Analyse des Données

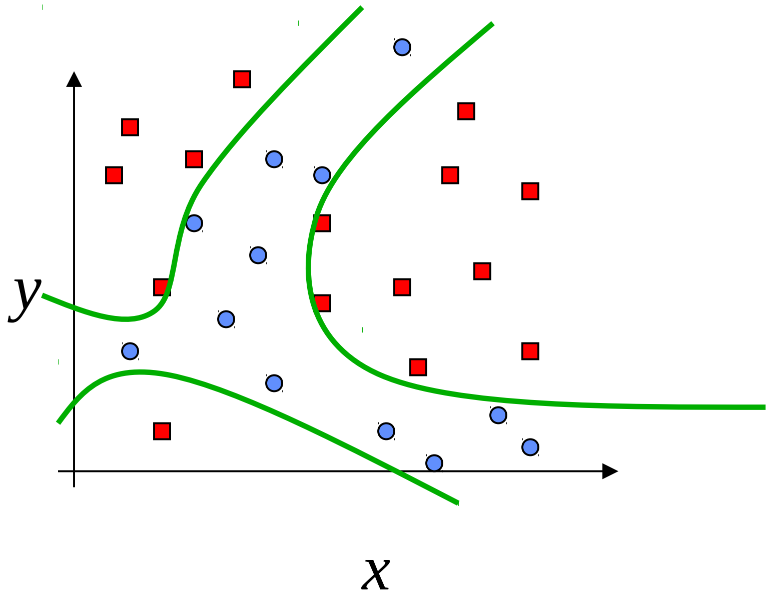
Master 2 IMAFA

Andrea G. B. Tettamanzi
Université Nice Sophia Antipolis
UFR Sciences - Département Informatique
andrea.tettamanzi@unice.fr

Séance 3

Classification and Prediction

Modélisation



prédiction

$z = M(x, y)$

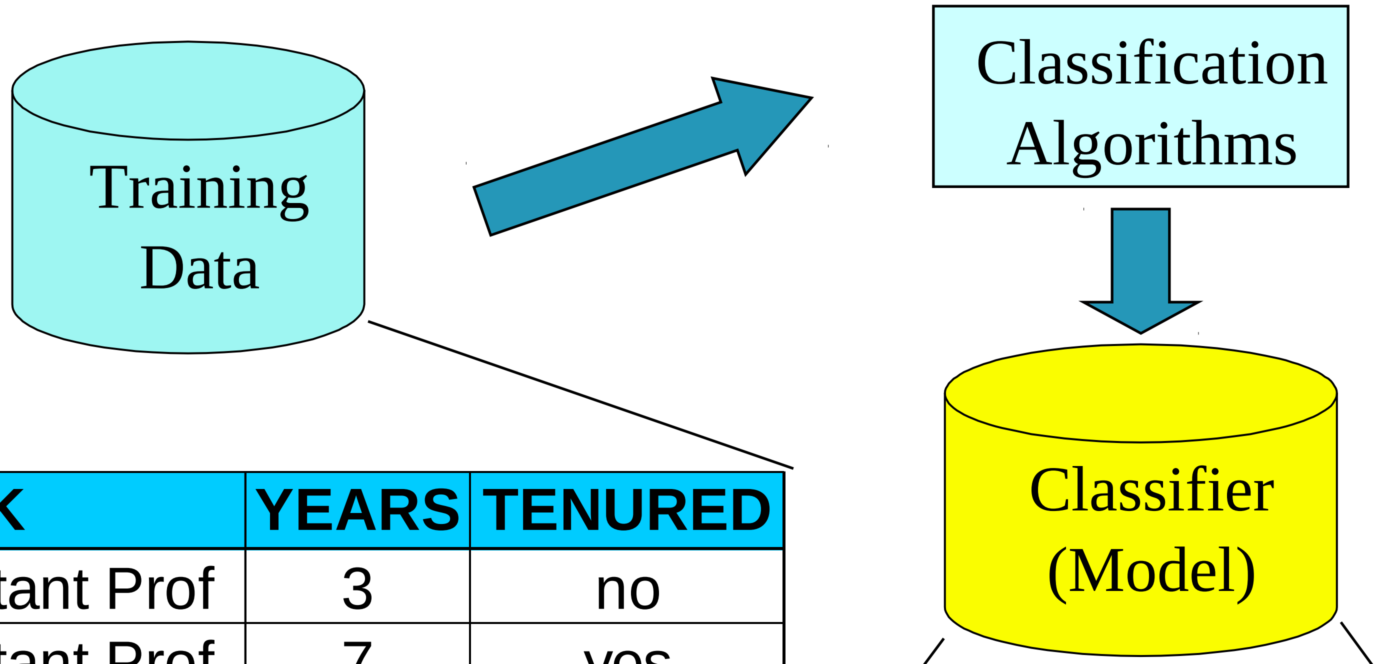
Variables connues

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. Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit approval
  - Target marketing
  - Medical diagnosis
  - Fraud detection

