Classification et prédiction

Classification vs. Prédiction

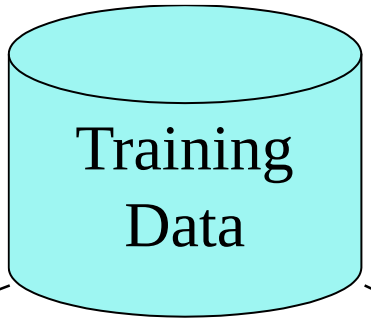
- Classification:
  - Classifier les données (construire un modèle) en se basant sur un ensemble où l'on connaît déjà l'association données-classes (*training set: ensemble d'apprentissage*)
- Prédiction:
  - Modéliser des valeurs connues pour prédire des valeurs inconnues

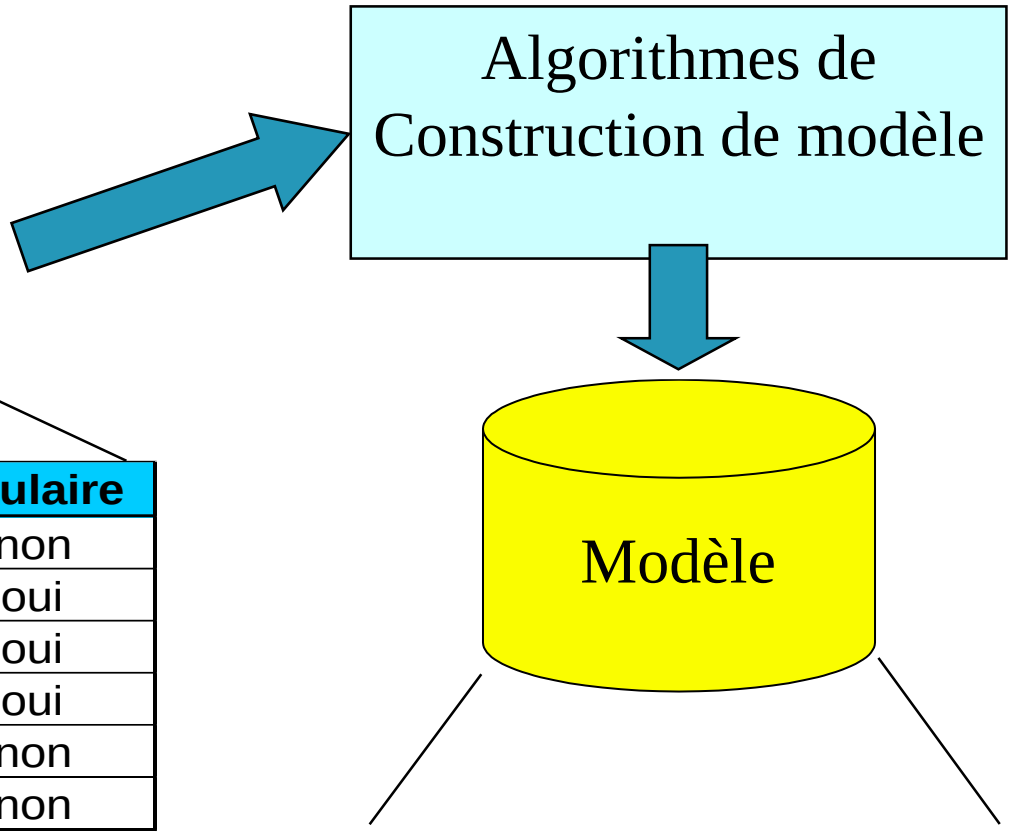
Classification— Processus à deux étapes

- Construction du modèle:
  - Chaque tuple (exemple) est supposé appartenir à une classe comme spécifié par le label de l'attribut "Classe"
  - Les données sont partagées en 2 sous ensembles
    - Le modèle (construit sur le 1er sous ensemble) est représenté par des règles de classification, arbres de décisions …
    - Estimer la pertinence sur le 2ème sous ensemble
      - Comparer les labels de classe de l'ensemble avec ce que prévoit le modèle
      - Le pourcentage de tuples qui sont correctement classifiés par le modèle donne une mesure de la précision

Construction du modèle (1):

Modèle qui *prédit* la valeur de l'attribut « Titulaire » en fonction des autres variables



