Apprentissage automatique symbolique : intérêts et limites

Isabelle Tellier

Lattice, université Paris 3 - Sorbonne Nouvelle

Motivations

Pourquoi parler d'apprentissage symbolique ?

- apprentissage automatique $\neq$ statistiques !
- domaine ancien mais encore actif
- nettement moins enseigné, connu, diffusé... que les modèles statistiques
- liens historiques forts avec l'IA
- applications en TALN
- sources d'idées pour le TALN actuel ?

Plan

1. Introduction : motivations
2. Inférence Grammaticale (IG)
3. Programmation Logique Inductive (PLI)
4. Comparaison, perspectives

Apprenabilité à la limite (Gold 67) : cadre général

- le critère d'apprenabilité concerne une classe de grammaires et pas une unique grammaire
- une classe est apprenable s'il existe un algorithme d'apprentissage qui identifie n'importe lequel de ses membres
- métaphore des enfants pouvant apprendre toutes les LN
- entrées de l'algorithme Φ : des séquences (ou structures) appartenant (ou pas) à une langue L de la classe
- cible : une grammaire formelle G générant L ($L(G) = L$)
- le processus d'apprentissage est potentiellement infini :

exemples :	e_1	e_2	e_3	e_4	e_5	...
	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	...
hypothèses de Φ :	G_1	G_2	G_3	G_4	G_5	...

Apprenabilité à la limite (Gold 67) : définition

- une classe de grammaires $\mathcal{G}$ est apprenable ssi :
- il existe un algorithme Φ tel que :
  - pour toute grammaire $G \in \mathcal{G}$
  - pour toute présentation d'exemples de $L(G)$ à Φ
    qui énumère $L(G)$
  - Φ se stabilise ("à la limite") sur une grammaire hypothèse G_0
  - telle que $L(G_0) = L(G)$
- si aucun algorithme de ce genre n'existe, $\mathcal{G}$ n'est pas apprenable

Apprenabilité à la limite (Gold 67) : quelques résultats

- avec des exemples positifs et négatifs : toute classe de grammaires récursivement énumérable est (trivialement) aprenable
- avec des exemples positifs seulement : si une classe de grammaires peut générer toutes les langues finies et un moins une langue infinie, alors elle n'est pas apprenable
- example : $\Sigma = \{a\}$, l'ensemble des langues finies sur Σ est $\mathcal{L}$
  - la classe cible est $\mathcal{L} \cup \{a^*\}$
  - soit la séquence d'exemples : aaa, a, $aaaaaaaaaaa$, a, aa,...
  - si l'algorithme suppose la langue est finie, il ne trouve jamais a^*
  - si l'algorithme choisit de généraliser à a^*, il risque de surgénéraliser (il ne recevra jamais de contre-exemple)

  $\Longrightarrow \mathcal{L} \cup \{a^*\}$ est non apprenable par exemples positifs seuls

Apprenabilité à la limite (Gold 67) : héritage

- conséquence directe de ce premier résultat : aucune classe de la hiérarchie de Chomsky n'est apprenable par exemples positifs seuls
- mais des classes apprenables transverses ont été trouvées : Angluin 80, Kanazawa 98
- les algorithmes intéressants prennent tous la forme suivante :
  - construire une grammaire "minimale" générant exactement les exemples
  - tant qu'on reste dans la classe cible : appliquer un opérateur de généralisation

Apprentissage des languages k-réversibles (Angluin 80)

- un langage est k-réversible ssi le miroir de son automate minimal déterministe est déterministe avec anticipation k
- pour tout k fixé, la classe est apprenable par exemples positifs :
  - construction du Prefix Tree Acceptor (PTA)
  - tant que possible : fusion des états violant la k-réversibilité

Exemple d'exécution (0-réversible)

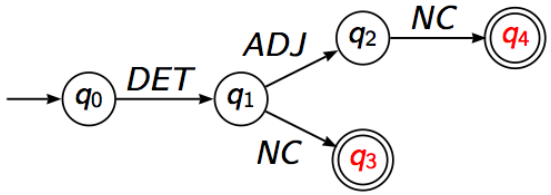


Figure: PTA(S_+)

