UNIVERSITÉ CÔTE D'AZUR

Introduction à l'Intelligence Artificielle (L2 Portail Sciences et Technologies)

Andrea G. B. Tettamanzi
Laboratoire I3S – Équipe SPARKS
andrea.tettamanzi@univ-cotedazur.fr

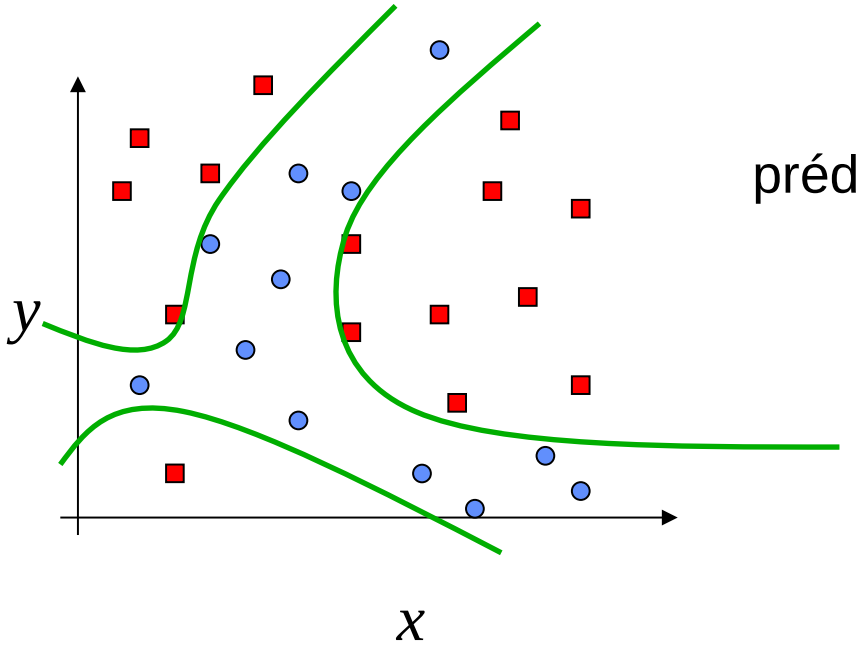
univ-cotedazur.fr

Séance 8
Apprentissage supervisé

Plan pour cette séance

- Classification et prédiction
- Évaluation des modèles
- Survol de quelques méthodes
  - Arbres de décision
  - Classification bayésienne
  - Systèmes de règles
  - Logique floue

Apprentissage supervisé



prédiction/classification

$z = M(x, y)$

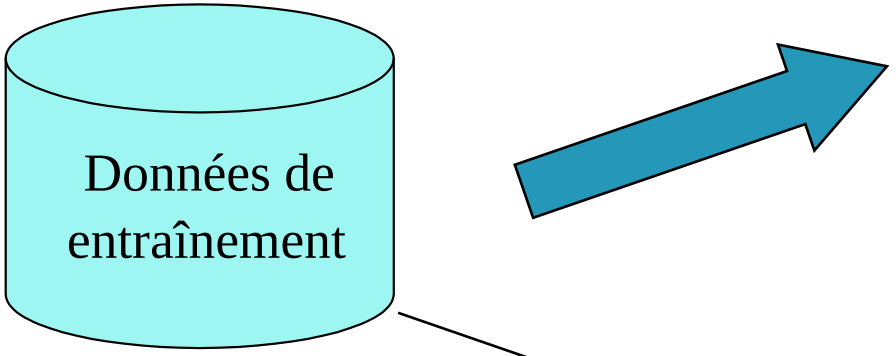
Variables connues

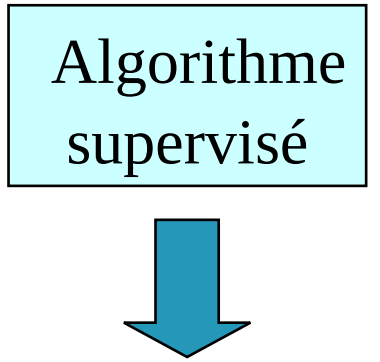
M est la loi qui lie les variables x, y et z.
Étant donné un échantillon de n-uplets (x, y, z),
on cherche la loi qui les “explique”.

Classification vs. Prédiction

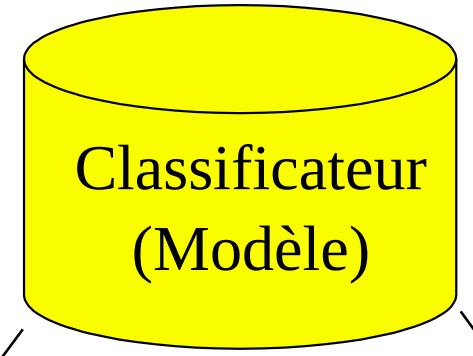
- Classification
  - prédit les étiquettes de classes catégorielles (discrètes ou nominales)
  - classifie les données (construit un modèle) sur la base du jeu de données d'entraînement et des valeurs (étiquettes de classe) dans un attribut de classification et l'utilise pour classifier de nouvelles données
- Prédiction
  - modélise des fonctions à valeur continue, c'est-à-dire prédit les valeurs inconnues ou manquantes
- Applications typiques
  - Autorisation de crédit
  - Marketing ciblé
  - Diagnostic médical
  - Détection des fraudes

