Introduction à l'Extraction de Connaissances

Chapitre IV Part 1 : Exploration & les méthodes

Bayes, Arbre de Décision

Alexandre Saidi
Master -Informatique
ECL - LIRIS - CNRS

Octobre 2017

Introduction et Rappels

Rappel : EC est un domaine multi disciplinaires

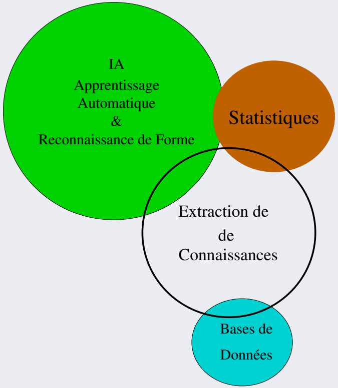
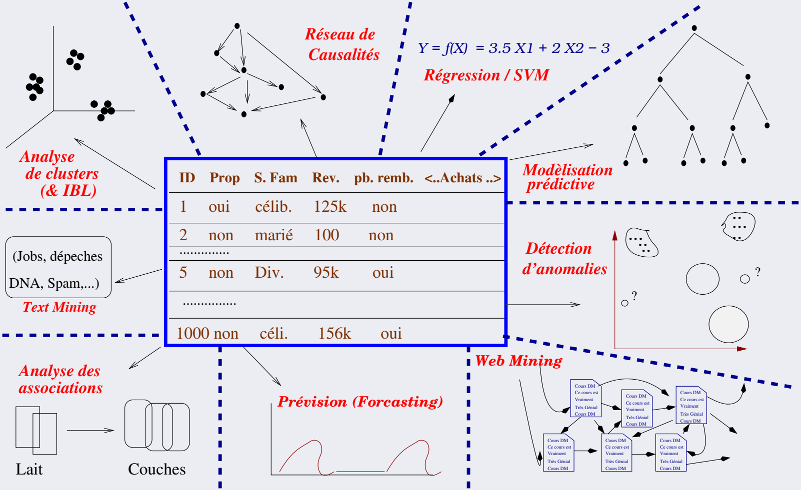


FIGURE 1: ECD issue des disciplines importantes

Introduction et Rappels (suite)

Rappel : principales méthodes de l'ECD



Introduction et Rappels (suite)

Rappel : le processus ECD

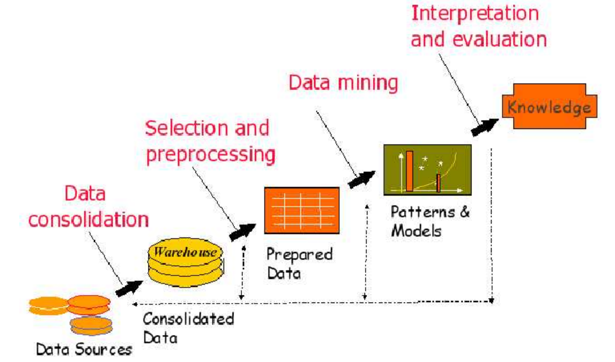


FIGURE 2: Processus

Notes sur ce document

- Techniques pratiques de base utilisées en Extraction de Connaissances
  - Méthodes basiques exploratoires
  - Méthodes statistiques (Bayésiennes)
  - Arbres de Décision
  - Règles de classification
  - Règles d'association
  - Méthodes à base de noyaux (Kernel) : SVM
  - Clustering, etc.

N.B. : Les chapitres 2 et 3 (un seul fichier) expliquent différentes formes des entrées (data) et sorties du processus DM (concepts).
➛ A lire !

Faire simple d'abord !

- **Constat** : les idées simples et basiques marchent souvent bien !
- Principe des méthodes basiques :

  Vérifier si un seul attribut décrit tout dans une BD.

  ➢ Les autres attributs seraient redondants, pas assez utiles (ou discriminants), indépendants, participant de manière égale aux résultats.

- Dans les méthodes basiques et simples, on cherche :
  - Une simple structure logique appropriée (peu d'attributs / règles)
  - Des relations suffisantes / corrélation entre certains attributs
  - Une combinaison linéaire des attributs peut être suffisante

    ➢ avec éventuellement des pondérations
  - Si exploration positive, une méthode à base d'instances peut être utilisée

    ➢ notion de distance

Faire simple d'abord ! (suite)

Inconvénients de la simplicité :

- Le succès de la méthode dépend du domaine !
- BDs différentes → concepts différents (**biais** vs. **variance**)
- Une méthode qui cherche telle régularité peut en rater une autre plus intéressante/plus simple/plus claire...
- On peut avoir une équation linéaire entre des attributs numériques
- Etc.

Pour ces raisons :

- Il faut être prudent !
- Les méthodes simples utilisées au stade d'exploration (de tâtonnement).

Inférence de règles rudimentaires

- A la recherche d'une structure rudimentaire ➛ un attribut suffit.
- La **méthode 0R**
- La **méthode 1R** génère un arbre (de décision) d'un seul niveau :
  - Produit une classification simple
  - Donnant un ensemble de règles testant un seul attribut.
- Cas simple : seulement des attributs nominaux
  - Une branche par valeur d'attribut
  - Chaque branche affecte la classe la plus fréquente
  - On cherche le meilleur attribut qui **représente** l'ensemble d'apprentissage
- Evaluation (Taux d'erreur) :

  La proportion d'instances qui n'appartiennent pas à la classe majoritaire de la branche correspondante.

  ➛ On choisit l'attribut qui réduit le taux d'erreur.

Inférence de règles rudimentaires (suite)

Algorithme de principe trivial :

Pour tout attribut A_i
  Pour toute valeur V_{ij} de A_i, envisager une règle de la manière suivante :
    - Compter le nombre de fois où chaque classe apparaît
    - Trouver la classe C_k la plus fréquente
    - Par cette règle, affecter la classe C_k à $< A_i, V_{ij} >$ (attribut-valeur)
  Calculer la taux d'erreur des règles
Choisir les règles avec le taux d'erreur minimum

- Méthode simple et naïve, donne de bons résultats (selon la BD).

➛ Le choix du meilleur attribut peut se faire par différentes méthodes

Inférence de règles rudimentaires (suite)

- Application à l'exemple "météo" :

Outlook	Tmp.	Hum.	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

- Classement sur la dernière colonne ("play" yes/no)

Outlook	Tmp.	Hum.	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
...	...	...	...	...

Inférence de règles rudimentaires (suite)

La méthode 1R appliquée à l'exemple "Météo" :

Outlook	Tmp.	Hum.	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

TABLE 1: BD exemple "météo"

Att.	Règle	Err.	Tot. err.
Outlook	Sunny → No	2/5	
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	4/14
Temp.	Hot → No*	2/4	
	Mild → Yes	2/6	
	Cool → Yes	1/4	5/14
Humidity	High → No	3/7	
	Normal → Yes	1/7	4/14
Windy	False → Yes	2/8	
	True → No*	3/6	5/14

TABLE 2: Règles **1R** de l'exemple "météo"

- Un astérisque (*) : choix aléatoire sur une égalité (*play : yes/no*)
- **Meilleures règles** : 1er et 3e paquets (4/14) → on garde l'une ou l'autre

- N.B. : en cas de valeur manquante (pour un attribut) :
  ➢ une nouvelle valeur = "manquante", "absente", etc.
- Si attributs numériques : discrétisation

Discrétisation des attributs numériques

Supposons un attribut numérique *température* dans la BD "météo" $\in [0..200]$

- On trie les instances selon l'attribut numérique à discrétiser :

Temp :	64	65	68	69	70	71	72	72	75	75	...
Play :	yes	no	yes	yes	yes	no	no	yes	yes	yes	...

➛ On place un *point de rupture* à chaque fois que la classe change

- Dans l'exemple, 8 partitions sur l'ensemble des valeurs de *Température*

➛ Tout changement de classe implique une partition :

Temp : 64 65 68 69 70 71 72 72 75 75 85

yes | no | yes yes yes | no no | yes yes yes | no | yes yes| no

N.B. : un test par partition (tout changement de classe) ➛ 8 tests ici

➛ erreur minimale mais beaucoup de tests + fort risque d'*overfitting*.

Discrétisation des attributs numériques (suite)

Problème de sur-apprentissage (*overfitting ou sur-adaptation*)

- Il y a "overfitting" quand on colle trop aux données (Variance élevé).

Variance : quand la méthode colle trop aux données, elle intègre dans le modèle obtenu le bruit aléatoire des données d'apprentissage plutôt que les sorties prévues.
Biais : peut venir d'une méthode (algorithme d'apprentissage) qui manque de relations pertinentes entre les données en entrée et les sorties prévues **(sous-apprentissage)**.

- Overfitting si un attribut a un grand nombre de valeurs différentes :
  - Si on associe une partition à chaque changement de classe, on diminue le taux d'erreur mais le modèle construit n'est pas exploitable (trop d'erreur de test avec une temp. non rencontrée).

Ce problème se pose pour toutes les méthodes ! !

- **Un cas limite** : attribut avec une valeur différente par instance (date, id, ...)

Discrétisation des attributs numériques (suite)

- Partitions (suite) : *Points de Rupture*

64 | 65 | 68 69 70 | 71 72 | 72 75 75 | 80 | 81 83 | 85

yes | no | yes yes yes | no no | yes yes yes | no | yes yes | no

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

64.5 66.5 70.5 72 73.5 77.5 80.5 84.0

➢ Les deux 72 posent **problème** : même valeur mais classes différentes.

Solution :

- Déplacer la rupture 72 en 73.5 → une partition avec 2 × "no" et 1 "yes".
- **1R** attribue la classe **majoritaire** (ici "no") à la partition.
- Un test en moins mais une erreur en plus !

Discrétisation des attributs numériques (suite)

Pour éviter un changement fréquent de classes :

- ◦ Imposer un **nombre minimum** d'instances (e.g. 3) par classe majoritaire dans chaque partition.

- Exemple : si on a

yes | no | yes yes | yes no no | yes yes | yes ...

- ◦ L'instance voisine (ici le précédente) est aussi yes, on l'inclue :

yes | no | yes yes yes | no no | yes yes | yes ...

➛ On a des partitions à 2 instances (il en faut 3 de la classe majoritaire) !

➛ Pas trop gênant car la classe est homogène (mais introduit un test de plus)

- **Autre traitement** pour diminuer les variations :

On fusionne 2 partitions si elles ont la même classe majoritaire.

➛ Augmente le taux d'erreur mais diminue les tests.

Discrétisation des attributs numériques (suite)

Retour au traitement des partitions de l'exemple météo :

64 |	65 |	68	69	70 |	71	72 |	72	75	75 |	80 |	81	83 |	85
yes |	no |	yes	yes	yes |	no	no |	yes	yes	yes |	no |	yes	yes|	no
↑	↑			↑		↑	↑		↑	↑		↑	
64.5	**66.5**			**70.5**		**72**	**73.5**		**77.5**	**80.5**		**84.0**	

- On (l'outil) fusionne la frontière 72
  ➛ On respecte le minimum 3 pour la classe majoritaire

yes |	no |	yes	yes	yes |	no	no	yes	yes	yes |	no |	yes	yes|	no

- L'outil de partitionnement a un paramètre nb_partitions (*bins*, ici = 3)
  ➛ Il fusionne là où il peut (en minimisant l'erreur) pour arriver à :

64	**65**	**68**	**69**	**70**		**71**	**72**	**72**	**75**	**75**		**80**	**81**	**83**	**85**
Yes	**No**	**Yes**	**Yes**	**Yes**	|	**No**	**No**	**Yes**	**Yes**	**Yes**	|	**No**	**Yes**	**Yes**	**No**

➛ A droite, le seuil 3 non respecté (pas 3 instances majoritaires)

Discrétisation des attributs numériques (suite)

N.B. : les dernières partitions (pour *bins=3*) :

64	**65**	**68**	**69**	**70**		**71**	**72**	**72**	**75**	**75**	|	**80**	**81**	**83**	**85**
Yes	**No**	**Yes**	**Yes**	**Yes**	|	**No**	**No**	**Yes**	**Yes**	**Yes**	|	**No**	**Yes**	**Yes**	**No**

➢ Toute discrétisation n'est pas forcément bonne !

- Si *bins=2* :
  - La partition de droite ne peut pas être fusionnée avec celle du milieu
    - ➢ Car la décision finale serait "Yes" (partout !)
    - ➢ la classe de la partition de droite sera **"no"**
  - Et on fusionne la partition de gauche et celle du milieu

Discrétisation des attributs numériques (suite)

Donc, avec table précédente :

64	**65**	**68**	**69**	**70**	**71**	**72**	**72**	**75**	**75**	|	**80**	**81**	**83**	**85**
Yes	**No**	**Yes**	**Yes**	**Yes**	**No**	**No**	**Yes**	**Yes**	**Yes**	|	**No**	**Yes**	**Yes**	**No**

- Pour ce partitionnement, la règle sur l'attribut "Température" sera :

$$\boxed{\begin{array}{l} \textit{If Temperature} \leq 77.5 \textit{ then Play} = \textit{yes}\,; \\ \textit{If Temperature} > 77.5 \textit{ then Play} = \textit{non}\,; \end{array}}$$

➛ Taux d'erreur : 5/14

☞ Dans cet exemple :

- La méthode 0-R aurait donné le même taux d'erreur (5/14)
- 1-R aussi !

Discrétisation des attributs numériques (suite)

Remarques : sur le dernier tableau

64	65	68	69	70		71	72	72	75	75		80	81	83	85
Yes	No	Yes	Yes	Yes	|	No	No	Yes	Yes	Yes	|	No	Yes	Yes	No

- Un autre partitionnement peut donner :

64	65	68	69	70	|	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	|	No	No	Yes	Yes	Yes	No	Yes	Yes	No

➛ On respecte le seuil de 3 (de la classe majoritaire)

➛ On aura le même taux d'erreur ($= 5/14$)

☞ Ne pas tout fusionner → *Sine qua nihil praeceptum ad "temperature"*!

- Les outils (cf. Weka utilisé pour les BEs) sont paramétrés par :

  - le nombre de partitions et

  - le nombre de classes majoritaires (à respecter).

☞ Le partitionnement fait perdre de l'information.