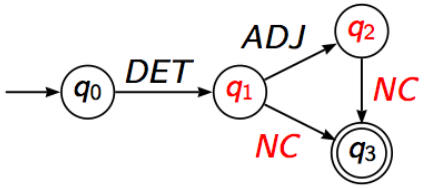


Figure: Premier pas de 0-RI (résultat de 1-RI)

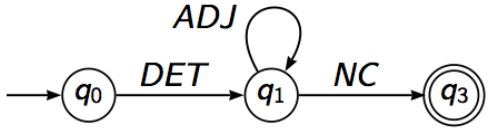
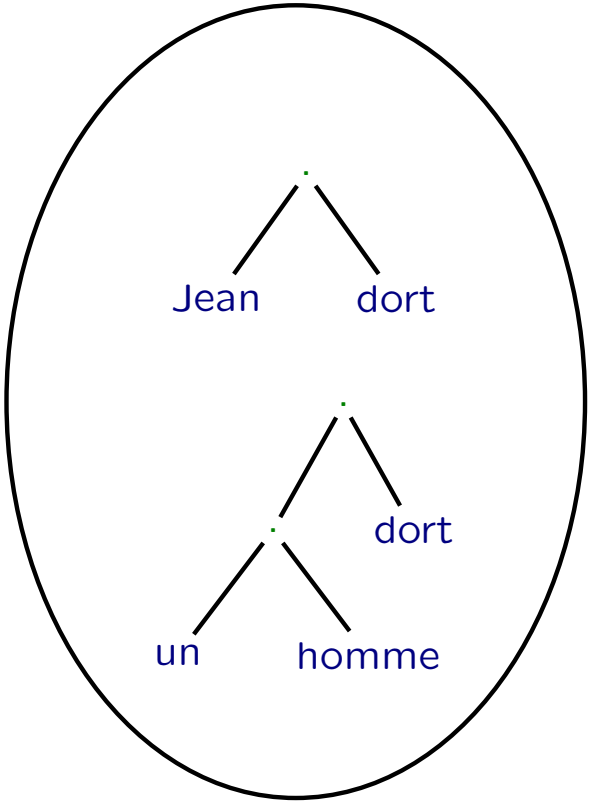


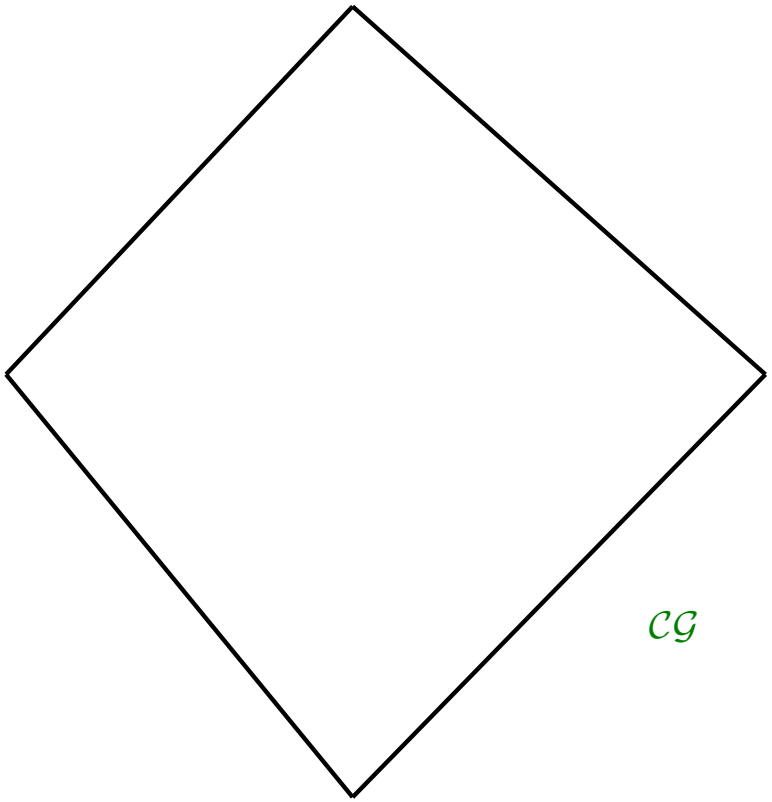
Figure: Résultat de 0-réversible

Apprentissage à partir de structures (Sakakibara, Kanazawa...)