Étape 1 : Construction du modèle



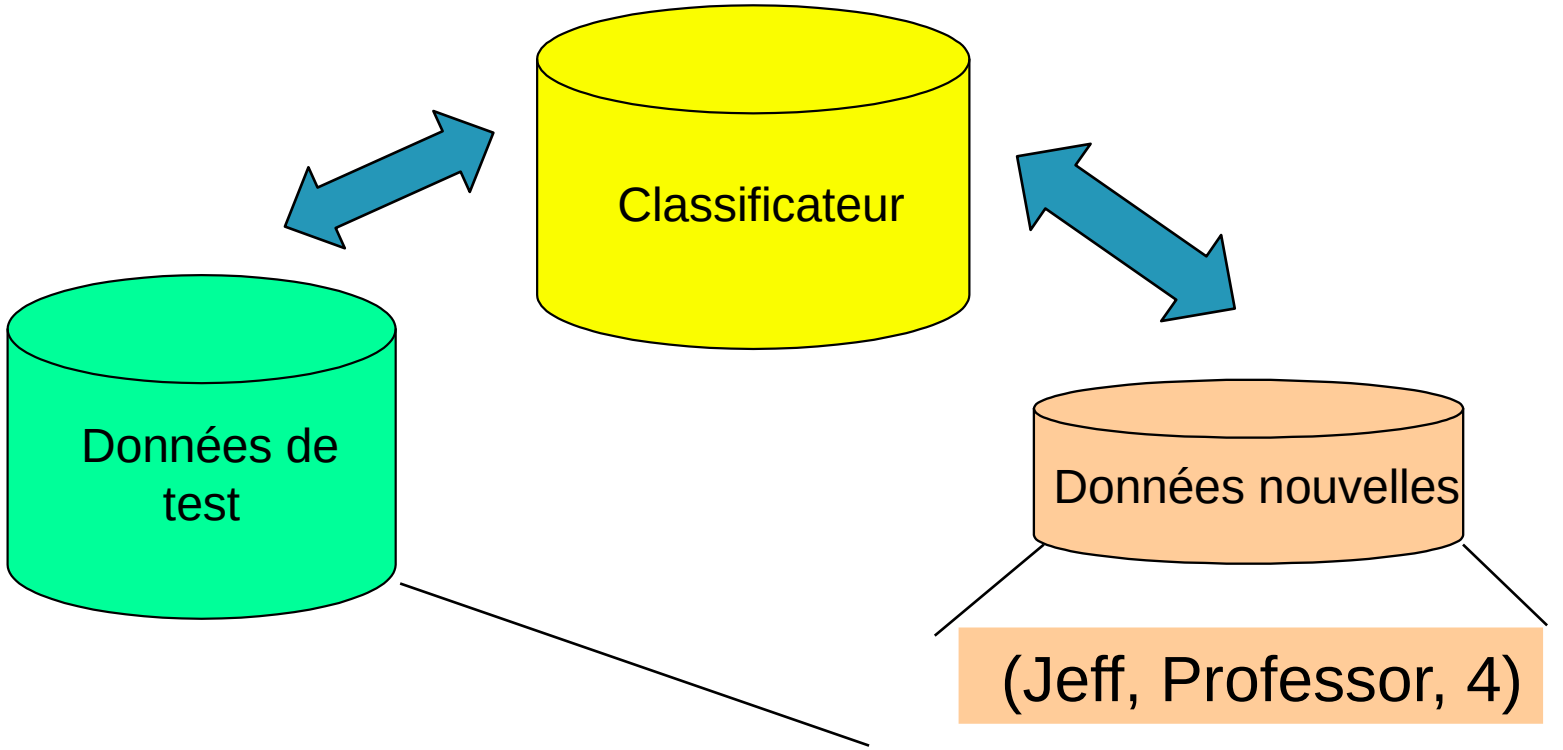


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no



SI rank = ‘professor’
OU years > 6
ALORS tenured = ‘yes’