Résultats Weka

- Weka envisage ce tableau des 1-R possibles (avec l'erreur de chacune)

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temperature	≤ 77.5 → Yes	3/10	5/14
	> 77.5 → No*	2/4	
Humidity	≤ 82.5 → Yes	1/7	3/14
	> 82.5 and ≤ 95.5 → No	2/6	
	> 95.5 → Yes	0/1	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

FIGURE 3: 1R appliquée à l'exemple "Météo" (données discrétisées)

Remarques : la règle sur la *Temperature* donne **5 erreurs** ;

➛ moins bien que la règle sur "Outlook" (4 erreurs) ou "Humidity" (3/15).

Résultats Weka (suite)

- Pour améliorer le taux d'erreur :

  Weka choisit finalement une règle sur l'attribut **Humidity** :

Humidity :	$\leq 82.5 \rightarrow$ **yes**
	$> 82.5 \quad \& \quad \leq 95.5 \rightarrow$ **no**
	$> 95.5 \rightarrow$ **yes**

  3 erreurs ➛ le meilleur résultat "1-R" pour cette BD.

- Rappel : si un attribut numérique a des valeurs manquantes :

  ➛ Une catégorie supplémentaire est créée (0, -1, ∞, etc.)

  ➛ On évite l'influence de cette valeur particulière sur la discrétisation par :
  - L'exclusion de cette valeur particulière de la discrétisation (pb. d'ordre)
  - La discrétisation des seules instances sans cette valeur manquante.

Résultats Weka (suite)

Bilan sur 1R :

- Holte 1993 : a appliqué "1-R" à 60 BDs avec *cross-validation*
  - ➛ Le nombre minimum de classes majoritaires dans une partition = 6
  - ➛ Résultats comparables, voire meilleurs que des méthodes plus sophistiquées (sur certaines BDs).

Privilégier le principe "Le plus simple d'abord" (simplicity first)
➛ Au moins pour commencer (phase d'analyse exploratoire)
➛ Et se faire une idée de la base d'exemples

- Autre cas de *simplicity first* : données temporelles avec "saisonalité" :
  - ◦ Savoir que la consommation d'un produit augmente à telle période (et le reste = linéaire) vs. un modèle très complexe non linéaire.

Modélisation Statistiques

- On utilise (potentiellement) tous les attributs (en même temps)
- **Les hypothèses** : les attributs sont
  - d'importance égale, de distribution *normale*
  - sont statistiquement indépendantes (vis à vis de la *classe*)
    - ➢ Indépendance = les connaissances sur la valeur d'un attribut particulier ne disent rien sur la valeur d'un autre attribut (pour une classe connue).

      Exemples de l'indépendance conditionnelle :

      "ma pelouse mouillée" ← "il pleut" → "la pelouse du voisin mouillée"

      ➢ Si je sais qu'il a plu, savoir que "la pelouse du voisin mouillée" ne m'apprend rien sur l'information "ma pelouse mouillée"

      Ou : **"Tremblement de T." → "alarme" ← "Cambriolage"**
  - et de distribution *Normale*.

Modélisation Statistiques (suite)

- La réalisation de ces hypothèses mènerait à une distribution équitable :
  - ➢ P.ex. sur oui/non dans le cas bi-classes ! Qui a dit "*Deus alea non ludit*" ?(A.E.)

☞ Dans le cas Bayésien général, l'ensemble de ces hypothèses ne sont presque jamais réalisées, mais le schéma donne en pratique de très bons résultats !

- **Rappel de la BD Météo** (pour la suite)

Outlook	**Temp.**	**Humidity**	**Windy**	**Play**
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

TABLE 3: Rappel BD exemple "Météo"

La probabilité conditionnelle de Bayes

- L'hypothèse H et l'évidence E basée sur H :

$$\boxed{\mathrm{Pr[H \mid E]} = \frac{Pr[E|H] \times Pr[H]}{Pr[E]}}$$

- **Pr[H | E]** : l'événement **H** conditionné par l'événement/l'évidence **E**.
  - la proba *a posteriori* de H (proba de H connaissant E)
- **Pr[E | H]** : la *vraisemblance* de E (connaissant H) dont on connaît souvent la loi
- **Pr[H]** : la probabilité *a priori* de H :

  → Pr(H) sans connaître E (quelques soient les attributs)

  ➢ Ex. "météo" : sans rien savoir à propos du jour que l'on veut classer pour **H=yes** : 9/14 (de "yes" quelques soient les attributs)
- Considérations théoriques et la loi de H.

La probabilité conditionnelle de Bayes (suite)

Pour comprendre : un Exemple simple de calcul de Bayes :

Genre (G)	% Possède Carte Crédit	% du genre ayant fait un défaut de paiement
Mas.	60	55
Fem.	40	35

- On fait un tirage aléatoire d'un détenteur de CB qui a fait <u>défaut</u>.

➛ Question : quelle probabilité que ce soit une Femme ?

$P(Genre = Fem|Defaut = oui)$ ↙ Pr[Oui | Fem] : combien "être femme" contribue à "faire défaut"

$$= \frac{P(Defaut=oui|Genre=Fem) \times P(Genre=Fem)}{P(Defaut=oui|Genre=Fem)P(Genre=Fem) + P(Defaut=oui|Genre=Mas.)P(Genre=mas)}$$

$$= \frac{0.35 \times 0.40}{0.35 \times 0.40 + 0.55 \times 0.60} = 0.30$$

➛ Et 70% pour les hommes : $P(Genre = Mas|Defaut = oui) = \frac{0.55 \times 0.60}{0.35 \times 0.40 + 0.55 \times 0.60} = 0.70$

☞ Le dénominateur (commun) sert à normaliser les valeurs.

La probabilité conditionnelle de Bayes (suite)

Retour à la "météo" :

- Soit à calculer Pr[Yes | E] avec E :

Outlook	Temperature	Humidity	Windy	Play
Sunny	Cool	High	True	? ?

- **Hypothèse** (naïve) de Bayes (indépendance) : l'évidence E se décompose ici en ses composantes (attributs) indépendantes p/r à la classe :

$$\Pr[\mathbf{H} \mid \mathrm{E}] = \frac{Pr[E|H] \times Pr[H]}{Pr[E]} \qquad \text{avec } E =< E_1, E_2, ..., E_n >$$
$$= \frac{Pr[E_1|H] \times Pr[E_2|H] \times \cdots \times Pr[E_n|H] \times Pr[\mathbf{H}]}{Pr[E]}$$

  - Avec H : "play=yes", les E_i représentent les 4 autres attributs :

$$\Pr[\mathbf{yes} \mid \mathrm{E}] = \frac{Pr[Outlook|yes] \times Pr[Temp|yes] \times Pr[Hum|yes] \times Pr[Windy|yes] \times Pr[\mathbf{yes}]}{Pr[E]}$$

- Remarque : pour la rigueur, une notation telle que $Pr[Outlook|yes]$ veut dire $Pr[Outlook = une\_val\_de\_outlook|Play = yes]$

Application à l’exemple météo

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

- A new day:

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Likelihood of the two classes

For “yes” = 2/9 × 3/9 × 3/9 × 3/9 × 9/14 = 0.0053

For “no” = 3/5 × 1/5 × 4/5 × 3/5 × 5/14 = 0.0206

Conversion into a probability by normalization:

P(“yes”) = 0.0053 / (0.0053 + 0.0206) = 0.205

P(“no”) = 0.0206 / (0.0053 + 0.0206) = 0.795

FIGURE 4: Probabilités conditionnelles pour les données ”météo”

Application à l'exemple météo (suite)

On ne peut multiplier les probabilités que sous l'hypothèse de l'indépendance.

- La formule de Bayes calcule la conditionnelle via la *jointe* (puis normalisée)
- Pour une nouvelle instance à classer, **l'évidence E** sera :

Outlook	Temperature	Humidity	Windy	Play
Sunny	Cool	High	True	? ?

- Quelle est la probabilité de "yes" pour la nouvelle instance (E ci-dessous) ?

$$\mathbf{Pr[\ yes \mid E\ ] =}$$
$$\mathbf{Pr[Outlook = Sunny \mid yes] \times Pr[Temperature = Cool \mid yes] \times}$$
$$\mathbf{Pr[Humidity = High \mid yes] \times Pr[Windy = True \mid yes]} \times \frac{Pr[Yes]}{Pr[E]}$$
$$= \frac{2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14}{Pr[E]}$$

➛ On fait le même calcul pour $Pr[Play = no|E] = \dfrac{3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14}{Pr[E]}$

N.B. : Ici, $Pr[E] = Pr[E|yes]Pr(yes) + Pr[E|no]Pr(no)$ sert à la normalisation.

Application à l'exemple météo (suite)

- **Récapitulatif de l'exemple "météo"**

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

- A new day:

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Likelihood of the two classes

For "yes" = 2/9 × 3/9 × 3/9 × 3/9 × 9/14 = 0.0053

For "no" = 3/5 × 1/5 × 4/5 × 3/5 × 5/14 = 0.0206

Conversion into a probability by normalization:

P("yes") = 0.0053 / (0.0053 + 0.0206) = 0.205

P("no") = 0.0206 / (0.0053 + 0.0206) = 0.795

FIGURE 5: Probabilités conditionnelles pour les données "météo"

Remarques sur la méthode Bayésienne

- **Avantages de Bayes** : Méthode simple, résultats intéressants,
  - S'améliore si suppression d'attributs redondants
    ➤ suppression des dépendances (cf. l'hypothèse)
  - Rappel : techniques statistiques simples de calcul de la dépendance :

    - Si $P(A \wedge B) = P(A) \times P(B)$ alors **indépendance statistique**
    - Si $P(A \wedge B) > P(A) \times P(B)$ alors **corrélation positive**
    - Si $P(A \wedge B) < P(A) \times P(B)$ alors **corrélation négative**

    N.B. : le test χ^2 ou le coeff. de corr. ($\frac{\sigma_{AB}}{\sigma_A.\sigma_B}$) révèlent le même type de relation.

- **Inconvénients** : Bayes fonctionne mal si les valeurs d'un attribut particulier ne sont pas associées à toutes les valeurs de classe finale (dans la BD).
  - E.g. si outlook=sunny toujours associé avec play=no
    ➤ Pr[yes | sunny]=0 qui multiplie ... = 0
    ➤ probabilité finale = 0 ➤ "sunny" a un droit de veto.
  - Une solution : **Estimateur de Laplace** (ou lissage)
    ➤ ajustement (calcul des probabilités à partir des fréquences).

Remarques sur la méthode Bayésienne (suite)

Ajustement et Correction du droit de veto :

➛ Changer la fréquence d'un attribut de $\frac{0}{x}$ en $\frac{\epsilon}{x'}$ $\rightarrow$ évite le pb. !

- Exemple ("météo") :

  pour "yes", on a *sunny* 2/9 fois, *overcast* 4/9, *rainy* 3/9

  - On ajoute 1 à chaque numérateur puis 3 au dénominateur :

  ➛ Les valeurs ajustées *équivalentes* : *3/12, 5/12 et 4/12.*

- Dans cette technique standard appelée **Estimateur ou Lissage de Laplace** :

  ➛ On considère l'ajout d'une petite constante α (par défaut, $\alpha = 0$)

  ➛ Pour l'exemple : $\dfrac{2+\alpha/3}{9+\alpha}$, $\dfrac{4+\alpha/3}{9+\alpha}$, $\dfrac{3+\alpha/3}{9+\alpha}$

  ➛ On avait pris $\alpha = 3$ ci-dessus.

  ➛ Un α élevé signale l'importance des poids p/r aux nouvelles évidences,

  ➛ Un petit α dénote une moindre influence.

Remarques sur la méthode Bayésienne (suite)

- **Lissage de Laplace général :**

  au lieu de diviser α à 3 parts égales (3 car *outlook* est ternaire) :

  ➛ on peut utiliser $\frac{2+\alpha.p_1}{9+\alpha}$, $\frac{4+\alpha.p_2}{9+\alpha}$, $\frac{3+\alpha.p_3}{9+\alpha}$ (avec $p_1 + p_2 + p_3 = 1$)

  ➛ p_i : probabilité *a priori* de outlook à être =*sunny*, *overcast* ou *rainy* (pour "yes").

  ➛ On obtient la **formule complète de Bayes** avec des probabilités *a priori* pour tout ce qui figure dans les calculs.

- **Inconvénient de Laplace :** ces poids sont difficiles à fixer.

  Dans la pratique, si le nombre d'instances disponibles est suffisant, les probabilité s *a priori* (les p_i et α) ont peu d'influence.

  ➛ on estime les fréquences en utilisant *l'estimateur de Laplace* et en initialisant tous les compteurs (α) à 1 au lieu de 0 (cas naïf).

Valeurs nominales manquantes

- Un des avantages de Bayes : les valeurs manquantes posent peu de problème.

- Exemple : si la valeur de *outlook* est souvent manquante (dans la BD) :

  → En *apprentissage* : l'instance spécifique n'est pas inclue dans le calcul des fréquences

  → En *test* : le calcul omet simplement cet attribut

vraisemblance de "yes" = $3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0238$
vraisemblance de "no" = $1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0343$

Pr(yes) = 0.0238 / (0.0238 + 0. 0343) = 41%
Pr(no) = 0.0343 / (0.0238 + 0.0343) = 59%

  - Problème atténué : *outlook* manque dans les 2 classes
  - Ici, les probabilités sont plus élevées

Valeurs Numériques dans Bayes

- **Remarque** : si la *température* est une mesure continue, la probabilité d'avoir *exactement 66 degré* (ou exactement une valeur comme 63.1415) est nulle.

- On utilise la fonction de densité de probabilité (*PDF*)= la probabilité pour qu'une quantité soit dans une région proche de x (à $\pm\varepsilon/2$ près).

- Le sens réel de la *PDF* : quelque soit $f(x)$ la loi (distribution) de x :

$$\boxed{\mathbf{Pr}[\mathbf{x}-\tfrac{\varepsilon}{\mathbf{2}} \;<\mathbf{x}\;<\;\mathbf{x}+\tfrac{\varepsilon}{\mathbf{2}}] = \int_{\mathbf{x}-\varepsilon/\mathbf{2}}^{\mathbf{x}+\varepsilon/\mathbf{2}} \mathbf{f}(\mathbf{t}).\mathbf{dt} \;\approx\; \varepsilon\;.\;\mathbf{f}(\mathbf{x})}$$

- Plus généralement, on a : $\boxed{Pr[a \le x \le b] = \int_a^b f(t).d(t)}$ $\qquad f(.) =$ la loi

- N.B. : dans Bayes, ε est omis des calculs des *vraisemblances* car il serait annulé lors du calcul des probabilités.