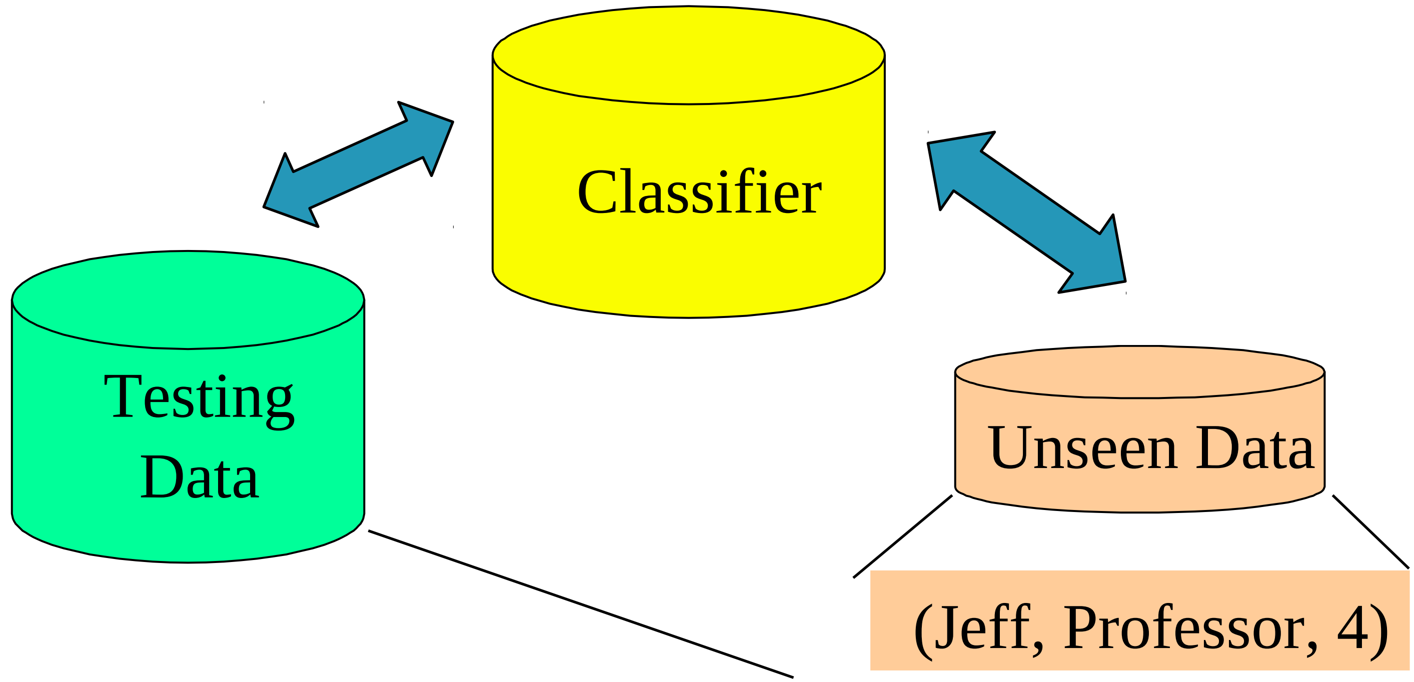
Step 1: Model Construction



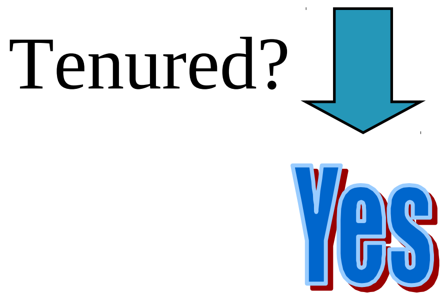
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Step 2: Using the Model in Prediction



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



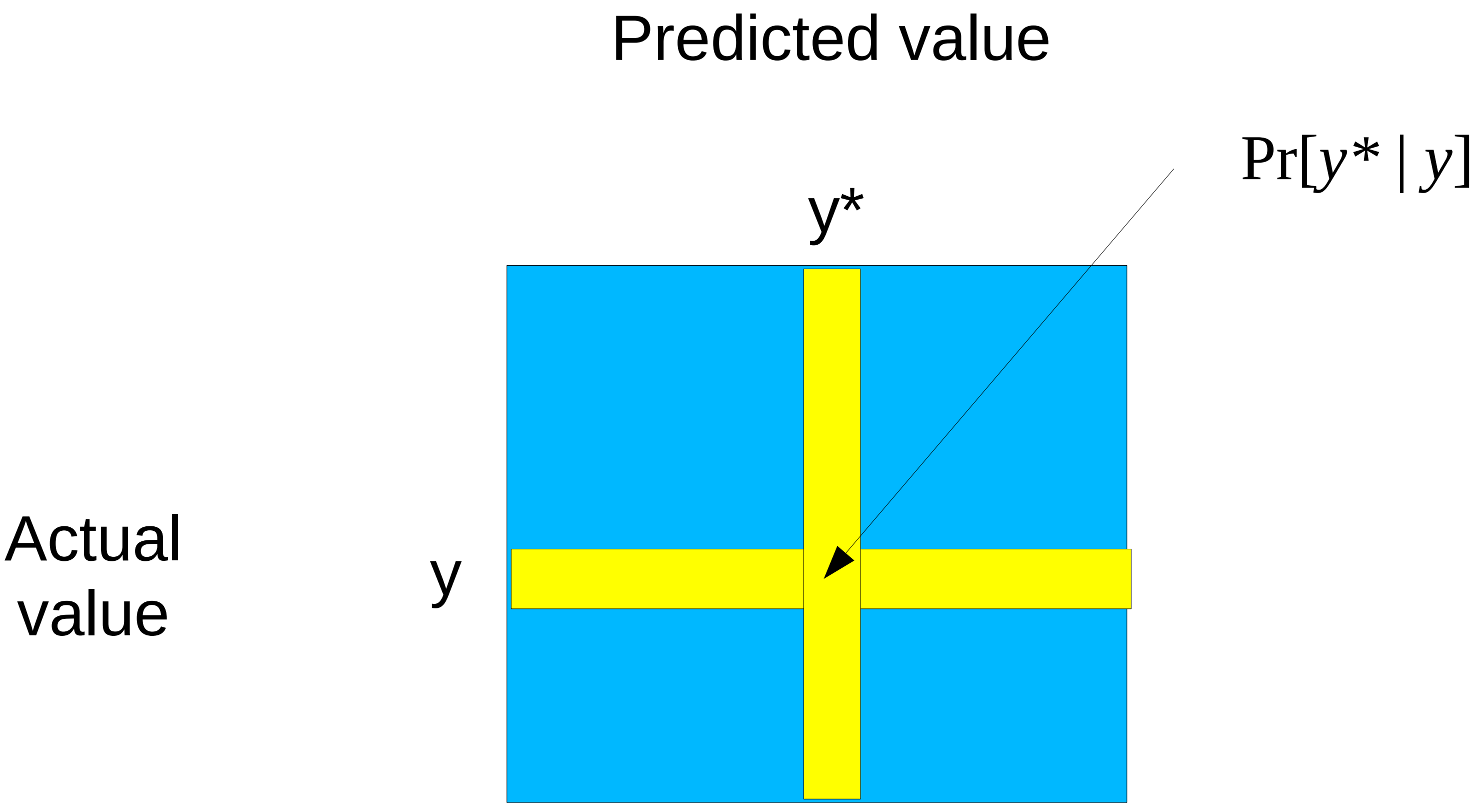
Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Evaluating Classification Methods

