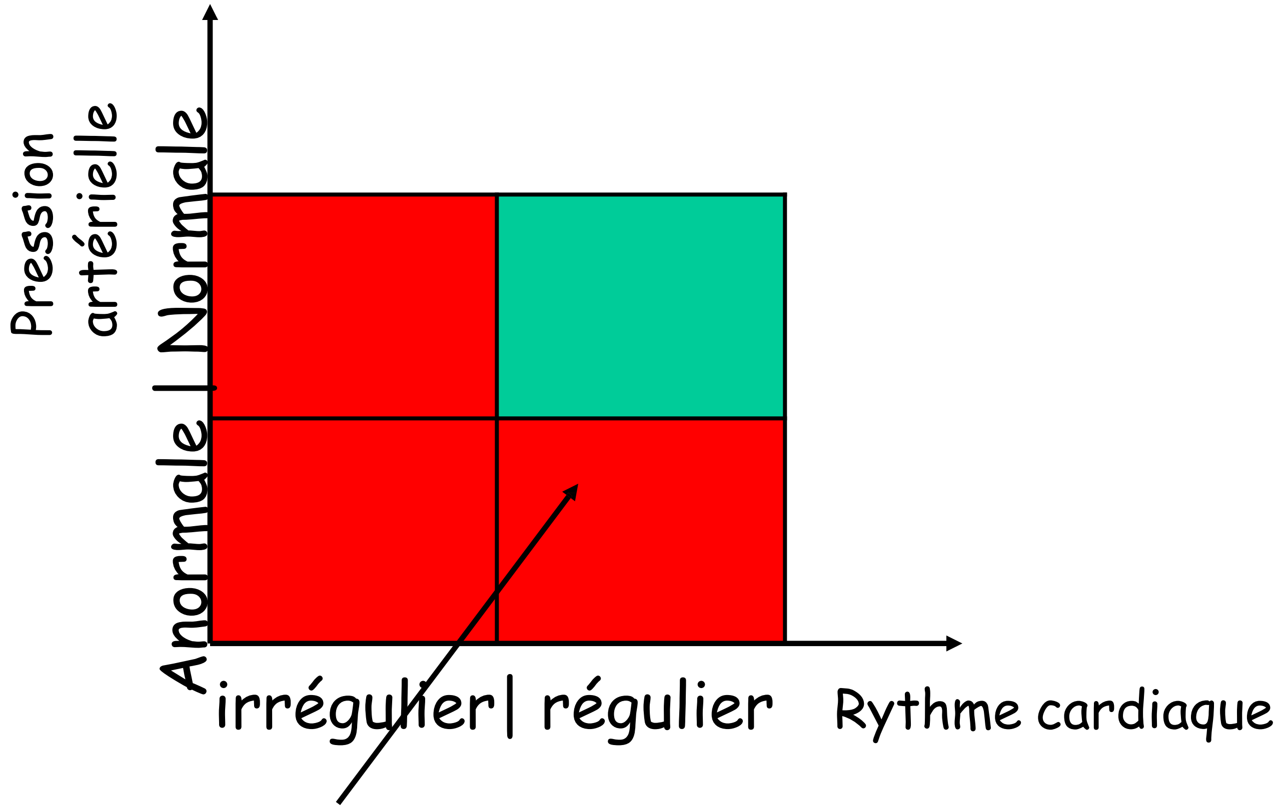
# Arbres de décision (classification)

## Utilisation en classification



# Arbres de décision (classification)

## Classes selon l'état du patient




Ex : Pierre = [ régulier anormale]

Classe de Pierre = Malade

# Arbres de décision (classification)

## Algorithme de construction d'un arbre de décision

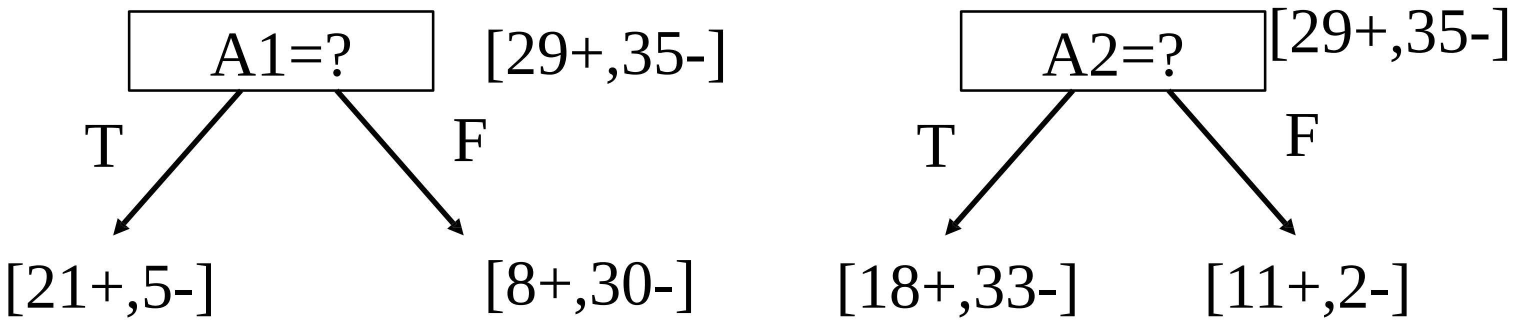
1) Choisir le “meilleur” attribut

2) Diviser l’ensemble d’apprentissage suivant les valeurs de l’attribut choisi

3) Répéter les étapes 1 et 2 de manière récursive jusqu’à ce que tous les objets soient correctement classés.

# Arbres de décision (classification)

## Comment choisir le meilleur attribut ?




- Obtenir un petit arbre :
  - Maximiser la séparation des classes à chaque étape, i.e. rendre les ensembles “successeurs” aussi pures que possible
  - ⇒ Ceci favorise les chemins courts dans l’arbre

# Arbres de décision (classification)

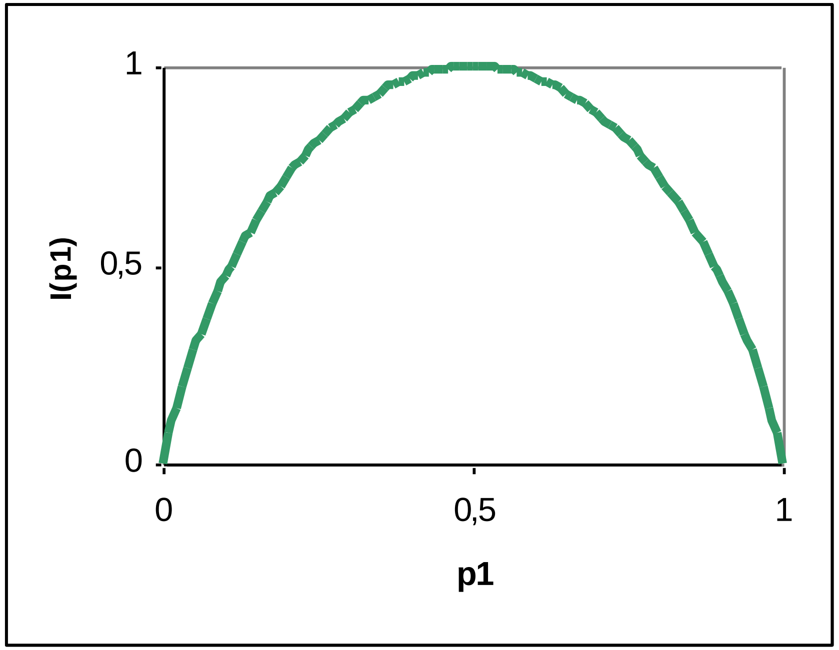
## Mesure d'impureté

- Entropie de Shannon :

✓ $H(S) = \sum_{ci \epsilon C} -P(ci) log2 P(ci)$

✓ Pour 2 classes, $p_1=1-p_2$

✓ $\mathrm{P(ci)} = \frac{|ci|}{|S|}$




- L'entropie mesure l'impureté, l'incertitude, la surprise…
- La réduction de l'entropie s'appelle : gain d'informations

# Arbres de décision (classification)

## Impureté

- Soit $S$ un ensemble d'objets, $p_j$ les proportions d'objets de classe $j$ ($j$=1,…,$J$) dans $S$,
- On définit une mesure d'impureté $H(S)$ qui satisfait les conditions suivantes :
  - $H(S)$ est minimum seulement quand $p_i$=1 et $p_j$=0 pour $j \neq i$ (tous les objets sont dans la même classe)
  - $H(S)$ est maximum seulement quand $p_j$ =1/$J$ (il y a exactement le même nombre d'objets de toutes les classes)
  - $H(S)$ est symétrique par rapport à $p_1,\ldots,p_J$

# Arbres de décision (classification)

## Réduction de l'impureté

Etant donnée un vecteur de caractéristiques $\vec{f}$, en utilisant les valuers d'une caractéristique $f_j$, on peut diviser l'ensemble de donnée *S* en plusieurs sous ensembles groupés dans un ensemble $S_j$. Le gain d'information est mesuré en se basant sur la différence entre l'entropie originale de *S* et celle après sa division en se basant sur une caractéristique $f_j$.

