

Apprentissage automatique de règles

Synthèse de lectures

(Séminaire sur l'apprentissage automatique)

de

Benoit Lavoie

benoit@benoit-lavoie.ca

Programme de Doctorat en Informatique Cognitive

Université du Québec à Montréal

5 avril 2006

Table des matières

1. INTRODUCTION	1
2. APERÇU DE L'APPRENTISSAGE DE RÈGLES	1
3. RÈGLES PROPOSITIONNELLES ET RÈGLES DE LOGIQUE D'ORDRE UN	1
4. ALGORITHMES DE COUVERTURE SÉQUENTIELLE.....	2
CARACTÉRISTIQUES DES ALGORITHMES DE COUVERTURE SÉQUENTIELLE	2
L'ALGORITHME DE FÜRNKRANZ (1999): UN ALGORITHME GÉNÉRAL DE COUVERTURE SÉQUENTIELLE	2
PROCÉDURE PRINCIPALE DE L'ALGORITHME DE COUVERTURE SÉQUENTIELLE DE FÜRNKRANZ (1999)	3
PROCÉDURE DE RECHERCHE POUR LA MEILLEURE RÈGLE UTILISÉE DANS L'ALGORITHME DE COUVERTURE SÉQUENTIELLE DE FÜRNKRANZ (1999).....	4
5. BIAIS UTILISÉS POUR L'APPRENTISSAGE DE RÈGLES	6
BIAIS RELIÉS AU LANGAGE	6
BIAIS DE RECHERCHE RELIÉS AUX ALGORITHMES	6
BIAIS DE RECHERCHE RELIÉS AUX STRATÉGIES.....	7
BIAIS DE RECHERCHE RELIÉS AUX HEURISTIQUES D'ÉVALUATION	8
BIAIS DE CONTRÔLE DE LA SURSPÉCIALISATION DES HYPOTHÈSES (OVERFITTING AVOIDANCE)	9
6. APPRENTISSAGE D'ARBRES DE DÉCISION VERSUS APPRENTISSAGE DE RÈGLES.....	10
7. APERÇU DE TROIS SYSTÈMES ET D'UNE TECHNIQUE D'APPRENTISSAGE DE RÈGLES	11
SYSTÈME CN2	11
SYSTÈME FOIL.....	12
SYSTÈME GOLEM	12
TECHNIQUE D'APPRENTISSAGE DE RÈGLES DE LOGIQUE D'ORDRE UN BASÉE SUR LA RÉOLUTION INVERSÉE	12
8. AVANTAGES ET INCONVÉNIENTS DES TECHNIQUES D'APPRENTISSAGE DE RÈGLES	13
9. RÉFÉRENCES	13

1. Introduction

Ce document présente une synthèse de travaux portant sur l'apprentissage de règles de façon automatique et supervisée, faisant ressortir les grandes tendances dans ce domaine. Ce travail a été réalisé dans le cadre d'un séminaire sur l'apprentissage automatique (DIC9380) du programme de Doctorat en Informatique Cognitive de l'Université du Québec à Montréal (UQAM).

2. Aperçu de l'apprentissage de règles

Dans le contexte de l'apprentissage automatique (Lounis, 2006a), l'apprentissage de règles se définit généralement comme l'induction automatique et supervisée, à partir de données, d'une fonction cible spécifiée par des règles de type *si [prémisses] alors [conclusion]* (Mitchell, 1997; Lounis, 2006b). Les représentations à base de règles ont généralement l'avantage d'être considérées comme les représentations étant les plus expressives parmi celles utilisées en apprentissage automatique et les plus compréhensibles pour l'humain (Mitchell, 1997).

3. Règles propositionnelles et règles de logique d'ordre un

Les techniques d'apprentissage de règles peuvent être catégorisées par le type de règles inférées (Mitchell, 1997; Lounis, 2006b). On distingue deux grandes catégories de règles inférées par ces techniques: les règles propositionnelles (ou d'ordre zéro) et les règles de logique d'ordre un. Les règles propositionnelles sont les plus simples. Ces règles peuvent lier des propositions (formées de noms et de valeurs d'attributs ou de constantes) avec certains opérateurs logiques (conjonction, disjonction, et implication) et peuvent appliquer des opérations de comparaison ($<$, $>$, $=$, \leq , \geq) sur les valeurs des attributs et des constantes. Ces règles ne font toutefois pas référence à des variables ou à des relations génériques entre attributs et constantes (i.e. relations autres que les opérateurs logiques et les opérateurs de comparaison).