- Accuracy
  - classifier accuracy: predicting class label
  - predictor accuracy: guessing value of predicted attributes
  - More sophisticated measures: Confusion matrix, ROC curve
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Confusion Matrix



Classifier Accuracy Measures

	C_1	C_2
C_1	True positive	False negative
C_2	False positive	True negative

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

- Accuracy of a classifier *M*, acc(*M*): percentage of test set tuples that are correctly classified by the model *M*
  - Error rate (misclassification rate) of *M* = 1 – acc(*M*)
  - Given *m* classes, $CM_{i,j}$, an entry in a confusion matrix, indicates # of tuples in class i that are labeled by the classifier as class j
- Alternative accuracy measures (e.g., for cancer diagnosis)
  - sensitivity = t-pos/pos /* true positive recognition rate */
  - specificity = t-neg/neg /* true negative recognition rate */
  - precision = t-pos/(t-pos + f-pos)
  - accuracy = sensitivity * pos/(pos + neg) + specificity * neg/(pos + neg)
  - This model can also be used for cost-benefit analysis

Predictor Error Measures

- Measure predictor accuracy: measure how far off the predicted value is from the actual known value
- **Loss function**: measures the error b/w y_i and the predicted value y_i'
  - Absolute error: $|y_i - y_i'|$
  - Squared error: $(y_i - y_i')^2$
- Test error (generalization error): the average loss over the test set
  - Mean abs error: $\frac{1}{d}\sum_{i=1}^{d}|y_i - y_i'|$ Mean squared error: $\frac{1}{d}\sum_{i=1}^{d}(y_i - y_i')^2$
  - Relative abs error: $\frac{\sum_{i=1}^{d}|y_i - y_i'|}{\sum_{i=1}^{d}|y_i - \bar{y}|}$ Relative sq error: $\frac{\sum_{i=1}^{d}(y_i - y_i')^2}{\sum_{i=1}^{d}(y_i - \bar{y})^2}$

The mean squared-error exaggerates the presence of outliers

Evaluating the Accuracy of a Classifier or Predictor (I)

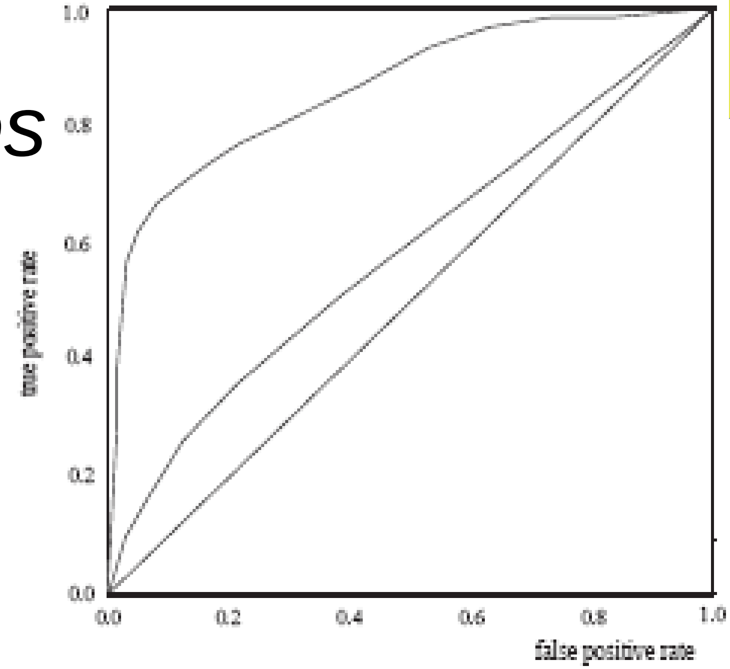
- Holdout method
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- Cross-validation (k-fold, where k = 10 is most popular)
  - Randomly partition the data into k mutually exclusive subsets, each approximately equal size
  - At i-th iteration, use Di as test set and others as training set
  - Leave-one-out: k folds where k = # of tuples, for small sized data
  - Stratified cross-validation: folds are stratified so that class distribution in each fold is approx. the same as that in the initial data

Evaluating the Accuracy of a Classifier or Predictor (II)

- Bootstrap
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several boostrap methods, and a common one is **.632 boostrap**
  - Suppose we are given a data set of d tuples. The data set is sampled d times, with replacement, resulting in a training set of d samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data will end up in the bootstrap, and the remaining 36.8% will form the test set (since $(1 – 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedue k times, overall accuracy of the model:

