Apprentissage d'ensembles de règles

C. Rouveirol
MASTER MICR

Sommaire

- Approche générer et tester descendante
  - Exemple : ID3 (déjà vu au cours 1 et 2)
  - Apprentissage de règles (DNF)
    - Apprentissage de règles par couverture
  - Algorithme de couverture
    - Boucle externe : Algorithme générique
    - Boucle interne : Algorithme générique
    - Exemples détaillés d'algorithmes : CN2 (Clark 90)
- Algorithme ascendant « guidé par les données »
- Algorithme descendant « guidé par les données »

Sources

- Transparents de J. Fürnkranz sur Inductive Rule Learning (article Roc'n'Rule)
- Transparents A. Osmani, Apprentissage Symbolique, M2 MICR 2006-2007
- Transparents E. Alphonse, M2 EID, 2006-2007

Apprentissage d'ensembles de règles (DNF)

Pourquoi des ensembles de règles ?

- arbres de décision souvent complexes et difficiles à interpréter
- problème de réplication de sous-arbres avec arbres de décision, ce qui partitionne inutilement l'espace des instances et pose des difficultés à la recherche heuristique
- les ensembles de règles ordonnés à k littéraux par règles plus expressif que les arbres de profondeur k
- se généralise bien à la logique d'ordre un

Apprentissage d'ensembles de règles (DNF)

Comment les apprendre ?

- Apprendre directement l'ensemble de règles
  - intérêt théorique mais très peu utilisé
- Apprendre une règle à la fois : stratégie de couverture

L'origine de la stratégie de couverture est due à Michalski (1969) (Algorithme AQ). Le terme Separate-and-conquer dû à Pagallo et Haussler (1990) est simplement une interprétation de la stratégie de couverture.

Variantes des algorithmes de couverture

- Couverture séquentielle ou simultanée?
  - Séquentielle: apprend une hypothèse à la fois
  - Simultanée: apprend un arbre (plusieurs règles en parallèle)
- Ascendant ou descendant?
- Générer et tester ou guidé par les données/exemples ?
  - Générer et tester: explore un espace d'hypothèses bien formées (ordonnées)

    Guidé par les données: utililse les exemples pour modifier son hypothèse courante
- Post-pruning des règles
- Quelles heuristiques ?
  - Entropie,prédiction, laplacien...

Recherche descendante dans l'espace des hypothèses

Deux types d'opérateurs :

- «générer-tester» : fondés sur la structure de H seulement

  ρ (h) = h' avec h' $\in$ H et h' $\leq_h$ h
- «dirigés par les données» : utilisation d'un exemple négatif pour spécialiser

  ρ (h,e-) = h' avec h' $\in$ H et h' $\leq_h$ h et not couvre(h',e-)
- On appelle *couverture inférieure* de h l'ensemble d'hypothèses:

  CI(h) = {h' $\in$ H | h' $\leq_h$ h et il n'existe pas de h'' $\in$ H t.q. h' $\leq_h$ h'' $\leq_h$ h }

Apprentissage de règles par couverture

Tous les algorithmes d'apprentissage de règles dits « *separate-and-conquer* » ou « *covering* » partagent le même principe :

- Chercher une règle qui couvre une partie des exemples positifs
- Enlever les exemples positifs couverts de la base d'apprentissage initiale
- Recommencer récursivement le processus jusqu'à ce qu'il n'y ait plus d'exemples positifs à couvrir

A la fin de ce processus, chaque exemple de la base d'apprentissage est couvert par au moins une règle

Algorithme générique de couverture (boucle externe)

```
Procédure Couverture(Exemples)        % renvoie une hypothèse cohérente
    Hypothèse = T  % élément universel
    Tantque Positif(Exemples) ≠ ∅
        Règle = ApprendreMeilleureRègle(Exemples)
        Hypothèse = Hypothèse ∪ Règle
        Exemples = Exemples \ ExemplesCouverts(Règle, Exemples)
    Renvoyer Hypothèse
```

Les algorithmes d'apprentissage par couverture diffèrent au niveau de la méthode implantée pour **ApprendreMeilleureRègle** (Exemples).