- N.B. : la fonction de "densité de probabilité" pour un événement est liée à la probabilité (mais n'est pas tout à fait la même chose) ➟ P. Ex., même si $\int_{\mathbb{R}} f(t).d(t) = 1$, $\quad f(t)$ peut être > 1.

Valeurs Numériques dans Bayes (suite)

- Hypothèse de Bayes : les numériques ont (toutes) une distribution de probabilités **Normale** (**Gaussienne**)

- La PDF (pour une loi normale $N(\mu, \sigma)$) : $$\boxed{\mathbf{f(x)} = \frac{\mathbf{1}}{\sqrt{\mathbf{2}\pi}\ \sigma}\ \mathbf{e}^{-\frac{(\mathbf{x}-\mu)^{\mathbf{2}}}{\mathbf{2}\sigma^{\mathbf{2}}}}}$$

  Avec la moyenne : $\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$ et σ l'écart type où : $\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{\mathbf{n-1}}$

  ⇨ Le "-1" sur "n" concerne **le degré de liberté** dans les instances

- N.B. : les numériques manquantes n'interviennent pas dans μ et σ.

- **Cas données mixtes** : pour calculer la probabilité de la classe d'une nouvelle instance, on utilisera
  - la PDF pour les numériques et
  - la fréquence pour les nominaux (énumérés, catégoriels, discrets).

Valeurs Numériques dans Bayes (suite)

Exemple météo avec des données mixtes :

	outlook	temperatu	humidity	windy	play
1	outlook	temperatu	humidity	windy	play
2					
3	sunny	85	85	FALSE	no
4	sunny	80	90	TRUE	no
5	overcast	83	86	FALSE	yes
6	rainy	70	96	FALSE	yes
7	rainy	68	80	FALSE	yes
8	rainy	65	70	TRUE	no
9	overcast	64	65	TRUE	yes

.

Application : prévision pour une nouvelle instance (à classer) :

Outlook	Temperature	Humidity	Windy	Play
Sunny	66	90	True	? ?

Pour *play="yes"*, on obtient $\mu = 73$ et $\sigma = 6.2$ pour la Température ../..

Valeurs Numériques dans Bayes (suite)

- La table des calculs ("météo", attributs mixtes) :
  - ➢ PDF pour les numériques et fréquence pour les nominaux.

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3		83	85		86	85	False	6	2	9	5
Overcast	4	0		70	80		96	90	True	3	3		
Rainy	3	2		68	65		80	70					
				...	...		...	...					
Sunny	2/9	3/5	*mean*	73	74.6	*mean*	79.1	86.2	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	*std dev*	6.2	7.9	*std dev*	10.2	9.7	True	3/9	3/5		
Rainy	3/9	2/5											

FIGURE 6: Probabilités pour les données "météo" (numériques et nominaux)

Valeurs Numériques dans Bayes (suite)

outlook	temperature	humidity	windy	**play**
sunny	66	90	true	?

- La PDF pour la "Température" (play="yes" si Temp.=66) :

$$\mathbf{f(Temperature = 66|yes) = \frac{1}{\sqrt{2\pi}\ .\ 6.2}\ e^{-\frac{(66-73)^2}{2\ .\ 6.2^2}} = 0.0340}$$

➛ De manière analogue, la densité de proba de play="yes" si Humidité=90 :

$$\mathbf{f(Humidity = 90|yes) = 0.0221}$$

- Donc, pour la nouvelle instance :

Vraisemblance de "yes" = $2/9 \times 0.0340 \times 0.0221 \times 3/9 \times 9/14 = 0.000036$

De même :

Vraisemblance de "no" = $3/5 \times 0.0291 \times 0.0380 \times 3/5 \times 5/14 = 0.000136$

Valeurs Numériques dans Bayes (suite)

- A partir de ces eux valeurs de f, on a les probabilités :

$$\boxed{\begin{aligned} \Pr(\text{"yes"}) &= \frac{0.000036}{0.000036 + 0.000136} = 20.9\% \\ \Pr(\text{"no"}) &= \frac{0.000136}{0.000036 + 0.000136} = 79.1\% \end{aligned}}$$

  - Valeurs très proches des calculs précédents :
    - ➢ la température 66 est proche de "cool" et l'humidité 90 proche de "high".

Avantages de la Bayésienne Naïve

- Simple avec une sémantique claire pour représenter, apprendre et utiliser des connaissances probabilistes.
- Rivalise avec d'autres classifieurs sur les mêmes BDs.
- Dans Bayes, la classification <u>n'a pas besoin</u> d'estimations <u>précises</u> des probabilités si le max de probabilité est affecté à la bonne classe
- L'hypothèse de "distribution normale" est raisonnable,
- On peut traiter les attributs mixtes.
- Si valeurs numériques manquantes, le calcul de μ et de σ sont uniquement basés sur celles présentes.
- <u>L'indépendance</u> ? traitement préalable et recherche de corrélation.

Inconvénients de la Bayesinenne Naïve

- Il y a des BDs pour lesquelles Bayesienne naïve ne marche pas
  ➛ Pb. si les attributs sont réellement dépendants.
- Les attributs redondants sabotent le processus d'Apprentissage.
  - Exemple : dans "météo", si on a un autre attribut avec les mêmes valeurs que la "Température", cet attribut aura un effet multiplié :
    - Toutes ses probabilités seront mises **au carré**
      → **beaucoup d'influence**.
    - Pire : si *Température* est répétée 10 fois, elle remporte la décision finale.

  ➛ La dépendance entre les attributs réduit le pouvoir de Bayes Naïve
- Amélioration en sélectionnant un sous ensemble intéressant d'attributs

Inconvénients de la Bayesinenne Naïve (suite)

- une autre **restriction** de Bayes : l'hypothèse de **"distribution normale"** pour les données numériques.

  ➢ Beaucoup d'attributs ne sont pas *Normalement* distribués.

  ◦ On peut utiliser d'autres distributions pour les attributs numériques
    ➢ calcul de la vraisemblance, (voir l'addendum)

- Si on suspecte que ce n'est pas une distribution *normale* et si l'on **ne connaît pas la distribution**

  → des méthodes **"kernel density estimation"**

  → plus compliquées mais ne font pas d'hypothèse sur la distribution

- Rappel : on peut toujours discrétiser les valeurs numériques.
- A propos des réseaux Bayesiens.... (voir plus loin)

Ex2 : Bayes Naïve en classification de documents

- Chaque doc représente une instance d'une classe (*Topic*) de documents.
  ➢ Par ex, la classe des (journaux) d'info, de sports, de spam, ...

- Les docs sont caractérisés par les mots qui les constituent.

- **Deux méthodes basiques :**

  1- Une méthode basique et naïve : traiter la présence/absence d'un mot, puis de décider sur ces simples fréquences des mots ;

  2- La Bayésienne Naïve = rapide et efficace.
    ➢ Mais elle ne tient pas compte du nombre d'occurrences d'un mot qui peut être importante pour la classification.

- Pour tenir compte des fréquences des mots, on applique une forme modifiée de Bayes naïve : *multi-nominal Naïve Bayes* (**MNB**).
  ➢ Dans **MNB**, les docs sont considérés comme des *sacs-de-mots* :
  Un doc : un ensemble pouvant contenir plusieurs fois le même mot.

Ex2 : Bayes Naïve en classification de documents (suite)

• MNB s'appuie sur une distribution *multinomiale* pour la classification.

• Pour cette distribution, la probabilité qu'un doc E (composé de n_E mots clefs, voir ci-dessous) soit d'une classe H est :

$$Pr[H|E] \propto Pr(H) \prod_{j=1}^{n_E} Pr[w_j|H]$$

➢ w_j : les mots (clefs du dico) des documents de la catégorie H

➢ $Pr[w_j|H]$ est la proba que w_j figure dans les docs de la catégorie H
= combien w_j contribue à ce que H soit (la vraie) classe de E.

➢ $Pr(H)$: probabilité *a priori* qu'un doc soit de la catégorie H.

• Le but est de trouver la meilleure classe H pour E :
➢ celle la plus vraisemblable = celle avec une probabilité *a posteriori* maximum (MAP = *maximum a posteriori*).

Ex2 : Bayes Naïve en classification de documents (suite)

- $w_1, ...w_j...w_{n_E}$ sont les mots clefs dans E et n_E = nombre de ces mots dans E

  ➛ P.ex, $w_1, ...w_j...w_{n_E}$ pour un doc avec une seule phrase :

  "ECL et EML sont ensemble dans un Bateau"

  sera *<ECL, EML, ensemble, Bateau>* avec n_E=4 (une fois *tokenisé*)

- $\hat{Pr}$ sera une estimation de Pr (à partir d'une base d'apprentissage).
- Utiliser *log* pour ne pas perdre de la précision dans les multiplications.
- On choisira le maximum de $log(\hat{Pr}[H|E]) \propto log(\hat{Pr}(H)) + \sum_{j=1}^{n_E} log(\hat{Pr}[w_j|H])$

  ➛ Cette somme indique "combien" le doc. E peut être de la classe H.

Comme dans Bayes :

- $log(\hat{Pr}[w_j|H])$ donne la valeur de l'indicateur w_j pour désigner la classe H
- et $log(\hat{Pr}(H))$ indique la fréquence relative de la classe H :

  ➛ plus la classe est fréquente, plus elle a la chance d'être choisie. ../..

Ex2 : Bayes Naïve en classification de documents (suite)

Estimation de $\hat{P}r(H)$ **et** $\hat{P}r[w|H]$:

• On utilisera MLE (maximum de vraisemblance) est ici simplement la fréquence relative dans la base d'apprentissage :

• $\hat{P}r(H) = \frac{N_H}{N} = \frac{\text{nombre de documents dans la classe } H}{\text{le nombre total des documents du corpus}}$

• L'estimation pour $\hat{P}r[w_j|H]$ est la fréquence relative du mot w_j dans les documents de la classe H.

➛ $\hat{P}r[w|H] = \frac{T_{Hw}}{\sum_{w' \in V} T_{Hw'}}$.

➛ T_{Hw} est le total de toutes les occurrences du mot w dans la base d'apprentissage (dans le vocabulaire des *mots clefs* V).

Ex2 : Bayes Naïve en classification de documents (suite)

- Rappel : on est indép de la position des mots dans les docs (pas d'ordre)
  - ➛ On ne calcule donc pas différentes estimations pour différentes positions
  - ➛ Si un mot apparaît 2 fois, alors $\hat{Pr}[w|H]$ sera identique pour les 2 occ.
  - ➛ P. ex. les documents $\{Ecully\ Dardilly\ Ecully\}$ et $\{Ecully\ Ecully\ Dardilly\}$

    sont considérés identiques et les mots répétés ont le même poids.

- Pour le problème du *veto de zéro* : *Lissage de Laplace*

$$\hat{Pr}[w|H] = \frac{T_{Hw} + 1}{\sum_{w' \in V}(T_{Hw'} + 1)} = \frac{T_{Hw} + 1}{(\sum_{w' \in V} T_{Hw'}) + B}$$

  où B est la constante du lissage de Laplace = ici <u>la taille du vocabulaire</u>.

  - ➛ L'ajout de 1 signifie une *a priori* uniforme (comme si chaque mot apparaissait une seule fois dans chaque classe).

  ➤ Voir Bayes pour la généralisation du lissage.

Ex2 : Bayes Naïve en classification de documents (suite)

Un exemple : soit les documents

ID	Ensemble	les mots du document	classe H="Chine" ?
1	Apprentissage	Chinois Pékin Chinois	oui
2	Apprentissage	Chinois Chinois Shanghai	oui
3	Apprentissage	Chinois Macao	oui
4	Apprentissage	Tokyo Japon Chinois	no
5	Test	Chinois Chinois Chinois Tokyo Japon	?

- On a également, pour $H = \text{"} Chine \text{"}$, $\hat{Pr}(H) = 0,75$ et $\hat{Pr}(\bar{H}) = 0,25$
- Calcul des probabilités conditionnelles :

$$\hat{Pr}(\text{"} Chinois \text{"} | H) = \frac{5+1}{8+6} = \frac{3}{7}$$

$$\hat{Pr}(\text{"} Tokyo \text{"} | H) = \hat{Pr}(\text{"} Japon \text{"} | H) = \frac{0+1}{8+6} = \frac{1}{14}$$

$$\hat{Pr}(\text{"} Chinois \text{"} | \bar{H}) = \frac{1+1}{3+6} = \frac{2}{9}$$

$$\hat{Pr}(\text{"} Tokyo \text{"} | \bar{H}) = \hat{Pr}(\text{"} Japon \text{"} | \bar{H}) = \frac{1+1}{3+6} = \frac{2}{9}$$

➢ On utilise les dénominateurs $(8+6)$ et $(3+6)$ car la longueur des textes de la classe $H = \text{"} Chine \text{"} = 8$ et celle de $\overline{H} = 3$ et

➢ La constante B du lissage de Laplace $= 6$ (= nb. termes dans le vocab.)

Ex2 : Bayes Naïve en classification de documents (suite)

- On aura $\hat{Pr}(\text{"}Chine\text{"}|d_5) \propto 3/4.(3/7)^3.1/14.1/14 \approx 0,0003$
  Et $\hat{Pr}(\overline{\text{"}Chine\text{"}}|d_5) \propto 1/4.(2/9)^3.2/9.2/9 \approx 0,0001$

- NB : pour passer de $\approx$ à $=$, on normalise en divisant ces valeurs par leur somme :

$$\hat{Pr}(\text{"}Chine\text{"}|d_5) = \frac{0,0003}{0,0003+0,0001} = 75\%$$

Et

$$\hat{Pr}(\overline{\text{"}Chine\text{"}}|d_5) = \frac{0,0001}{0,0003+0,0001} = 25\%$$

- **Conclusion** : le document de test (d_5) appartient à la classe $H = \text{"}Chine\text{"}$.

  ➢ Ici, les 3 occ de l'indicateur positif ("Chinois") dans d_5 prennent le dessus sur les 2 occ des indicateurs négatifs ("Japon" et "Tokyo").

☞ L'hypothèse de l'indépendance des mots dans la phrase !
  ➢ Une réponse : utilisation de Bi / Digrammes, Trigrammes, ...

Addendum : Probabilité et Densité

- Pourquoi : $\boxed{Pr[a \leq x \leq b] = \int_a^b f(t).d(t)}$?

