Spécialité DAC **Module ASWS** **TD/TME du 12 décembre 2018**

1. Utiliser Orange: construction arbres de décision

- Télécharger et installer Orange
- Sur l'interface graphique :
  - « File »
  - « Data Table »
  - Construire un fichier .tab avec l'exemple des lentilles vu en cours (classe = lenses)

age	**prescription**	**astigmatic**	**tear_rate**	**lenses**
young	myope	no	reduced	none
young	myope	no	normal	soft
young	myope	yes	reduced	none
young	myope	yes	normal	hard
young	hypermetrope	no	reduced	none
young	hypermetrope	no	normal	soft
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	no	reduced	none
pre-presbyopic	myope	no	normal	soft
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	no	reduced	none
pre-presbyopic	hypermetrope	no	normal	soft
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	no	reduced	none
presbyopic	myope	no	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	no	reduced	none
presbyopic	hypermetrope	no	normal	soft
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

- Ouvrir le tableau avec « File »
- Visualiser la table avec « Data Table »
- Utiliser « Classification tree » pour construire un arbre de décision
  - Utiliser le gain d'information
- Recourir à « Classification tree graph » pour visualiser l'arbre obtenu
- Tester le classifieur avec « Test Learner » en ayant recours à la méthode « leave one out »

2. Règles d'association

- Construire, un fichier tab avec l'exemple donné page suivante
    - Utiliser support 0.4
    - Visualiser avec le « visualiseur de règles ». Profondeur 2, puis 3

a	b	c	d	e	f
1	1	1	0	0	1
1	1	0	1	0	1
0	1	1	0	1	0
1	1	1	0	0	1
1	0	1	0	1	0

- Construire les règles d'association avec l'ensemble « lentilles » vu à la question 1
    - on commencera avec un support min de 0,5, puis avec un support min de 0,3

3. Programmation ID3 en Python

- Reprendre le script Python donné sur le site
    a. Tester la classe appelée ensemble d'apprentissage
    b. Méthode valeur qui donne la valeur d'un attribut
    c. Méthode qui donne l'entropie d'un ensemble
    d. Construire un arbre de décision avec le choix de l'entropie qui minimise l'entropie, puis avec un choix aléatoire

4. Généralisation de termes

Ex1: généralisation de t_1 = g(h, f(3, 5), k(i)) et de t_2 = g(u, f(5, 5), k(8))
Ex2: généralisation de t_1 = g(h, f(3, 5), k(3)), g(u, f(5, 5), k(5)), T, E1, E2).

Ex3 : généraliser tous les couples (t_i, t_j)
t_1 = [pair(4), impair(3), impair(4 + 3)]
t_2 = [pair(8), impair(3), impair(8 + 3)]
t_3 = [pair(8), impair(5), impair(8 + 5)]
t_4 = [impair(5), pair(8), impair(8 + 5)]
t_5 = [impair(5), pair(8), impair(5 + 8)]
t_6 = [pair(8), impair(5), impair(5 + 8)]

Télécharger le programme Prolog 'generalisation_termes' et le faire fonctionner sur ces exemples.

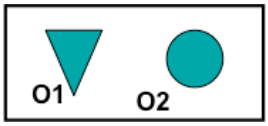
5. Construction du plus petit généralisé de deux clauses

a. Bouquets

C1: bouquet(1) :- contient(1,f1), contient(1, f2), rose(f1), jaune(f1), narcisse(f2), blanche(f2).
C2: bouquet(2) :- contient(2, f3), contient(2, f4), jonquille(f3), jaune(f3), rose(f4), blanche(f4).
C3: bouquet(3) :- contient(3, f5), contient(3, f6), jonquille(f5), rose(f6), blanche(f6).

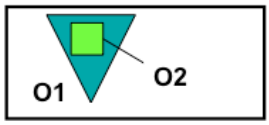
i. Calculer toutes les sélections de C1 et C2 et de C1 et C3.
ii. Calculer la généralisation des deux clauses C1 et C2 et la généralisation des deux clauses C1 et C3.
iii. A supposer que l'on introduise un prédicat binaire fleur(<type>, F) pour remplacer les prédicats unaire rose(F), narcisse(F) et jonquille(F), et que l'on utilise des connaissances du type inf(jonquille, narcisse) et inf(narcisse, narcisse) pour tenir compte du fait que les jonquilles sont des narcisses particulières. Montrer comment reformuler les exemples, c'est-à-dire C1, C2 et C3, en C'1, C'2 et C'3 afin d'utiliser ces connaissances au cours de la généralisation.
iv. Calculer la généralisation des clauses C'1 et C'3 – réécritures de C1 et C3 – à l'aide de l'algorithme de Plotkin – à défaut, donner au moins le type de généralisation que l'on doit obtenir.

b. Problèmes de Bongard

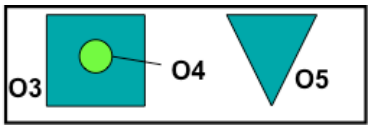


b1([pos(1), - contient(1,o1), -contient(1,o2), -triangle(o1), -pointe(o1,down), -circle(o2)]).

b2([pos(2), - contient(2,o3), -triangle(o3), -pointe(o3,down)]).
b3([pos(b1), -contient(b1,o1), -contient(o1,o2), -triangle(o1), -carré(o2)]).



b5([pos(b2), -contient(b2,o3), -carré(o3), -contient(o3, o4), -carré(o4)]).
i. Calculer toutes les sélections de b1 et b2 et de b3 et b5.
ii. Calculer la généralisation des deux clauses b1 et b2 et la généralisation des deux clauses b3 et b5.



iii. Transformer les clauses b3 et b5 pour prendre en compte la connaissance selon laquelle les carrés et les triangles sont des polygones.
iv. reprendre les questions i, ii et iii avec les clauses c1 et c2 puis c2 et c3
c1([pos(1), -contient(1,o1), -contient(o1,o2), -triangle(o1), -carre(o2)]).
c2([pos(2), -contient(2,o3), -carre(o3), -contient(o3, o4), -carre(o4), -contient(2, o5), -triangle(o5),
-contient(o5, o6), -cercle(o6)]).
c3([pos(3), -contient(3,o7), -carre(o7), -contient(o7, o8), -carré(o8)]).