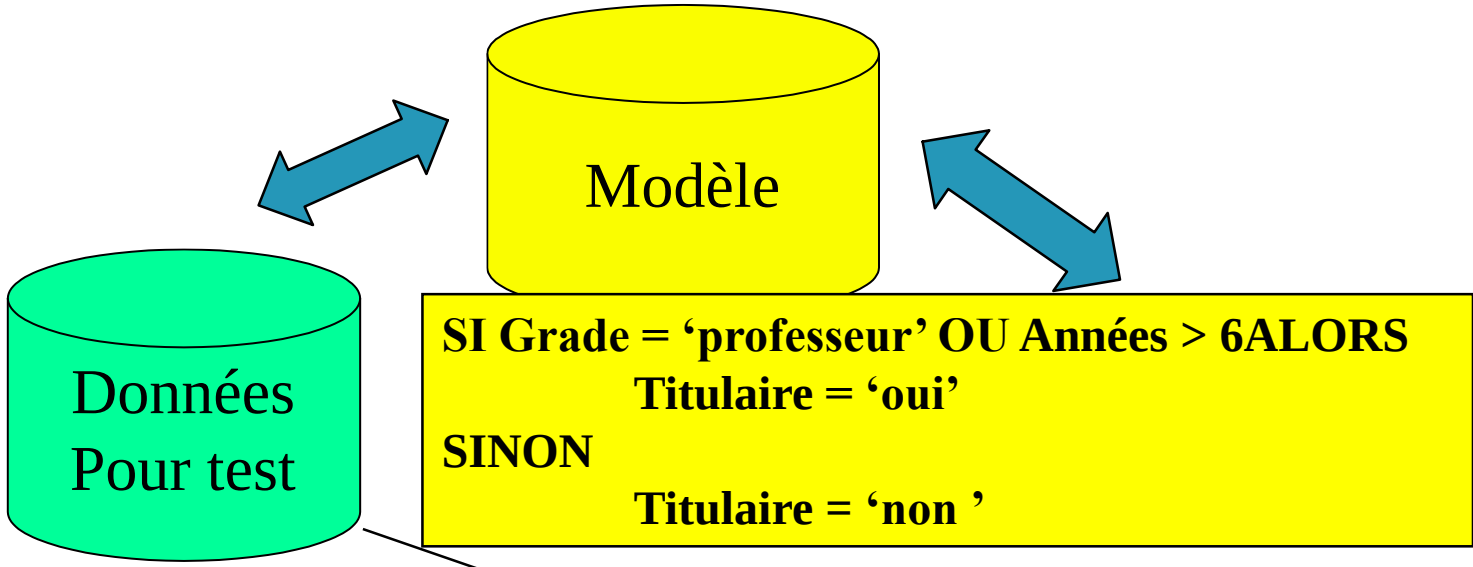


Nom	Grade	Années	Titulaire
David	Assistant	3	non
Marie	Assistant	7	oui
Jean	Professeur	2	oui
Jim	Prof. Associé	7	oui
Pierre	Assistant	6	non
Anne	Prof associé	3	non

SI Grade = 'professeur' OU Années > 6 ALORS
Titulaire = 'oui'
SINON
Titulaire = 'non '

Vérification de la pertinence (2):

Le modèle construit auparavant est testé sur le deuxième jeu de données



Tom	Assistant	2	non
Elise	Assistant	7	non
George	Professeur	5	oui
Josephe	Assistant	7	oui

Le modèle se trompe sur Elise et prédit la bonne valeur pour les autres:
Pertinence = 75%

Utilisation du modèle (3)

- (Pierre, Assistant, 15, ?)