$$IG(S, f_j) = H(S) - \sum_{S_{jk} \in S_j} P(S_{jk}) H(S_{jk})$$

Où:

$$P(S_{jk}) = \frac{|S_{jk}|}{|S|}$$

# Arbres de décision (classification)

## Exemple

| temps | température | humidité | vent | jouer |
|---|---|---|---|---|
| ensoleilé | chaude | haute | non | non |
| ensoleilé | chaude | haute | oui | non |
| nuageux | chaude | haute | non | oui |
| pluvieux | douce | haute | non | oui |
| pluvieux | fraîche | normale | non | oui |
| pluvieux | fraîche | normale | oui | non |
| nuageux | fraîche | normale | oui | oui |
| ensoleilé | douce | haute | non | non |
| ensoleilé | fraîche | normale | non | oui |
| pluvieux | douce | normale | non | oui |
| ensoleilé | douce | normale | oui | oui |
| nuageux | douce | haute | oui | oui |
| nuageux | chaude | normale | non | oui |
| pluvieux | douce | haute | oui | non |

On calcule la probabilité de chaque classe:

P(jouer=oui) = 9/14

P(jouer=non) = 5/14

On calcule, ensuite, l'entropie de l'ensemble des données:

H(S) = - P(jouer=oui) * log2(P(jouer=oui)) - P(jouer=non) * log2(P(jouer=non))

H(S) = - 9/14 * log2(9/14) - 5/14 * log2(9/14)

H(S) = 0.41 + 0.53 = 0.94

# Arbres de décision (classification)

## Exemple

Pour chaque caractéristique, on calcule le gain d'information.

**temps:**

Le caractéristique "temps" divise les données sur 3 sous ensembles. Voici le nombre des occurrences de chaque classe dans chaque sous-ensemble:

| temps | jouer (oui) | jouer (non) |
|---|---|---|
| ensoleilé | 2 | 3 |
| nuageux | 4 | 0 |
| pluvieux | 3 | 2 |

On calcule la probabilité de chaque ensemble:

- P(S_ensoleilé) = 5/14
- P(S_nuageux) = 4/14
- P(S_pluvieux) = 5/14

On calcule l'entropie de chaque ensemble:

- H(S_ensoleilé) = - 2/5 * log2(2/5) - 3/5 * log2(3/5) = 0.971
- P(S_nuageux) = - 4/4 * log2(4/4) - 0/4 * log2(0/4) = 0
- P(S_pluvieux) = - 3/5 * log2(3/5) - 2/5 * log2(2/5) = 0.971

Le gain d'information de la caractéristique "temps":

IG(S, temps) = H(S) - P(S_ensoleilé) * H(S_ensoleilé) - P(S_nuageux) * H(S_nuageux) - P(S_pluvieux) * H(S_pluvieux)

IG(S, temps) = 0.94 - 5/14 * 0.971 - 4/14 * 0 - 5/14 * 0.971

IG(S, temps) = 0.247

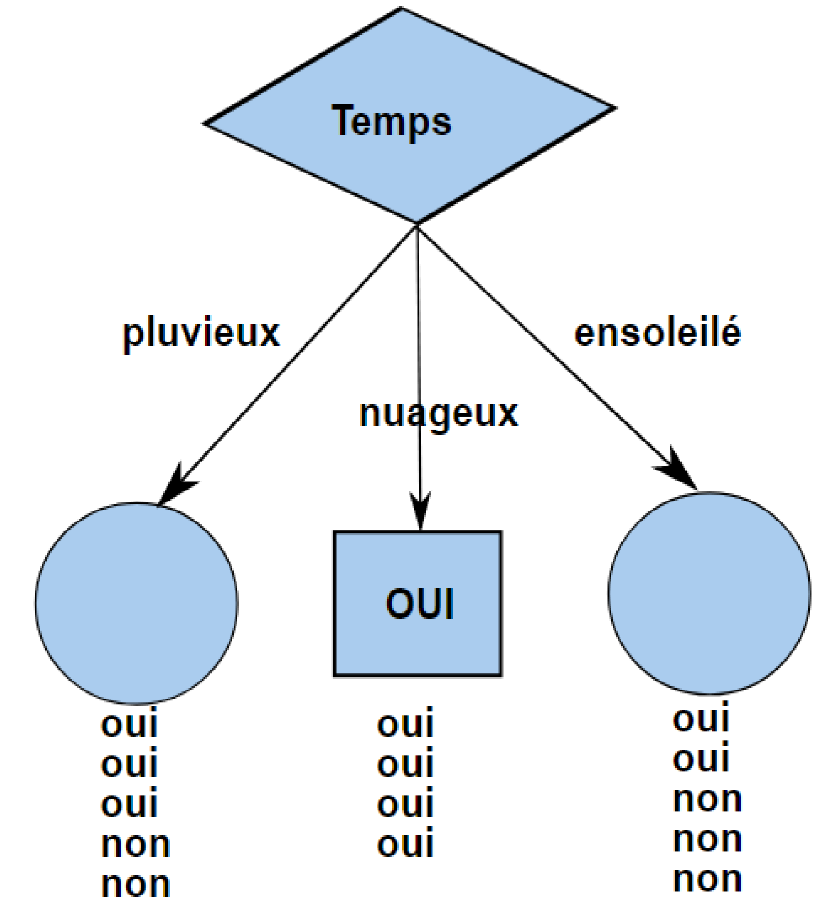
# Arbres de décision (classification)

## Exemple

En calculant le gain d'information des autres caractéristiques

| | temps | température | humidité | vent |
|---|---|---|---|---|
| IG | 0.247 | 0.029 | 0.152 | 0.048 |

Donc, la première caractéristique à vérifier dans l'arbre sera "temps". Comme l'entropie du temps étant "nuageux" est 0, cet ensemble contient des échantillons de la même classe. Donc, cet ensemble forme une feuille.

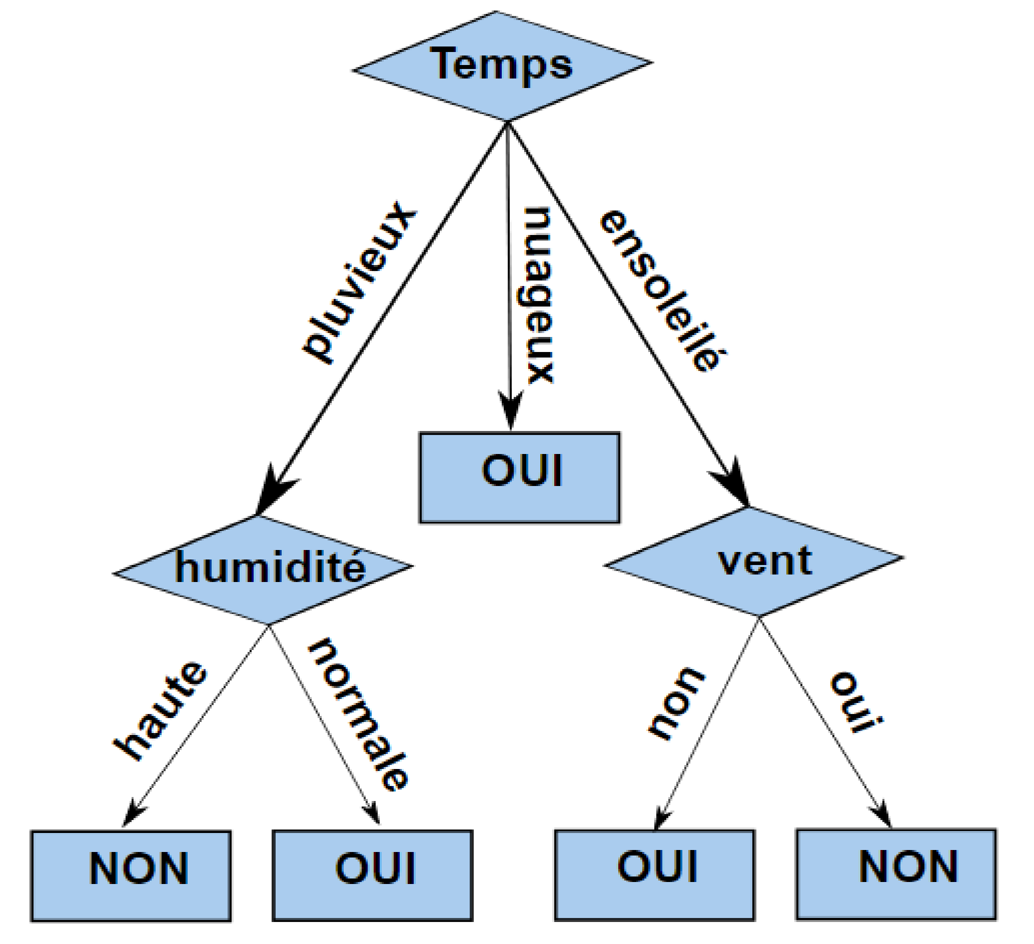



*Division des données selon la caractéristique "temps"*

# Arbres de décision (classification)

**Exemple**

On fait la même chose sur les sous ensembles.