Les règles de logique d'ordre un étendent le formalisme des règles propositionnelles en permettant la représentation de variables et de relations génériques (fonction ou prédicat) pouvant relier constantes (dont les attributs) et variables. De façon générale, les règles de logique supportées par les techniques d'apprentissage de règles sont basées sur des variantes des clauses de Horn d'ordre un. Ce type de clause est défini comme une disjonction de littéraux (termes positifs ou négatifs, chacun pouvant consister en une constante, une variable, une fonction ou un prédicat), et où un seul des termes est positif. Cette clause peut être reformulée en règle de format général *si [prémisses] alors [conclusion]*: la conclusion de cette règle est formée par le terme positif et les prémisses sont formées de la conjonction des autres termes mis sous forme positive. Les clauses de Horn (et plus généralement les règles de logique d'ordre

un) sont plus puissantes que les règles propositionnelles; l'utilisation de variables et la mise en relation des variables et/ou constantes (par les fonctions ou prédicats) dans les clauses de Horn permettent souvent de représenter des règles de façon plus concise qu'avec des règles propositionnelles. Ces fonctionnalités (variables et relations) permettent de spécifier des règles qui peuvent être impossibles ou difficiles à spécifier avec le formalisme des règles propositionnelles.

Comme indiqué plus haut, on distingue deux principales catégories de techniques d'apprentissage de règles: (i) les techniques d'apprentissage de règles propositionnelles et les techniques d'apprentissage de règles de logique d'ordre un. Des techniques pour chacune de ces catégories sont présentées à la Section 7. La prochaine section décrit un type d'algorithme général utilisé par plusieurs techniques d'apprentissage de règles.

4. Algorithmes de couverture séquentielle

Caractéristiques des algorithmes de couverture séquentielle

Les techniques d'apprentissage de règles sont généralement basées sur des algorithmes de couverture séquentielle (Mitchell, 1997; Lounis, 2006b). L'apprentissage de règles à partir de ce type d'algorithme peut être caractérisé par les trois étapes suivantes: (1) à partir d'un ensemble d'exemples d'entraînement, l'algorithme apprend une règle couvrant un certain nombre d'exemples positifs; (2) l'algorithme supprime ces exemples positifs de l'ensemble d'entraînement; (3) puis l'algorithme répète itérativement (1-3) jusqu'à ce qu'il ne reste plus d'exemple positif à couvrir.

L'algorithme de Fürnkranz (1999): un algorithme général de couverture séquentielle

Les caractéristiques mentionnées ci-haut peuvent s'appliquer aussi bien à des techniques d'apprentissage de règles propositionnelles qu'à des techniques d'apprentissage de règles de logique d'ordre un. Mitchell (1997, section 10.2) présente un algorithme de couverture séquentielle très souvent cité dans la littérature. Toutefois, le pseudocode de l'algorithme présenté par Mitchell n'est directement applicable que pour l'apprentissage de règles propositionnelles de façon descendante (voir plus bas pour des détails sur les biais d'apprentissage). Fürnkranz (1999) présente une version alternative de cet algorithme qui peut s'appliquer (i.e. s'instantier) aussi bien pour l'apprentissage de règles propositionnelles que pour l'apprentissage de règles de logique d'ordre un. De plus, l'algorithme permet de supporter différents biais de recherche dont les recherches d'hypothèses de règles de façon descendante ou ascendante. L'article de Fürnkranz décrit sommairement comment l'algorithme proposé peut être instantié pour des systèmes basés sur des techniques de règles propositionnelles telles que AQ de Michalski (citation – Mitchell, 1997) et CN2 (Clark & Niblett, 1989; Clark & Boswell, 1991) ou des techniques de règles de logique

d'ordre un tel que FOIL (Quinlan , 1990; Quinlan & Cameron-Jones, 1993) et GOLEM (Muggleton & Feng, 1990). Les deux prochaines sous-sections décrivent deux des procédures de l'algorithme de couverture séquentielle présenté par Fürnkranz (1999). Il est à noter que Fürnkranz n'utilise pas le terme de *couverture séquentielle* pour référer à son algorithme mais cet algorithme possède toutes les caractéristiques que l'on attribue aux algorithmes de couverture séquentielle: l'algorithme de Fürnkranz (1999) ressemble d'ailleurs à plusieurs égards à celui décrit par Mitchell (1997).