SI Grade = ‘professeur’ OU Années > 6ALORS
Titulaire = ‘oui’
SINON
Titulaire = ‘non ’

- Le modèle prédit : Titulaire = Oui

Apprentissage Supervisé vs non supervisé

- Apprentissage Supervisé (classification)
  - Supervision: les données d'apprentissage (observations) sont accompagnés par les labels indiquant leurs classes
  - Les nouvelles données sont classifiées en se basant sur le training set
- Apprentissage non supervisé (regroupement)
  - Le label de classe des éléments observés (training set) n'est pas connu
  - Le but est de déceler l'existence de classes ou groupes dans les données

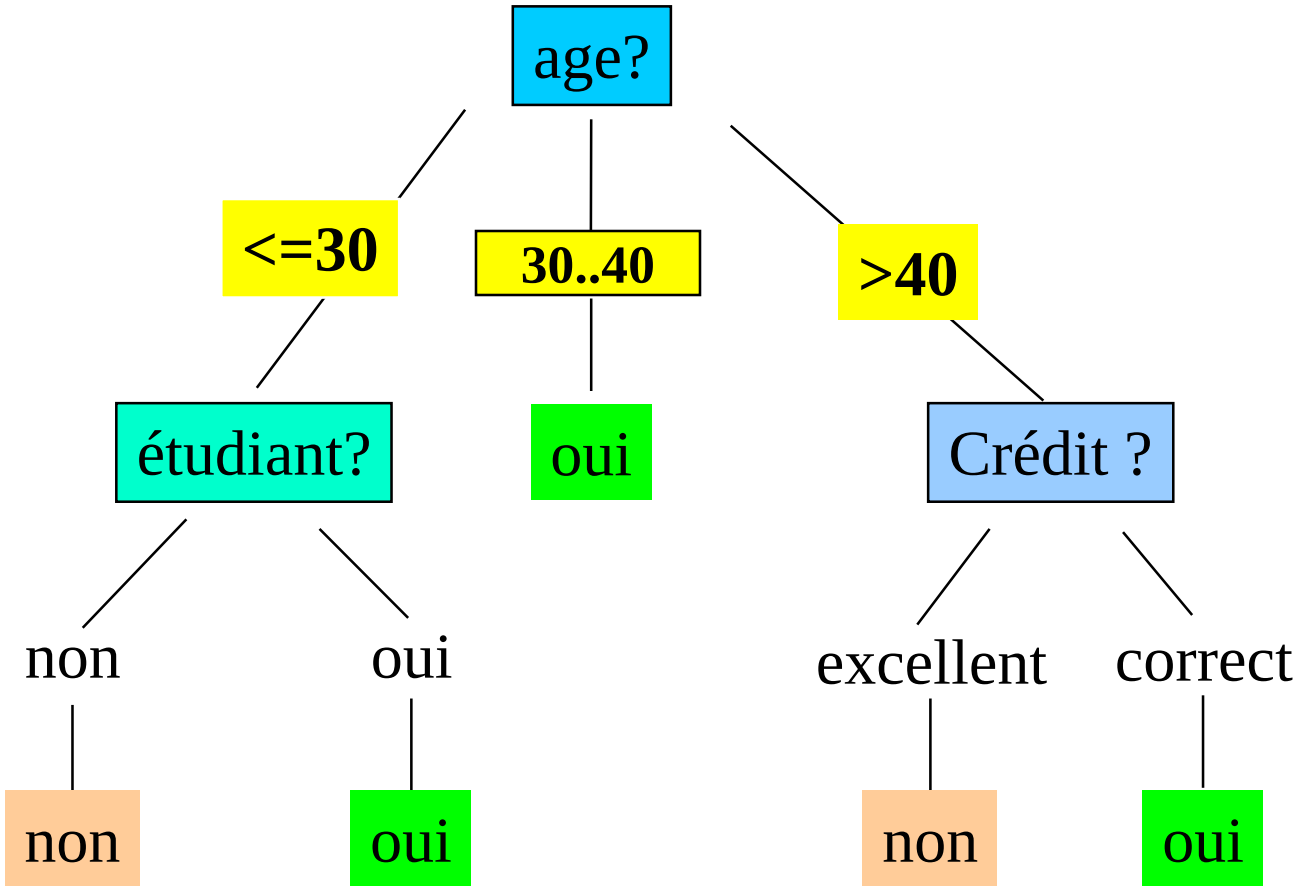
Classification avec arbres de décision

- Arbre de Décision
  - Les nœuds internes correspondent à des tests
  - Un arc correspond au résultat d'un test
  - Les nœuds feuilles représentent des classes
- La génération se fait en 2 phases
  - Construction de l'arbre
    - Au début tous les tuples se trouvent sur la racine
    - Partitionner les tuples récursivement en se basant à chaque fois sur un attribut sélectionné
  - Simplification de l'arbre
    - Identifier et supprimer les branches qui correspondent à des exceptions
- Utilisation:
  - Tester les attributs du tuple par rapport à l'arbre pour trouver la branche et qu'il satisfait donc sa classe