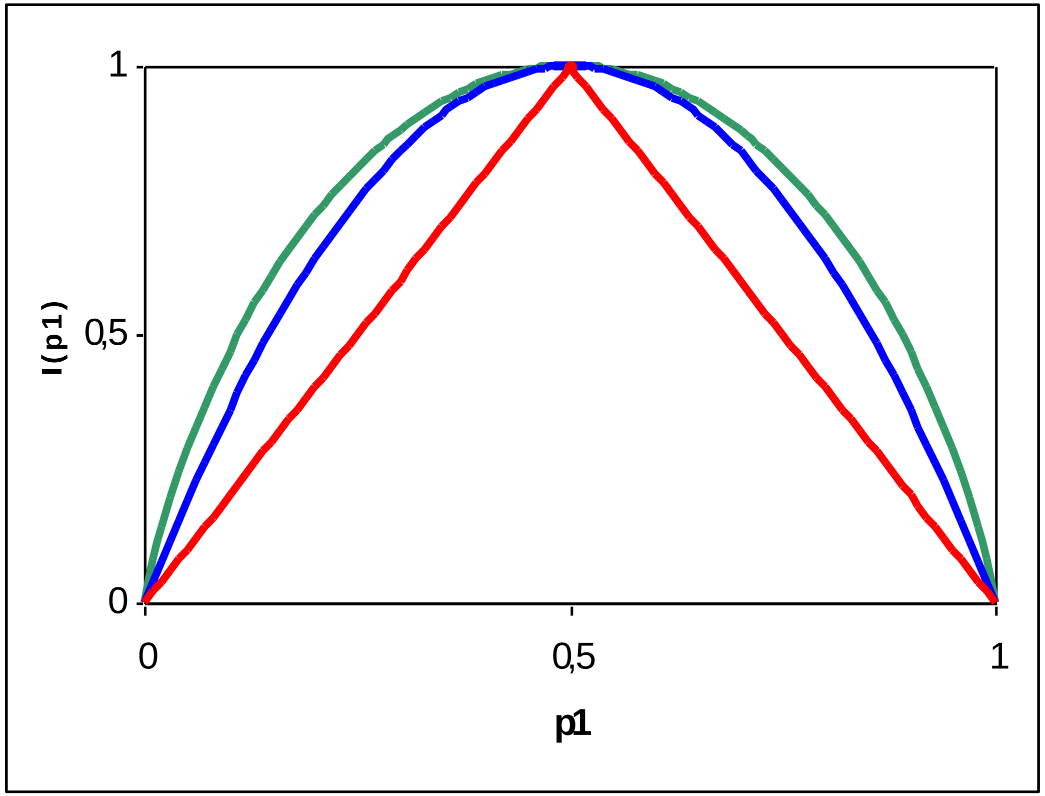

*L'arbre de décision en utilisant ID3*

# Arbres de décision (classification)

## Autres mesures d'impureté

- Index de Gini:
  - $I(S)=\sum_j p_j\,(1-p_j)$
- Taux d'erreurs de classification :
  - $I(S)=1-\max_j p_j$
- Cas de deux classes

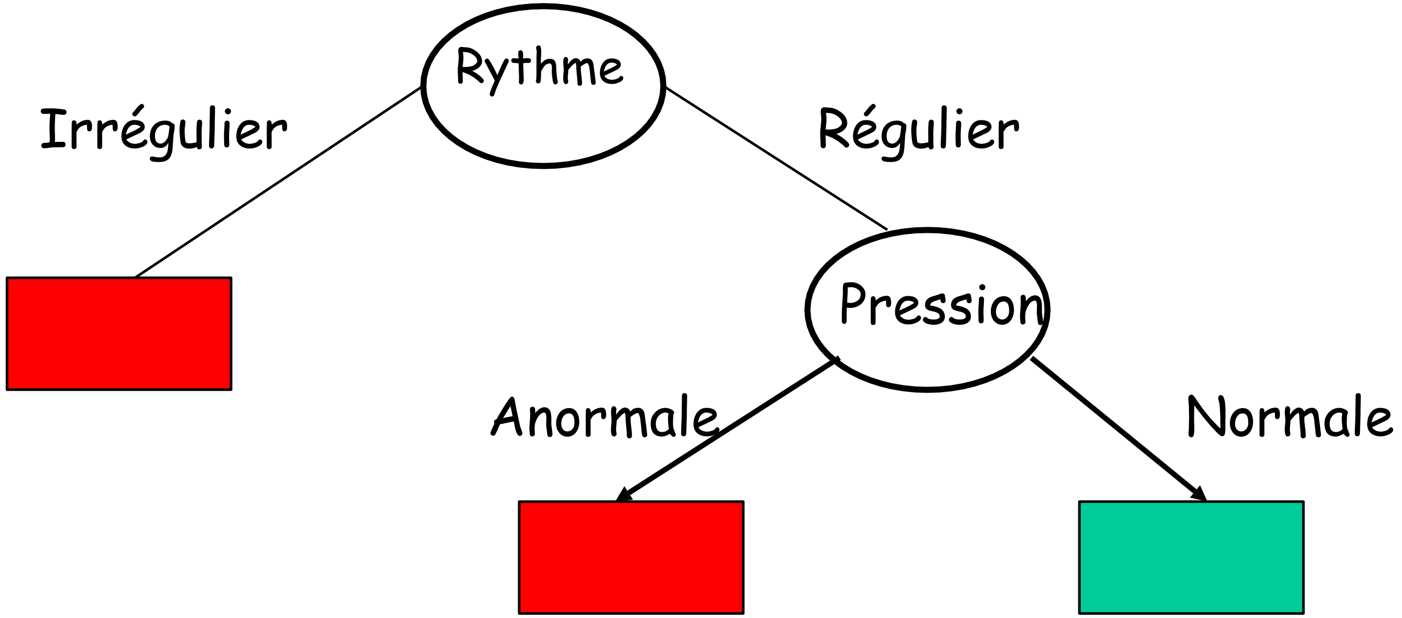
(après normalisation)



# Arbres de décision (classification)

## Arbre de décision

Un arbre de décision définit une fonction de classification dans l'espace (l'ensemble) d'entrée



# Arbres de décision (classification)

## Algorithme de construction d'un arbre binaire (récursif)

1. Soit S l'ensemble de données courant
Calculer les gains d'informations pour chaque attribut
Choisir l'attribut qui apporte le plus d'information
Construire le nœud avec cet attribut

2. Partager les données S en 2 sous-ensembles : S(gauche) et S(droit)
Tels que les données qui vérifient le test sur l'attribut vont à gauche
Les données qui ne vérifient pas le test vont à droite

3. Si S(gauche) contient uniquement des exemples d'une classe alors
arrêter la construction à gauche sinon aller en 1

4 idem du côté droit

# Arbres de décision (classification)

## Attributs numériques : comment les traiter ?

**Exemple** : température, pression, longueurs,…

**Solution** : poser une question du type $x > a$ avec $a$ valeur à déterminer

On définit l'information mutuelle par :

$\Delta I(S, X{:}a) = I(S) - I(S,X{:}a)$

Avec :

$I(S,X{:}a) = P(X < a)\, I(S,X<a) + P(x >= a) I(S, x >= a)$

$$\boxed{\Delta I(S, X) = \max_a \{I(S,X{:}a)\}}$$

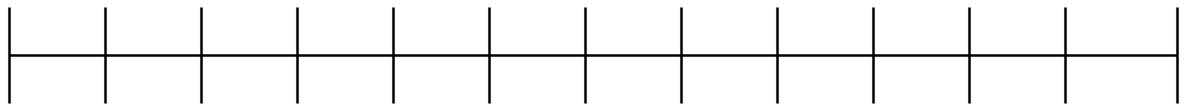
# Arbres de décision (classification)

Attributs numériques

Comment choisir a ?

On prend par exemples toutes les valeurs entre MinX et MaxX avec un pas de (MaxX-MinX)/m. m définit la granularité avec laquelle On veut étudier les données