Procédure principale de l'algorithme de couverture séquentielle de Fürnkranz (1999)

La Figure 1 de ce document illustre l'algorithme de la procédure principale de l'algorithme de couverture séquentielle décrit par Fürnkranz (1999). Cette procédure appelée SEPARATEANDCONQUER correspond à la procédure SEQUENTIAL-COVERING décrite par Mitchell (1997), mais spécifiée de façon plus générale.

```

procedure SEPARATEANDCONQUER(Examples)
  Theory =  $\emptyset$ 
  while POSITIVE(Examples)  $\neq$   $\emptyset$ 
    Rule = FINDBESTRULE(Examples)
    Covered = COVER(Rule,Examples)
    if RULESTOPPINGCRITERION(Theory,Rule,Examples)
      exit while
    Examples = Examples \ Covered
    Theory = Theory  $\cup$  Rule
  Theory = POSTPROCESS (Theory)
  return(Theory)

```

Figure 1. Procédure principale de l'algorithme de couverture séquentielle de Fürnkranz (1999)

Cet algorithme décrit à un niveau global le traitement pour l'apprentissage de règles (aussi bien propositionnelles que de logique d'ordre un). Cet algorithme réfère à plusieurs procédures dont l'instantiation variera d'une technique d'apprentissage à une autre. Certaines de ces procédures sont équivalentes à celles retrouvées dans l'algorithme de Mitchell (e.g. FINDBESTRULE décrite plus bas). Par contre, d'autres procédures ne se retrouvent pas dans l'algorithme de Mitchell (e.g. RULESTOPPINGCRITERION, POSTPROCESS, etc.); ces procédures qui ne sont utilisées que dans certaines techniques doivent être considérées comme étant optionnelles.

La procédure SEPARATEANDCONQUER illustrée à la Figure 1 initialise à vide une théorie (ou l'ensemble résultant des règles apprises). Puis, tant qu'un ensemble donné d'exemples contient des exemples positifs, les opérations suivantes sont appliquées séquentiellement: l'algorithme (1) recherche la meilleure règle (FINDBESTRULE); (2) identifie les exemples couverts par cette règle (COVER); (3) vérifie

un critère d'arrêt de recherche (RULESTOPPINGCRITERION); (4) élimine les exemples couverts de l'ensemble courant d'exemples; et (5) ajoute la règle à l'ensemble courant de règles (théorie). Après l'itération, les règles peuvent être post-traitées. Selon les techniques, le post-traitement peut consister à appliquer des procédures de post-élagage de règles (voir Section 5), ajouter des règles par défaut, etc.

Procédure de recherche pour la meilleure règle utilisée dans l'algorithme de couverture séquentielle de Fürnkranz (1999)

La Figure 2 illustre procédure `FINDBESTRULE` utilisée par procédure principale de l'algorithme de couverture séquentielle décrit par Fürnkranz (1999; voir Figure 1). `FINDBESTRULE` tente de trouver la meilleure règle pour un ensemble donné d'exemples à partir de critères de sélection et d'évaluation d'hypothèses de règles. Cette procédure correspond à la procédure `LEARN-ONE-RULE` présentée par Mitchell (1997) mais formulée de façon à pouvoir être appliquée par une plus grande variété de techniques d'apprentissage de règles; la procédure `LEARN-ONE-RULE` est moins générique dans le sens où elle assume l'apprentissage de règles propositionnelles basé sur une technique de fouille d'hypothèses de règles allant des plus génériques aux plus spécifiques.

```

procedure FINDBESTRULE(Examples)
  InitRule = INITIALIZERULE(Examples)
  InitVal = EVALUATERULE(InitRule)
  BestRule = <InitVal,InitRule>
  Rules = {BestRule}
  while Rules ≠ ∅
    Candidates = SELECTCANDIDATES(Rules, Examples)
    Rules = Rules \ Candidates
    for Candidate ∈ Candidates
      Refinements = REFINERULE(Candidate, Examples)
      for Refinement ∈ Refinements
        Evaluation = EVALUATERULE(Refinement, Examples)
        unless STOPPINGCRITERION(Refinement, Examples)
          NewRule = <Evaluation,Refinement>
          Rules = INSERTSORT(NewRule, Rules)
          if NewRule > BestRule
            BestRule = NewRule
      Rules = FILTERRULES(Rules, Examples)
  return(BestRule)