Apprentissage de règles par couverture

Beaucoup d'algorithmes de cette famille sont développés pour les différents langages d'hypothèses. Parmi lesquels :

DNF : **AQ** et AQ15 (Michalski69 et 86), PRISM (Cendrowska87), PFOIL (Mooney95), JoJo(Fensel&Wiese93), **RIPPER** (Cohen95), …

CNF : PFOIL-CNF (Mooney1995), ICL (de Raedt95)

Decisionlists : **CN2 (Clark & Niblett90)**, PREPEND (Webb&Brkie93), FOILDL (Mooney95) …

Programmes logiques : INDUCE (Michalski80), **FOIL(Quinlan90), PROGOL (Muggleton95)**, FOCL(Pazzani92)

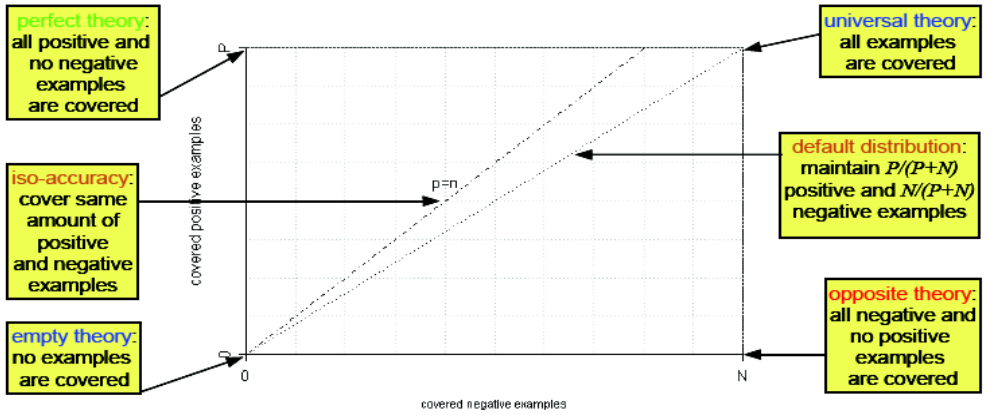
Règles de régression : RULE (Weiss93), FORS (Karalie95) …

Apprentissage d'une règle (boucle interne)

```
Procedure ApprendreMeilleureRègle(Exemples)
    MeilleureRègle = T
    ListeOuvert = {T}
    Tantque ListOuvert ≠ ∅
        tmpListeOuvert = ∅
        Pour tout h ∈ ListeOuvert
            ListOuvert = ListeOuvert \ h
            Pour chaque ρ(h) = h', h' ∈ H
                Si h' est correcte alors          % solution potentielle
                    Si évaluation(h')> évaluation(MeilleureRègle)
                        alors  MeilleureRègle = h'
                Sinon Ajouter h' dans tmpListOuvert
        Élaguer tmpListOuvert                     % heuristique
            ListeOuvert = tmpListeOuvert
    FinTantque
    Retourner MeilleureRègle
```

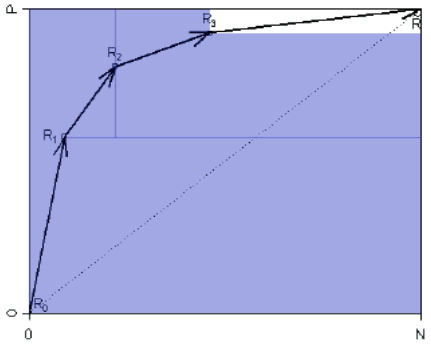

Coverage Spaces

- good tools for visualizing properties of covering algorithms
  - each point is a theory covering p positive and n negative examples



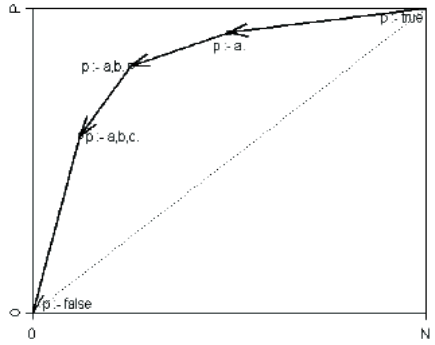
Covering Strategy

- Covering or Separate-and-Conquer rule learning learning algorithms learn one rule at a time
- This corresponds to a path in coverage space:
  - The empty theory R_0 (no rules) corresponds to (0,0)
  - Adding one rule never decreases p or n because adding a rule covers *more* examples (generalization)
  - The universal theory R+ (all examples are positive) corresponds to (N,P)