$\Rightarrow$ soit X associé à la fonction f_X : $X \rightsquigarrow f_X$

- On sait par ailleurs (voir plus haut) : $\Pr[X \geq a] = \int_{\infty}^{a} f_X(t)d(t)$

$\Rightarrow$ $\Pr[a \leq X \leq b] = Pr[X \leq b] - Pr[X \leq a]$ pour le segment ab, $a < b$

$\Rightarrow$ $$= \int_{-\infty}^{b} f_X(t)d(t) - \int_{-\infty}^{a} f_X(t)d(t)$$

on développe le 1er terme

$\Rightarrow$ $$= \int_{-\infty}^{a} f_X(t)d(t) + \int_{a}^{b} f_X(t)d(t) - \int_{-\infty}^{a} f_X(t)d(t)$$

$\Rightarrow$ $$= \int_{a}^{b} f_X(t)d(t)$$

Addendum : Probabilité et Densité (suite)

Remarques et rappels :

On a : $Pr[A \cap B] = Pr[A|B].Pr[B] = Pr[B|A].Pr[A] \Rightarrow Pr[A|B] = \frac{Pr[A \cap B]}{Pr[B]}$

$$\boxed{Pr[A|B] = \frac{Pr[B|A].Pr[A]}{Pr[B]}}$$

et, en cas d'indépendance des B_i :

$$Pr[B|A] = Pr[B_1|A] \times Pr[B_2|A] \ldots Pr[B_n|A] \text{ (Bayes)}$$

- Pour les numériques : $\int_{x-\varepsilon/2}^{x+\varepsilon/2} f(t).dt \approx \varepsilon \, . \, f(x)$

  Par exemple, pour $x = 21$: $\int_{21-\varepsilon/2}^{21+\varepsilon/2} f(t).dt \approx \varepsilon \, . \, f(21)$

  Et, pour le calcul de la part température dans l'ex. météo :

  $Pr[Temp = 21|play = yes] \sim f(21)$. avec f : la fonction de densité.

Addendum : Probabilité et Densité (suite)

De la PDF à la proba pour x centrée

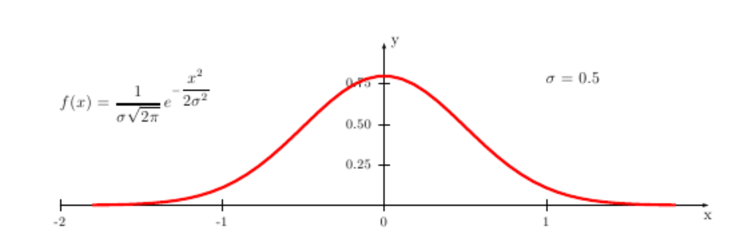


FIGURE 7: La fonction de densité pour une variable centrée ($\mu = 0$) avec $\sigma = 0.5$

- Rappel : $\int_{-\infty}^{+\infty} f(t)d(t)$ d'une manière générale, pour une probabilité.
- Cette intégrale (CDF) représente une probabilité :
  - ➛ Pourquoi l'intégrale ci-dessus (courbe) vaut 1 ? ../..

Addendum : Probabilité et Densité (suite)

- Soit $$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\ e^{-\dfrac{x^2}{2\sigma^2}} = \frac{1}{\sigma\sqrt{2\pi}}\ e^{-\left(\dfrac{x}{\sigma\sqrt{2}}\right)^2},$$
- Posons (changement de variable) $y = \dfrac{x}{\sigma\sqrt{2}}$ ➛ $dy = \dfrac{dx}{\sigma\sqrt{2}}$ ➛ $dx = \sigma\sqrt{2}.dy$

  ➛ On aura :

$$\int_{-\infty}^{\infty} f(x)\ dx = \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}}\ e^{-\left(\dfrac{x}{\sigma\sqrt{2}}\right)^2}.dx = \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}}\ e^{-y^2}.\sigma\sqrt{2}.dy$$

$$\text{car } dx = \sigma\sqrt{2}.dy$$

- Or, on sait que $\displaystyle\int_0^{\infty} e^{-x^2} dx = \frac{\sqrt{\pi}}{2}$ (appelé Intégrale de la Gaussienne)
  - Et donc : $\displaystyle\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$
- Donc, $\displaystyle\int_{-\infty}^{\infty} f(x)\ dx = \sigma\sqrt{2}\ \frac{1}{\sigma\sqrt{2\pi}}\ \int_{-\infty}^{\infty} e^{-y^2}.dy = \frac{1}{\sigma\sqrt{2\pi}}\ \sigma\sqrt{2}\ \sqrt{\pi} = 1$ ■

Addendum : : exemples de calculs Bayesiens

- La formule de Bayes dans un contexte (background) c :

$$\boxed{Pr[H|E,c] = \frac{Pr[E|H,c] \times Pr[H|c]}{Pr[E|c]}}$$

Voyons deux exemples ./..

Addendum : Exemple 1 (station services)

• Dans une station-service, on connaît les différentes probabilités d'avoir de 0..k clients dans un délai de 15 minutes (loi binomiale) :

x_i	0	1	2	3	4
$Pr[X = x_i]$	0.1	0.2	0.4	0.2	0.1

• On sait aussi : la probabilité qu'un client entré demande du Diesel = 0.4.

1- **Quelle est** la loi conditionnelle (la vraisemblance) de Y pour $X = x_i$?

- ◦ i.e. la probabilité pour que k demandes de diesel sachant x_i entrés (en 15 min) ?

2- **Quelle est** la loi du couple (X,Y), celle de Y,

3- Combien d'entrées (loi de X) sachant k demandes de Diesel ?... ../..

Addendum : Exemple 1 (station services) (suite)

Solution :
- On fixe $X = x_i$ dont chacun (entré) a une proba de 0.4 de demander du diesel.
- Le nombre de ces personnes est donné par la variable aléatoire binomiale $\beta(x_i, 0.4)$.

1 - On a la vraisemblance de k demandes de diesel si x_i clients sont entrés :

$$Pr[Y = k \ demandes \ de \ \ Diesel \mid X = x_i] = C_{x_i}^{k} \ \ 0.4^{k} \ \ 0.6^{x_i - k} \quad si \ k \leq x_i \quad (et = 0 \ sinon)$$

➤ Par exemple, la proba d'une demande de diesel <u>si</u> un client est entré $= 0.4$
➤ Et la proba d'une demande de diesel <u>si</u> deux clients sont entrés $= 0.48$

2 - La loi du couple (X, Y) sera :

$$Pr(Y = k, X = x_i) = Pr(X = x_i, Y = k) = Pr[Y = k|X = x_i] \times Pr(X = x_i)$$

3- Et enfin :

$$Pr(X = x_i \mid Y = k) = \frac{Pr(X = x_i, Y = k)}{Pr(Y = k)} = \frac{Pr[Y = k|X = x_i] \times Pr(X = x_i)}{Pr(Y = k)}$$

Addendum : Exemple 2 (Robert)

- Robert veut ouvrir une boutique franchisée de trottinettes (enseigne *Teufteuf*)
- Son affaire ne sera viable que s'il a 25% de saturation de marché
- Il fait une étude locale sur 20 clients : 5 ont bien l'intention d'achat (25%) :
  - Mais il doute !
  - Il demande des chiffres à la maison mère ...
- Les données de la maison mère :

Taux de Saturation	% des sociétés
0,10	0,05
0,15	0,05
0,20	0,20
0,25	0,20
0,30	0,40
0,35	0,10
	Total=100

- La question de Robert : Quelle est sa chance d'être au moins dans les 20% qui saturent le marché à 25% (étant donné le sondage !) ?
  - quelle chance d'être dans les 70% des enseignes qui ont un taux $\geq 25\%$?

Addendum : Exemple 2 (Robert) (suite)

- On utilise l'inférence Bayésienne : $P(H|E) = \frac{P(E|H) \times P(H)}{P(E)}$
- De la théorie de la *distribution binomiale* :

  ➤ Si la probabilité d'un événement dans une seule tentative est p, alors la probabilité pour que k de ces événements arrivent dans n tentative est :

$$P(k) = \frac{n!}{k!(n-k)!} * p^k * (1-p)^{(n-k)}$$

  ➤ Par exemple : la vraisemblance pour que 5 des 20 personnes (25%) soient *clients*, pourvue que Robert soit dans la catégorie des 20% d'enseignes saturant 25% du marché est :

$$P(k=5|p_{0.20}) = \frac{20!}{5!(20-5)!} * (0.25)^5 * (0.75)^{15} = 0.20233$$

  ➤ N.B. : 20 exemples est peu.

  ➛ Plus il y en a, plus les probas a priori auront du poids.

- Le tableau suivant résume les calculs (p_i = la colonne gauche) .../ ...

Addendum : Exemple 2 (Robert) (suite)

Événement (saturation) $\mathbf{p}_i$	Probabilité a priori $\mathbf{P_0(p_i)}$	vraissem- -blance $\mathbf{P}(k=5\|p_i)$	Probabilité jointe $\mathbf{P}(k=5\|p_i) * P_0(p_i)$	Proba a posteriori $\mathbf{P_1(p_i)} = P(p_i\|k=5)$ $\frac{Pr[k=5\|p_i] \times P_0[p_i]}{P[k=5]}$
0,10	0,05	0.032	0.0016	0.0100
0,15	0,05	0.103	0.005	0.0031
0,20	0,20	0.174	0.035	0.2100
0,25	0,20	0.202	0.04	$\boxed{0.2430}$
0,30	0,40	0.179	0.0715	$\boxed{0.4300}$
0,35	0,10	0.127	0.0127	$\boxed{0.0760}$
Totaux	1.00	0.8177	0.1664=**P(k=5)**	0.999

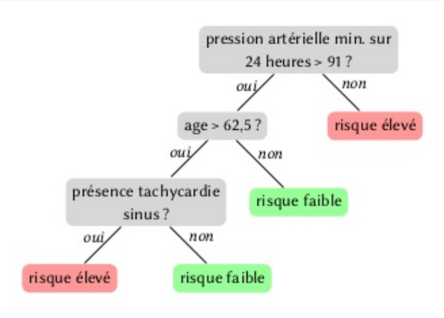
∘ Presque 75% (somme des $\boxed{\dots}$) de chance pour que Robert soit dans >= 25%

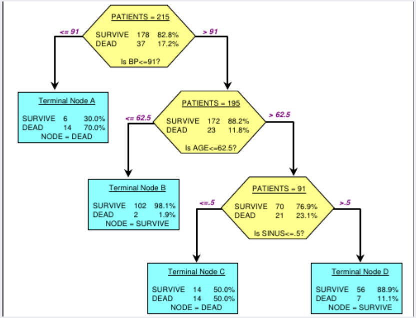
• Ce calcul permet de tenir compte à la fois des données de la maison mère (*a priori*)·et du sondage local.

➛ La maison mère laissait une proba de 70%, le sondage augmente cette proba.

Introduction aux Arbres de Décision

- Un exemple d'AD dans le domaine des maladies cardiovasculaire :
  - ➢ Prédiction d'une 2nde attaque après une 1e et la mort dans les 30 jours :





  - ➢ BP (Blood Pressure : la Tension artérielle), Sinus (de la courbe tachycardie) et Age sont des **attributs** du dossier médical.
  - ➢ Les rectangles verts : la **décision**.

Introduction aux Arbres de Décision (suite)

- Stratégie utilisée : *Diviser et Régner* (Divide & Conquer)
- **Le principe de la construction d'un arbre de décision** :

1. Sélectionner un attribut A (sauf l'attribut *classe*),
   ➛ Soit V_i les différentes valeur de A
2. Le placer à la racine et créer une branche pour chaque V_i
   ➛ un sous ensemble de la BD par valeur d'attribut (une branche)
3. Répéter le processus récursivement pour chaque branche en considérant les instances qui atteignent cette branche
4. Sur tout noeud :
   - si toutes les instances de ce noeud ont la même classe, alors arrêter le développement de ce noeud.
   - si tous les attributs ont été utilisés (depuis la racine jusqu'à ce noeud), alors arrêter le développement de cette partie de ce noeud.
   - Sinon, trouver un autre attribut et diviser ce noeud en branches.

Critères de choix d'un attribut

Rappel de la B.D. "Météo" :

Outlook	Tmp.	Hum.	Windy	Play
Sunny	Hot	High	False	*No*
Sunny	Hot	High	True	*No*
Overcast	Hot	High	False	*Yes*
Rainy	Mild	High	False	*Yes*
Rainy	Cool	Normal	False	*Yes*
Rainy	Cool	Normal	True	*No*
Overcast	Cool	Normal	True	*Yes*
Sunny	Mild	High	False	*No*
Sunny	Cool	Normal	False	*Yes*
Rainy	Mild	Normal	False	*Yes*
Sunny	Mild	Normal	True	*Yes*
Overcast	Mild	High	True	*Yes*
Overcast	Hot	Normal	False	*Yes*
Rainy	Mild	High	True	*No*

Critères de choix d'un attribut (suite)

Exemple "météo" avec 4 attributs ➛ 4 possibilités de racine

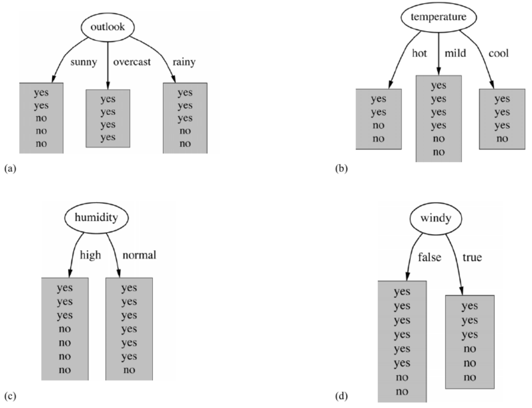


FIGURE 8: Les racines possibles de l'arbre de décision pour la BD. "Météo"

Critères de choix d'un attribut (suite)

Hypothèse : pour préciser les classes de l'ensemble E d'instances rattaché à un noeud N, on a besoin d'un mot de L bits (L est locale et non cumulative).

➛ Donner la classe de chaque instance, une par une, nécessite un mot long !