```

Figure 2.

Procédure de sélection de la meilleure règle de l'algorithme de couverture séquentielle de Fürnkranz (1999)

La procédure `FINDBESTRULE` consiste aux opérations suivantes:

- (1) Une règle initiale (InitRule) est d'abord instanciée (INITIALIZERULE). Dans le cas des techniques dont la fouille se fait allant d'hypothèses de règles plus génériques vers de plus spécifiques (cas général trouvé dans des systèmes comme CN2 et FOIL – voir détails plus bas), l'initialisation se fait en sélectionnant la règle la plus générique possible (quelque chose est vrai inconditionnellement). Dans le cas des techniques dont la fouille se fait allant d'hypothèses de règles plus spécifiques vers de plus génériques (e.g. le système GOLEM), l'initialisation peut se faire à partir de règles très spécifiques tirées directement des exemples. L'initialisation de règle dans la procédure LEARN-ONE-RULE présentée par Mitchell (1997) assume une recherche descendante et fait l'initialisation à partir d'une règle générique.
- (2) La règle initiale est évaluée (EVALUATERULE) selon un critère donné (e.g. gain d'information ou autre – voir détails plus bas). Cette règle et son résultat d'évaluation sont sauvegardés dans une variable indiquant la meilleure règle courante (BestRule) et est insérée dans une liste de règles (Rules) utilisées pour la fouille. Selon les techniques employées (e.g. différentes versions du système CN2) cette liste de règles peut être ordonnée ou non.
- (3) L'algorithme effectue une fouille itérative à partir de la liste de règles courantes. À chaque itération:
 - (3.1) Un sous-ensemble des règles candidates est sélectionné (SELECTCANDIDATES);
 - (3.2) Les règles candidates sont enlevées de la liste de règles;
 - (3.3) Chaque règle candidate est ensuite raffinée (REFINERULE), produisant un ensemble de raffinements (les types et le nombre de raffinements variant selon la technique employée). Chacun de ces raffinements est ensuite évalué (EVALUATERULE). Si un raffinement n'est pas filtré par un certain critère (STOPPINGCRITERION), il est ajouté à la liste de règles à explorer, et il remplace la meilleure règle courante si son évaluation est meilleure;
 - (3.4) Les règles courantes peuvent optionnellement être filtrées; selon les biais de recherche reliés aux algorithmes (voir détails à la Section 5), une ou plusieurs règles peuvent être raffinées au cours de la prochaine itération.
- (4) La meilleure règle courante est retournée.

La procédure FINDBESTRULE et les sous-procédures qu'elle utilise sont implémentées de différentes façons selon les techniques employées. La section 5 décrit quelques uns des biais utilisés par les

techniques d'apprentissage de règles. La section 7 décrit quelques caractéristiques pour trois systèmes d'apprentissage de règles (CN2, FOIL et GOLEM) qui sont basés sur un algorithme de couverture séquentielle. Cette section présente également une technique d'apprentissage basée sur la résolution inversée et qui est utilisée par GOLEM.