Top-Down Hill-Climbing

- Successively extends a rule by adding conditions
- This corresponds to a path in coverage space:
  - The rule `p:-true` covers all examples (universal theory)
  - Adding a condition never increases p or n (specialization)
  - The rule `p:-false` covers no examples (empty theory)



- which conditions are selected depends on a *heuristic function* that estimates the quality of the rule

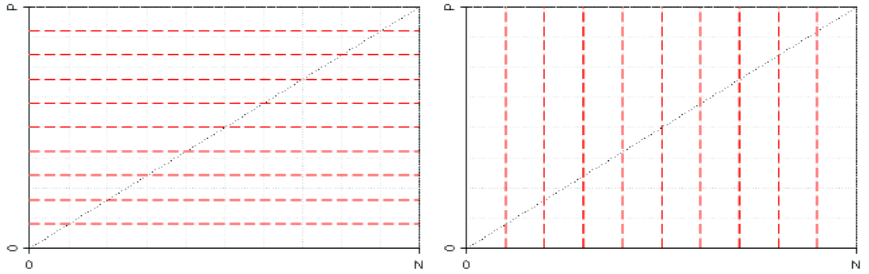
Rule Learning Heuristics

- Adding a rule should
  - increase the number of covered negative examples as little as possible (do not decrease *consistency*)
  - increase the number of covered positive examples as much as possible (increase *completeness*)
- An evaluation heuristic should therefore trade off these two extremes
  - Example: Laplace heuristic $h_{Lap} = \frac{p+1}{p+n+2}$
    - grows with $p \to \infty$
    - grows with $n \to 0$
  - Note: Precision is not a good heuristic. Why?

$$h_{Prec} = \frac{p}{p+n}$$

Isometrics in Coverage Space

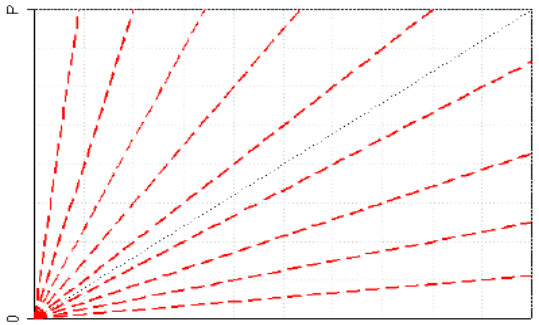
- Isometrics are lines that connect points for which a function in p and n has equal values
  - *Examples*: Isometrics for heuristics $h_p = p$ and $h_n = -n$



Precision (Confidence)

$$h_{Prec} = \frac{p}{p+n}$$

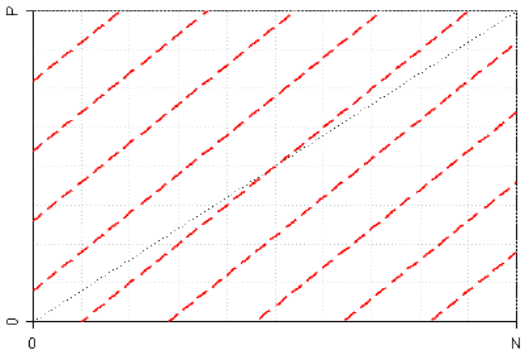
- *basic idea:*
  percentage of positive examples among covered examples
- *effects:*
  - rotation around origin (0,0)
  - all rules with same angle equivalent
  - in particular, all rules on P/N axes are equivalent