Valeurs de l'attribut X



MinX et MaxX

# Arbres de décision (classification)

## Compromis profondeur de l'arbre/généralisation

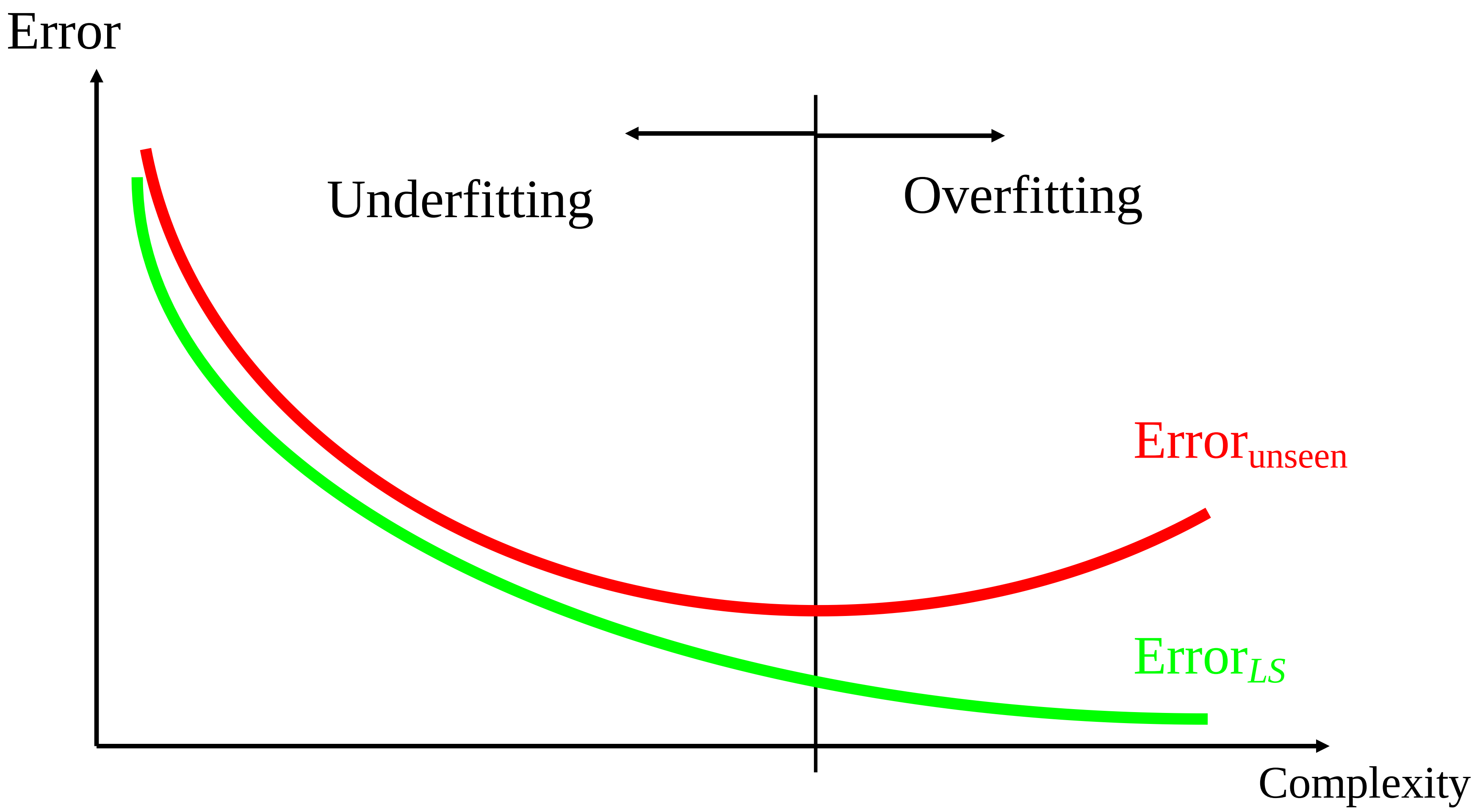
Arbre complet : feuilles "pures"
-> risque d'overfitting

**Solution** : arrêter l'arbre suivant un nb minimal d'exemples dans une feuille

***Autre solution*** : construire complètement puis élaguer suivant un ensemble de validation (distinct de l'ensemble d'apprentissage)

# Arbres de décision (classification)

## Overfitting




En pratique, $\text{Error}_{\text{unseen}}(T)$ est estimée à partir d'un ensemble de test.

# Arbres de décision (classification)

## Comment élaguer ?

- La méthode la plus simple est d'utiliser un ensemble de test, distinct de l'ensemble d'apprentissage.

- On parcourt l'arbre construit en comparant à chaque nœud intermédiaire l'erreur effectuée sur les données présentes
- si on s'arrêtait à ce nœud et la somme des erreurs effectuées sur les deux nœuds fils.
- Si l'adjonction des deux nœuds fils n'apporte pas d'amélioration, on les supprime.

# Arbres de décision (classification)

## Elagage d'un arbre de décision

L'arbre maximal construit selon la méthode que nous avons décrite risque de sur-apprendre les données d'apprentissage.

Un algorithme de post-élagage consiste à remettre en cause chaque nœud de l'arbre en utilisant :

- soit un ensemble de validation
- soit la validation croisée.

# Arbres de décision (classification)

## Exemple d'algorithme d'élagage

Notations :
$T_k$ arbre à k feuilles en moins

$T_{max}$ : arbre construit à partir de S (maximal).

**Idée de l'élagage :**
On construit un ensemble d'arbres à partir de $T_{max}$
avec de moins en moins de feuilles.

# Arbres de décision (classification)

## Elagage d'un arbre de décision

<u>On utilise un ensemble de validation différent de l'ensemble d'apprentissage.</u>

```
Elaguer(T_max) :
K←0
T_k←T_max
TANT QUE T_k a plus d'un nœud FAIRE
  POUR chaque nœud v de T_k FAIRE
   calculer w(T_k,v) sur exemples appr. ou valid.
  FIN_POUR
  choisir v_m = tel que w(T_k,v) soit minimum
  T_k+1: T_k où v_m a été remplacé par une feuille
  k←k+1
FIN_TANT_QUE
```

Pour finir, on choisit parmi {Tmax, T1, … Tn} l'arbre qui a la plus petite erreur de classification sur l'ensemble de *validation*

# Arbres de décision (classification)

## Critère en chaque nœud

**Soit l'arbre T, et v un de ses nœuds, et :**

- **MC(T,v) = nb d'exemples Mal Classés par v dans T**
- **$MC_{ela}$(T,v) = nb d'exemples Mal Classés par v dans l'arbre T *<u>élagué à v</u>***
- **n(T) = nb de feuilles de T**
- **nt(T,v) = nb de feuilles du sous-arbre de T sous nœud v**

**ALORS on prend comme critère à minimiser :**

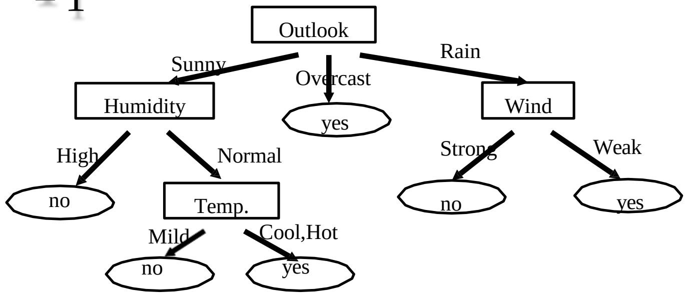
$$w(T,v) = (MC_{ela}(T,v)-MC(T,v))/(n(T)*(nt(T,v)-1))$$

**→ Prise en compte simultanée de l'erreur et de la complexité de l'arbre**

# Arbres de décision (classification)

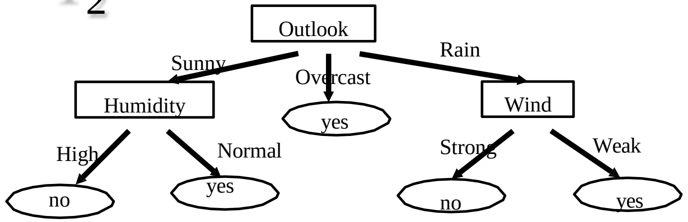
## Exemple d'élagage

$T_1$



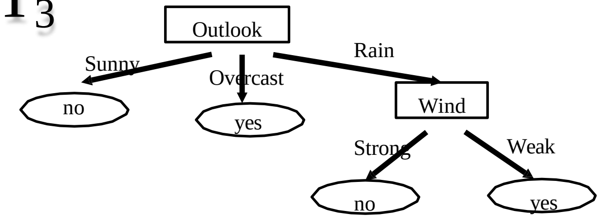

$Error_{GS}$=0%, $Error_{VS}$=10%

$T_2$



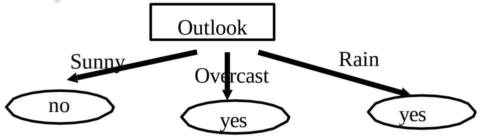

$Error_{GS}$=6%, $Error_{VS}$=8%

$T_3$




$Error_{GS}$=13%, $Error_{VS}$=15%

$T_4$




$Error_{GS}$=27%, $Error_{VS}$=25%

$T_5$


$Error_{GS}$=33%, $Error_{VS}$=35%

# Arbres de décision (classification)

## Comment évaluer un classifieur ?

- mesurer à quel point un classifieur est "performant"
- comparer/choisir entre deux classifieurs

# Arbres de décision (classification)

## Evaluation des classifieurs Méthode 1

**Erreur de classification en apprentissage :**

$Err_{app}$ = (nb d'ex d'app mal classes)/(nb total de données d'apprentissage)

**Erreur de classification en test :**
$Err_{test}$ = (nb d'ex de test mal classes)/(nb total de données de test)

# Arbres de décision (classification)

## Validation croisée Méthode 2

On divise l'échantillon d'apprentissage en B sous-ensembles $D_b$ de même taille

On applique la méthode d'estimation ou d'apprentissage sur chaque sous-ensemble $D - D_b$ : on mesure le risque empirique pour chaque ensemble $D_b$ restant.

$$Err_{cv}(h) = (1/B) \sum_b Err_b(h)$$

*Estimation par validation croisée*

$Err_b(h)$ : erreur moyenne de h sur $D_b$

Lorsque B= n, la méthode s'appelle : leave-one-out
(on n'en laisse qu'un de côté)