Training set

age	salaire	etudiant	crédit	achète_ordinateur
<=30	élévé	non	correct	non
<=30	élévé	non	excellent	non
30…40	élévé	non	correct	oui
>40	moyen	non	correct	oui
>40	faible	oui	correct	oui
>40	faible	oui	excellent	non
31…40	faible	oui	excellent	oui
<=30	moyen	non	correct	non
<=30	faible	oui	correct	oui
>40	moyen	oui	correct	oui
<=30	moyen	oui	excellent	oui
31…40	moyen	non	excellent	oui
31…40	élévé	oui	correct	oui
>40	moyen	non	excellent	non

Output: Un arbre de décision pour "achète_ordinateur"



Création de l'arbre de décision

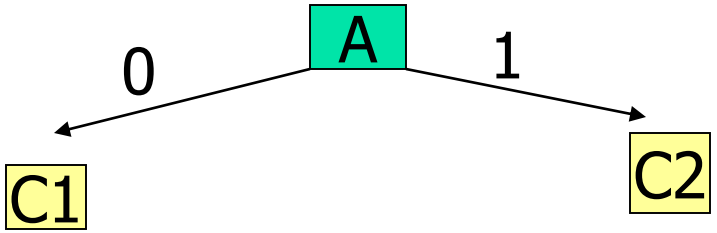
- L'arbre est construit top-down récursivement
  - Au début, tous les tuples sont sur la racine
  - Les attributs sont qualitatifs (discrétisation s'il le faut)
  - Les tuples sont ensuite partitionnés en fonction de l'attribut sélectionné
  - L'attribut de test est sélectionné en utilisant des heuristiques ex: gain informationnel (on y reviendra)
- Conditions d'arrêt du partitionnement
  - Tous les tuples d'un nœud se trouvent dans la même classe

Choix de l'attribut de partionnement (1)

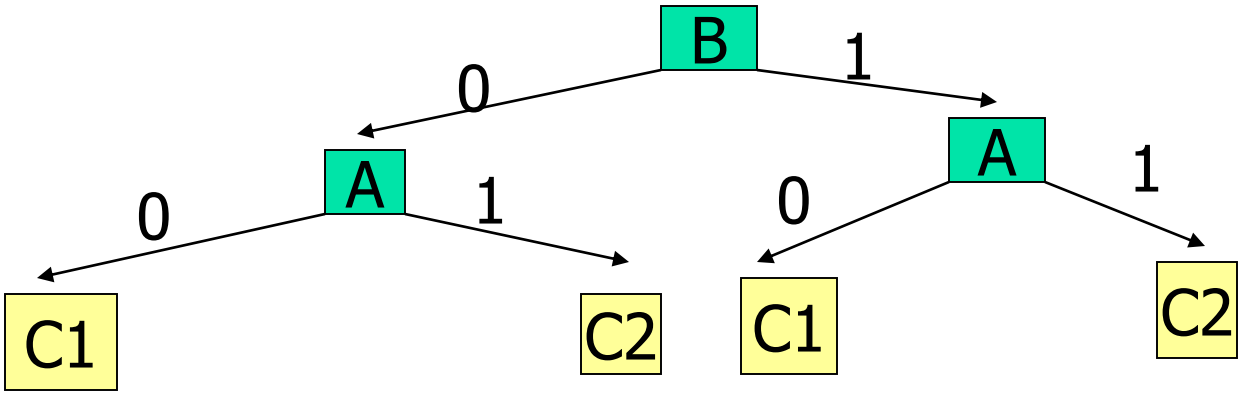
- Soit le training set suivant

A	B	Classe
0	1	C1
0	0	C1
1	1	C2
1	0	C2

Si c'est A qui est choisi en premier



Si c'est B qui est choisi en premier



Choix de l'attribut de partionnement (2)