exemples

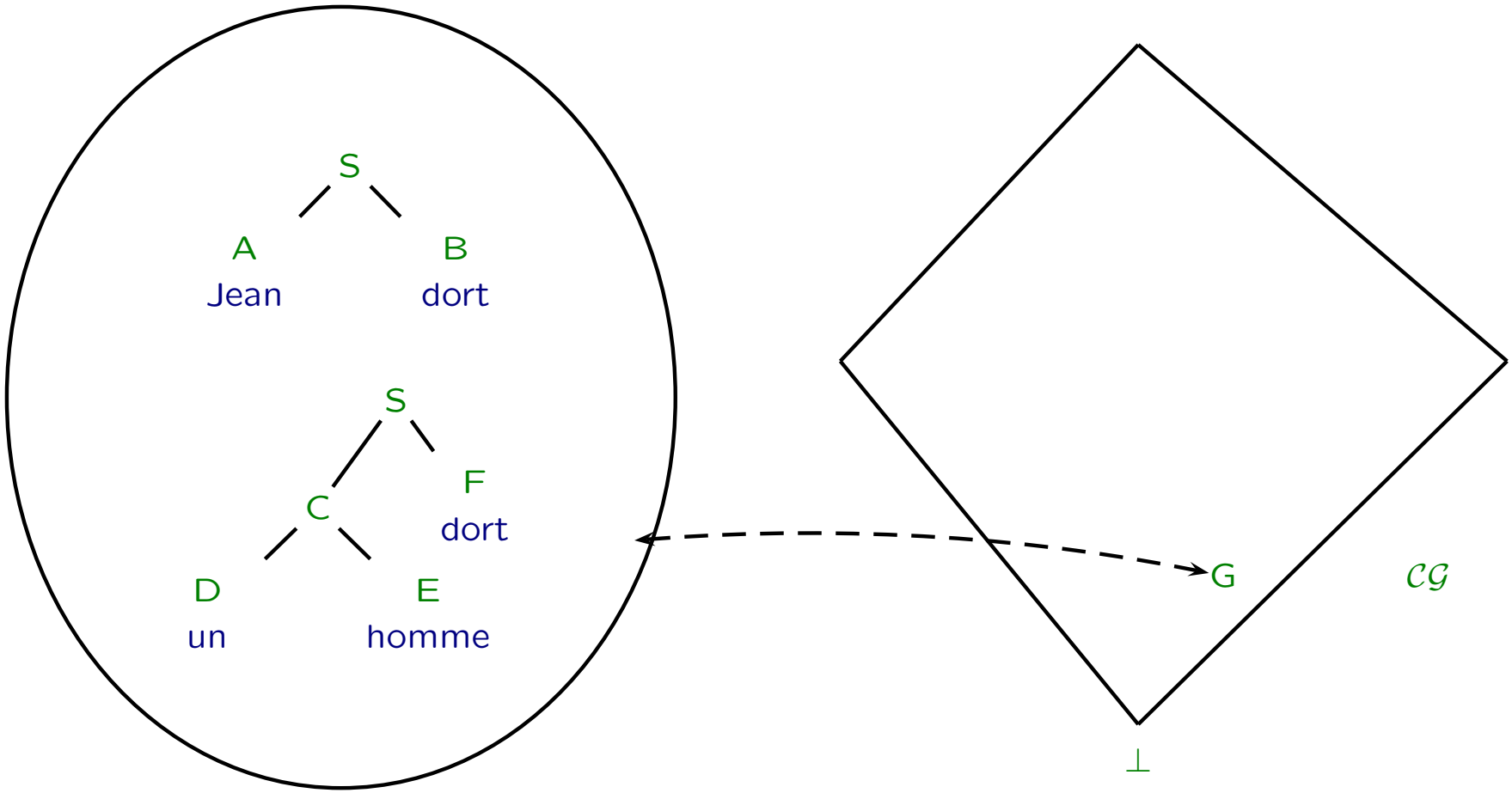


espace des grammaires



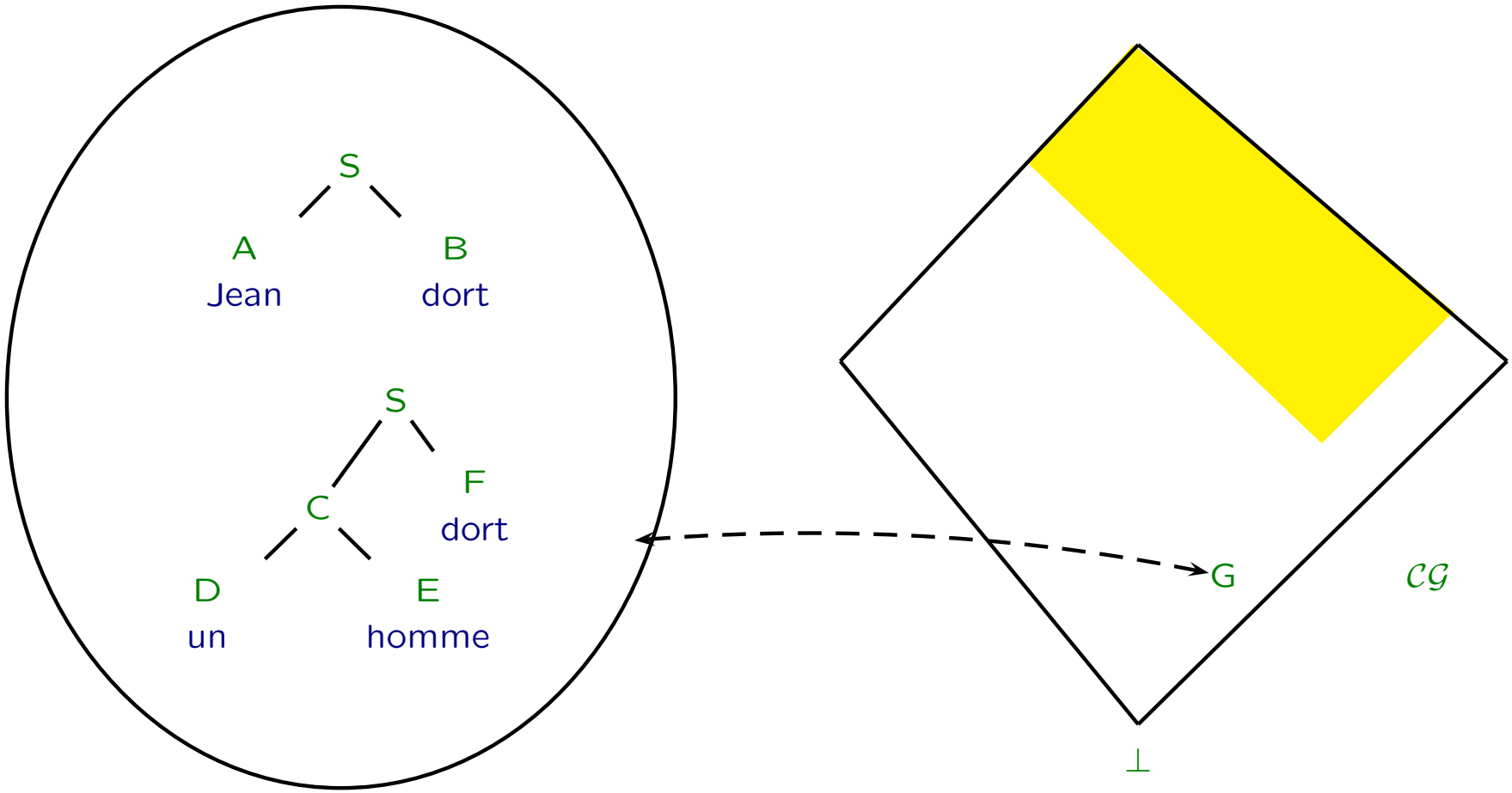
Inférence Grammaticale

Grammaire la moins générale générant exactement les exemples



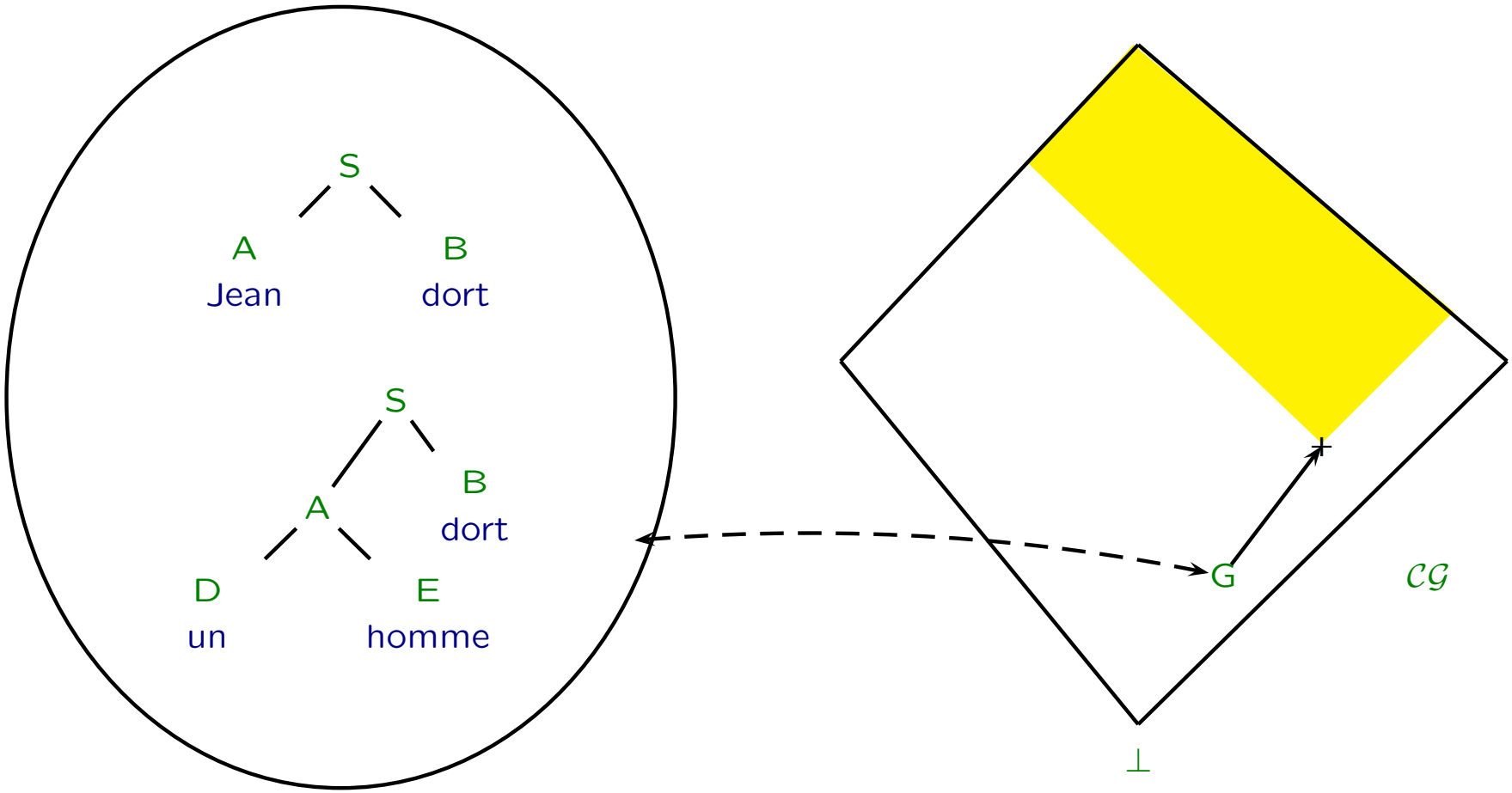
Inférence Grammaticale

Sous-classe des grammaires où on cherche une solution



Inférence Grammaticale

Unification de catégories <=> généralisation de la grammaire



Extensions

- autres modèles : apprentissage Probablement Approximativement Correct (PAC, Valiant 84)
- classes de grammaires plus riches, classes de graphes...
- hybridations : apprenabilité de classes d'automates/grammaires probabilistes

Application au TAL

- quelques tentatives en traduction automatique, morphologie...
- modèles théoriques d'apprentissage du LN avec de la sémantique (<=> structure) (Tellier...)