Accuracy

$$h_{Acc} = \frac{p+(N-n)}{P+N} \simeq p-n$$

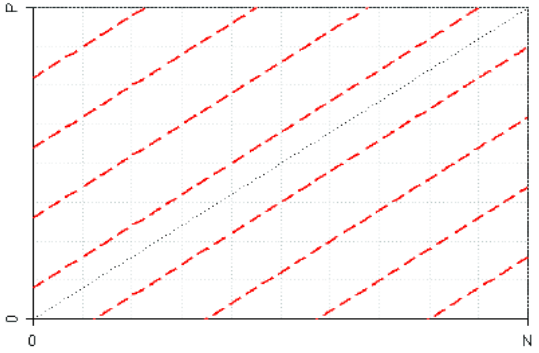
- *basic idea:*
  percentage of correct classifications (*covered positives* plus *uncovered negatives*)
- *effects:*
  - isometrics are parallel to 45º line
  - covering one positive example is as good as not covering one negative example



Weighted Relative Accuracy

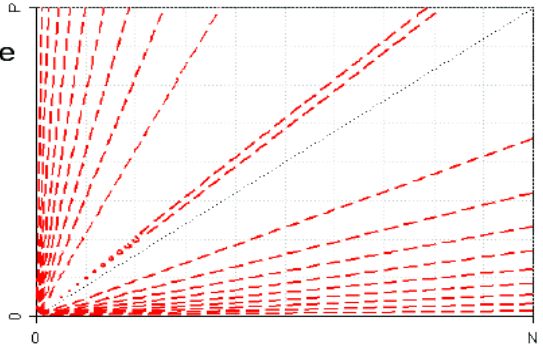
$$h_{Acc} = \frac{p+n}{P+N}\left(\frac{p}{p+n} - \frac{P}{P+N}\right) \simeq \frac{p}{P} - \frac{n}{N}$$

- *basic idea:*
  normalize accuracy with the class distribution
- *effects:*
  - isometrics are parallel to diagonal
  - covering $x\%$ of the positive examples is as good as not covering $x\%$ of the negative examples



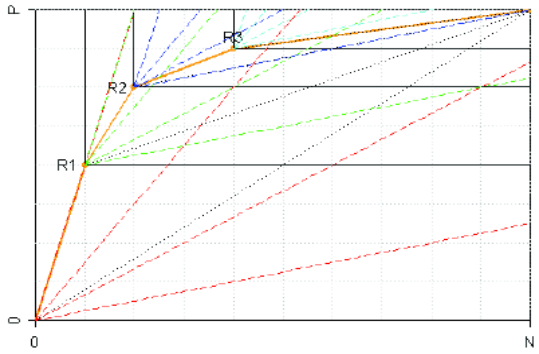
Entropy and Gini Index

- $h_{Ent} = -\left(\frac{p}{p+n}\log_2\frac{p}{p+n} + \frac{n}{p+n}\log_2\frac{n}{p+n}\right)$
- $h_{Gini} = 1 - \left(\frac{p}{p+n}\right)^2 - \left(\frac{n}{p+n}\right)^2 \simeq \frac{pn}{(p+n)^2}$
- *effects:*
  - entropy and Gini index are equivalent
  - like precision, isometrics rotate around (0,0)
  - isometrics are symmetric around 45º line
  - a rule that only covers negative examples is as good as a rule that only covers positives



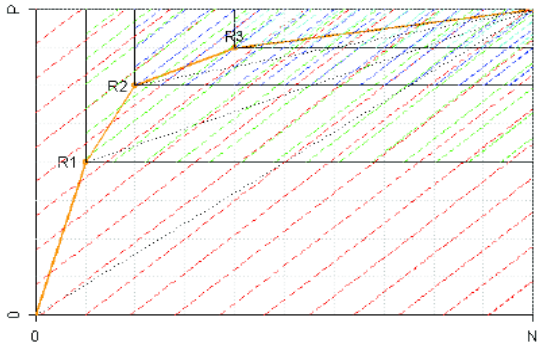
Optimizing Precision

- Precision tries to pick the steepest continuation of the curve
  - does not assume any costs



Optimizing Accuracy

- Accuracy assumes the same costs in all subspaces
  - a local optimum in a sub-space is also a global optimum in the entire space



RIPPER (Cohen, 95)

Généralisation du gain d'information utilisé dans ID3 et C4.5 à l'apprentissage de règles :

- Recherche : gloutonne
- Opérateur de spécialisation: ajout de descripteurs
  - A = v, A < v, A >= v
- Fonction d'évaluation : Gain d'information

$$Gain(L,R) \equiv p\left(lg\left(\frac{p_1}{p_1+n_1}\right) - lg\left(\frac{p_0}{p_0+n_0}\right)\right)$$

R: règle à spécialiser
L: littéral candidat à ajouter à R
p_0, n_0: positifs (négatifs) couverts par R
p_1, n_1: positifs (négatifs) couverts par R+L
p : positifs couverts par R et R+L

- Elagage de l'hypothèse

Autres algorithmes

- Recherche : gloutonne, faisceau, meilleur d'abord, ...
- Opérateur de spécialisation : dépend du langage des hypothèses
- Fonction d'évaluation : précision, Laplacien, m-estimate, ...