Étape 2: Utilisation du modèle

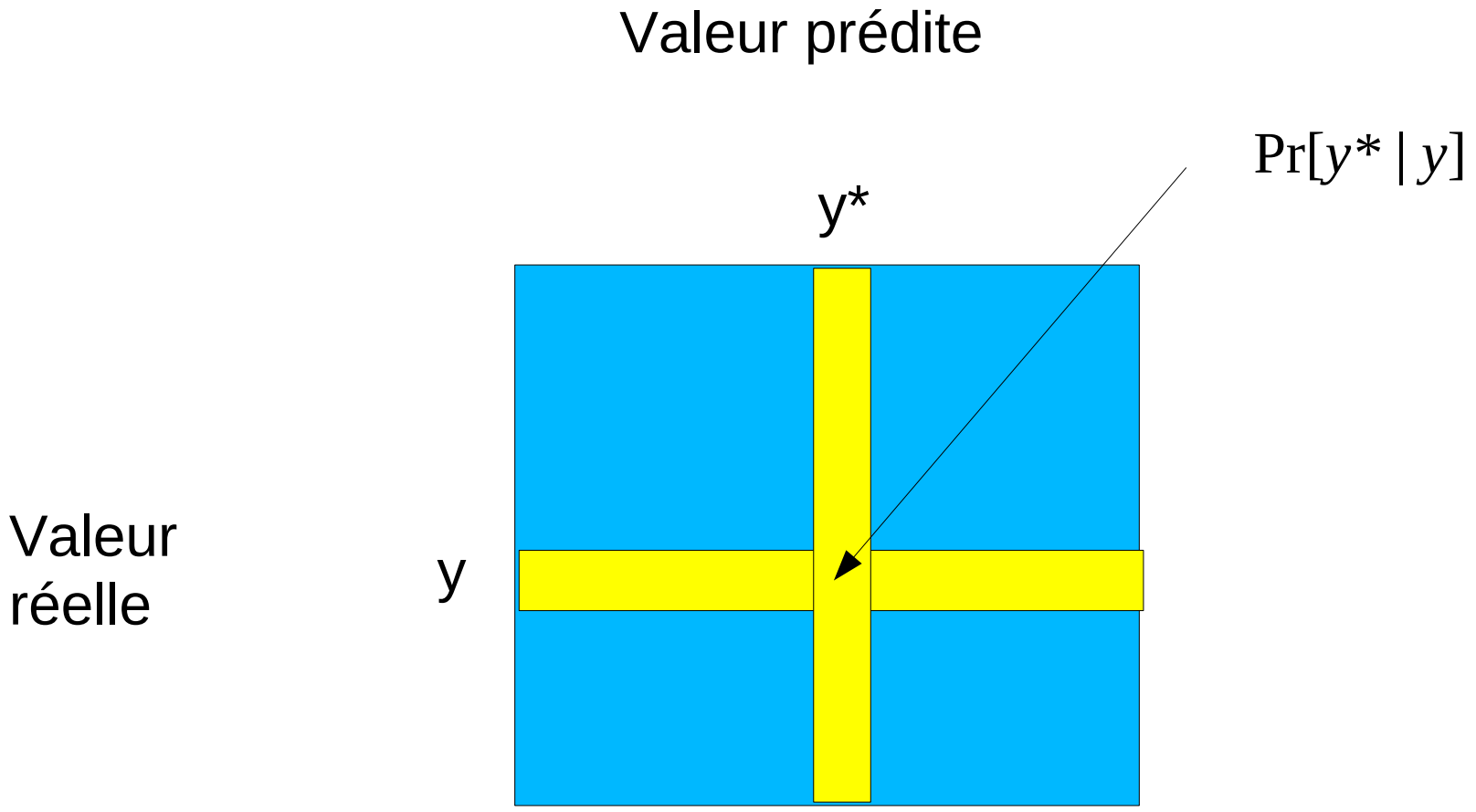


NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Évaluation des modèles

- Précision
  - Précision d'un classificateur : prédiction des étiquettes de classe
  - Précision d'un modèle de prédiction : prédiction des valeurs de la variable cible
  - Mesures plus sophistiquées : matrice de confusion, courbe ROC
- Rapidité
  - Temps d'entraînement
  - Temps pour obtenir une classification/prédiction
- Robustesse : capacité de gérer du bruit et des valeurs manquantes
- Passage à l'échelle : performance sur des grandes masses de données
- Interprétabilité : est-ce qu'on comprend comment le modèle parvient à ses conclusions ?
- Autres mesures, comme la qualité des règles (taille et complexité)

Matrice de confusion



Précision d'un classificateur

	C_1	C_2
C_1	vrai positif	faux négatif
C_2	faux positif	vrai négatif

classes	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.52

- Exactitude du classificateur *M*, acc(*M*): pourcentage des tuples de test qui sont classifiées correctement par le modèle *M*
  - Taux d'erreur de *M* = 1 – acc(*M*)
  - Étant données *m* classes, $CM_{i,j}$, une cellule de la matrice de confusion, indique le nb de tuples de la classe i qui sont étiquetées comme appartenant à la classe j
- Mesures de précision alternatives (p.ex., pour la diagnose d'une maladie)
  - sensitivité = v-pos/pos /* taux de reconnaissance des vrais positifs */
  - specificité = v-nég/nég /* taux de reconnaissance des vrais négatifs */
  - precision = v-pos/(v-pos + f-pos)
  - exactitude = sensitivité * pos/(pos + nég) + specificité * nég/(pos + nég)

Mesures d'erreur de prédiction

- EN gros, il s'agit de mesurer la distance entre la valeur de la prédiction et la vraie valeur connue
- **Fonction de perte :** mesure l'erreur entre y_i et la valeur de la prédiction y_i'
  - Erreur absolue : $|y_i - y_i'|$
  - Erreur quadratique : $(y_i - y_i')^2$
- Erreur de test (ou erreur de généralisation) : perte moyenne sur les données de test
  - Erreur moyenne absolue : $\frac{1}{d}\sum_{i=1}^{d}|y_i - y_i'|$ quadratique : $\frac{1}{d}\sum_{i=1}^{d}(y_i - y_i')^2$
  - Erreur relative absolue : $\frac{\sum_{i=1}^{d}|y_i - y_i'|}{\sum_{i=1}^{d}|y_i - \bar{y}|}$ quadratique : $\frac{\sum_{i=1}^{d}(y_i - y_i')^2}{\sum_{i=1}^{d}(y_i - \bar{y})^2}$

  L'erreur quadratique moyen exagère l'effet de données anormales (outliers)

Techniques d’évaluation (1)

- Validation non-croisée (*holdout method*)
  - Le jeu de données est partitionné en deux ensembles indépendants
    - Données d’entraînement (p.ex., 2/3), pour la construction su modèle
    - Données de test (p.ex., 1/3) pour l’estimation de la précision
  - Échantillonage aléatoire : une variation de la méthode *holdout*
    - Répéter *holdout* k fois, précision = moyenne des précisions obtenues
- Validation croisée (à k blocs, où k = 10 est le choix le plus courant)
  - On partitionne les données en k ensembles mutuellement exclusifs, de taille égale (si possible, sinon similaire)
  - À l’i-ème itération, on utilise le bloc i pour test et les autres pour entraînement
- Leave-one-out: k blocs, où k = nombre d’objets, pour des jeux de données petits
- Validation croisée stratifiée : blocs construits de telle sorte que la distribution des classes y soit identique

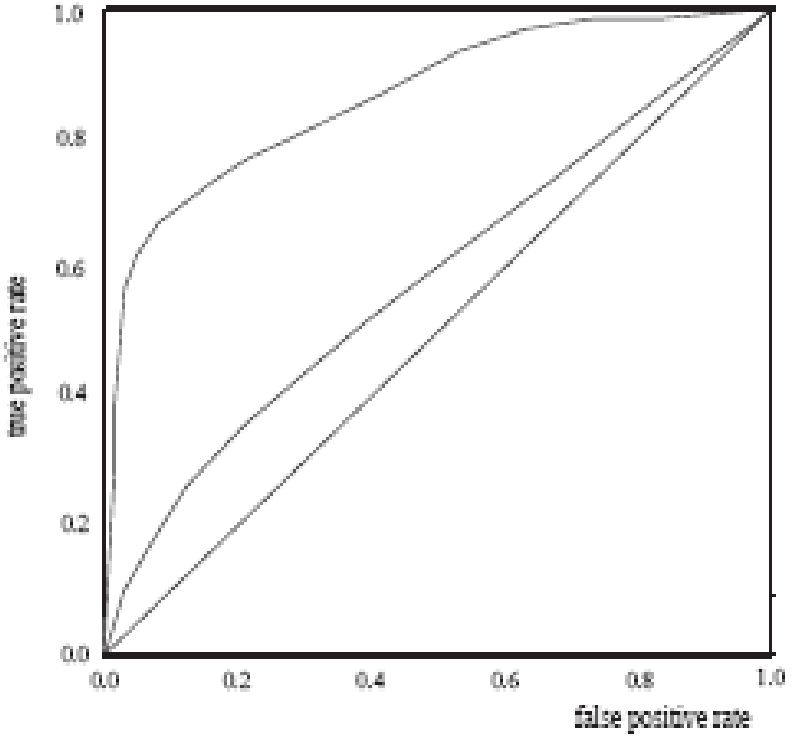
Techniques d’évaluation (2)