- On choisit un attribut A (avec des valeurs $V_1..V_p$) qui scinde E en $E_1..E_p$.
- Soit $E_i \subseteq E$ les instances de la i^{eme} branche (valeur V_i de A = noeud N_i),
  - mot de L_i bits pour donner les classes de tous les éléments de E_i,
  - on pose $L_A = \sum_i L_i$, on a $L_A \leq L$ (L = nbr bits avant de scinder sur A)

  ➛ L_i reflète l'hétérogénéité / homogénéité des instances, contient la part d'incertitude dans la prédiction de la classe d'une instance.

  ➛ L_i est <u>minimale</u> si toutes les instances sont d'une même classe.

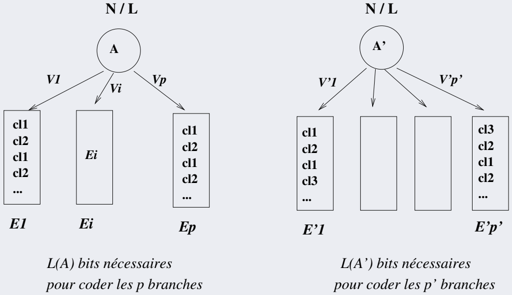
  ➛ L_i est <u>maximale</u> si chaque instance est d'une classe différente.

- On fait le même raisonnement avec un autre attribut A' sur le même noeud N
- **Choisir l'attribut** A si $L_A < L_{A'}$, sinon choisir A' :

  ➛ sur le noeud N, si A permet de définir la classe des instances avec moins de bits (hétérogénéité réduite), alors on préfère A à A'.

Critères de choix d'un attribut (suite)

Représentation graphique :



> Ici, on a une distribution de probas. sur les instances d'un noeud ;
> on veut calculer **l'information nécessaire** pour prédire la classe d'une (nouvelle) instance.

Les longueurs L et L_A non cumulatives ➛ optimum local vs. optimum global ?

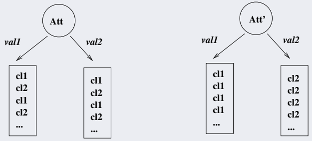
Critères de choix d'un attribut (suite)

- **But** : déterminer l'attribut A tel que $\sum_i L_i$ soit minimale.

  ➛ Celui qui donne **l'arbre le + petit** (en taille/hauteur), réduit l'erreur, ...

- Une bonne heuristique : la notion de **Pureté** (uniformité) des classes des noeuds
  - si toutes les instances du noeud sont d'une même classe alors on aura une pureté maximale (dispersion minimale),
  - si elles sont toutes de classes différentes ➛ pureté minimale

- Une *pureté* (uniformité) plus élevée diminue l'incertitude de la classification
  - L'attribut Att disperse davantage les instances (augmente l'incertitude d'affecter les classes) → incertitude maximum
  - Par contre, avec Att', les classes des instances sont peu dispersées → incertitude minimum



Répartition "non pure" — **incertitude maximum**

Répartition "pure" — **incertitude minimum**

✌ Cas idéal : une seule classe dans chaque branche (cf. noeud Att')

Critères de choix d'un attribut (suite)

Comment faire ?

• Soit la longueur L pour donner la classe de chaque instance d'un noeud **avant** de scinder sur un attribut A (vs. A')

• Observer comment les valeurs de A et de A' dispersent les instances dans des paquets de pureté diverses (donnent les L_i)

• Le gain (du choix d'un attribut A) $= L - \sum_i L_i$
➛ *l'incertitude avant division en paquets - l'incertitude après division*

➛ **maximiser ce gain = minimiser l'incertitude après division**
= augmenter la pureté

• Représenter une info. → Représenter l'info + une incertitude → nbr. de bits néc.

• Supposons disposer de la fonction **info()** qui nous donne L_i (et donc $\sum_i L_i$)
➛ **info(.)** tient compte du nbr. de chaque classe présente dans chaque branche.

Critères de choix d'un attribut (suite)

Utilisation de la fonction info() dans l'exemple météo :

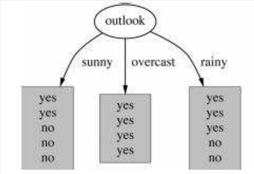
- Pour la fig. ci-dessous, le nombre de "yes"/"no" des noeuds : *[2,3], [4,0] et [3,2]*

➤ Le nbr de bits (valeur de l'info) de ces noeuds (v. détails + loin) :

info([2,3]) = 0.971 bits
info([4,0]) = 0.0 bits incertitude nulle !
info([3,2]) = 0.971 bits

☞ Pour le 1e & 3e noeud, l'incertitude est presque maximale ($\in$ [0..1], pour un cas bi-classes)

- On calcule la moyenne de ces valeurs en tenant compte du nombre d'instances de chaque branche : **5**, **4** et **5**



*info([2,3], [4,0], [3,2]) = (5/14) * 0.971 + (4/14) * 0 + (5/14) * 0.971 = 0.693 bits.*

➤ Cette moyenne (0.693 bits) = la quantité **moyenne** d'information nécessaire pour spécifier la classe d'une nouvelle instance pour un arbre de décision avec l'attribut "outlook" à la racine.

Critères de choix d'un attribut (suite)

Rappel de la BD "Météo"

Outlook	Temp.	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

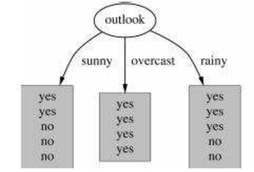
➛ on a 9 noeuds "yes" et 5 "no"

Rappel : le gain de A est la différence entre *avant* et *après* la division (pour A)

Critères de choix d'un attribut (suite)

- A la racine (avant tout choix d'attribut), on a 9 noeuds "yes" et 5 "no"

$$\boxed{info([9,\ 5]) = 0.940\ bits.}$$



◦ On a eu : info("Outlook") = 0.693
→ L'arbre de "outlook" est responsable d'un gain <u>(avant - après)</u> d'information de 0.247 bits car :

$$\boxed{\text{gain(outlook) = info([9,5]) - info([2,3], [4,0], [3,2]) = 0.940 - 0.693 = 0.247 bits.}}$$

- Si on scinde suivant "Outlook", <u>on diminue</u> l'incertitude de 0.247 bits

Interprétation de ce gain (quantifiée) : c'est la quantité d'information apportée par la création d'une branche sur l'attribut "outlook" (à la racine).

➛ C'est la <u>contribution</u> de *outlook* pour départager les 14 instances de la BD

➛ C'est la longueur en bits gagnée pour énoncer les classes des instances ../..

Critères de choix d'un attribut (suite)

- **La méthode du choix du meilleur attribut** : faire les calculs de gain sur chaque attribut et choisir celui qui maximise le gain.

  ☞ Pour avoir le *max(avant - après)*, on prend *min(après)*

- Le calcul de ce gain pour les 4 attributs possibles :

gain(outlook) = 0.247
gain(Température) = 0.029
gain(Humidité) = 0.152
gain(Windy) = 0.048

  Le maximum de gain pour scinder l'arbre à la racine : "outlook"

  ➛ le choix pour lequel la branche fille est la plus "pure" possible.

- On continue récursivement sur chaque noeud crée.

Choix des autres attributs

- Les possibilités de branches sachant "Outlook= sunny" :

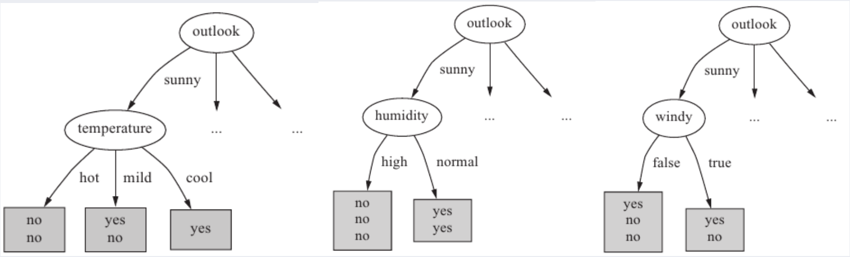


FIGURE 9: Examen des 3 attributs restants sur la branche "sunny" (avec 5 instances)

Choix des autres attributs (suite)

Évidence : une nouvelle division sur "outlook" donnera un gain nul !

- Les gains des 3 attributs restants (pour le 2e niveau de l'arbre) :

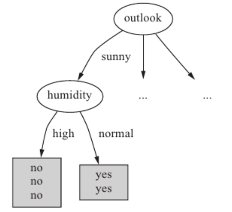
gain(Temperature) = 0.571
gain(Humidity) = 0.971
gain(Windy) = 0.020

→ On choisit "Humidity" pour scinder la branche "sunny" ...

→ Ensuite : plus rien à diviser : terminé pour cette branche !

- Après la dispersion sur "Humidity", il n'y a plus d'incertitude :
➛ L'utilisation d'un autre attribut est sans effet !

- Après la dispersion sur "Humidity", il n'y a plus d'incertitude :
➛ On affecte les classes à ces deux feuilles



Choix des autres attributs (suite)

- L'arbre de décision final :

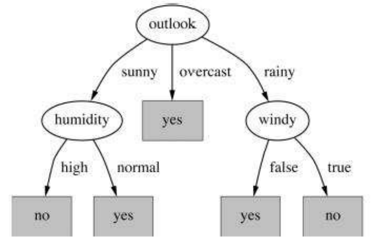


FIGURE 10: l'arbre de Décision final pour "Météo"

☞ Dans un AD, la profondeur (longueur) d'une branche est proportionnelle à la réduction de l'hétérogénéité des instances qui "descendent" le long de cette branche :

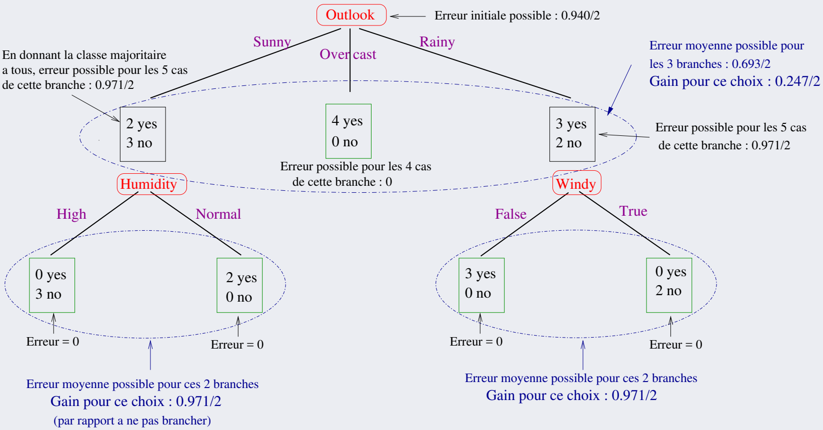
➢ Les branches les plus courtes relative aux autres branches d'un AD (cf. celle du milieu ci-dessus) contiennent les noeuds les plus homogènes

➢ A l'inverse, les branches longues ont "tenté", par des tests successifs sur les attributs, de réduire cette hétérogénéité.

Choix des autres attributs (suite)

Entropie : une autre interprétation des choses

Avant toute chose : 9 yes et 5 no
En donnant la classe majoritaire a toutes les instances
on se trompera de 0.940 / 2 (< 1/2 car on n'a que 5/14 erreurs)



Le calcul de l'information

Propriétés requises de la fonction "info(..)" :

- Elle devrait avoir les propriétés suivantes (cf. BD. "Golf/Météo") :
  - Si tout est de la même classe (e.g. tout est "yes" et le nombre "no" = 0), l'incertitude est **minimale** = 0 (l'information nécessaire =0) ;
  - Si le nombre "yes" = nombre de "no" ➛ l'incertitude sera **maximale**.
  - Plus généralement (pour N classes) : incertitude maximum si toutes les classes sont présentes de manière égale
- Être applicable aux situations multi-classes (plus de 2 classes)
- Être Calculable par étapes (sans ordre entre les étapes) :
  ➛ i.e. on doit avoir : *info([2,3,4]) = info([3,2,4])*
  - *info([2,3,4]) = info([2,7]) + (7/9) × info([3,4])* → ici [3,4] ensemble
  - *= info([3,2,4]) = info([3,6]) + (6/9) × info([2,4])* → ici [2,4] ensemble

Le calcul de l'information (suite)

Une fonction satisfait ces propriétés : l'*entropie* (la *valeur d'information*)

$$\boxed{entropie(p_1, p_2, ..., p_n) = -p_1 \times log\ p_1 \quad - p_2 \times log\ p_2 \quad ... \quad - p_n \times log\ p_n}$$

- Les arguments p_i sont des fractions (fréquences) et $\sum_i p_i = 1$.
- Pourquoi '-' : "log" des fractions p_i est négatif, l'entropie est positive
- 'log' en base 2 $\rightarrow$ résultat en nombre de bits

> Étant donné une distribution de probabilités (ici fréquences), la quantité d'**information nécessaire pour prédire un évènement** est *l'entropie de la distribution*

- Exemple de calcul (un cas à 3 classes) :

  info([2,3,4]) = entropie(2/9, 3/9, 4/9) =1.53

➢ L'entropie donne cette information nécessaire <u>en nombre de bits</u>

Le calcul de l'information (suite)

L'**entropie** mesure *l'incertitude* et permet de quantifier le caractère aléatoire d'une distribution de probabilités.

- Elle permet de mesurer <u>l'incertitude relative à l'appartenance</u> des objets aux différentes classes.

- Elle permet, empiriquement, de donner une idée de l'erreur (calculée par la fonction **info**) si on devait donner la classe majoritaire (voire une classe aléatoire) à toutes les instances d'un noeud.

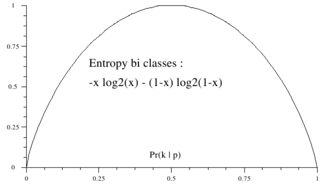
- Lorsque tous les objets appartiennent à une seule classe, l'incertitude est <u>nulle</u>.

Le calcul de l'information (suite)

- Étant données une position p dans l'arbre et c classes que l'on cherche à prédire, l'entropie associée à p est donnée par

$$H(p) = -\sum_{k=1}^{c} Pr(k|p) log_2(Pr(k|p))$$

➛ Noter $H(0) = H(1) = 0$, $H(0.5) = 1$



La fonction entropie et la courbe d'entropie pour 2 classes C_0 et C_1 (log base2 , axe x=Pr(k|p))

Le calcul de l'information (suite)

- La propriété de la décision multi-niveaux (soit $p+q+r=1$) :

$$\boxed{entropie(p,q,r) = entropie(p,q+r) + (q+r) \times entropie(\frac{q}{q+r}, \frac{r}{q+r})}$$

➢ p, q, r sont des proportions issues d'une division en branches.