Exemple avec heuristique de précision

$$hAcc(règle, exemples) = \frac{p(règle)}{p(règle)+n(règle)}$$

\+ 1: a,b
2: b,c
\- 3: c,d
4: d,e

hAcc(T) = 2/4
hAcc(a)=1/1
hAcc(b)=2/2
hAcc(c)=1/2
hAcc(d)=0/2
hAcc(e)=0/1

Sélectionner a et enlever les exemples couverts :

\+ 2: b,c
\- 3: c,d
4: d,e

hAcc(a) =0/0
hAcc(b)=1/1
hAcc(c)=1/2
hAcc(d)=0/2
hAcc(e)=0/1

On sélectionne la DNF : a ∨ b

CN2 (Clark&Boswell 1991)

- Apprend une liste de décision (Rivest, 1987)
- Une liste de décision:conjonction (bornée) associée à une classe. Les listes sont ordonnées. La dernière conjonction, true, représente la règle par défaut de la liste de décision.
- Exemple: la liste de décision suivante représente x1 XOR x2
  - (x1x2, 0)
  - (x1,not(x2)),1)
  - (not(x1)not(x2),0)
  - (true,1)

CN2 (Clark&Boswell 1991)

- Recherche : faisceau (taille 5)
- Opérateur de spécialisation : A=v, A
- Recherche heuristique : dans la version originale

$$h_{ent} = -(\frac{p}{p+n}\log_2\frac{p}{p+n} + \frac{n}{p+n}\log_2\frac{n}{p+n})$$

- Puis, évolution vers :

$$LaPlace(règle, exemples) = \frac{p(règle)+1}{p(règle)+n(règle)+NumClasses}$$

Exemple CN2

\+ **1: a,b**
2: b,c
\- **3: c,d**
4: d,e

L(T) =3/6
L(a)=2/3
L(b)=3/4
L(c)=2/4
L(d)=1/4
L(e)=1/3

On sélectionne le faisceau=2. On garde a, b. On continue la recherche a ∧ b, a ∧ c, b ∧ c. La meilleure règle obtenue est: si b alors classe = +
-

\- **3: c,d**
4: d,e

L(T) =3/3
L(c)=2/3
L(d)=3/3
L(e)=2/3

La meilleure règle obtenue est:

si d alors classe = -.

Stratégie d'apprentissage descendante «dirigée par les données»

- Un système emblématique de la stratégie DDD : AQ (Michalski et al. 78, 80, 83)

Procédure :

1. Choisir un exemple dans E+ comme graine
2. Rejeter les exemples négatifs un à un en produisant uniquement les raffinements qui couvrent la graine

- $\rho(h,e^-)$ dépend de la graine => $\rho_g(h,e^-)$

Opérateurs de spécialisation «dirigés par les données»

Soit $g = (a_1, \ldots a_i, \ldots a_n)$ la graine, $e^- = (b_1, \ldots b_i, \ldots b_n)$ le négatif à rejeter :

- opérateur nominal :
  - $a_i = v$ et $b_i \neq v$ => $a_i = v$
  - $a_i = v$ et $b_i = v'$ => $a_i \neq v$
- opérateur numérique :
  - $a_i = v$ et $b_i = v'$ => $a_i \in]-\infty, v'[$ si $v' > v$ et sinon $a_i \in]v', +\infty[$

Des opérateurs définis pour les hiérarchies, les ensembles (Michalski, 83)

Le système DDD AQ

- Recherche : faisceau
- Opérateur de spécialisation : opérateurs DDD
- Fonction d'évaluation : LEX (fonction lexicographique)
  - Maximiser le nombre d'exemples couverts par la règle
  - Minimiser le nombre d'exemples négatifs couverts
  - Minimiser le nombre d'attributs de la règle
  - ...