# Arbres de décision (classification)

## valuation par mesure de Receiver Operator Characteristics (ROC)

**Origine** : Théorie de la détection de signaux Egan 1975

Utilisé dans le domaine de la prévision de risque dans le domaine Médical depuis 35 ans

Comportements des systèmes de diagnostic Swets 1988

Visualisation dans le domaine médical Beck et Schultz 1986

Diverses applications à l'évaluation de classifieurs pour le data-mining :

**Provost et Fawcett 1996-97 (KDD et ICML)**

# Arbres de décision (classification)

## Notations pour un classifieur bi-classes

Pour un exemple x
la réponse du classifieur est : Y ou N
Sa « vraie » classe est Pos ou Neg

# Arbres de décision (classification)

## Terminologie des courbes ROC

Taux de vrais positifs True Positive rate (TP) : en ordonnée

TP = p(Y | Pos) » positifs correctement classés / nbre total de positifs

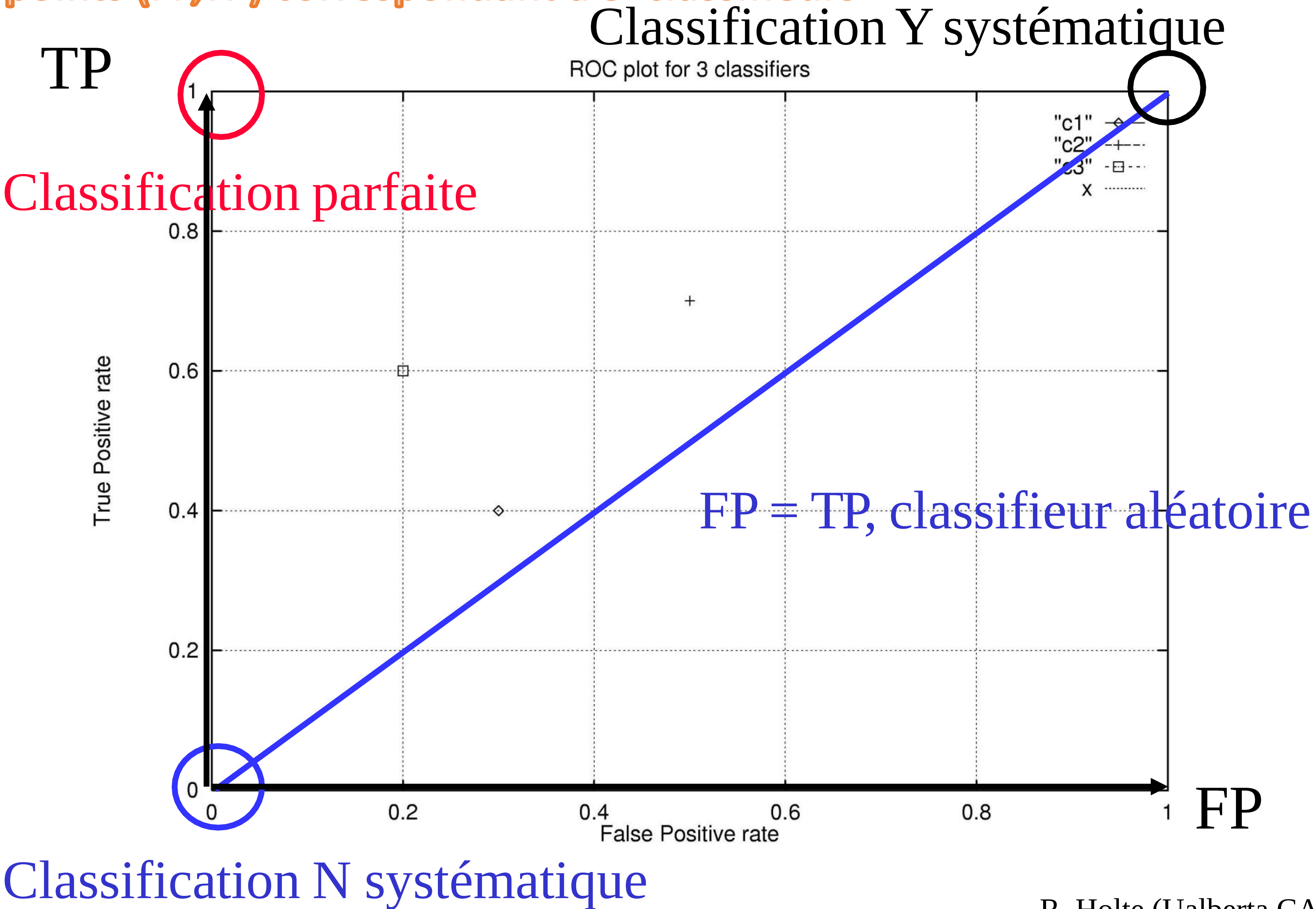
Taux de faux positifs, False Positive (FP) : en abscisse

FP = p(Y|Neg) » négatifs incorrectement classés / nbre total de négatifs

**Remarque :** ces quantités sont indépendantes des distributions des classes !

# Arbres de décision (classification)

## Exemple de 3 points (FP,TP) correspondant à 3 classifieurs




R. Holte (Ualberta,CA)

# Arbres de décision (classification)

## Courbe ROC

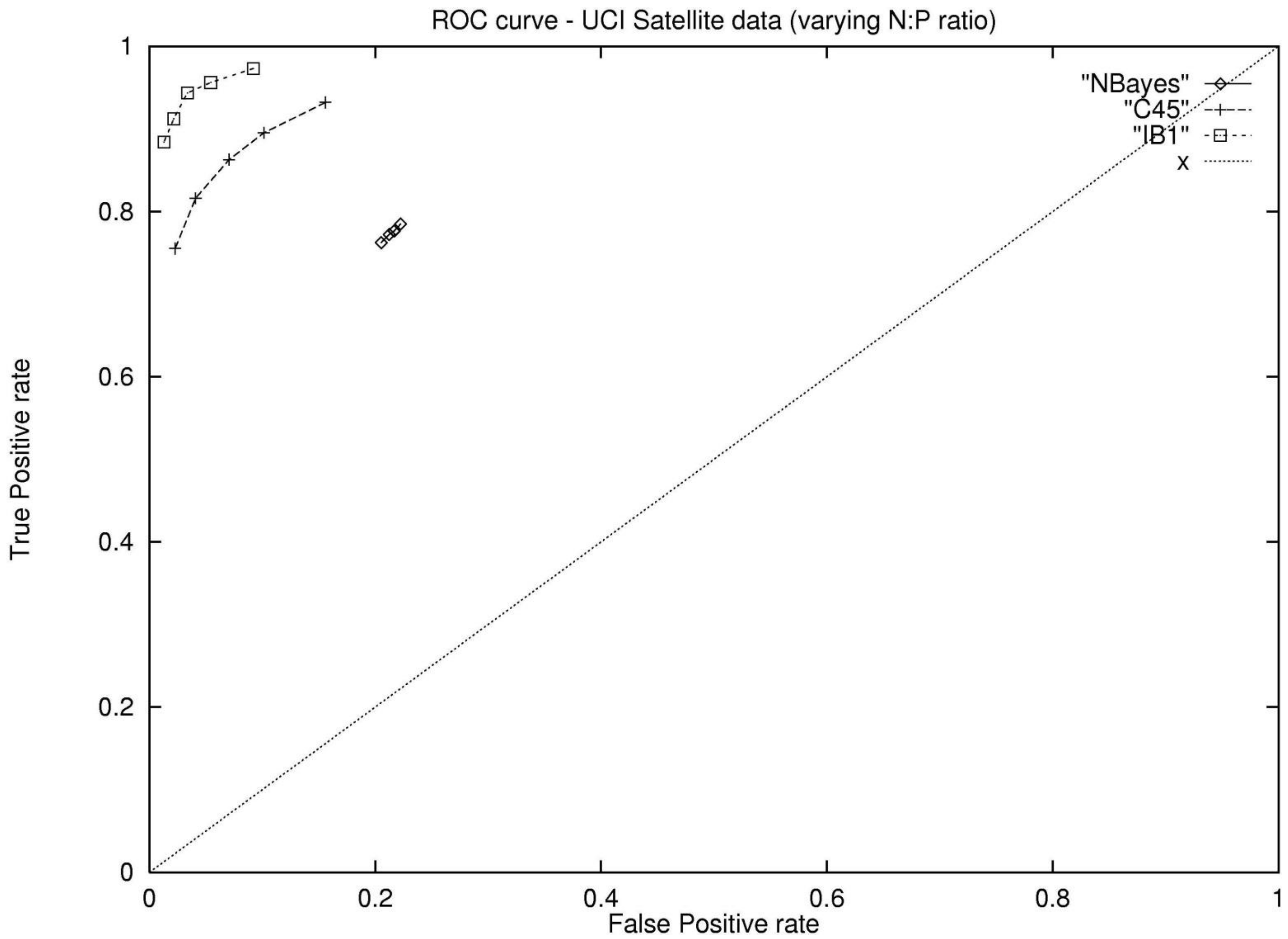
en abscisse les Faux positifs FP ,
en ordonnée les Vrais positifs TP

tracer les points ( FP,TP) correspondants à différents paramétrages d'un système d'apprentissage

ex:
déplacer le seuil d'acceptation d'un classifieur à sortie continue (si $P(Y|x)$ > seuil alors classe Pos)