Inférence Grammaticale

Intérêts

- on se place dans un cadre bien défini
- lisibilité, interprétabilité (en principe) du résultat
- on peut démontrer des théorèmes d'apprenabilité (plus de précision, rappel, F-mesure...)
- on comprend mieux ce que signifie "apprendre" (généraliser mais pas trop)

Limites

- approche trop sensible aux erreurs
- conditions théoriques (classe cible, exemples disponibles...) pas connues/vraies en pratique
- algorithmes pas toujours efficaces

Plan

1. Introduction : motivations
2. Inférence Grammaticale (IG)
3. Programmation Logique Inductive (PLI)
4. Comparaison, perspectives

Introduction : vocabulaire de la programmation logique

- constantes : Alain, Anne, Bart, Carine...
- variables : $X, Y, Z..$
- prédicats : $femme_1(), homme_1(), parent_2(,)$...
- fonction : $age_1()$
- terme : constante | variable $|fonction_k(terme_1, ..., terme_k)$
- littéral : $(\neg)predicat_k(terme_1, , ..., terme_k)$
- clause : disjonction de littéraux
- clause de Horn : avec au plus un littéral positif (<=> règle)
  ex : $grand\_parent(X, Z) \vee \neg parent(X, Y) \vee \neg parent(Y, Z)$
  ou : $grand\_parent(X, Z) \Leftarrow parent(X, Y) \wedge parent(Y, Z)$

Introduction : but

- données de départ : des exemples prédicatifs

  $parent(Alain, Bart), parent(Anne, Bart)$

  $parent(Bart, Carine)$

  $grand\_parent(Alain, Carine), grand\_parent(Anne, Carine)$

- objectif : apprendre automatiquement la définition d'un concept cible (prédicat avec variables) du genre $\forall X, Y, Z$

  $grand\_parent(X, Z) \Longleftarrow parent(X, Y) \wedge parent(Y, Z)$

- récursivité possible : $\forall X, Y, Z$

  $ancetre(X, Y) \Longleftarrow parent(X, Y)$

  $ancetre(X, Z) \Longleftarrow parent(X, Y) \wedge ancetre(Y, Z)$

Introduction : Reformulation du problème

Idée : l'induction est le contraire de la déduction !

- étant donnés :
  - un ensemble d'exemples $D = \{\langle X_i, f(X_i)\rangle\}$ (f est la cible)
  - un "background knowledge" B (ex : liste de prédicats utilisables, règles du 1^{er} ordre, etc.)
- apprendre à partir de D connaissant B signifie :
  - trouver une hypothèse h telle que :
  - $\forall\langle X_i, f(X_i)\rangle \in D$ on a : $B \wedge h \wedge X_i \vdash f(X_i)$ (où $\vdash$ est l'implication logique)
- difficulté : il existe beaucoup de h satisfaisant ce critère $\Longrightarrow$ heuristique nécessaire pour en choisir une (MDL)
- B servira à "guider la recherche"

Le principe de résolution

- substitution=fonction : variable $\rightarrow$ terme
  naturellement étendue aux littéraux et aux clauses
  ex : $L = parent(X, Y),\ \sigma\ :\ X \rightarrow\ Alain, Y\ \rightarrow Bart$
  $L\sigma = \sigma(L) = parent(Alain, Bart)$
- unification : deux littéraux L_1 et L_2 sont unifiés par une substitution σ si $L_1\sigma = L_2\sigma$
- l'unifieur le plus général (mgu) entre deux littéraux L_1 et L_2 est un unifieur σ (unique à un renommage près) tel que pour tout autre unifieur θ il existe une substitution γ telle que $\sigma\gamma = \theta$

Le principe de résolution

- Etant données deux clauses :
  - $C_1 : H_1 \Longleftarrow A, a$
  - $C_2 : H_2 \Longleftarrow b$

  où H_1, a et b : conjonctions (éventuellement vides) d'atomes,
  A un littéral
- C_1 peut être résolue avec C_2 si A et H_2 peuvent s'unifier :
  on note σ un plus grand unifieur ($A\sigma = H_2\sigma$)
- La clause résolvante est $Res(C_1, C_2) : H_1\sigma \Longleftarrow b\sigma, a\sigma$
- Exemple (avec a et b vides) :
  $C_1 : enfant(X, Y) \Longleftarrow parent(Y, X)$
  $C_2 : parent(Alain, Bart)$
  $\sigma(X) = Bart, \sigma(Y) = Alain$
  $Res(C_1, C_2) : enfant(Bart, Alain)$