- Une simplification technique :

Exemple pour une division à 3 branches :

$$\begin{aligned}
\textbf{info([2,3,4])} &= \textbf{entropie(2/9, 3/9, 4/9)} \\
&= \textbf{-2/9 . log 2/9 - 3/9 log 3/9 - 4/9 log 4/9} \\
&= \textbf{[-2 log2 - 3 log3 - 4 log4 + 9 log9] / 9 .} \\
&= \textbf{1.53} \qquad \textit{log en base 2}
\end{aligned}$$

N.B. : pour simplifier les calculs, ne pas simplifier *log9* !

Addendum : Remarques sur l'entropie

$$\textbf{info([a,b])} = \textbf{entropie}\left(\frac{a}{a+b}, \frac{b}{a+b}\right)$$
$$= -\ \frac{a}{a+b} \;.\; \textbf{log}\ \frac{a}{a+b} \;-\; \frac{b}{a+b} \;.\; \textbf{log}\ \frac{b}{a+b}$$
$$= \textbf{[-a log(a) - b log(b) + (a+b) * log(a+b)] / (a+b)}$$

.

- Un exemple de calcul

⇒ Pour l'exemple "météo", on avait *info([9,5])=0.940* :

info([9,5]) = entropy(9/14, 5/14)

= (-9*log9 - 5*log5+14*log14)/14

= 0.94

Addendum : Remarques sur l'entropie (suite)

Résumé des propriétés et exemples :

- **entropy([a,b])** = entropy([b,a])
- **entropy([0, x])** = 0 (0 multipliera un log0) ➛ une seule classe
- **entropy([x,x])** = 1 ➛ instances équi-réparties

- **info([a,b,c])** = entropy(a/S, b/S, c/S) avec S = a+b+c
  = -a/S * log a/S - .... - c/S * log c/S (simplification ci-dessus)

- **info([a,b], [c,d])** = $\dfrac{a+b}{a+b+c+d}$ * info([a,b] + $\dfrac{c+d}{a+b+c+d}$ * info([c,d])

- **info([a,b,c])** = info([a, (b+c)] + (b+c)/(a+b+c) * info([b,c])
  - Ex. : **info([2,3,4])** = info([2,7]) + (7/9) * info([3,4]) (données "météo")
  - Même chose que : $entropie(p, q, r)$

$$= entropie(p, q+r) + (q+r) \times entropie(\frac{q}{q+r}, \frac{r}{q+r})$$

⇒ Exemple (données "Météo", voir p. svte pour [4,0]) :

info([2,3], [4,0], [3,2]) = (5/14) * info([2,3]) + (4/14) * info([4,0]) + (5/14) * info([3,2])

Addendum : Remarques sur l'entropie (suite)

Exemples (sur la BD Météo) :

Calcul des valeurs utilisées dans la construction de l'Arbre de Décision de "Golf/Météo" :

- Outlook = "Sunny" :
  info([2, 3]) = entropy(2/5, 3/5) = -2/5 log(2/5) - 3/5 log(3/5) = 0.971 bits
- Outlook = "Overcast" :
  info([4, 0]) = entropy(4/4, 0/4) = entropy(1, 0) = -1 log(1) - 0 log(0)
  = 0 bits (*log(0)* non défini)
- Outlook = "Rainy" :
  info([3, 2]) = info[2, 3] = 0.971 bits (comme pour "Sunny")
- L'information attendue pour l'attribut "Outlook" (sachant les 3 valeurs ci-dessus) :
  info([3, 2], [4, 0], [2, 3]) = $(5/14) \times 0.971 \times (4/14) \times 0 + (5/14) \times 0.971$
  = 0.693 bits

Les attributs dispersants

- Problème : les attributs très diviseurs (très "branchants"/dispersants)
  - Avec un grand nombre de valeurs (e.g. Ident, Date, Heure, ...)

    → Les sous ensembles crées seront pourtant "pures"
  - Cas extrême : une valeur différente pour chaque instance (e.g. ID code)
- Sur ce type d'attributs
  - Le calcul du gain est biaisé et **favorise l'attribut branchant**
  - Favorise le sur-apprentissage par *sur-adaptation* (*overfitting*)
  - Le calcul d'un véritable gain d'information fiable devient compliqué.

Les attributs dispersants (suite)

- Exemple "Météo" avec un ID code "dispersant"

ID code	Outlook	Temp.	Humidity	Windy	Play
a	Sunny	Hot	High	False	No
b	Sunny	Hot	High	True	No
b	Overcast	Hot	High	False	Yes
d	Rainy	Mild	High	False	Yes
e	Rainy	Cool	Normal	False	Yes
f	Rainy	Cool	Normal	True	No
g	Overcast	Cool	Normal	True	Yes
h	Sunny	Mild	High	False	No
i	Sunny	Cool	Normal	False	Yes
j	Rainy	Mild	Normal	False	Yes
k	Sunny	Mild	Normal	True	Yes
l	Overcast	Mild	High	True	Yes
m	Overcast	Hot	Normal	False	Yes
n	Rainy	Mild	High	True	No

TABLE 4: BD exemple "météo"

Les attributs dispersants (suite)

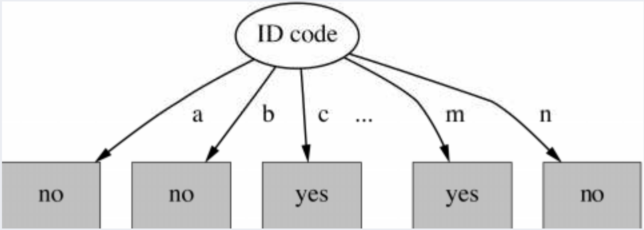


FIGURE 11: Branchement sur ID code de l'Arbre de Décision pour "météo"

- L'entropie de la division :
  info([0,1]) + info([0,1]) + info([1,0]) + ... + info([0,1]) = 0
  - Le gain est maximum pour l'attribut ID code : 0.940 - 0 = **0.940**
    → (0.940 = l'information à la racine avec 9 "yes" / 5 "no")

Les attributs dispersants (suite)

- **Remarques**
  - Le code identifie l'instance et détermine sa classe **sans ambigüité**
    ➛ (= pure).
  - Il donne le meilleur gain
    ➛ l'ID sera choisi inévitablement comme meilleur attribut de division !
  - Mais ce branchement ne permet pas de prédire la classe de nouvelles instances
  - Ne donne rien sur la structure de la connaissance et de la décision.
  - Une solution (à ce gain biaisé) :
    la mesure par le **"ratio de gain"** réduit son biais

Les attributs dispersants (suite)

Mesure Ratio de Gain :

- Basé sur le **nbr et la taille** des divisions faites par un attribut.
  - Correction du gain par l'**information intrinsèque** de la division
  - Sans considérer aucune information sur la classe,
    Mais seulement le nombre d'instances dans chaque branche.

- L'**information intrinsèque** (*Split info*) :
  L'entropie de la distribution des instances en branches
  - L'information (nbr. bits) nécessaire pour dire qu'une instance *suit telle branche*

- Dans Fig 11 (avec ID code) : toutes les feuilles auront une instance :

  ➢ info intrinsèque : **info([1,1,....,1]) = -1/14 × log 1/14 × 14 = log 14 = 3.807**

  = le nombre de bits nécessaires pour *déterminer la branche de chaque instance.*

  ➢ moins de 4 bits nécessaires pour les 14 exemples de "Météo" (ici 3.807).

Les attributs dispersants (suite)

- Plus il y a de branches, plus l'information intrinsèque est grande.
- **Le ratio de gain** pour un attribut $A = \boxed{\dfrac{Gain\ de\ A}{Info\ Intrinsèque\ de\ A}}$

  ➛ L'importance de l'attribut diminue quand l'information intrinsèque augmente

- Exemple "Météo" :
  - $info\ intrinsèque(\text{"}ID\ code\text{"}) = 3.807$

    et $info(\text{"}ID\ code\text{"}) = 0.940$
  - Ratio de gain de "ID Code" $= \frac{Gain\ de\ \text{"}ID\ Code\text{"}}{Info\ Intrinsèque\ de\ \text{"}ID\ Code\text{"}}$

    $= \frac{0.940}{3.807} = 0.246$

Les attributs dispersants (suite)

Application à l'exemple "Météo" :

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.362
Gain ratio: 0.247/1.577	0.156	Gain ratio: 0.029/1.362	0.021
Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

TABLE 5: Ratio de Gain pour les attributs de l'exemple "Météo" (Outlook est meilleur)

Les attributs dispersants (suite)

Exemple (*outlook* de "Météo") :

Rappel : on avait calculé

$$\boxed{gain(\text{"Outlook"}) = info([9,5]) - info([2,3],[4,0,[3,2]) = 0.940 - 0.693 = 0.247 \text{ bits}}$$

➛ Info intrinsèque de "Outlook" (*SANS faire attention* aux classes finales) :

$$\boxed{split\ info([5,4,5]) = 1.577}$$

D'où :

$$\boxed{ratio\ de\ gain = \frac{0.247}{1.577} = 0.156}$$

• Rappel : l'information intrinsèque (split info) est plus grande pour un attribut "trop dispersant" (e.g. ID code).

Les attributs dispersants (suite)

Classification dans l'exemple "Météo" (par ratio) :

- "Outlook" l'emporte mais "Humidity" est maintenant <u>très proche</u> :
- "Humidity" divise l'ensemble en 2 parties (7 et 7) au lieu de 3 (pour outlook)

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.362
Gain ratio: 0.247/1.577	0.156	Gain ratio: 0.029/1.362	0.021
Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

TABLE 6: Ratio de Gain pour les attributs de l'exemple "Météo" (Outlook est meilleur)

Les attributs dispersants (suite)

Insuffisances et Inconvénients du Ratio de gain :

- Dans la BD. "météo", l'attribut "ID code" (si on le garde !) avec un ratio de 0.246 **sera préféré** aux 4 autres attributs.
  - ➛ Même si cet *avantage* est grandement réduit par le calcul du ratio.
- Dans la pratique, un test empêche de scinder sur un tel attribut "inutile".
- A contrario, le ratio de gain peut sur-compenser un attribut :
  - ➛ On risque de préférer un attribut uniquement parce que son *split info* est bien moindre (donc ratio plus élevé)

Une solution pratique : si le ratio de gain important, ne choisir l'attribut que s'il a un gain d'information supérieur à la moyenne des gains d'information.

- Le ratio sacrifie l'élégance et la clarté théorique du critère de gain d'information.

Les attributs dispersants (suite)

Un autre inconvénient (à surveiller) :

• Les attributs qui scindent en beaucoup de branches posent le problème de *Fragmentation* menant à un trop grand arbre.

Problème de Fragmentation :

• Les mesures (*entropie*, *Gini*, etc) peuvent provoquer de la **fragmentation** :

- ○ Le nombre d'instances devient plus petit quand on traverse l'AD
- ○ Sur un noeud, ce nombre pourraient être trop petit pour toute mesure statistique.

➛ **Une solutions** : recours à l'élagage (regroupement de sous arbres sous la contrainte d'erreur acceptable)

Pratique des ADs

- La construction d'AD est une procédure d'induction.
- L'algorithme de **Hunt** donne les étapes de cette induction par une approche descendante (via une stratégie "diviser et régner")

- Soit D_t l'ensemble atteignant un noeud t, $C = \{c_1, .., c_k\}$ les classes.

- Si D_t est un ensemble vide alors t est une feuille étiquetée par la classe par défaut c_d - Si D_t contient des instances qui appartiennent à la même classe c_t alors t est une feuille étiquetée par c_t - Si D_t qui contient des instances qui appartiennent à plus d'une classe, * Utiliser un attribut pour scinder le données en petits sous-ensembles. * Récursivement appliquer la même procédure à chaque sous-ensemble.

☞ Les critères de gain et de ratio de gain sont des mesures parmi d'autres.

- L'algorithme **ID3** développé par Ross Quinlan (cf. BE)
  - ID3 → développement de **C4.5**
  - **C4.5** traite les attributs numériques, les valeurs manquantes et données bruitées + génération de règle de à partir de l'arbre (cf. BE, Weka).

Pratique des ADs (suite)

Remarques :

- Lors de l'affectation d'une classe à une feuille d'une AD :
  - Le critère principal est la "pureté" du noeud
  - Pour décider de la classe affectée à une feuille si son homogénéité n'est pas totale, on peut :
    - ➢ Choisir la classe la mieux représentée (majoritaire) ;
    - ➢ Affecter la classe *a posteriori* la plus probable (au sens Bayes) si les probabilités *a priori* sont connues ;
    - ➢ Affecter la classe la moins couteuse si les couts des mauvais classements sont connus.

Approches similaires

Les méthodes les plus utilisées pour construire l'arbre :

- Elles varient selon le critère de choix d'attribut sur un noeud

- *ID3* utilise l'*Entropie de Shannon* (et le *Gain*) vue ci-dessus

$$Gain(P, Att) = Entropie(P) - \sum_{v \in valeurs(Att)} \frac{|P_v|}{|P|}.Entropie(P_v)$$

  P est le noeud parent (avant partitionnement sur Att)

- *C4.5* utilise (en plus) le *ratio de gain*

- **Les méthodes similaires** (utilisant des mesures différentes) :
  - ➢ *CART* utilise la mesure *Gini* et divise toujours en 2 branches.
  - ➢ *SLIQ, SPRINT* utilisent la mesure : *erreur de classification*

CART et la mesure GINI

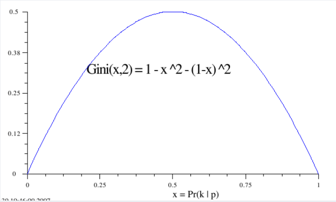
CART :

Classification and Regression Trees (arbres de classification **binaires**)

- Arbres **binaires** (plus simples à comprendre) par la mesure **Gini**
- Les valeurs des attributs ternaires peuvent être regroupées en 2 paquets (3 pour quaternaire, ...)
- L'indice (d'*impureté*) *Gini* mesure la diversité : il exprime
  - ➢ avec quelle fréquence une instance aléatoire serait mal classée si on lui affectait une classe aléatoire dans la distribution de la BD.