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

Model Selection: ROC Curves



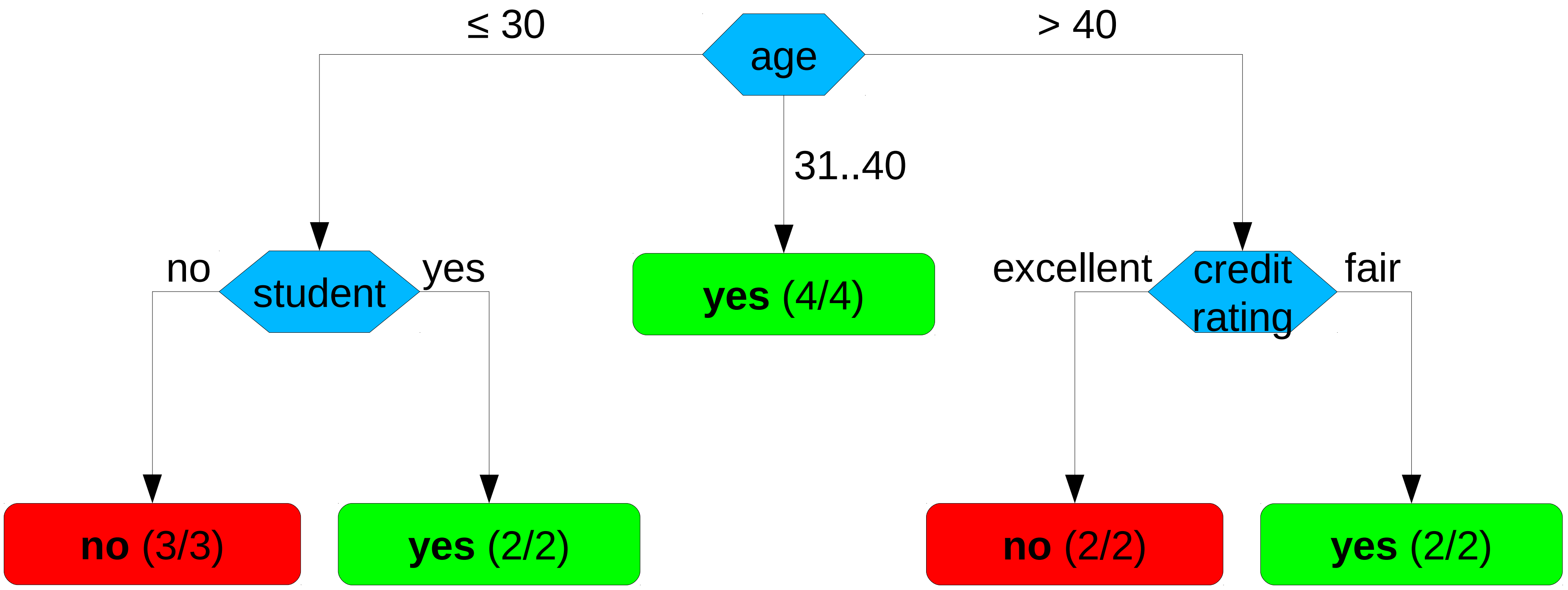
- ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model

•Vertical axis represents the true positive rate
•Horizontal axis rep. the false positive rate
•The plot also shows a diagonal line
•A model with perfect accuracy will have an area (AUC) of 1.0

Decision Tree Induction: Training Dataset

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

Output: A Decision Tree for “buys_computer”



Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i, D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

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

- Information needed (after using A to split D into v partitions) to classify D:

$$H_A(D) = \sum_{i=1}^{v} \frac{|D_j|}{|D|} \cdot H(D_j)$$

- Information gained by branching on attribute A:

$$\mathrm{Gain}(A) = H(D) - H_A(D)$$

Gini index (CART, IBM IntelligentMiner)

- If a data set D contains examples from n classes, gini index, $gini(D)$ is defined as

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

where p_j is the relative frequency of class j in D

- If a data set D is split on A into two subsets D_1 and D_2, the *gini* index $gini(D)$ is defined as

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

- Reduction in Impurity:

$$\Delta\mathrm{gini}(A) = \mathrm{gini}(D) - \mathrm{gini}_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node

Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on $\chi 2$ test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistics: has a close approximation to $\chi 2$ distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

Bayesian Classification: Why?

- **A statistical classifier:** performs probabilistic prediction, i.e., predicts class membership probabilities
- **Foundation:** Based on Bayes' Theorem.
- **Performance:** A simple Bayesian classifier, naïve Bayes classifier, has comparable performance with decision tree and selected neural network classifiers
- **Incremental:** Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- **Standard:** Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayes' Theorem: Basics

- Let X be a data sample ("evidence"): class label is unknown
- Let H be a hypothesis that X belongs to class C
- Classification is to determine $P(H|X)$, the probability that the hypothesis holds given the observed data sample X
- $P(H)$ (prior probability), the initial probability
  - E.g., X will buy computer, regardless of age, income, …
- P(X): probability that sample data is observed
- $P(X|H)$ (posteriori probability), the probability of observing the sample X, given that the hypothesis holds
  - E.g., Given that X will buy computer, the prob. that X is 31..40, medium income

Bayes' Theorem

- Given training data **X**, posterior probability of hypothesis *H*, P(*H*|**X**), follows Bayes' Theorem

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

Informally, this can be written as

posterior = likelihood x prior/evidence

- Predicts **X** belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes
- Practical difficulty: requires initial knowledge of many probabilities, significant computational cost

Towards Naïve Bayes Classifiers

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector X = (x1, x2, …, xn)
- Suppose there are m classes C1, C2, …, Cm.
- Classification is to derive the maximum posteriori, i.e., the maximal P(Ci|X)

- This can be derived from Bayes' theorem

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

- Since P(X) is constant for all classes, we need only maximize

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

Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

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

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical, $P(X_k|C_i)$ is the # of tuples in C_i having value X_k for A_k divided by $|C_{i, D}|$ (# of tuples of C_i in D)
- If A_k is continous-valued, $P(X_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

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

and $P(x_k|C_i)$ is

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

Naïve Bayesian Classifier: Training Dataset

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

Data sample
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

age	income	student	credit_rating	_comp
<=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

Naïve Bayesian Classifier: An Example

- $P(C_i)$: 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”)

Avoiding the 0-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

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

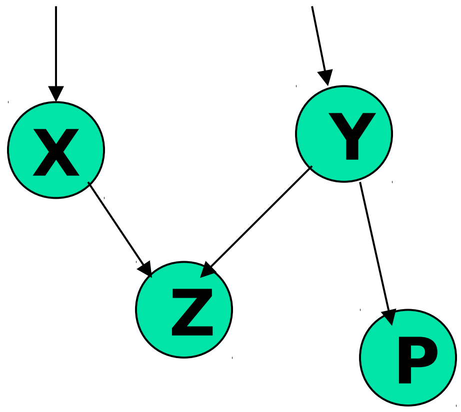
- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10),
- Use Laplacian correction (or Laplacian estimator)
  - Adding 1 to each case
    - Prob(income = low) = 1/1003
    - Prob(income = medium) = 991/1003
    - Prob(income = high) = 11/1003
  - The “corrected” prob. estimates are close to their “uncorrected” counterparts

Naïve Bayesian Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.
    - Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