- Bootstrap
  - Fonctionne bien avec de petits jeux de données
  - Les échantillons du jeu de données d’entraînement sont échantillonnés uniformément avec remplacement
  - c'est-à-dire que chaque fois qu'un tuple est sélectionné, il a autant de chances d'être sélectionné à nouveau et ajouté au jeu de données d’entraînement
- Plusieurs méthodes de *boosting*, dont la plus courante est le *boosting* 0,632
  - Supposons que l'on nous donne un jeu de données de *d* tuples. Le jeu de données est échantillonné *d* fois, avec remplacement, ce qui donne un jeu d’entraînement de *d* échantillons. Les tuples qui ne sont pas entrés dans l'ensemble de formation finissent par former l'ensemble de test. Environ 63,2 % des données originales se retrouveront dans le *bootstrap*, et les 36,8 % restants formeront le jeu de test (puisque (1 - 1/d)d ≈ e-1 = 0,368)
  - On répète la procédure d'échantillonnage *k* fois, précision globale du modèle :

$$acc(M) = \sum_{i=1}^{k} (0.632 \times acc(M_i)_{test\_set} + 0.368 \times acc(M_i)_{train\_set})$$

Sélection du modèle : courbes ROC

- ROC (*Receiver Operating Characteristics*)
- Comparaison visuelle de modèles de classification
- Technique empruntée de la théorie de la détection des signaux
- Montre le compromis entre sensibilité (taux des v-pos) et spécificité (taux des f-pos)
- La superficie sous la courbe ROC est une mesure de l'exactitude du modèle
- Trier les tuples de test par ordre décroissant : celle qui a plus de chances d'appartenir à la classe positive au sommet du classement
- Plus la courbe s'approche de la diagonale (donc plus la superficie s'approche de 0.5), moins le modèle est précis

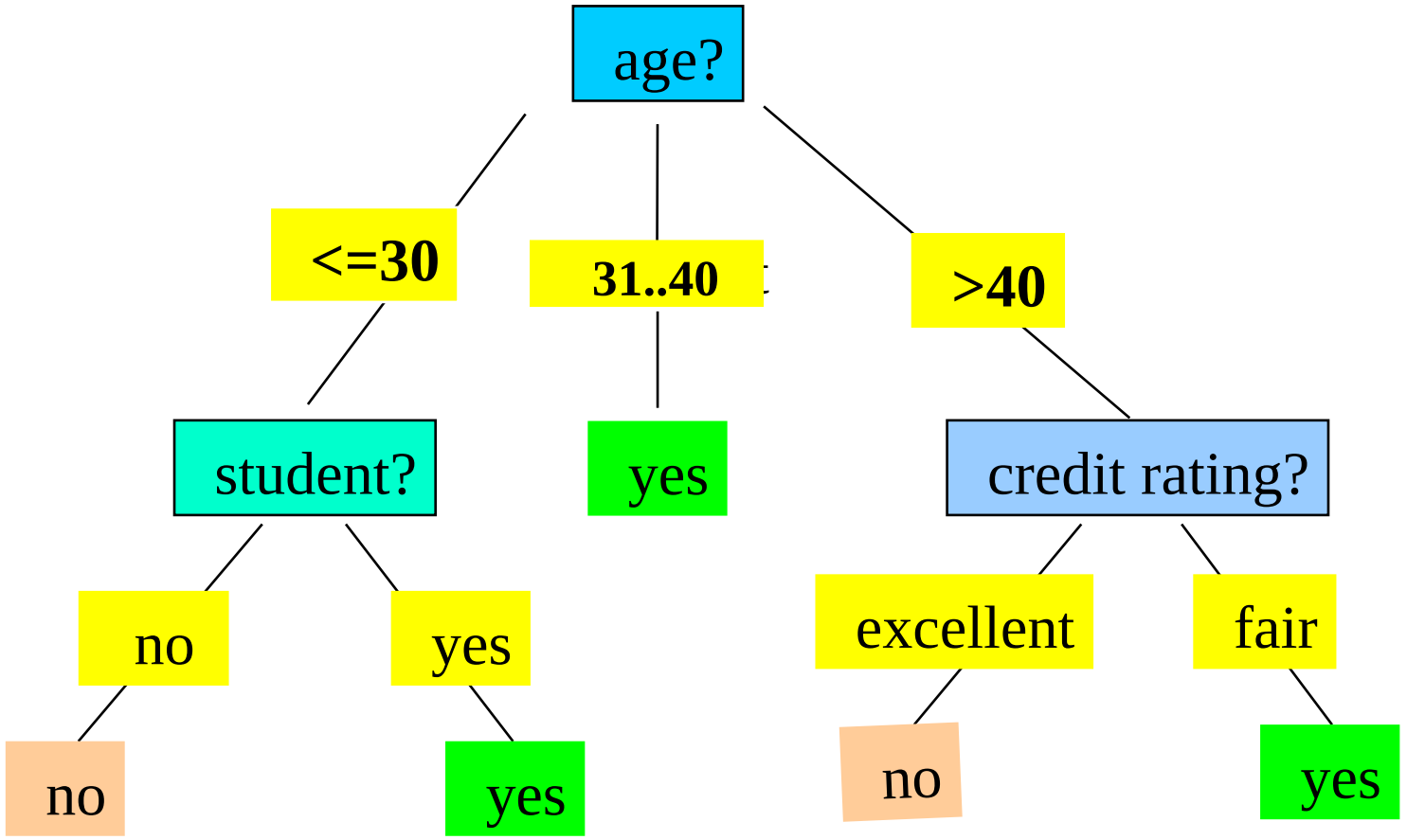


- En ordonnées : taux des vrais positifs
- Abscisses : taux des faux positifs
- On dessine aussi la diagonale
- Un modele parfaitement précis aura une superficie en dessous de la courbe (*area under the curve*, AUC) de 1.0

Induction d'arbres de décision : données d'entraînement

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Résultat : un arbre de décision pour “*buys_computer*”



Algorithme pour l'induction d'arbres de décision

- Méthode de base (un algorithme gourmand)
  - L'arbre est construit de manière récursive, de haut en bas (diviser pour mieux régner)
  - Au départ, tous les exemples d'entraînement sont dans la racine
  - Les attributs sont catégoriques (s'ils sont continus, on les discrétise à l'avance)
  - Les exemples sont partitionnés de manière récursive en fonction de certains attributs
  - Les attributs de test sont sélectionnés sur la base d'une mesure heuristique ou statistique (par exemple, le gain d'information)
- Conditions pour arrêter le partitionnement
  - Tous les échantillons pour un nœud donné appartiennent à la même classe
  - Il n'y a pas d'autres attributs pour un partage ultérieur – dans ce cas, le vote à la majorité est utilisé pour classer la feuille
  - Il n'y a plus d'échantillons