- Un arbre de décision représente la suite de questions à poser pour pouvoir classifier un nouvel exemple.
- Le but consiste à obtenir une classification en posant le moins possible de questions
- Dans l'exemple précédent, on dira que l'attribut A apporte plus d'information, respectivement à la classification des exemples, que B
- Nous avons donc besoin de quantifier l'information apportée par chaque attribut

Notions sur la théorie de l'information(1)

- Intuitivement : Plus un événement est probable, moins il nous apporte d'information
  - Exemple : Vous êtes dans le désert et on vous annonce que le lendemain, il fera beau. C'est un événement très probable, ce message n'apporte donc presque aucune information

- La quantité d'information associée à un événement x sera considérée comme une fonction croissante sur son *improbabilité*

$$h(X) = f\left(\frac{1}{\text{Proba}(X)}\right)$$

- Un événement certain apporte une quantité d'information nulle, ainsi *f(1)* doit être nulle

Notions sur la théorie de l'information(2)

- La réalisation de 2 événements indépendants apporte une quantité d'information égale à la somme de leurs informations respectives, i.e

$$h(X,Y) = f\left(1/\text{Proba}(X,Y)\right) = f\left(1/\text{Proba}(X) * \text{Proba}(Y)\right) = h(X) + h(Y)$$

- C'est la fonction log en base 2 qui a été choisie. Ainsi,

$$h(X) = -\log_2\left(\text{Proba}(X)\right)$$

- La fonction *h* satisfait les 2 conditions: croissante et l'info de deux événements indépendants est la somme des infos

Notions sur la théorie de l'information(3)

- Supposons maintenant qu'il y a deux classes, *P* et *N*
  - Soit *S* un ensemble qui contient *p* éléments de *P* et *n* éléments de *N*
  - La probabilité qu'un élément soit dans P est *p/(p+n)*
  - La quantité d'information nécessaire pour décider si un élément quelconque de *S* se trouve dans *P* ou dans *N* est définie par

$$I(p, n) = -p \log_2 \frac{p}{p+n} - n \log_2 \frac{n}{p+n}$$

Cas particulier

- Supposons que p soit nul. Cela veut dire que I(n,p)=0:
  - p=0 et log(p/[n+p])=-∞ le produit donne 0 (pour être précis, la limite du produit tend vers 0 quand p tend vers 0)
  - log(n/[n+p])=0 donc le produit donne 0
- Ce qui est conforme à l'intuition: On n'a pas besoin d'info pour décider si un élément est dans N ou P; on est sûr qu'il est dans N

Intuition de l'expression I(n,p)

- Chaque élément apporte une information qui est
  - $-\log\left(\frac{p}{n+p}\right)$ si il est dans P
  - $-\log\left(\frac{n}{n+p}\right)$ si il est dans N
- Si l'on fait le total des infos, on obtient I(n,p)

Gain d'information et arbre de décision

- Supposons qu'en utilisant l'attribut A, S est partitionné en {S_1, S_2 , …, S_v} (ça veut dire que A prend v valeurs)
  - Si S_i contient p_i *tuples* de *P* et n_i tuples de *N*, l'entropie, ou la quantité d'information nécessaire pour classifier les objets de tous les sous arbres S_i est

$$E(\mathrm{A}) = \sum_{i=1}^{v} I(p_i, n_i)$$

  - L'entropie mesure la « quantité de désordre » qui reste après le choix de A
- L'information de codage gagnée en utilisant A sera

$$Gain(\mathrm{A}) = I(p, n) - E(\mathrm{A})$$

Intuition derrière Gain(A)

- Pour classer les éléments dans S_i, nous avons besoin d'une quantité d'info égale à $I(n_i,p_i)$
- Pour classer les éléments dans tous les S_i, on fait la somme des $I(n_i,p_i)$ ce qui donne E(A)
- On sait que pour classifier les éléments, nous avons besoin d'une quantité d'info égale à I(n,p)
- Suite au pationnement des n+p éléments selon les valeurs de A, nous aurons besoin d'une quantité d'info égale à E(A)
- Donc, il nous manquera que I(n,p)-E(A) pour pouvoir classer

Application à l'exemple

- Il y a 2 classes C1 (P) et C2 (N)
- En choisissant A, S est partitionné en S_1 et S_2
- p_1=2, n_1=0, p_2=0 et n_2=2