Stratégie d'apprentissage descendante «dirigées par les données»

- Remarque fondamentale : le nombre de raffinement possibles d'une hypothèse dépend du nombre de valeurs d'attributs en commun entre la graine et l'exemple à rejeter
- Définition d'un ordre total sur les négatifs : on traite les plus proches d'abord
- Si le négatif et la graine ne diffèrent que d'un attribut alors G ne fragmente pas : une nuance critique (ou *near-miss* de Winston, 75)

A Pathology for Top-Down Learning

- Parity problems (e.g. XOR)
  - r relevant binary attributes
  - s irrelevant binary attributes
  - each of the $n = r + s$ attributes has values 0/1 with probability ½
  - an example is positive if the number of 1's in the relevant attributes is even, negative otherwise
- Problem for top-down learning:
  - by construction, each condition of the form $a_i = 0$ or $a_i = 1$ covers approximately 50% positive and 50% negative examples
  - irrespective of whether a_i is a relevant or an irrelevant attribute
  - → top-down hill-climbing cannot learn this type of concept
- Typical recommendation:
  - use *bottom-up learning* for such problems

Stratégies d'apprentissage «dirigées par les données»

- Approche ascendante : on parcourt par généralisation à partir de l'élément nul pour trouver un élément de S
  - Algorithme par moindre-généralisé

Stratégie d'apprentissage ascendante

Opérateurs de raffinement:

- «générer-tester» : fondés sur la structure de H seulement
  - δ (h) = h' avec h' $\in$ H et h $\leq_h$ h'
- «dirigés par les données» : utilisation d'un exemple positif pour généraliser
  - δ(h,e+) = h' avec h' $\in$ H et h $\leq_h$ h' et couvre(h',e+)
- On appelle couverture supérieure de h l'ensemble d'hypothèses : CS(h) = {h' $\in$ H | h $\leq_h$ h' et il n'existe pas de h" t.q. h $\leq_h$ h" $\leq_h$ h' }

Opérateur de généralisation «dirigé par les données»

- Adaptation de l'opérateur de moindre-généralisé au traitement des exemples d'apprentissage

=> astuce de la représentation unique (*single representation trick, Dietterich et al, 1982*) : on plonge l'espace des instances dans l'espace des hypothèses. Un exemple peut toujours être vu comme la règle la plus spécifique couvrant uniquement cette exemple

Opérateur de généralisation «dirigés par les données»

- Soient deux hypothèses h1 et h2 $\in$ H
  - δ (h1,h2) = h1 $\vee$ h2 = h' avec h' $\in$ H et h1 $\leq_h$ h' et h2 $\leq_h$ h'
- Plotkin (70) a le premier montré l'intérêt du calcul de la borne supérieure d'un ensemble pour le problème de l'apprentissage. On parle de moindre-généralisé ou de **lgg** (least-general-generalization) ou de **msg** (most-specific generalization)
- La lgg permet de généraliser par plus petit pas (par définition)
- Son calcul dépend du poset utilisé comme espace d'hypothèses
- Elle est définie uniquement entre hypothèses

Opérateur de généralisation «dirigé par les données»

Moindre-généralisé en langage attribut-valeur

Soit $e_j = (a_{j1}, \ldots a_{ji}, \ldots a_{jn})$ la description de l'exemple e_j

$lgg(e_i,e_j)=h = (b_1, \ldots b_k \ldots b_n)$, t.q. pour tout k allant de 1 à n

Si $a_{ik} = a_{jk}$ alors $b_k = a_{ik}$

Sinon

Si a_{ik} est de type nominal, alors b_k =? (une valeur indéterminée)

Si a_{ik} est de type continu, alors b_k = le plus petit intervalle contenant a_{ik} et a_{jk}

Opérateur de généralisation «dirigés par les données»

Moindre-généralisé :

Exemple : e_1 = (age=24, sexe=F, taille=1.8, yeux=M)

e_2 = (age=26, sexe=H, taille=1.8, yeux=M)