Heuristique pour la sélection des attributs : Gain d'information

- On sélectionne les attributs donnant le plus haut gain d'information
- Soit p_i la probabilité qu'un tuple arbitraire en D appartienne à la classe C_i, estimée par $|C_{i,D}|/|D|$
- Information moyenne (entropie) nécessaire pour classifier un tuple en D :

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information nécessaire (après avoir utilisé A pour partager D en v partitions) :

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

- Gain d'information grâce à l'attribut A :

$$Gain(A) = Info(D) - Info_A(D)$$

Indice de Gini

- Pour un jeu de données *D* qui contient exemples de *n* classes, l'indice de Gini, *gini*(*D*) est défini par

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

où p_j est la fréquence relative de la classe *j* en *D*

- Si *D* est partagé suivant A en deux sous-ensembles D_1 et D_2, l'indice de Gini *gini*(*D*) est défini par

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2}{|D|} gini(D_2)$$

- Réduction de l'impureté :

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- L'attribut qui fournit le plus petit $gini_{part}(D)$ (ou la plus grande réduction de l'impureté) est choisi pour partitionner le nœud

Surapprentissage et élagage de l'arbre

- Surapprentissage : un arbre de décision induit peut surapprendre les données d'entraînement
  - Trop de branches, dont quelques unes pourraient réfléter du bruit ou des anomalies
  - Mauvaise généralisation sur des données nouvelles
- Deux approches pour éviter le surapprentissage
  - Pré-élagage : arrêter la construction de l'arbre prématurément—on ne partitionne pas un nœud si la mesure choisie ne s'améliore pas assez (au dessus d'un seuil fixé)
    - Choix d'un seuil approprié pas évident
  - Post-élagage : on élimine des branches d'un arbre complètement formé—on obtient une séquence d'arbres réduits
    - On utilise des données différentes de celles d'entraînement pour décider lequel est le meilleur

Classification bayésienne

- **Un classificateur statistique** : il effectue une prédiction probabiliste, c'est-à-dire qu'il prédit les probabilités d'appartenance à une classe
- **Fondation** : Basée sur le théorème de Bayes.
- **Performance** : Un classificateur bayésien simple, le classificateur naïf de Bayes, a des performances comparables à celles des arbres de décision et de certains classificateurs basés sur les réseaux de neurones
- **Incrémentale** : chaque exemple d'entraînement peut augmenter/diminuer progressivement la probabilité qu'une hypothèse soit correcte – les connaissances préalables peuvent être combinées avec les données observées
- **Standard** : Même lorsque les méthodes bayésiennes sont difficiles à calculer, elles peuvent fournir une norme de prise de décision optimale par rapport à laquelle d'autres méthodes peuvent être mesurées

Notions de base sur le théorème de Bayes

- Soit X un échantillon de données : l'étiquette de classe est inconnue
- Soit H l'hypothèse que X appartienne à la classe C
- La classification consiste à déterminer P(H | X), la probabilité que l'hypothèse soit correcte, étant donné l'échantillon de données observées X
- P(H) (probabilité *a priori*), la probabilité initiale
  - Par exemple, X achètera un ordinateur, quels que soient son âge, ses revenus, etc.
- P(X) : probabilité que les données de l'échantillon soient observées
- P(X | H) (probabilité *a posteriori*), la probabilité d'observer l'échantillon X, étant donné que l'hypothèse H est correcte
  - Par exemple, étant donné que X va acheter un ordinateur, la probabilité que X soit de 31..40, revenu moyen, etc.

Théorème de Bayes

- Étant données les données d'entraînement **X**, la probabilité a posteriori de l'hypothèse *H*, P(*H* | **X**), suit la formule de Bayes

$$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})}$$

Informellement, ceci peut s'écrire comme

a posteriori = vraisemblance x a priori / évidence

- On prédit que **X** appartienne a C_i ssi la probabilité $P(C_i \mid \mathbf{X})$ est la plus haute parmi toutes les $P(C_k \mid \mathbf{X})$ pour toutes le *k* classes
- Difficulté pratique : cela réquiert une connaissance initiale de beaucoup de probabilités, ce qui signifie un coût de calcul

Classification bayesienne naïve

- Soit D un jeu de données d'entraînement de tuples avec leurs étiquettes de classe, et soit chaque tuple représenté par un vecteur de n attributs **X** = (x1, x2, …, xn)
- Supposons qu'il y ait *m* classes C1, C2, …, Cm.
- La classification se fait sur la base de la plus grande probabilité a posteriori, c-à-d, du P(Ci | **X**) maximum

  On peut calculer ceci grâce au Théorème de Bayes :

$$P(C_i \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C_i)P(C_i)}{P(\mathbf{X})}$$

- Puisque P(**X**) est constante pou toute classe, il suffit juste de maximiser

$$P(\mathbf{X} \mid C_i)P(C_i)$$

Classificateur naïf bayésien

- Simplification : on suppose que les attributs soit conditionnellement indépendant :

$$P(\mathbf{X} \mid C_i) = \prod_{k=1}^{n} P(X_k \mid C_i)$$

- Ceci réduit beaucoup la complexité des calculs : il suffit juste de prendre en compte la distribution de chaque classe
- Si A_k est catégorielle, $P(X_k \mid C_i)$ est le nombre de tuples de C_i ayant la valeur X_k pour A_k, divisé par $|C_{i,D}|$ (nombre de tuples de C_i en D)
- Si A_k est à valeurs continues, $P(X_k \mid C_i)$ est généralement calculé sur la base d'une distribution gaussienne avec une moyenne μ et un écart-type σ et $P(x_k \mid C_i)$ est

$$\mathcal{N}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(\mathbf{X} \mid C_i) = \mathcal{N}(X_k; \mu_{C_i}, \sigma_{C_i})$$

Classificateur naïf bayésien : données entraînement

Classes :
C1:buys_computer = ‘yes’
C2:buys_computer = ‘no’

Tuple donné :
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

age	income	student	credit_rating	com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Classificateur naïf bayésien : exemple

- $P(C_i)$: P(buys_computer = “yes”) = 9/14 = 0.643
  P(buys_computer = “no”) = 5/14= 0.357

- Compute $P(X|C_i)$ for each class
  P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222
  P(age = “<= 30” | buys_computer = “no”) = 3/5 = 0.6
  P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
  P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4
  P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667
  P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2
  P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667
  P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
$P(X|C_i)*P(C_i)$: P(X|buys_computer = “yes”) * P(buys_computer = “yes”) = 0.028
P(X|buys_computer = “no”) * P(buys_computer = “no”) = 0.007