• **Mesure Gini** d'impureté (cas bi-classes) :
Étant données une position p dans l'arbre et c classes que l'on cherche à prédire, l'entropie associée à p est donnée par :

$$
\begin{aligned}
Gini(p) &= 1 - \sum_{k=1}^{c} [Pr(k|p)]^2 \\
&= 2 \sum_{k<k'} Pr(k|p) Pr(k'|p)
\end{aligned}
$$



Gini et sa courbe pour 2 classes (axe x=Pr(k|p))

CART et la mesure GINI (suite)

Exemples de calcul de GINI (sur un noeud p) :

- *Gini* mesure la qualité de partitionnement
- Un cas bi-classes (C_1 et C_2) avec sur chaque noeud (ici p) différents nombres d'instances ventilées par un attribut B dans chaque classe (notées en face).
- Indice de Gini pour 4 cas différents $\quad Gini(p, A) = 1 - \sum_{k=1}^{c} [Pr(k|p)]^2$

C_1	0	$P(C_1) = \frac{0}{6} = 0 \qquad P(C_2) = \frac{6}{6} = 1$
C_2	6	$Gini = 1 - P(C_1)^2 - P(C_2)^2 = 1 - 0 - 1 = 0$
C_1	1	$P(C_1) = \frac{1}{6} \qquad P(C_2) = \frac{5}{6}$
C_2	5	$Gini = 1 - P(C_1)^2 - P(C_2)^2 = 0.278$
C_1	2	$P(C_1) = \frac{2}{6} \qquad P(C_2) = \frac{4}{6}$
C_2	4	$Gini = 1 - P(C_1)^2 - P(C_2)^2 = 0.44$
C_1	3	$P(C_1) = \frac{3}{6} \qquad P(C_2) = \frac{3}{6}$
C_2	3	$Gini = 1 - P(C_1)^2 - P(C_2)^2 = 0.5$

TABLE 7: Calcul de l'indice de Gini pour 4 cas différents

➛ Dans le 3e cas : 0.44 chance de se tromper en donnant une classe aléatoire.

CART et la mesure GINI (suite)

- L'intérêt d'un attribut A :
  - L'indice *Gini* permet de mesurer la **qualité de partitionnement** (*split quality*) au niveau des partitions :
  - Similaire à l'entropie (pour $\frac{n_i}{n}$), pour un noeud p partitionné en k partitions, la qualité de split est calculée par :

$$Gini_{split}(p, A) = \sum_{i=1}^{k} \frac{n_i}{n}\ GINI(i) \qquad A \text{ divise } p \text{ en } k \text{ branches}$$

Avec : n_i = nbr d'instances au niveau de la partition i,
n = nbr d'instances du noeud p, $\sum_{i=1}^{k} n_i = n$

- Comme pour l'entropie, l'attribut dont le $Gini_{split}$ est **moindre** est le **meilleur** car il maximise le gain :

$$Gain(p, A) = Gini(p) - Gini_{split} = Gini(p) - \sum_{v \in valeurs(A)} \frac{|p_v|}{|p|} \cdot Gini(p_v)$$

où $|p|$ = taille de p

Exemple de CART et Gini

Exemple : mesure de l'intérêt d'un attribut à 2 valeurs (cas bi-classes) :

- 12 instances (dont 6 dans C_1, 6 dans C_2 au départ),
- A donne 2 partitions N1 et N2 (7 dans N1 dont 5 dans C1)
- On mesure $Gini_{split}$ sur l'attribut A

➛ Les partitions <u>les plus larges et les plus pures</u> l'emportent :

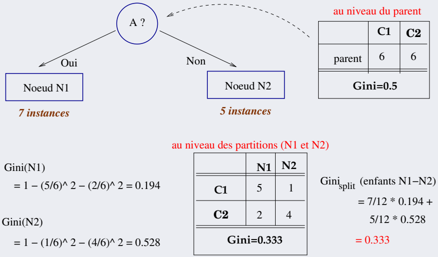


FIGURE 12: Calcul $Gini_{split}$ pour décider de l'intérêt d'un attribut A (dont le gain=0.5-0.333)

Exemple de CART et Gini (suite)

- **Cas de la BD Météo** : calcul de *Gini* pour l'attribut "windy" à la racine :

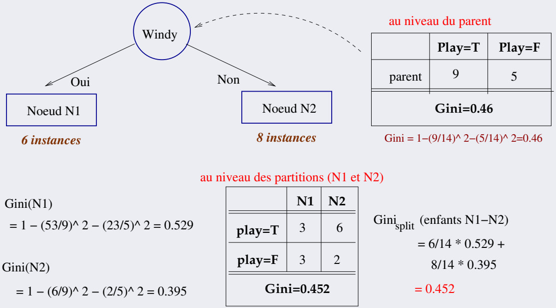


FIGURE 13: Calcul $Gini_{split}$ sur l'attribut *Windy* dans météo

Le Gain sera $0.460 - 0.452 = 0.008$: pas très intéressant !

- Rappel : pour *ID3*, le gain de *Windy* par l'entropie était 0.048

Ex. CART et attribut ternaire

Traitement par CART des attributs ternaires :

- Rappel : CART construit un arbre binaire.
- Exemple : 10 instances, 2 classes, et
  un attribut ternaire *modèle voiture* $\in \{familial, sport, luxe\}$.

	familial	sport	luxe
C1	1	2	1
C2	4	1	1
Gini	=	0.393	

- Par la méthode CART, on peut calculer Gini avec $\{V_1, V_2\}$ vs. V_3

  (ou toute autre combinaison).

  ➛ Ici :

  - La combinaison $\{sport, luxe\}$ et $\{familial\}$ donne *Gini=0.4*
  - La combinaison $\{familial, luxe\}$ et $\{sport\}$ donne *Gini=0.419*
  - ...

Bilan Méthode CART

- Utilisant *GINI*, *Gini Split* et *Gain*, la méthode CART construit :

  ◦ un **Arbre de Classification** où les attributs sont à valeurs catégorielles (ensemble de valeurs mutuellement exclusives et exhaustives)

  ◦ un **Arbre de Régression** pour les numériques (à valeurs continues).

  ◦ pour des données mixtes (cf. Temp. dans "météo"), CART traite des attributs numériques par discrétisation (vue plus haut) et en observant le *Gain* pour différents nombres de partitionnement des valeurs continues.

  → On retient la discrétisation qui maximise le Gain.

☞ Voir détails de CART (lorsque la classe = un réel) plus loin.

- Traitement de valeurs manquantes :
  ➛ Remplacer les valeurs manquantes par le *mode* (la valeurs la plus répétée)
  ➛ Attribution de probabilité (d'être présente) aux valeurs manquantes.

CART et la BD Météo

- On applique CART (division binaire, classe binaire) à l'exemple Météo.
- **Résultats** : 9 instances bien classées, 5 mal classées (erreur de 5/14)
  ➛ On remarque les divisions uniquement binaires (sous Weka, 10-XV) :

```
outlook = sunny ou rainy
  humidity = high
    outlook = sunny ou overcast : no     Ici, seul "sunny" s'applique
    outlook != sunny ou overcast : yes
  humidity != high
    windy = TRUE : yes
    windy != TRUE : yes
outlook !=sunny ou rainy : yes
```

- Rappel : pour la même base de données :
  - ID3 donne (seulement) 2 erreurs sur 14,
  - C4.5 donne 7 erreurs sur 14.

Méthodes SLIQ et SPRINT

- Autre mesure dans les Arbres de Décision : *Erreur de Classification*

  ➛ utilisée dans les méthodes SLIQ et SPRINT

- Ex. de calcul de l'Erreur de Classification au noeud p divisé en i branches :

  $Error(p) = 1 - max(Pr[i|p])$ ← ex. mesure linéaire

- **Cas / valeurs extrêmes :**

  ➛ Maximal (i.e. $1 - 1/n_c$) quand les instances sont équi-réparties entre toutes classes produisant le minimum d'information intéressante.

  ➛ Minimal (i.e. 0) quand les instances appartiennent toutes à une même classe produisant maximum d'information intéressante.

Erreur de Classification → meilleure = 0, pire = équi-réparties

$$\boxed{Error(t) = 1 - \max_i P(i\,|\,t)}$$

C1	0
C2	6

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	1
C2	5

P(C1) = 1/6 P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

C1	2
C2	4

P(C1) = 2/6 P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

- GINI est proche de la mesure de l'erreur de classification (linéaire dans le cas précédent).

- Il a été démontré que la mesure Gini minimise l'erreur de classification pour cette méthode.

Arbre de Classification probabiliste

- **Autre mesure** : probabilité d'être dans une classe donnée.
  - ➢ Donne lieu à la méthode **Arbre de Classification probabiliste** (*Class Probability Trees*)

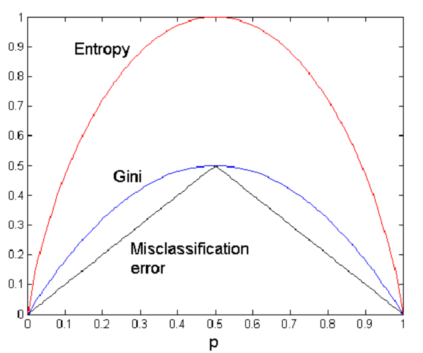
- **Idée** : calculer la probabilité d'être dans une certaine classe plutôt que de grouper des instances dans une même classe (cf. ID3, C45 et CART).

- **Exemple** : Diagnostic médical selon les maladies D1, D2, D3 :
  - ➢ on peut calculer la probabilité pour chacune des instances d'appartenir à une des classes (e.g. à l'aide de l'erreur *Moindre Carré*).

☞ Voir plus loin méthodes Gaussiennes et EM (*Expectation Maximization* = *espérance-maximisation*).

Comparaison des 3 mesures

- Rappel des mesures utilisées : *Entropie*, *Gini* et *Erreur de classification*
- Cas bi-classes :



Arbre de régression et CART

- Le cas où la valeur à prédire (y) est un nombre réel
  - Les attributs explicatifs peuvent être numériques (ou pas).
  - L'arbre de décision devient un **arbre** dit **de régression**.

Construction de l'arbre de régression :

- On procède de la même manière (que pour un AD) sauf que le critère pour scinder en branche tiendra compte par exemple de la **variance** de y :
  ➢ faire en sorte qu'après la division sur un attribut, la distribution de la variance dans les noeuds soit telle que *la somme des variances soit minimale*.

La condition d'arrêt (comme pour les ADs) :

- Si le nombre d'instances sur un noeud = 1
- Ou si les instances appartiennent à une *même classe*

Le choix du meilleur attribut au niveau d'un noeud N :

- Discrétisation des valeurs réelles guidée par la variance.

☞ L'arbre de régression de CART ne sera donc pas forcément binaire.

Arbre de Régression : Choix d'attribut

• Etant donné les instances (x, y), $y \in \mathbb{R}$, CART utilise le critère de **variance** pour choisir le meilleur attribut de partage (à valeurs réelles).

• Un attribut A avec différentes valeurs v produit une partition $T = \cup_{v_A} T_{v_A}$, chaque sous-ensemble ayant sa propre variance $V(T_{v_A})$.

• La variance attendue après une division sur l'attribut A (pour une instance (x, y) tirée uniformément au hasard dans T) est alors $V_A = \sum_{v_A} \frac{|T_{v_A}|}{|T|} V(T_{vA})$

☞ L'attribut A^* qui minimise cette somme est heuristique-ment le meilleur.

Algorithme de choix d'attribut de CART (voir addendum +loin) :

Pour tout attribut A avec les valeurs V_i
Scinder les instances du **noeud N** *selon les valeurs V_i*
$\sigma^2(A, i)$ = *la variance(y) des instances T_i descendues dans la branche i*
$\sigma^2(A) = \alpha \sum_i \sigma^2(A, i)$ % α : *la proportion* $\frac{N_i}{N}$
Fin Pour
Choisir A tel que $\sigma^2(A)$ soit minimum

☞ Cet algorithme ne s'applique pas uniquement aux arbres binaires.

• On évite de construire un arbre avec une profondeur importante (*overfitting*).

Utilisation de l'arbre de CART

Décision (prédiction de la classe réelle y) d'une nouvelle instance :

- Cas d'arbre binaire (les valeurs des attributs en 2 partitions).

- Une fois l'arbre t construit, la régression d'une nouvelle instance x explore l'arbre t selon la démarche suivante (la même que pour les ADs) :

- **Pour un arbre binaire :**

```
Regresser(x, t) : renvoie y ∈ ℝ la classe de l'instance x
    si t est une feuille (soit T_f)          % les (T_f) : instances de l'ensemble d'apprentissage
    alors renvoie la moyenne des valeurs de y de T_f
    sinon
        si t = noeud(i, v, t_left, t_right)       % choix de branche pour le ième attribut
        alors si x[i] ≤ v                         % x[i] est le ième attribut de x
              alors renvoie Regresser(x, t_left)
              sinon                                                   x[i] > v
                 renvoie Regresser(x, t_right)
        sinon                                           t = noeud(i, {v → t[v]})
          renvoie Regresser(x, t[x[i]])     % Autres cas : sélection le ième attribut de x
fin si
```

Utilisation de l'arbre de CART (suite)

Détails :

- L'exploration aboutit à une feuille T_f (contenant 1 à plusieurs instances) et la décision est à prendre sur l'ensemble $Y_f = \{y | \exists (x, y) \in T_f\}$.

- La valeur prédite est la moyenne $\overline{y}$ de Y_f,

➢ Cette prédiction sera d'autant meilleure que la distribution des $y \in Y_f$ est concentrée autour de y.

➢ C-à-d. : plus la dispersion des $y \in Y_f$ est grande, plus la prédiction sera mauvaise (erreur plus élevée).

- Dans le cas de CART :

➢ on utilise la variance $\overline{\sigma}^2$ des $y \in Y_f$ pour mesurer cette dispersion,

➢ et par ce biais estimer la "qualité" de la feuille f.

Remarque : pourquoi minimiser la variance ?

Bonnes raisons de la minimisation de la variance :

• Les instances (x, y) sont des réalisations i.i.d. des variables aléatoires (X, Y) et peuvent éventuellement être corrélées.

➛ Les (x, y) de la BD. ont été *générés* indépendamment selon une loi de probabilité inconnue P (*générative*).

• De même, les classes $y \in Y_f$ sont des réalisations i.i.d. d'une certaine variable aléatoire Y_f :

➛ la réalisation de Y conditionnée par les événements $X_i \leq v$ *ou* $X_i > v$ testés le long de la branche menant à f.