A	B	Classe
0	1	C1
0	0	C1
1	1	C2
1	0	C2

- E(A)=(I(2,0)+I(0,2))
  - I(2,0)=-log(1)-0*log(0)=0
  - I(0,2)=0
  - E(A)=0
- Gain(A)=I(2,2)-E(A)
  - I(2,2)=-2*log(2/4)-2*log(2/4)=4
- Gain(A)=4

Application à l'exemple

- En choisissant B, S est partitionné en S_1 et S_2
- $p_1=1$, $n_1=1$, $p_2=1$ et $n_2=1$

- E(B)=(I(1,1)+I(1,1))
  - I(1,1)=-log(1/2)-log(1/2)=2
  - E(B)=4

- Gain(B)=I(2,2)-E(B)=0

- Il vaut mieux choisir A!!

A	B	Classe
0	1	C1
0	0	C1
1	1	C2
1	0	C2

S1

S2

Gain d'information: Exemple

- Classe P: achète_ordinateur = "oui"
- Classe N: achète_ordinateur = "non"
- I(p, n) = I(9, 5) =13,16
- L'entropie de l'attribut *age*:

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	4,855
30…40	4	0	0
>40	3	2	4,855

$$E(age) = I(2,3) + I(4,0) + I(3,2) = 9,71$$

Ainsi,

Gain(age)=I(p,n)-E(age)=3,45

C'est l'attribut qui maximise le gain

Remarquer que le salaire n'a pas du tout été utilisé

Extraction de règles de classification

- De la forme SI-ALORS
- Chaque chemin partant de la racine et atteignant une feuille donne lieu à une règle
- Chaque paire attribut-value le long d'un chemin forme une conjonction
- Les feuilles constituent la classe
- Exemple

SI *age* = "<=30" ET *étudiant* = "*non*" ALORS
achète_ordinateur = "*non*"

SI *age* = "<=30" ET *étudiant* = "*oui*" ALORS
achète_ordinateur = "*oui*"

Problème de l'overfitting

- En appliquant la méthode décrite jusque là, on obtient des arbres qui classent correctement les exemples du training set
  - Aucune erreur (normalement)
- Mais rien ne dit qu'ils seront efficaces pour l'autre partie des exemples
- Lorsque l'arbre « colle trop au training set » on parle d'overfitting
- Pour résoudre le problème, on va autoriser des erreurs sur le training set pour obtenir des arbres assez généraux

Généraliser l'arbre induit

- 2 Approches
  - Prepruning: ne pas découper un nœud si le partage fait basculer la mesure de pertinence en dessous d'un certain seuil.
    - Par exemple si le gain est inférieur à un certain seuil
    - Difficile de choisir un seuil approprié
  - Postpruning: supprimer des banches d'un arbre déjà construit. Obtenir un ensemble d'arbres réduits
    - Utiliser un ensemble de données différent du training set pour choisir le meilleur arbre réduit

Classification Bayésienne

- Prédiction en termes de probabilité : Prédit plusieurs hypothèses en les pondérant par leurs probabilités
- Etant donné un objet O, la méthode consisite à calculer la probabilité d'appartenance de O à chaque classe, puis choisir celle qui maximise cette valeur
- Standard: Même s'il s'avère que les méthodes bayésiennes se révèlent intractables, elles peuvent être considérées comme étalon pour mesurer la correction d'autres méthodes

Théoreme de Bayes

- Soit le *training set D, la probabilité aposteriori de l'hypothèse h, P(h|D)* suit le théorème de Bayes

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- MAP (maximum posteriori) hypothesis

$$h_{MAP} \equiv \max_{h \in H} P(h \mid D) = \max_{h \in H} P(D \mid h)P(h).$$

- Difficulté pratique: on a besoin de connaître initialement plusieurs probabilités et un temps de calcul non négligeable

Classification Bayésienne

- Le problème de classification peut être formalisé en utilisant les probabilités a-posteriori:

- P(C|X) = prob. que $X=<x_1,\ldots,x_k>$ soit de la classe C.

- Ex. P(classe=N | temps=soleil,vent=vrai,…)

- Affecter à X la classe C tel que P(C|X) est maximal

Estimer les probabilités a-posteriori

- Théoreme de Bayes :

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

- P(X) est la même pour toutes les classes
- P(C) = fréquence relative des éléments de C
- C telle que P(C|X) est maximum =

  C telle que P(X|C)·P(C) est maximum
- Problème: calculer P(X|C) est infaisable !