Therefore, X belongs to class (“buys_computer = yes”)

Problème de la probabilité nulle

- La classification naïve bayésienne requiert que chaque probabilité conditionnelle soit non-nulle. Sinon, la probabilité prédite sera nulle !

$$P(\mathbf{X} \mid C_i) = \prod_{k=1}^{n} P(X_k \mid C_i)$$

- Exemple : soit un jeu de données de 1000 tuples, avec income=low (0), income= medium (990), et income = high (10),
- On utilise la correction laplacienne
  - On ajoute 1 à chaque cas
    - Pr(income = low) = 1/1003
    - Pr(income = medium) = 991/1003
    - Pr(income = high) = 11/1003
  - Ainsi, les estimations de probabilité « corrigées » sont proches de leur contreparties « brutes » ; en outre, elle ne sont jamais nulles

Remarques sur Naïve Bayes

- Avantages
  - Facile à coder
  - De bons résultats dans la plupart des cas
- Inconvénients
  - Hypothèse de fond : indépendance statistique des classes, ce qui rarement est vrai, et qui donc entraîne une perte de précision
  - Dans la réalité, il existe des dépendances entre les variables
    - Exemples : patients d'un hôpital : âge, familiarité, etc.
    - Symptômes: fièvre, tous, etc. ; Maladie : tumeur, diabète, etc.
    - Ces dépendances ne peuvent pas être modélisées par un classificateur naïf bayésien
- Comment gérer ces dépendances ?
  - Avec les **réseaux bayésiens** (mais on n'a pas le temps de les traites ici)

Règles SI-ALORS pour la classification

- On vise à représenter des connaissance sous forme de règles SI-ALORS (IF-THEN)

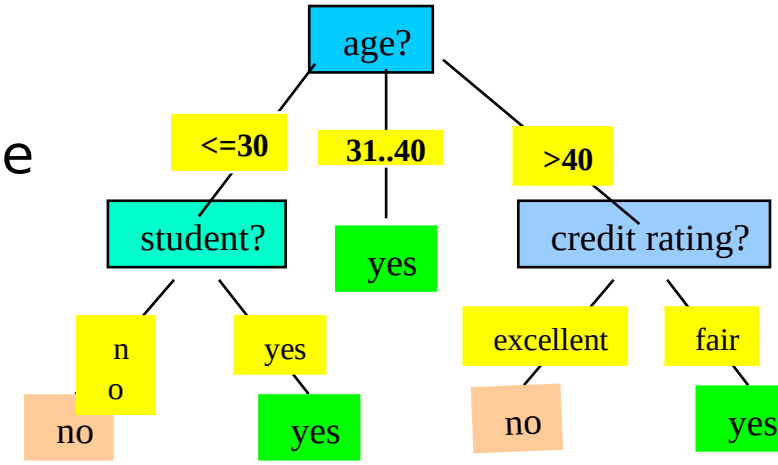
  R: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes

  - Antécédents - conséquent
- Évaluation d'une règle: *couverture* et *précision*
  - n_{covers} = nombre de tuples couvert par R
  - $n_{correct}$ = nombre de tuples correctement classifié par R

  coverage(R) = n_{covers} /|D| /* D: jeu de données d'entraînement */

  accuracy(R) = $n_{correct}$ / n_{covers}
- Si plus d'une règle est déclenchée, il faut **résoudre le conflit**
  - Tri par taille : on affecte la plus haute priorité aux règles ayant les conditions les plus strictes (c-à-d, celles qui *testent le plus d'attributs*)
  - Tri par classe : on donne plus de priorité aux règles qui reconnaissent les classes où une erreur coûterait plus cher
  - On se base sur l'ordre propre des règles (**liste de décision**): les règles sont organisées en une liste de priorité, suivant quelque mesure de qualité des règles ou le jugement des experts

Extraction de règles d'un arbre de décision



- Les règles sont plus faciles à comprendre que des gros arbres
- Une règle è créée pour chaque chemin de la racine à une feuille
- Les couples attribut-valeur le long d'un chemin sont la condition ; la feuille donne la classe
- Règles mutuellement exclusives et exhaustives

- Exemple: on extrait les règles à partir de l'arbre de décision *buys_computer* :

IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*

IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*

IF *age* = mid-age THEN *buys_computer* = *yes*

IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *yes*

IF *age* = young AND *credit_rating* = *fair* THEN *buys_computer* = *no*

Extraction de règles à partir des données

- Algorithmes de couverture séquentielle : extraction des règles directement à partir des données d'entraînement : FEUILLE, AQ, CN2, RIPPER
- Les règles sont apprises de manière séquentielle, chacune pour une classe donnée. Une règle pour Ci couvrira de nombreux tuples du Ci mais aucun (ou peu) des tuples des autres classes
- Étapes :
  - Les règles sont apprises une à la fois
  - Chaque fois qu'une règle est apprise, les tuples couverts par les règles sont supprimés
  - Le processus se répète sur les tuples restants, sauf condition d'arrêt, par exemple lorsqu'il n'y a plus d'exemples d'entraînement ou lorsque la qualité d'une règle renvoyée est inférieure à un seuil spécifié par l'utilisateur
- Comparaison avec l'induction d'arbres de décision : apprendre un ensemble de règles simultanément

Learn-One-Rule

- On commence par la règle la plus générale possible : condition = vide
- On ajoute de nouveaux attributs en adoptant une stratégie gourmande de profondeur d'abord
  - Choisir celui qui améliore le plus la qualité de la règle
  - Mesures de qualité des règles : tenir compte à la fois de la couverture et de la précision
- Foil-gain (dans FOIL & RIPPER) : évalue l'info_gain en prolongeant l'état