# Arbres de décision (classification)

## Exemple de 3 classifieurs comparés selon leur courbe



# Arbres de décision (classification)

## Evaluation de classifieurs par courbes précision/rappel

**Origine** : recherche d'information, Salton et McGill 83

**La classe 1 est la classe pertinente pour la requête**

Rappel : R = TP
largeur du résultat de la recherche

Précision : Pr = tp/(tp+fp)
qualité du résultat de la recherche

AVEC TP = tp/p
p : positifs (tous ceux de classe 1)
tp : true positive (prédits 1 et de classe 1)
fp : false positive (predits 1 mais de classe 0)

# Arbres de décision (classification)

## Un critère unique ?

- On voudrait que R et P soient maximaux

- Comment avoir une mesure unique ?
  - - fixer des valeurs de rappels, mesurer la précision
  - - moyenne
  - - F-mesure :

    - $(b^2 + 1)\ PR\ F_b(R,P) \quad =$

    ---

    - $b^2\ P + R$

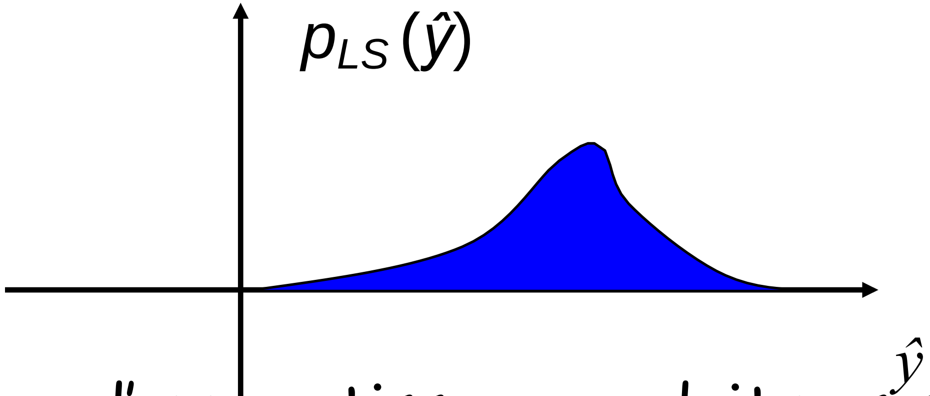
- en modifiant b on favorise l'un ou l'autre des critères

Combiner pour améliorer les performances et diminuer la sensibilité aux données

## Métodes d' ensembles

# Arbres de décision (classification)

## Bon algorithme d'apprentissage



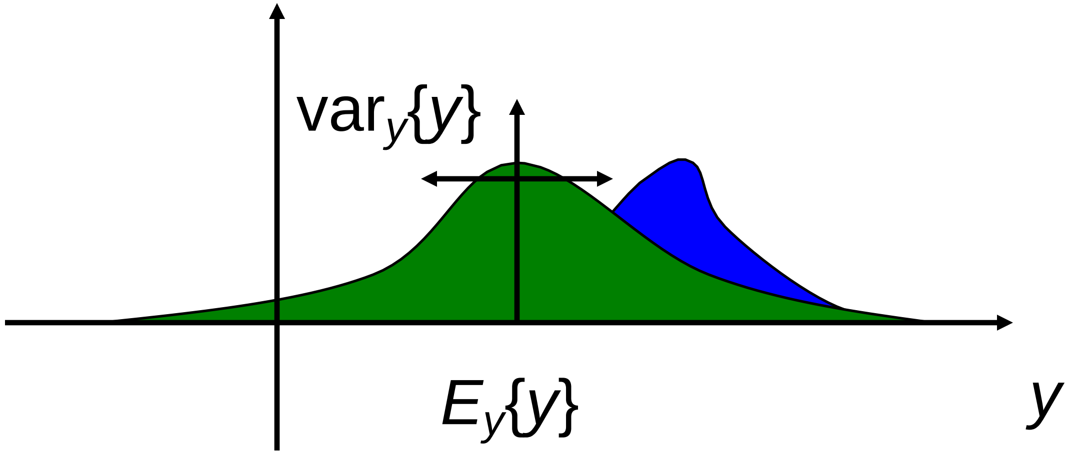

Un bon algorithme d'apprentissage ne doit pas seulement être bon sur un ensemble d'apprentissage mais aussi sur tous les ensembles d'apprentissage (de taille N).
$\Rightarrow$ on veut minimiser :

$$E = E_{LS}\{E_y\{(y - \hat{y})^2\}\}$$

Analysons cette erreur en détail

# Arbres de décision (classification)

## Décomposition Biais/variance (2)



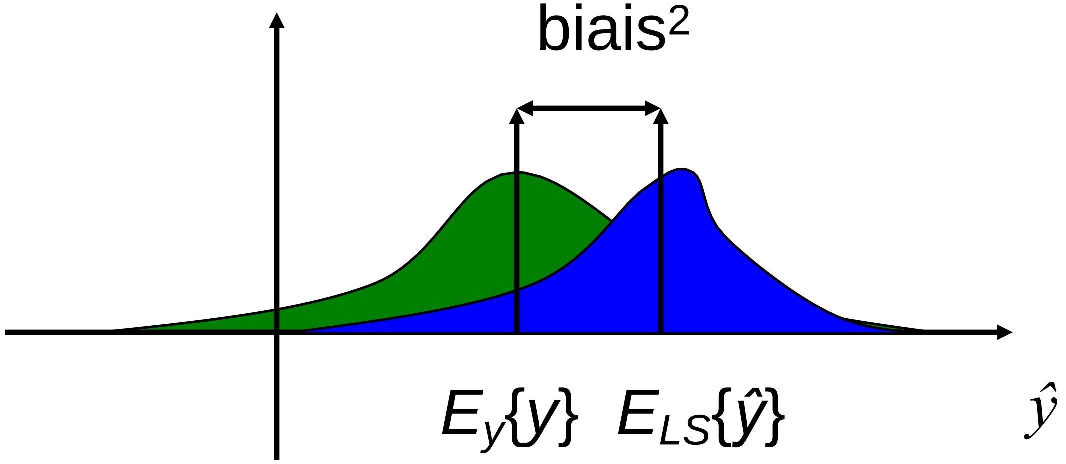

$$E = E_y\{(y - E_y\{y\})^2\} + E_{LS}\{(E_y\{y\} - \hat{y})^2\}$$

= erreur residuelle = minimal attainable error

= $var_y\{y\}$

# Arbres de décision (classification)

## Décomposition Biais/variance (4)



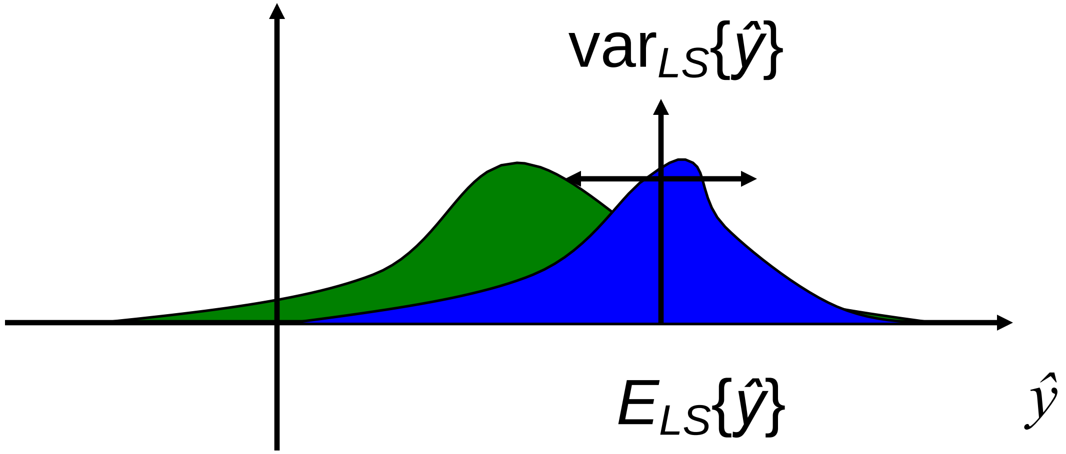

$E$= var$_y\{y\}$ + $(E_y\{y\}-E_{LS}\{\hat{y}\})^2$ + …

$E_{LS}\{\hat{\boldsymbol{y}}\}$ = modèle "moyenne "(sur tous les LS)

biais² = Erreur entre Bayes et le modèle "moyenne"