La θ-subsomption

- On dit qu'une clause C_1 subsume ("est plus générale que") une clause C_2 si : il existe une substitution θ telle que $C_1\theta \subseteq C_2$ (tous les littéraux de $C_1\theta$ apparaissent dans C_2)
- Propriété fondamentale : si C_1 θ-subsume C_2 alors $C_1 \vdash C_2$ mais la réciproque est fausse
- Exemple :

  $C_1 \ : \ fille(X,Y) \ \Longleftarrow \ parent(Y,X) \ \wedge \ femme(X)$

  $C_2 \ : \ fille(Carine,Bart) \ \Longleftarrow \ parent(Bart,Carine) \ \wedge$

  $femme(Carine) \ \wedge \ homme(Bart)$

  avec $\theta(X) = Carine$, $\theta(Y) = Bart$, on a : C_1 θ-subsume C_2

Propriétés

- Le principe de résolution est sain et complet pour l'implication et peut être "inversé"
- La θ-subsomption est décidable (mais NP-complète)
- La θ-subsomption peut être utilisée comme opérateur de généralisation dans un espace de recherche
- Il existe de nombreuses variantes de la θ-subsomption

Algorithme PROGOL [Muggleton 95]

- Approche descendante fondée sur l'implication inverse et la θ-subsomption
- Algorithme de type A^* avec une mesure de compression comme heuristique
- Les exemples positifs et négatifs sont fournis explicitement
- Biais sémantiques :
  - on spécifit le littéral à employer en tête de clause
  - on spécifie les littéraux utilisables dans le corps des clauses
- Background knowledge B (définition des littéraux utilisables dans les clauses) donné sous forme de clauses Prolog complètes

Algorithme PROGOL [Muggleton 95]

- L'utilsateur spécifie l'espace d'hypothèses (prédicats, fonctions, types et formats des arguments)
- WHILE il existe des exemples positifs DO
  - POUR chaque exemple positif e DO
    - construire la clause C_1 la plus spécifique qui implique e
    - trouver une clause C_2 qui θ-subsume C_1 telle que la mesure de compression soit maximale
    - retirer tous les exemples couverts par C_2
  - END (POUR)
- END (WHILE)

Extensions

- propositionnalisation, logique de description...
- hybridation avec les statistiques, apprentissage relationnel
- apprentissage de prédicats (Muggleton 2013)

Application au TAL

- apprentissage de règles en extraction d'information : le système Rapier (Califf & Mooney 03)
- quelques hybridations (Claveau...)

Intérêts

- lisibilité du résultat
- les règles résultat équivalent (sous certaines conditions) à un programme Prolog ($\Longrightarrow$ grande expressivité)

Limites

- espace de recherche énorme

  $\Longrightarrow$ “biais” nécessaires, par exemple :
  - hypothèses supplémentaires : toute donnée absente est négative (Prolog)
  - connaissance du domaine : les prédicats utilisables
  - heuristiques : choix des hypothèses
- phénomène de "transition de phase"

Plan

1. Introduction : motivations
2. Inférence Grammaticale (IG)
3. Programmation Logique Inductive (PLI)
4. Comparaison, perspectives

Comparaison, perspectives

Quelques points communs

- explicitent les conditions de possibilité de l'apprentissage
- avec des garanties de résultats
- la cible est un objet complexe
- parcours dans des espaces de recherche structurés
- stratégies de type : "moindre généralisé"
- exemples disponibles : positifs seuls (dur !) ou positifs/négatifs (plus facile)
- résultat (en principe) lisible, interprétable
- au prix d'une grosse complexité algorithmique
- mauvais passage à l'échelle, sensibilité au bruit, erreurs...
- hybridations...
- domaines encore actifs (conférences ICGI, ILP)

Comparaison, perspectives

Quelques perspectives

- hybridations possibles avec apprentissage statistique (déjà existantes)
  - objets hybrides : automates probabilistes, PCFG, réseaux...
  - méthodes hybrides
  - pour traiter les cas rares mais importants (améliorer les macro-averages !)
- devraient faire partie de la culture générale en apprentissage automatique