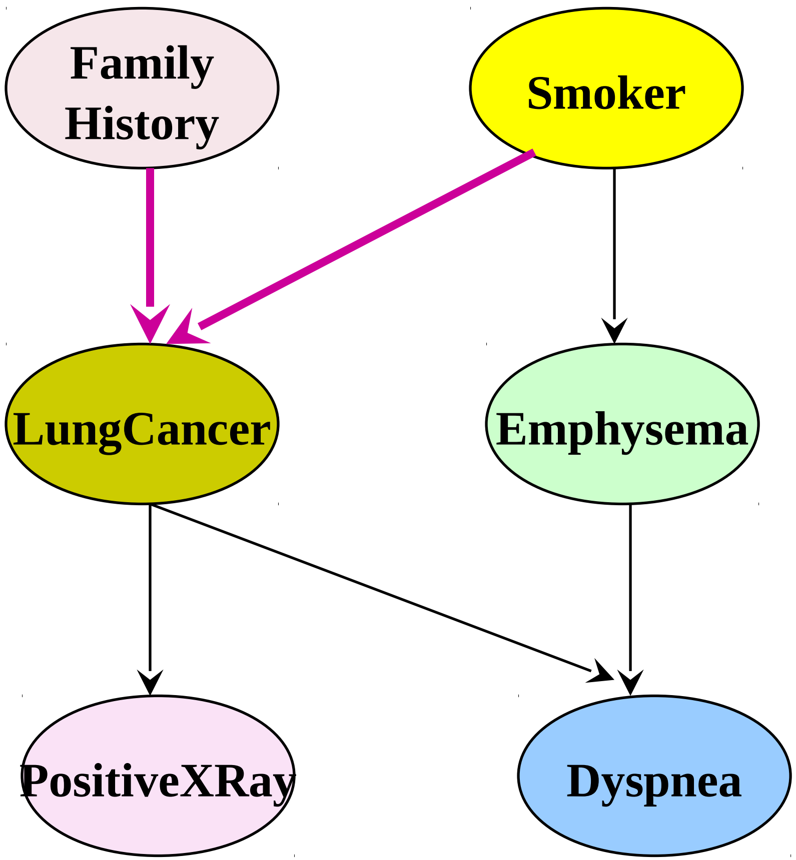
Bayesian Belief Networks

- Bayesian belief network allows a *subset* of the variables conditionally independent
- A graphical model of causal relationships
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution



- ❑ Nodes: random variables
- ❑ Links: dependency
- ❑ X and Y are the parents of Z, and Y is the parent of P
- ❑ No dependency between Z and P
- ❑ Has no loops or cycles

Bayesian Belief Network: An Example



The **conditional probability table** (**CPT**) for variable LungCancer:

	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

CPT shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of **X**, from CPT:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(Y_i))$$

Training Bayesian Networks

- Several scenarios:
  - Given both the network structure and all variables observable: learn only the CPTs
  - Network structure known, some hidden variables: gradient descent (greedy hill-climbing) method, analogous to neural network learning
  - Network structure unknown, all variables observable: search through the model space to reconstruct network topology
  - Unknown structure, all hidden variables: No good algorithms known for this purpose
- Ref. D. Heckerman: Bayesian networks for data mining

Using IF-THEN Rules for Classification

- Represent the knowledge in the form of IF-THEN rules

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

  – Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: *coverage* and *accuracy*
  – n_{covers} = # of tuples covered by R

  – $n_{correct}$ = # of tuples correctly classified by R

  coverage(R) = n_{covers} /|D| /* D: training data set */

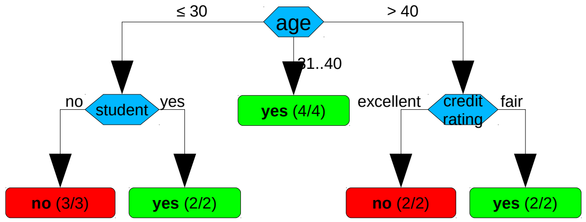
  accuracy(R) = $n_{correct}$ / n_{covers}
- If more than one rule is triggered, need **conflict resolution**
  – Size ordering: assign the highest priority to the triggering rules that has the "toughest" requirement (i.e., with the *most attribute test*)

  – Class-based ordering: decreasing order of *prevalence or misclassification cost per class*

  – Rule-based ordering (**decision list**): rules are organized into one long priority list, according to some measure of rule quality or by experts

Rule Extraction from a Decision Tree

- Rules are easier to understand than large trees
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive



- Example: Rule extraction from our *buys_computer* decision-tree

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*

Rule Extraction from the Training Data

- Sequential covering algorithm: Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned sequentially, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Steps:
  - Rules are learned one at a time
  - Each time a rule is learned, the tuples covered by the rules are removed
  - The process repeats on the remaining tuples unless termination condition, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold
- Comp. w. decision-tree induction: learning a set of rules simultaneously

How to Learn-One-Rule?

- Start with the most general rule possible: condition = empty
- Adding new attributes by adopting a greedy depth-first strategy
  - Picks the one that most improves the rule quality
- Rule-Quality measures: consider both coverage and accuracy
  - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition

$$\mathrm{FOIL\_Gain} = pos' \cdot \left(\log_2 \frac{pos'}{pos' + neg'} - \log_2 \frac{pos}{pos + neg} \right)$$

    - It favors rules that have high accuracy and cover many positive tuples
- Rule pruning based on an independent set of test tuples

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

    - Pos/neg are # of positive/negative tuples covered by R.
    - If FOIL_Prune is higher for the pruned version of R, prune R

Ensembles Flous

- Un ensemble « classique » est complètement spécifié par une fonction caractéristique $\chi : U \rightarrow \{0, 1\}$, telle que, pour tout $x \in U$,
  - $\chi(x) = 1$, si et seulement si x appartient à l'ensemble
  - $\chi(x) = 0$, autrement.
- Pour définir un ensemble « flou », on remplace χ par une fonction d'appartenance $\mu : U \rightarrow [0, 1]$, telle que, pour tout $x \in U$,
  - $0 \leq \mu(x) \leq 1$ est le degré auquel x appartient à l'ensemble
- Puisque la fonction μ spécifie complètement l'ensemble, on peut dire que μ « est » l'ensemble
- Un ensemble classique est un cas particulier d'ensemble flou !
- L'univers U est le référentiel de l'ensemble μ