Classification Bayésienne Naïve

- Hypothèse: indépendance des attributs

$$P(x_1,\ldots,x_k|C) = P(x_1|C)\cdot\ldots\cdot P(x_k|C)$$

- Si attribut A_i est qualitatif:
  $P(x_i|C)$ est estimée par la fréquence relative des éléments ayant la valeur x_i pour A_i et qui sont dans C

- Si attribut A_i est continu:
  $P(x_i|C)$ est estimé en utilisant la loi de Gauss (on suppose A_i suit une loi normale)

- Facile à calculer dans les deux cas

Example: estimer $P(x_i|C)$

Temps	Temperature	Humidite	Vent	Class
soleil	chaud	élevé	faux	N
soleil	chaud	élevé	VRAI	N
couvert	chaud	élevé	faux	P
pluie	tiede	élevé	faux	P
pluie	froid	normal	faux	P
pluie	froid	normal	VRAI	N
couvert	froid	normal	VRAI	P
soleil	tiede	élevé	faux	N
soleil	froid	normal	faux	P
pluie	tiede	normal	faux	P
soleil	tiede	normal	VRAI	P
couvert	tiede	élevé	VRAI	P
couvert	chaud	normal	faux	P
pluie	tiede	élevé	VRAI	N

P(p) = 9/14
P(n) = 5/14

Temps	
P(soleil|P) = 2/9	**P(soleil|N) = 3/5**
P(couvert|P) = 4/9	**P(couvert|N) = 0**
P(pluie|P) = 3/9	**P(pluie|N) = 2/5**
Température	
P(chaud|P) = 2/9	**P(chaud|N) = 2/5**
P(tiède|P) = 4/9	**P(tiède|N) = 2/5**
P(froid|P) = 3/9	**P(froid|N) = 1/5**
Humidité	
P(élevée|P) = 3/9	**P(élevée|N) = 4/5**
P(normale|P) = 6/9	**P(normale|N) = 2/5**
Vent	
P(Vrai|P) = 3/9	**P(vrai|N) = 3/5**
P(faux|P) = 6/9	**P(faux|N) = 2/5**

Exemple: classifier X

- Soit X = <pluie, chaud, élevée, faux>
- P(X|p)·P(p) = P(pluie|p)·P(chaud|p)·P(élevée|p)·P(faux|p)·P(p) = 3/9·2/9·3/9·6/9·9/14 = 0.010582
- P(X|n)·P(n) = P(pluie|n)·P(chaud|n)·P(élevée|n)·P(faux|n)·P(n) = 2/5·2/5·4/5·2/5·5/14 = 0.018286
- X est classifié en N (ne pas jouer au tennis)

L'hypothèse d'indépendance

- Rend le calcul possible
- Problème: en pratique, les attributs (variables) sont souvent corrélés
- Solution :
  - Réseaux Bayesien, utiliser le raisonnement Bayésien en tenant compte des relations causales qui existent entre attributs

La méthode des k plus proches voisins (*k*-Nearest Neighbor)

- Variables numériques: un objet=point dans espace à n dimensions.
- Utilisation de la distance pour définir le plus proche voisin.
- Etant donné O, on cherche ses k plus proches voisins. Ensuite, on lui affecte la classe la plus fréquente dans cet ensemble (ou la moyenne, s'il s'agit d'une variable continue)

Variation de l'algorithme

- Pondérer les voisins: Plus un voisin est proche, plus son poids est grand
- On peut considérer la formule de poids suivante

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

- Calculer la moyenne pondérée

Régression

- Régression Linéaire: $Y = \alpha + \beta X$
  - 2 paramètres , α et β spécifient une droite. Ils sont estimés en utilisant les données disponibles.
  - Utilisation de la méthode des moindres carrés avec comme données, les couples (Y_1, X_1) (Y_2, X_2) …
- Régression multiple: $Y = b0 + b1\ X1 + b2\ X2$.
  - Estimer b0, b1, b2 …
  - Plusieurs fonctions non linéaires peuvent être transformées de la sorte.