# Arbres de décision (classification)

## Décomposition Biais/variance (5)

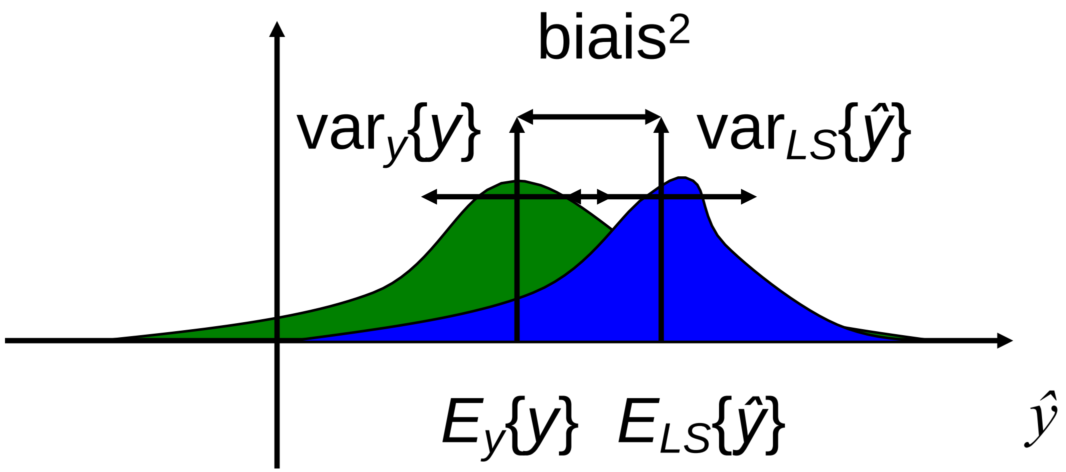



$E$= $var_y\{y\}$ + $biais^2$ + $\boldsymbol{E_{LS}\{(\hat{y}-E_{LS}\{\hat{y}\})^2\}}$

$var_{LS}\{\hat{\boldsymbol{y}}\}$ = estimation de la variance = conséquence du surapprentissage

# Arbres de décision  (classification)

## Biais/variance decomposition (6)




$$E = \mathrm{var}_y\{y\} + \mathrm{biais}^2 + \mathrm{var}_{LS}\{\hat{\boldsymbol{y}}\}$$

# Arbres de décision (classification)

## Les méthodes d'ensembles

- Combinent les prédictions (ou solutions) de plusieurs modèles construits avec un algorithme d'apprentissage pour améliorer les résultats.
- Deux familles importantes :
  - Techniques qui effectuent des moyennes
    - Construisent plusieurs modèles de façon indépendante et effectuent simplement la moyenne des résultats.
    - Ex: bagging, random forests (forêts aléatoires)
    - Décroissance de la variance des modèles
  - Algorithmes de type
    - Construisent plusieurs modèles de façon séquentielle
    - Ex: Adaboost, MART
    - Décroissance du biais

# Arbres de décision (classification)

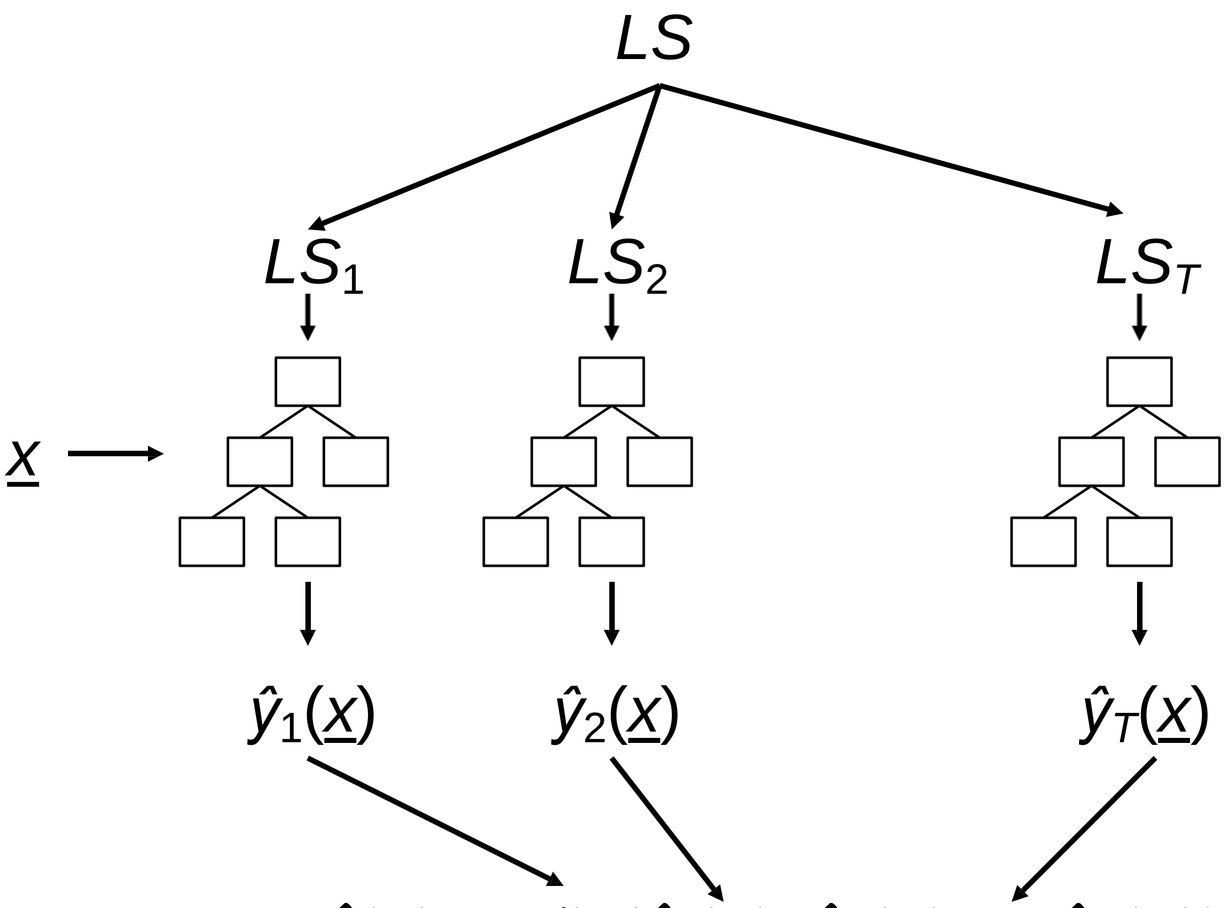
## Bagging (1)

$$E_{LS}\{Err(\underline{x})\}=E_{y|\underline{x}}\{(y-h_B(\underline{x}))^2\}+(h_B(\underline{x})-E_{LS}\{\hat{y}(\underline{x})\})^2+E_{LS}\{(\hat{y}(\underline{x})-E_{LS}\{\hat{y}(\underline{x})\})^2\}$$

- **Idée** : le modèle "moyenne" $E_{LS}\{\hat{y}(\underline{x})\}$ possède le même biais que la méthode d'origine mais une variance nulle.
- **Bagging** (Bootstrap ***AGG**regat**ING***) :
  - Pour calculer $E_{LS}\{\hat{y}(\underline{x})\}$, générer un nombre infini d'ensembles *LS* (de taille N)
  - Comme on a généralement un seul ensemble *LS*, on effctue un echantillonnage de type bootstrap à partir de *LS*
  - Bootstrap sampling = echantillonnage avec remplacement de *N* objects dans *LS* (*N* est la taille de *LS*)

# Arbres de décision (classification)

## Bagging (2)




En regression: $\hat{y}(\underline{x}) = 1/k.(\hat{y}_1(\underline{x})+\hat{y}_2(\underline{x})+\ldots+\hat{y}_T(\underline{x}))$

en classification: $\hat{y}(\underline{x})$ = la classe majoritaire dans $\{\hat{y}_1(\underline{x}),\ldots,\hat{y}_T(\underline{x})\}$