Représentation

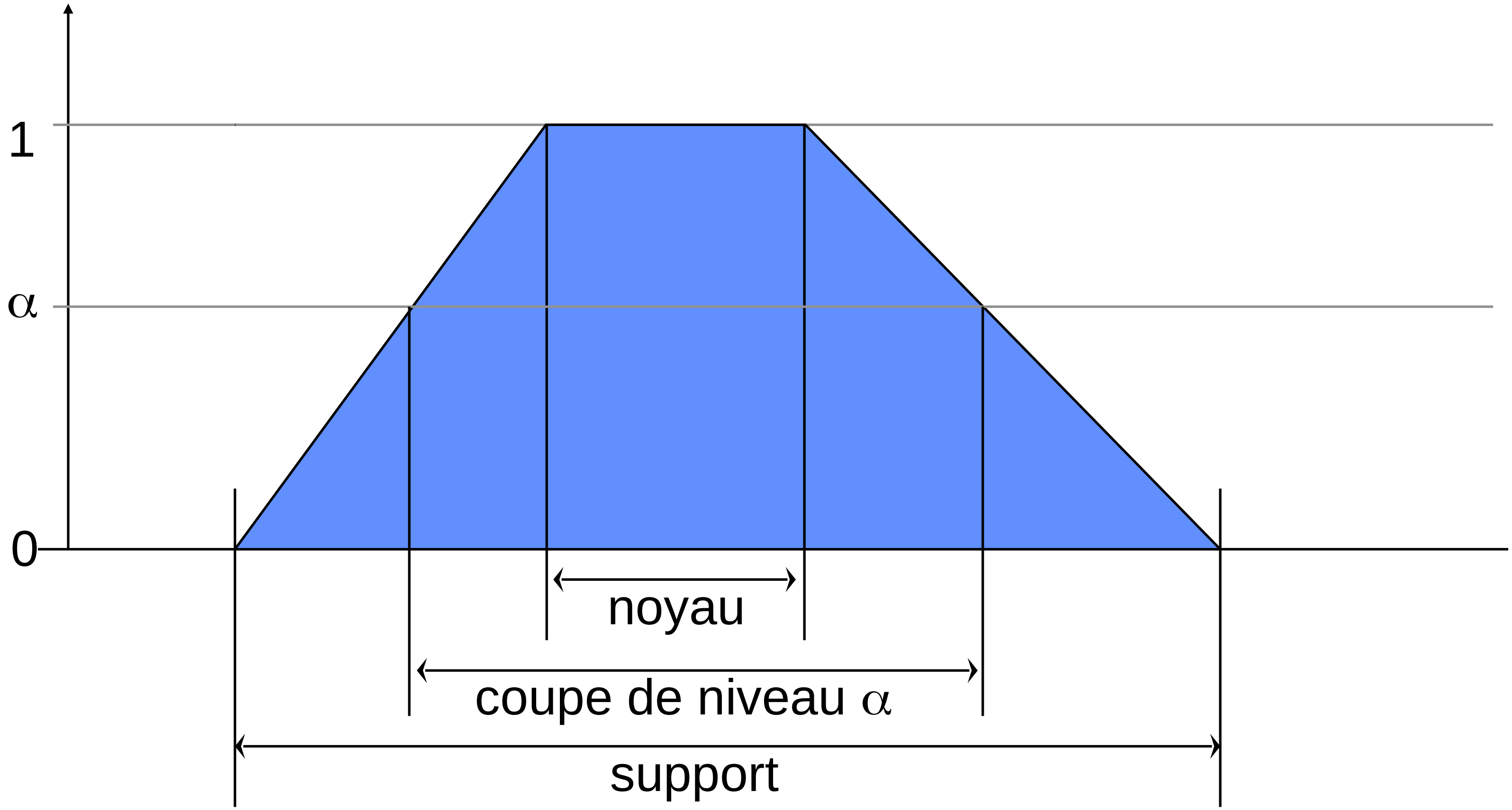
Référentiel fini : $A = \sum_{x \in U} \frac{\alpha_x}{x}$

MarqueAutoSportive = 0.8/BMW + 1/Ferrari + 0/Fiat + 0.5/Mercedes + …

Référentiel infini : $A = \int_{x \in U} \frac{\mu(x)}{x}$

$$\text{Chaud} = \int_{t=-273,15}^{+\infty} \frac{1/(1 - e^{\lambda(20-t)})}{t}$$

Ensembles flous



Opérations sur les ensembles flous

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

$$\begin{aligned}(A \cup B)(x) &= \max\{A(x), B(x)\} \\ (A \cap B)(x) &= \min\{A(x), B(x)\} \\ \bar{A}(x) &= 1 - A(x)\end{aligned}$$

Systèmes de règles floues

- Variables et valeurs linguistiques

- Clause floue :

  X is *A*

- Règle :

  IF *antécendant* THEN *conséquant*

- Méthodes de « déflouification »