• Dans ce contexte, pour y la classe de la feuille Y_f :

∘ y est un estimateur de $\mathbb{E}(Y_f)$ et

∘ **l'erreur quadratique** commise en prédisant y pour une nouvelle instance dans Y_f sera :

$$\begin{aligned}\mathbb{E}(Y_f - \overline{y})^2 &= \mathbb{E}(Y_f - \mathbb{E}(Y_f) + \mathbb{E}(Y_f) - \overline{y})^2 \\ &= (\overline{y} - \mathbb{E}(Y_f))^2 + \mathbb{E}(Y_f - \mathbb{E}(Y_f))^2\end{aligned}$$

➛ y et $\mathbb{E}(Y_f)$ sont ici constantes. ../..

Remarque : pourquoi minimiser la variance ? (suite)

- Soit donc l'erreur : $\mathbb{E}(Y_f - \overline{y})^2 = [\mathbb{E}(\overline{y} - \mathbb{E}(Y_f))]^2 + \mathbb{E}(Y_f - \mathbb{E}(Y_f))^2$

  ➢ A dte., le 2nd terme est la variance σ^2 de Y_f, dont $\overline{\sigma}^2$ est un estimateur.
  ➢ Le premier terme s'appelle *(biais)*^{2} = l'erreur d'approximation,
  ➢ Le 2nd *la variance* sur $\bigcup_{BDs.}$ = la sensitivité de la prédiction sur 1 BD donnée.

- Minimiser $\overline{\sigma}$ pour la feuille f vise donc à minimiser ce second terme.

☞ Mais aussi le premier car (n : taille de la BD, T : une partition) :

  ◦ Sachant que $\overline{y} = \frac{1}{k}\sum_{i=1}^{n} Y_f^{(i)}$ est une moyenne empirique des réalisations de Y_f , on a $\mathbb{E}_T(\overline{y}) = \mathbb{E}(Y_f)$ et $V_T(\overline{y}) = \frac{1}{n}\sigma^2$.

  ◦ A l'aide de *l'inégalité de Markov* (i.e. $P(|X - \mathbb{E}(X)| \geq \alpha) \leq \frac{\sigma^2}{\alpha^2}$), on pose :

$$P_T[(\overline{y} - \mathbb{E}(Y_f))^2 > \alpha] < \frac{\sigma^2}{k\alpha}$$

tout réel $\alpha \geq 0$, k : le nbr d'instances considérés

  ➢ **Donc** : réduire σ a pour effet de concentrer la distribution de $\overline{y} - \mathbb{E}(Y_f)$ en zéro.

Évaluation de CART

Évaluation du choix de l'attribut de partage :

Choix d'un attribut explicatif (comme pour numérique) :

- La variance attendue après une division sur l'attribut A (pour une instance (x, y) tirée uniformément au hasard dans T) est : $V_A = \sum_{v_A} \frac{|T_{v_A}|}{|T|} V(T_{vA})$

➛ L'attribut A qui minimise cette somme est heuristiquement le meilleur.

- De même pour le **calcul de l'erreur** : un attribut A produit une partition $T = \cup_{v_A} T_{v_A}$, chaque sous-ensemble T_{v_A} a sa propre erreur $e(T_{v_A})$.

➛ L'erreur attendue après un branchement sur cet attribut (pour une instance (x, y) tirée uniformément au hasard dans T) est alors

$$e_A = \sum_{v_A} \frac{|T_{v_A}|}{|T|} e(T_{v_A})$$

➛ L'attribut qui minimise cette somme est (heuristiquement) le meilleur.

Classification CART pour cas classe catégorielle

- CART peut être utilisée pour les classes non numériques (évt. discrétisées).
  - ➛ Lorsque la classe est discrète (catégorielle), l'algorithme de classification de CART (pour construire un AD) sera similaire à celui de la régression.

Ex : exploitation d'un arbre binaire CART (les valeurs des attributs en 2 partitions) :

```
Classifier(x, t)                              % Le même que l'algorithme de Hunt pour les ADs mais
    si t est une feuille                      % sans l'indice Gini (même pour les vars catégorielles)
    alors retourner la classe majoritaire     % majoritaire ≃ variance minimale ?
    sinon si t = noeud(i, v, t_left, t_right)
        si x[i] ≤ v
        alors retourner Classifier(x, t_left)
        sinon                                 % x[i] > v
        retourner Classifier(x, t_right)
    sinon                                     % t = noeud(i, {v → t[v]})
    retourner Classifier(x, t[x[i]])
fin si
```

☞ Rappel : on appelle également CART la méthode qui utilise l'indice de *Gini* dans la construction de l'AD.

- ➛ Ici : arbre de régression adapté à un cas de classification (sans GINI).

Rappel : on a vu (ci-dessus) l'arbre de régression dans le cas où la classe $y \in \mathbb{R}$

Classification CART pour cas classe catégorielle (suite)

Homogénéité des feuilles et Erreur

- L'arbre de classification idéal possède des feuilles homogènes (même classe).
  ➛ Pas toujours réaliste : l'homogénéité n'est souvent pas être totale (*pure*).

- 3 mesures utilisées pour quantifier l'homogénéité :
  soit $p_1, ..., p_c$ les fréquences relatives des classes $1..c$ dans T_f,
  et c^* la classe la plus fréquente.

1. **Entropie** : $e(T_f) = -\sum_c p_c \; log(p_c)$: un estimateur de l'entropie de la classe d'une instance de T_f tirée uniformément au hasard.
   ➛ C'est la mesure d'erreur utilisée dans les arbres ID3 (et en partie en C4.5).
2. **Taux d'erreur** : $e(T_f) = 1 - p_{c^*}$: taux erreur de classification sur l'ensemble d'apprentissage.
3. **Gini** : $e(T_f) = \sum_c p_c(1 - p_c)$: taux erreur de classification sur l'ensemble d'apprentissage d'un algorithme 'randomisé' qui renvoie la classe c avec la probabilité p_c (au lieu de toujours retourner la classe c^*).
   ➛ Cette mesure est souvent utilisée dans CART pour les attributs discrets.

Extension des arbres de décision

Les arbres de décision (ID3, C45, CART, ...) présentent plusieurs limitations :

- Le problème d'optimisation globale est NP-complet pour de nombreux critères d'optimalité :
  ➢ On utilise des heuristiques ;
- La procédure d'apprentissage d'un arbre de décision/de régression est <u>statique</u> :
  ➢ on ne peut pas apprendre de manière incrémentale de nouvelles instances qui viendraient s'ajouter à l'ensemble d'entraînement ;
- Elle est sensible au bruit et a une forte tendance à sur-apprendre
  ➢ i.e. à apprendre à la fois les relations entre les données et le **bruit** présent dans l'ensemble d'apprentissage.

Solutions (voir aussi plus loin la méthode *Ensemble*) :

- **Élagage** : travaille directement sur les arbres et procède à l'élagage.
- **Baguage** (bagging) : *collégial* plutôt que d'utiliser des prédicateurs individuels
  ➢ Relève d'un contexte dans lequel le sur-apprentissage et la sous-optimalité (dû aux heuristiques) posent moins de problèmes.

Extension des arbres de décision (suite)

Bagging (Bootstrap aggregating) :

➛ une méta-méthode à base de *Bootstarpping* puis d'*Agrégation*

Bootstarpping : un *bootstrap* d'un ensemble T est l'ensemble obtenu en tirant $|T|$ fois des éléments de T uniformément au hasard et avec remise.

➛ produit un nouvel ensemble T' (de la même taille que T) qui présente en moyenne $1-e^{-1} \approx 63\%$ des instances uniques différentes de T quand $|T| \succcurlyeq 1$.

Agrégation : on produit (ainsi) plusieurs bootstraps $T_1, ..., T_m$, chaque bootstrap T_i est utilisé pour entraîner un prédicteur t_i

➛ Par exemple, dans le cas des arbres de décision/régression :

- Pour une instance (x, y), on fait régresser chaque arbre, ce qui nous donne un ensemble de valeurs $y_1, ..., y_m$ prédites.

➛ Celles-ci sont alors agrégées en calculant leur moyenne $\hat{y} = \dfrac{1}{m}\sum_i y_i$.

- Le *bagging* corrige plusieurs défauts des arbres de décision dont :
  - *instabilité* : de petites modifications dans l'ensemble d'apprentissage peuvent entraîner des arbres très différents
  - *overfitting* : leur tendance à sur-apprendre.

Extension des arbres de décision (suite)

Les contreparties du Bagging :

- Une perte de lisibilité :
  ➢ les prédictions d'une forêt d'arbres issue de Bagging ne sont plus le fruit d'un raisonnement, mais un consensus de raisonnements potentiellement très différents.

- La corrélation entre les prédicteurs réduit les gains apportés par le Bagging.

- L'analyse théorique du *bagging* est difficile :
  ➢ on peut comprendre les améliorations apportées mais il reste difficile de modéliser et de mesurer son impacte (c'est un sujet de recherche).

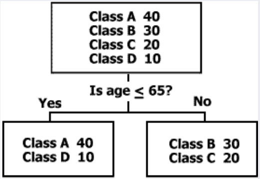
Extension des arbres de décision (suite)

Random Forests (RF) : une autre méthode collégiale

- Pour adapter les ADs au Bagging, dans l'algorithme de construction d'AD :
  - Au lieu de choisir le "meilleur" attribut, on échantillonne uniformément un *sous ensemble des attributs* parmi lequel on choisit les "meilleurs".
  - Dans le cas d'un arbre de régression, on choisit en général 1/3 des attributs dont on prendre les meilleurs.
- De cette manière, les arbres construits sont dé-corrélés
  - ➛ Rappel : la corrélation entre les prédicteurs réduit les gains apportés par le Bagging.
- La condition d'arrêt : les ADs seront les plus profonds possibles (dans la limite du temps de calcul) utilisant les attributs sélectionnés (dans RF) :
  - pour permettre au prédicteur de contenir un maximum de relations entre les données,
  - le sur-apprentissage et la sensibilité aux bruits seront compensées par le Bagging.

La variante twoing

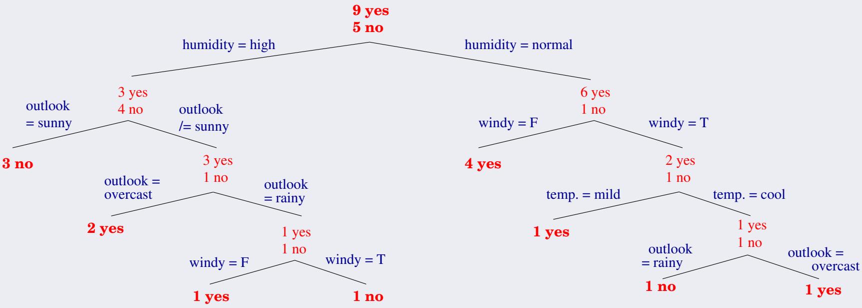
- Les méthodes ID3, C4.5, CART ... et similaires :
  - Choix d'un attribut maximisant la pureté (divisions binaires pour CART).
- Une variante pour créer un AD binaire : la méthode TWOING
  - Scinder les classes en 2 groupes couvrant chacun env. 50% des instances.
  - Ensuite, chercher le meilleur attribut qui permet de faire cette division.
  - Et ainsi de suite...



- Le *Twoing* marche plutôt dans le cas de données idéales !

La variante twoing (suite)

Exemple "Météo" : à chaque étape, on cherche des branches (presque) "balancées" : $\frac{1}{2}$ des instances $\rightarrow 2 \times 7$ instances à la racine, et ainsi de suite



	Outlook	Temperature	Humidity	Windy	Play
Test ➛	Sunny	Cool	High	True	? ?

- Pour le jour ci-dessus, la décision sera = No

  ➛ conforme au verdict de l'AD précédente et de la méthode BN.

La variante twoing (suite)

Une mesure souvent utilisée en Twoing :

- Soit P_l la proba qu'une instance du noeud courant soit dans la branche gauche,
- P_r : proba pour aller dans la branche droite
- t est le noeud à scinder

• *Twoing* utilise le **maximum** de la mesure ϕ suivante pour décider de la division binaire ($j|P_x$ a le même sens que en Entropie) :

$$\phi(t) = \frac{P_l\ P_r}{4}\left[\sum_j \left(Pr(j|P_l) - Pr(j|P_r)\right)\right]^2)$$

La (méta) méthode Ensemble & ADs

Généralisation des méta-méthodes :

• La méta méthode *Ensemble* (voir aussi plus loin pour les détails) utilise plusieurs modèles et procède à un vote majoritaire.

- les modèles sont construits chacun sur une partie des données.
- les Arbres de Décision peuvent être produits utilisant (plutôt) les mesures *Gini* et *Entropie*.

• La (méta) méthode *Ensemble* se décline sous 2 variantes principales : **Boosting** et **Bagging** (une déclinaison vue plus haut, voir aussi + loin).

- *Bagging* : on construit plusieurs (ici 2) arbres de décision et retient l'avis de celui qui semble le plus juste (le moins erroné).
  ➛ Ici, pas d'agrégation.

- *Boosting* : on génère une séquence (e.g. 2) de modèles sur des partitions de l'ensemble d'apprentissage avec différentes (pondérations de) distributions puis on combine les résultats.
  ➛ Ici, agrégation.

La (méta) méthode Ensemble & ADs (suite)

Exemples simples de la méthode Ensemble :

• Soit 2 ADs sur toutes les données par 2 méthodes différentes (*Bagging*) :

si l'arbre A classe une nouvelle instance dans la classe C_1 avec un risque d'erreur e_1 et l'arbre B classe cette instance dans la classe C_2 avec une erreur e_2 et que $e_1 < e_2$ alors on retient la classe C_1.

Un cas particulier de décision collégiale :

• Cas de combinaison de plusieurs modèles obtenus chacun sur une partie des données avec une distribution propre (*Boosting*).

Dans cette figure, un ensemble de 3 classifieurs linéaires (A,B,C) constitue conjointement le modèle.
➛ Les lignes en **gras** donnent l'**ensemble** qui classifie un nouvel exemple en utilisant le vote majoritaire de A, B et C.

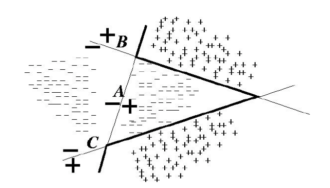


Table des matières

Table des matières (suite)