# Arbres de décision (classification)

## Bagging (3)

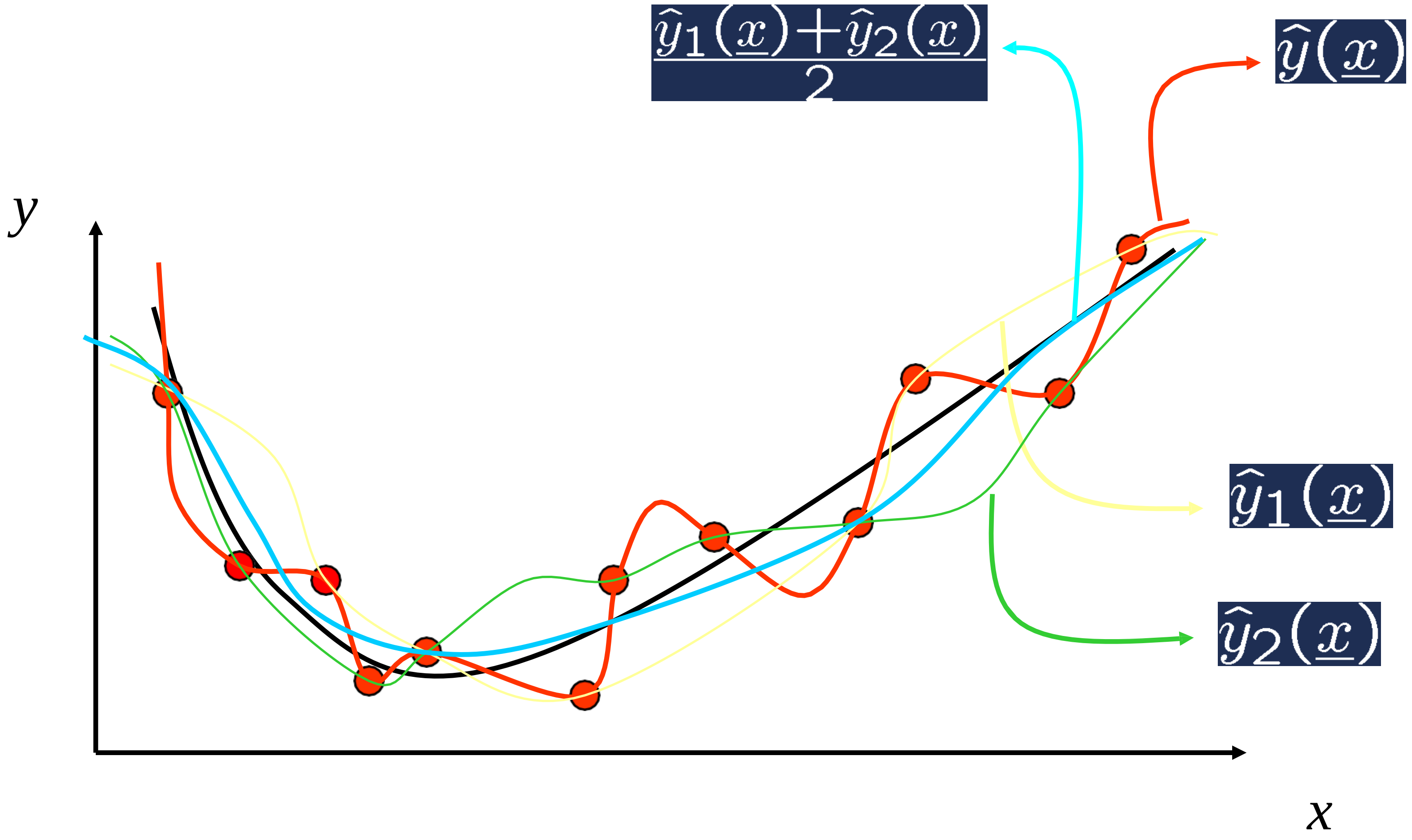
- Le bagging réduit beaucoup la variance sans augmenter beaucoup le biais.
- Application aux arbre de régression

| Method | E | Bias | Variance |
|---|---|---|---|
| 3 Test regr. Tree | 14.8 | 11.1 | 3.7 |
| Bagged (T=25) | 11.7 | 10.7 | 1.0 |
| Full regr. Tree | 10.2 | 3.5 | 6.7 |
| Bagged (T=25) | 5.3 | 3.8 | 1.5 |

- Réduction importante de la variance sans augmenter le biais (le modèle est bien sur beaucoup plus complexe qu'un simple arbre de décision)

# Arbres de décision (classification)

## Bagging (4)



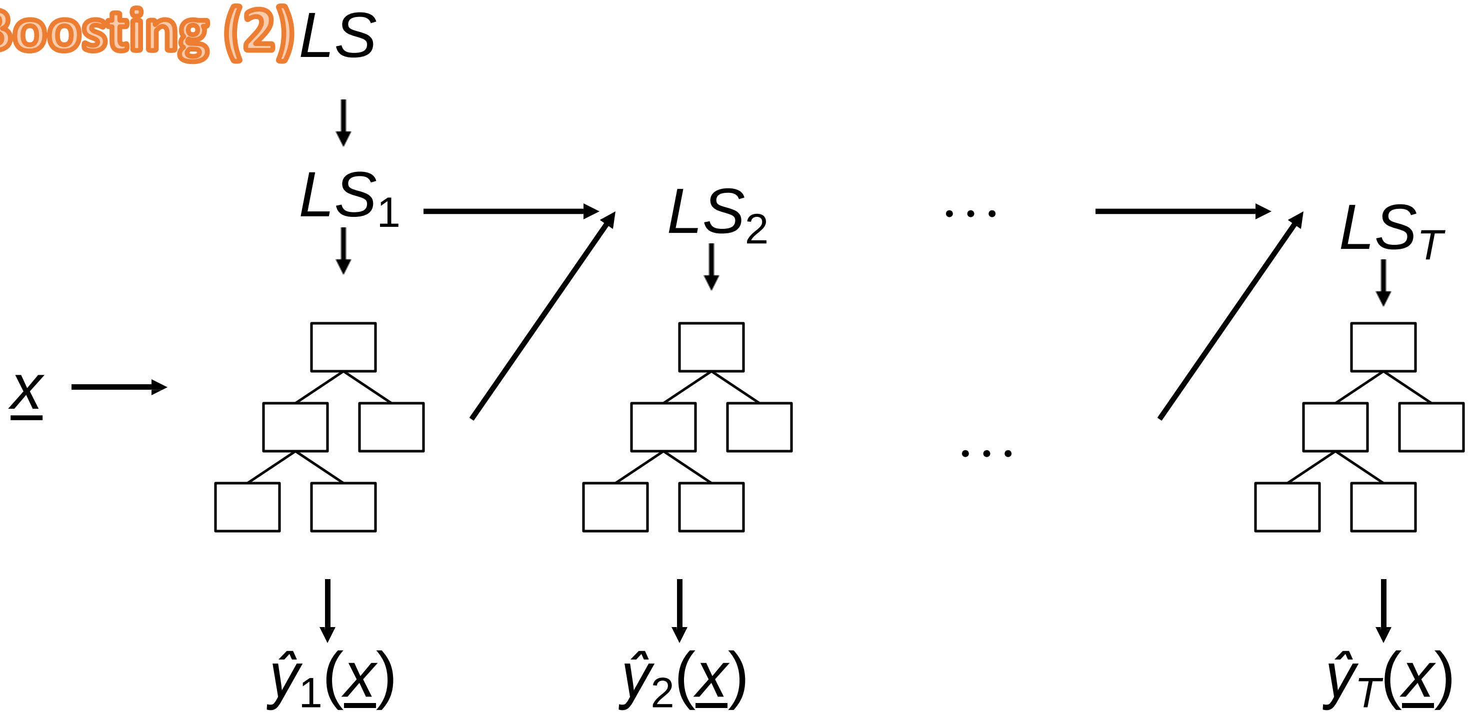
# Arbres de décision (classification)

## Méthodes de Boosting (1)

- La motivation du boosting est de combiner les sorties de plusieurs modèles « faibles » pour produire un modèle performant.
- Modèle faible = un modèle qui a un biais important (par exp, en classification, le modèle est légèrement meilleur qu'un classement aléatoire)
- Différences avec les méthodes d'ensembles précédentes :
  - Les modèles sont construits séquentiellement sur des versions modifiées des données.
  - Les prédictions des modèles sont combinées en utilisant des sommes/votes pondérés.

# Arbres de décision (classification)

## Méthodes de Boosting (2)




En régression: $\hat{y}(\underline{x}) = \beta_1.\hat{y}_1(\underline{x}) + \beta_2.\hat{y}_2(\underline{x}) + \ldots + \beta_T.\hat{y}_T(\underline{x}))$

En classification: $\hat{y}(\underline{x})$ = la classe majoritaire dans $\{\hat{y}_1(\underline{x}),\ldots,\hat{y}_T(\underline{x})\}$ pondérées par les poids $\{\beta_1,\beta_2,\ldots,\beta_T\}$

# Arbres de décision (classification)

## Adaboost (1)

- Supposons que l' algorithme d' apprentissage permet l' association des poids aux objets
- C' est le cas de plusieurs algorithmes d' apprentissage :
  - Pour les arbres, simplement prendre en compte les poids associés aux objets.
- A chaque étape, adaboost augmente les poids des exemples mal classés par le dernier modèle.
- Donc, l' algorithme se focalise sur les cas difficiles de l' ensemble d' apprentissage.
- Dans le vote majoritaire pondéré, adaboost donne une plus grande influence pour les modèles les plus précis.

# Arbres de décision (classification)

## Adaboost (2)

- Entrées : un algorithme d'apprentissage et un ensemble d'apprentissage $\{(x_i,y_i): i=1,\dots,N\}$
- Initialiser les poids $w_i=1/N$, $i=1,\dots,N$
- Pour $t=1$ à $T$
  - Construire un modèle $\hat{y}_t(\underline{x})$ avec l'algorithme d'apprentissage en utilisant les poids $w_i$
  - Calculer l'erreur pondérée :

$$err_t = \frac{\sum_i w_i I(y_i \neq \hat{y}_t(x_i))}{\sum_i w_i}$$

  - Calculer $b_t=\log((1-err_t)/err_t)$
  - Changer les poids :

$$w_i \leftarrow w_i \times \exp[\beta_m \times I(y_i \neq \hat{y}_t(x_i))]$$

# Arbres de décision (classification)

## Exemple d'application (Golub's microarray data)

- 72 objects, 7129 numerical attributes (gene expressions), 2 classes (ALL and AL)
- Leave-one-out error with several variants

| Method | Error |
|---|---|
| 1 decision tree | 22.2% (16/72) |
| Random forests (k=85,T=500) | 9.7% (7/72) |
| Extra-trees ($s_{th}$=0.5, T=500) | 5.5% (4/72) |
| Adaboost (1 test node, T=500) | 1.4% (1/72) |

- Variable importance with boosting