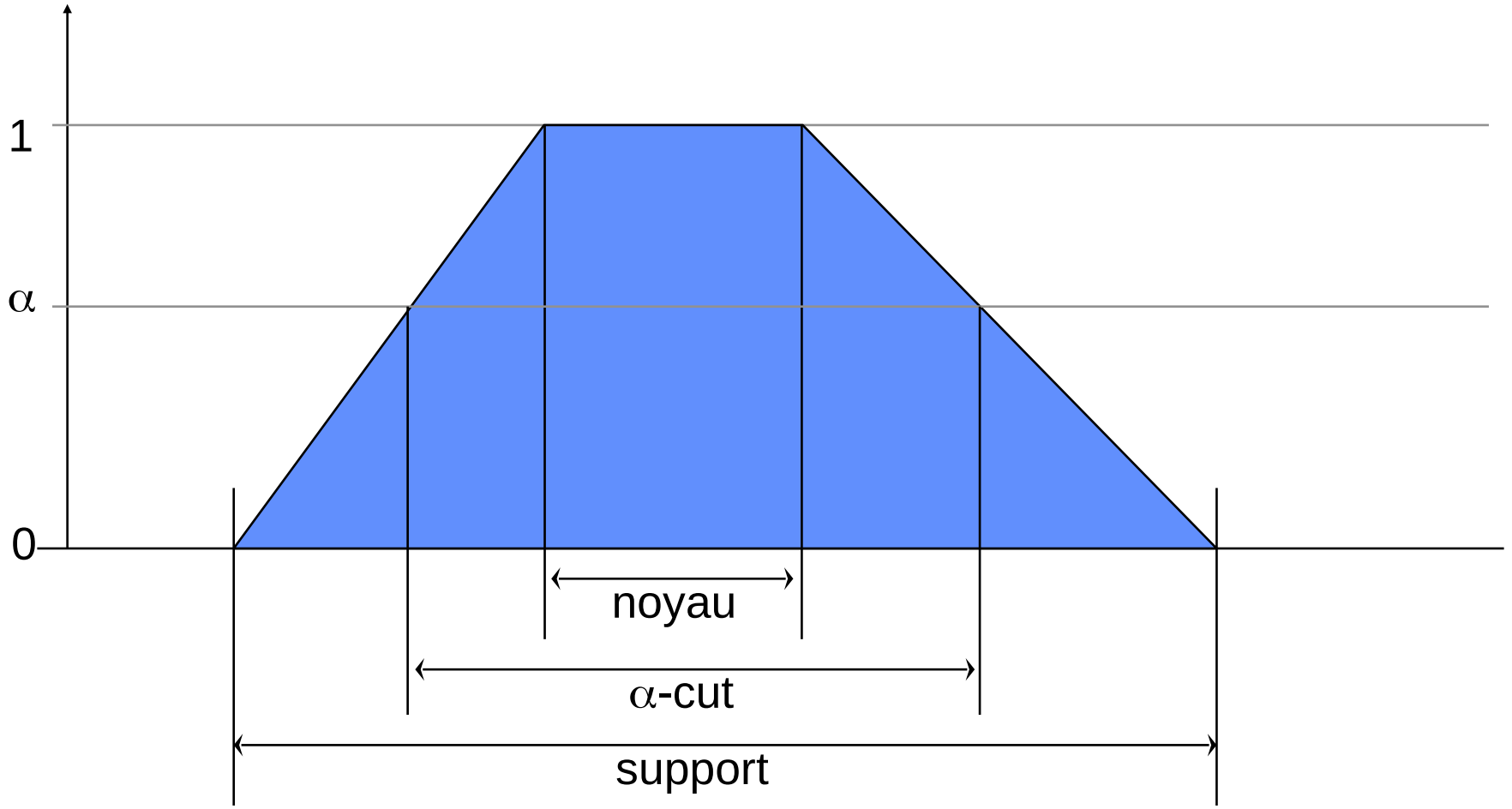
$$FOIL\_Gain = pos' \times (\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg})$$

  - Il favorise les règles qui ont une grande précision et couvrent de nombreux tuples positifs
- La taille des règles basée sur un ensemble indépendant de tuples tests

$$FOIL\_Prune(R) = \frac{pos - neg}{pos + neg}$$

  - Pos/neg sont le nombre de tuples positifs/négatifs couverts par R.
  - Si FOIL_Prune est plus élevé pour la version élaguée de R, élaguer R