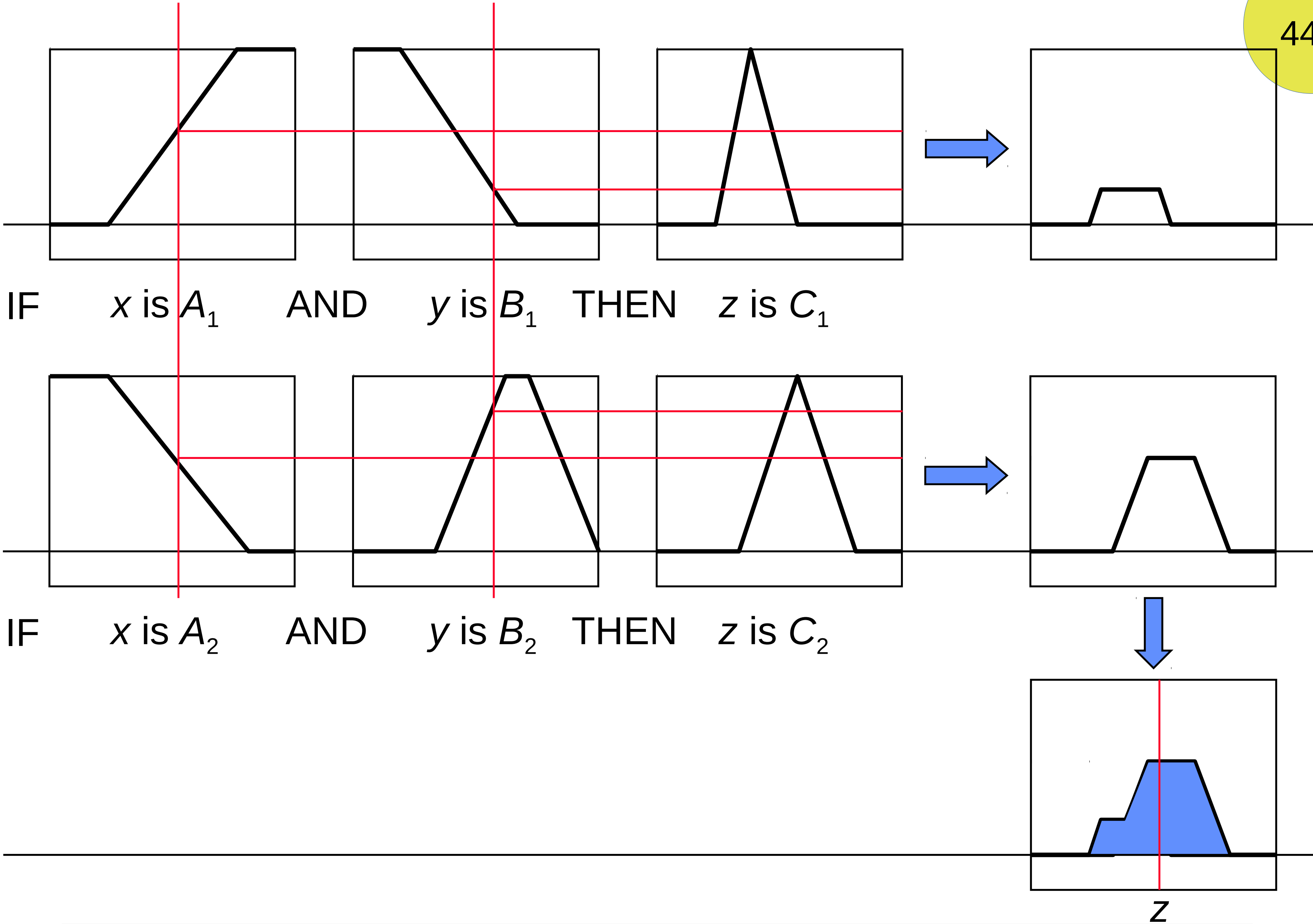
Inférence dans les systèmes de règles floues

Soit un ensemble de règles

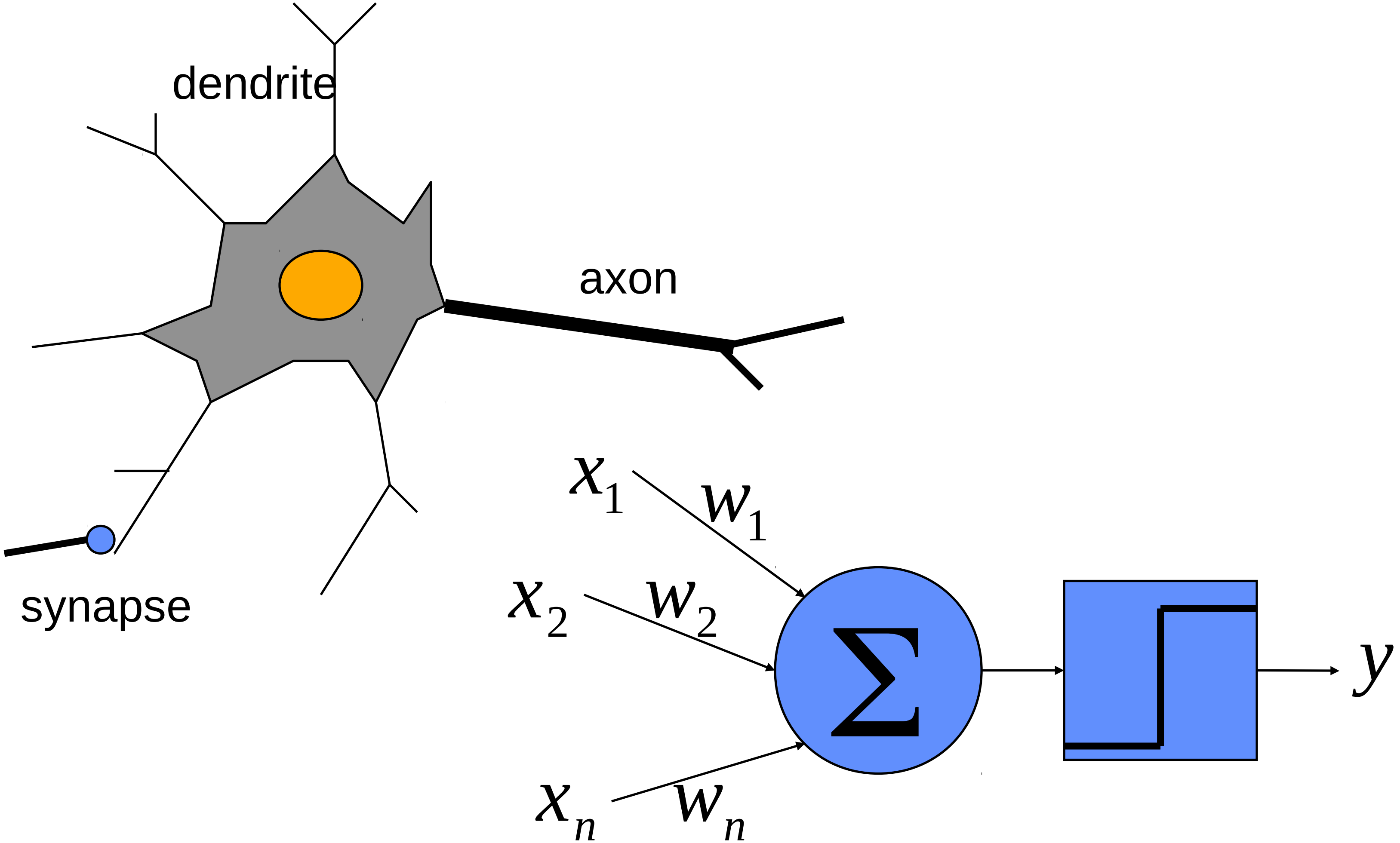
$$\begin{array}{ll} \text{IF } P_1(x_1, \ldots, x_n) & \text{THEN } Q_1(y_1, \ldots, y_m), \\ \vdots & \vdots \\ \text{IF } P_r(x_1, \ldots, x_n) & \text{THEN } Q_r(y_1, \ldots, y_m), \end{array}$$