$lgg(e_1,e_2)$ = (age=[24,26], sexe=?, taille=1.8, yeux=M) =h

Le même calcul peut se faire entre deux hypothèses ou entre une hypothèse et un exemple

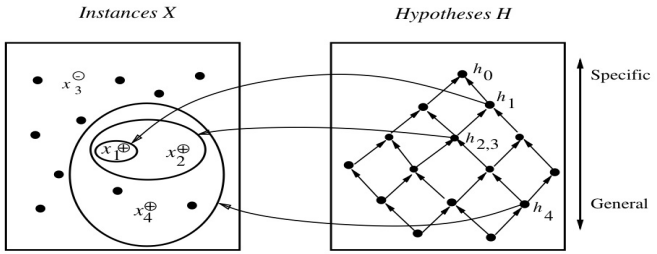
Exemple : e_3 = (age=27, sexe=F, taille=1.8, yeux=B)

$mg(h,e_3)$ = (age=[24,27], sexe=?, taille=1.8, yeux=?)

Stratégie ascendante guidée par les données

```
ApprendMeilleureRègle(E+,E-)
    MeilleureRègle = SélectionnerUnExemple(E+);
    Sol = MeilleureRègle;  ListeOuvert := MeilleureRègle
    while ListeOuvert ≠ ∅
        TempListeOuvert :=  ∅
        for each  h ∈ ListeOuvert   do
            for each e ∈ E+ - ext(h)  do
                TempListeOuvert := TempListeOuvert ∪ lgg(h,e)
            end for
            Candidats := Candidats  ∪ TempListeOUverte
        end for
        Sol := MeilleurEvaluation(Sol ∪ Candidats)
        Candidats := MaximalementSpécifiques(ElagageBU(Candidats, Sol))
        L:=   KmeilleursGuidageBU(Candidats, k)
    end while
 return(Sol)
 end
```

Stratégie ADD : FIND-S



X_1 <soleil chaud normal fort chaude stable oui > h_1 <soleil chaud normal fort chaude stable oui >

X_2 <soleil chaud forte fort chaude stable oui > h_2 <soleil chaud ? fort chaude stable oui >

X_3 <pluie froid forte fort chaude instable non > h_3 <soleil chaud ? fort chaude stable oui >

X_4 <soleil chaud forte fort fraîche instable oui > h_4 <soleil chaud ? fort ? ? oui >

Différents algorithmes ADD

Différenciés par le choix de la graine :

- le premier exemple (FIND-S)
- recherche par faisceau + choix aléatoire de paires d'exemples positifs (GOLEM, Muggleton et al. 92)
- recherche par faisceau + autant de graines que la taille du faisceau (ELENA, Brezellec et al. 93)
- autant de graines que d'exemples positifs + recherche du plus petit sous-ensemble de règles (GLOBO, Torre 99)

Conclusion sur les approches algorithmiques de l'apprentissage symbolique

- Algorithme de covering lorsque le concept à apprendre est caractérisé par un ensemble de règles.
- Le parcours est guidé par une heuristique (technique de parcours + fonction d'évaluation)
- Le parcours de l'espace de recherche se fait au moyen d'opérateurs de raffinement (ascendant ou descendant, «générer-tester» ou «dirigés par les données»)
- Un système d'apprentissage est caractérisé par sa stratégie d'apprentissage, son langage d'hypothèses, son type d'opérateurs, sa recherche heuristique

References

- Clark, P. and Boswell, R. Rule induction with CN2: some recent improvements. In Machine Learning - EWSL-91. Proceedings of the European Working Session on Learning., pages 151--163, 1991
- Clark, P. and Niblett, T. The CN2 induction algorithm. Machine Learning, 3(4):261--284, 1989
- Dietterich, T. G., London, R. L., Clarkson, K., and Dromey, G. . Learning and inductive inference. Chapter XIV in The Handbook of Artificial Intelligence, Vol. III, 323-512, William Kaufmann, 1982
- Fürnkranz J. and Flach P. A., "ROC'n Rule Learning: Towards a Better Understanding of Covering Algorithms", Machine Learning, 58 (1),39-77, 2005
- Plotkin, G. D. A note on inductive generalisation. Machine Intelligence 6:101—124, 1971.