Ensembles flous

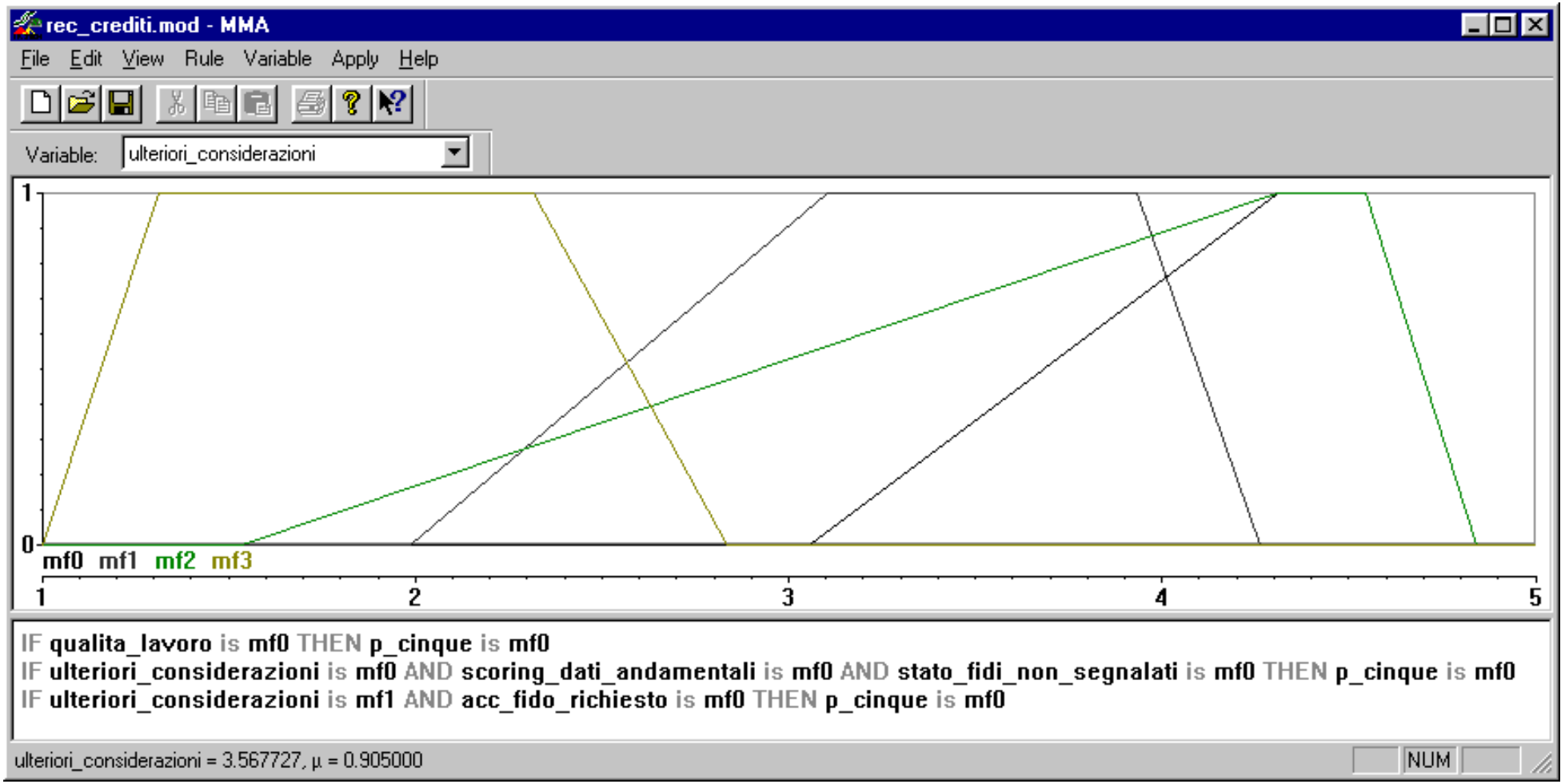


Operations sur les ensembles flous

- Extension des operations sur les ensembles classiques
- Normes et co-normes triangulaires
- Min et max sont un choix populaire

$$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\};$$
$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\};$$
$$\mu_{\bar{A}}(x) = 1 - \mu_A(x).$$

Systèmes de règles floues



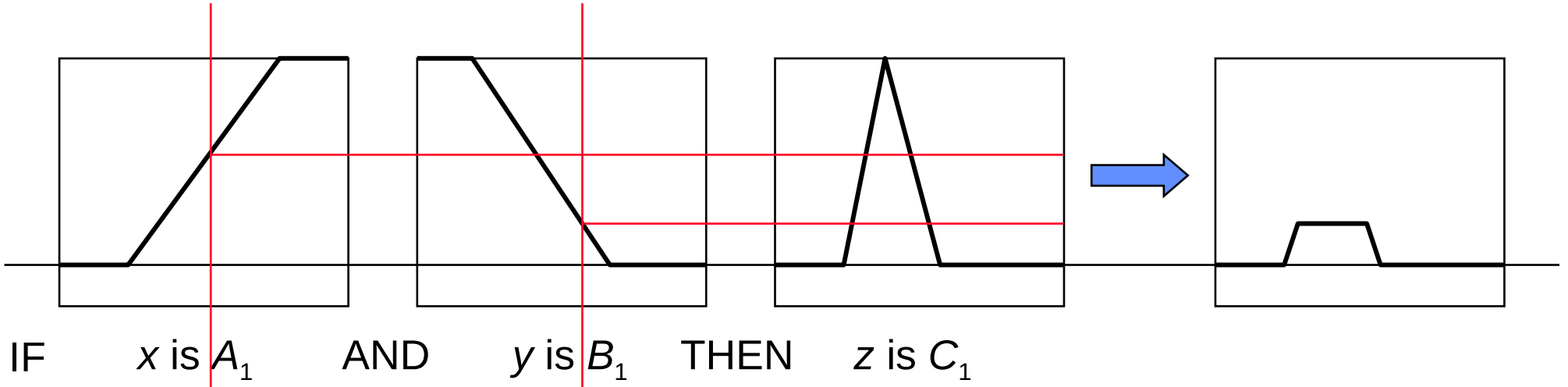
Inférence dans les systèmes de règles floues

Étant donné un ensemble de règles

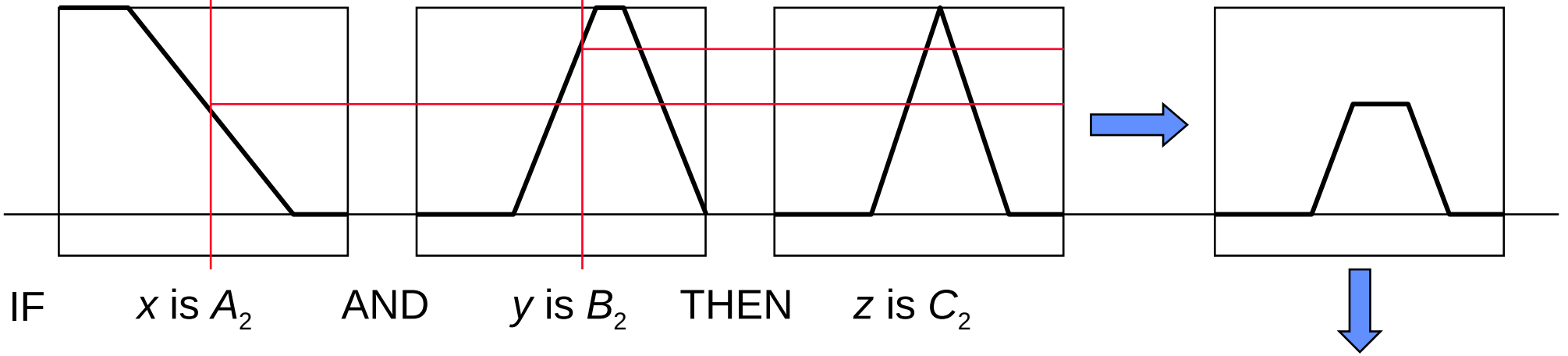
$$\begin{aligned} &\text{IF } P_1(x_1,\ldots,x_n) \text{ THEN } Q_1(y_1,\ldots,y_m),\\ &\qquad\vdots \qquad\qquad\qquad\vdots\\ &\text{IF } P_r(x_1,\ldots,x_n) \text{ THEN } Q_r(y_1,\ldots,y_m), \end{aligned}$$

L’ensemble flou des valeurs des variables dépendantes est

$$\begin{aligned} &\tau_R(y_1,\ldots,y_m;x_1,\ldots,x_n)\\ &\quad=\sup_{1\le i\le r}\min\{\tau_{Q_i}(y_1,\ldots,y_m),\tau_{P_i}(x_1,\ldots,x_n)\}. \end{aligned}$$



IF x is A_1 AND y is B_1 THEN z is C_1



IF x is A_2 AND y is B_2 THEN z is C_2