L'ensemble flou des valeurs des variables dépendantes est :

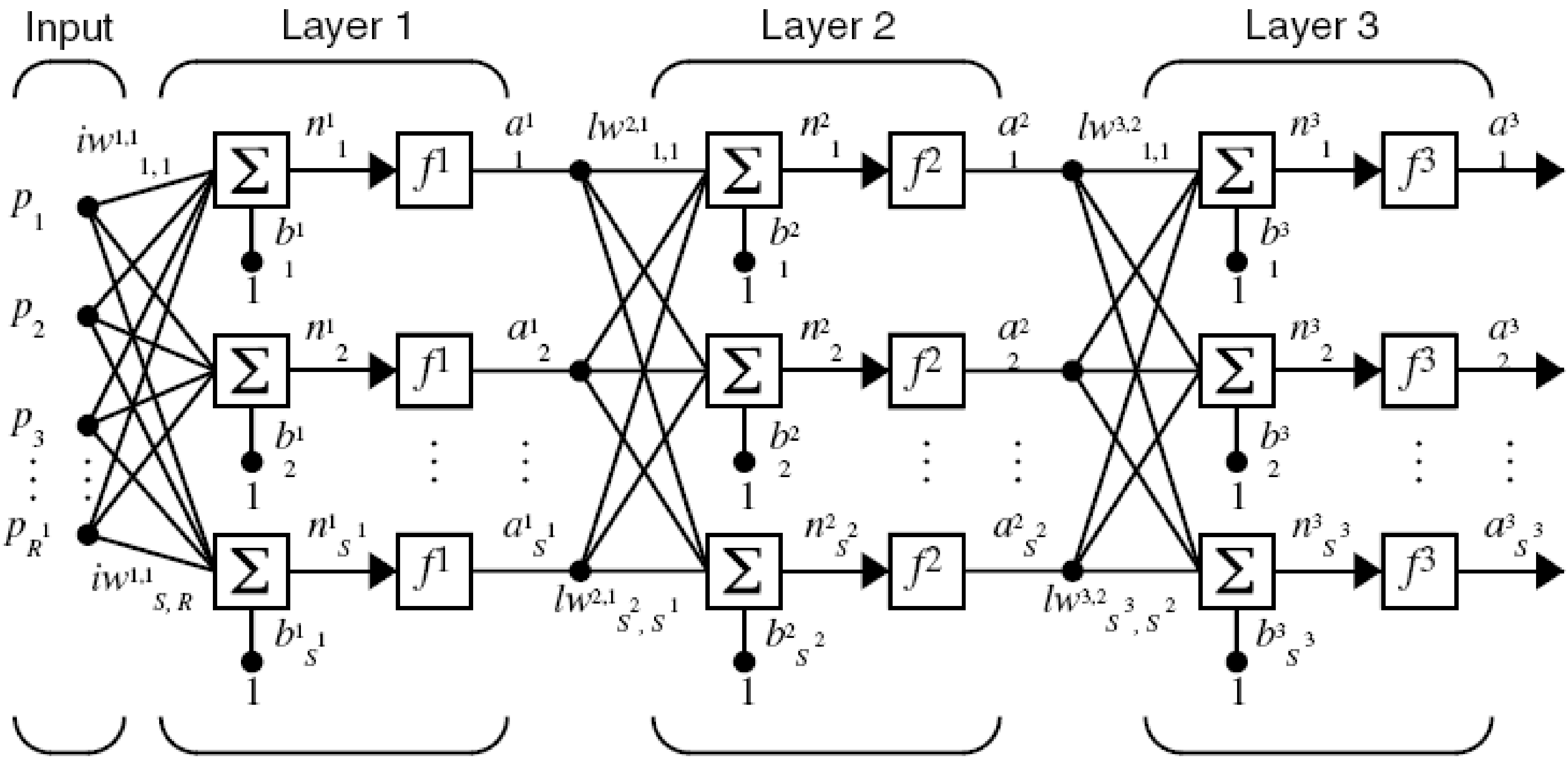
$$\begin{aligned} &\tau_R(y_1, \ldots, y_m; x_1, \ldots, x_n) \\ &\quad = \sup\nolimits_{1 \le i \le r} \min\{\tau_{Q_i}(y_1, \ldots, y_m), \tau_{P_i}(x_1, \ldots, x_n)\}. \end{aligned}$$


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
z

Réseaux de Neurones Artificiaux



Réseau Feed-Forward



$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{a}^1 + \mathbf{b}^2) \qquad \mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{a}^2 + \mathbf{b}^3)$$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\,\mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3)$$

Neural Network as a Classifier

- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or ``structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of ``hidden units" in the network
- Strength
  - High tolerance to noisy data
  - Ability to classify untrained patterns
  - Well-suited for continuous-valued inputs and outputs
  - Successful on a wide array of real-world data
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks

Backpropagation

- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to minimize the mean squared error between the network's prediction and the actual target value
- Modifications are made in the “backwards” direction: from the output layer, through each hidden layer down to the first hidden layer, hence “backpropagation”
- Steps
  - Initialize weights (to small random #s) and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

WEKA
The University of Waikato

- Collection d'algorithmes d'apprentissage automatique pour la fouille de données, open source
- Les algorithmes peuvent être utilisés comme ils sont ou appelés à partir d'un programme Java
- Weka contient des outils de
  - pré-élaboration de données
  - Classification
  - Régression
  - Regroupement (clustering)
  - Règles d'association
  - Visualisation
- Adapté pour développer des nouveaux algorithmes

Merci de votre attention