5. Biais utilisés pour l'apprentissage de règles

Les techniques d'apprentissage de règles utilisent des biais qui aident à guider et à simplifier la tâche d'apprentissage. Fürnkranz (1999) distingue trois types de biais décrits plus en détails plus bas dans cette section:

- *Biais reliés au langage*: ces biais concernent les contraintes reliées au langage d'expression et définissent l'espace de recherche.
- *Biais reliés à la recherche*: ces biais concernent les contraintes concernant la façon dont l'espace de recherche est fouillé. Ils incluent des biais de recherche liés aux algorithmes, aux stratégies, et aux heuristiques.
- *Biais reliés au contrôle de surspécialisation (overfitting avoidance)*: ces biais concernent les heuristiques utilisées afin d'éviter la surspécialisation des résultats (règles) d'apprentissage

Biais reliés au langage

Les biais reliés au langage portent sur des restrictions du formalisme des exemples et des règles d'apprentissage. De façon générale, le formalisme des exemples et des règles d'apprentissage sera restreint soit aux règles propositionnelles, soit aux règles de logique d'ordre un. Les règles propositionnelles peuvent être restreintes, par exemple au niveau des opérateurs utilisés pour les comparaisons et les énumérations de valeurs; par exemple le système AQ supporte un langage d'expression assez riche mais toutefois bien délimité (citation – Fürnkranz, 1999). Les règles de logique d'ordre un peuvent être limitées aux clauses de Horn par exemple, et restreintes au niveau des étiquettes utilisées pour les prédicats et fonctions des littéraux et au niveau des variables (voir les systèmes FOIL et GOLEM à la Section 7). Dans l'algorithme général présenté à la Section 4, certaines procédures peuvent être implémentées en fonction des biais reliés au langage: e.g. initialisation de règles (e.g. INITIALIZERULE), raffinement de règles (REFINERULE), etc.

Biais de recherche reliés aux algorithmes

Les biais de recherche reliés aux algorithmes portent sur les techniques d'exploration de l'espace de recherche. Fürnkranz (1999) décrit les algorithmes de recherche de type suivant: *hill-climbing* (escalade

de colline), *beam search* (recherche en faisceau), *best-first search* (recherche du meilleur en premier) et recherche stochastique. Dans l'algorithme général présenté à la Section 4, ces techniques de recherche peuvent être implémentées dans la procédure FINDBESTRULE par différents moyens dont l'itération sur les règles candidates, la sous-procédure REFINERULE, et la sous-procédure FILTERRULES.

La recherche *hill-climbing* est la technique la plus souvent employée; elle est supportée par exemple dans les techniques AQ, CN2, FOIL et GOLEM. Cette recherche consiste à sélectionner l'hypothèse de règle dont l'évaluation est la meilleure dans un espace donné et à poursuivre la recherche en raffinant cette règle récursivement. La recherche *hill-climbing* est limitée par les maximums locaux mais elle est souvent utilisée à cause de son efficacité.

La recherche *beam search* est également assez souvent employée; elle est supportée par exemple par AQ et CN2. Ce type de recherche ajoute à la recherche *hill-climbing* la possibilité d'explorer un nombre N (une ou plusieurs) des meilleures hypothèses de règles. Le nombre N correspond à la largeur du faisceau de recherche: la valeur 1 correspondant à une recherche *hill-climbing*. La recherche *beam search* a l'avantage de pouvoir réduire (sans généralement l'éliminer) la myopie relative de la recherche *hill-climbing* qui est limitée par les minimums locaux. La recherche *beam search* est toutefois plus coûteuse en temps de recherche.

La recherche *best-first search* sélectionne la meilleure hypothèse candidate ainsi que tout ses raffinements à moins qu'un critère d'élagage (sous-procédure STOPPINGCRITERION de la Figure 2) ne soit utilisé. En partant d'une hypothèse générale et en n'utilisant aucun critère d'élagage, la recherche *best-first search* peut trouver une règle optimale. Ce type de recherche est toutefois généralement très coûteux et n'est utilisé que dans relativement peu de techniques d'apprentissage de règles.

La recherche stochastique consiste généralement à introduire un certain degré d'aléatoire dans le raffinement de règle (sous-procédure REFINERULE de la Figure 2); cela vise à tenter d'éviter les minimums locaux tout en évitant également les recherches coûteuses. La recherche stochastique n'est utilisée que par peu de techniques d'apprentissage de règles.

Biais de recherche reliés aux stratégies

Les biais de recherche reliés aux stratégies se rapportent aux directions globales de recherche: recherche descendante (*top-down*), ascendante (*bottom-up*), ou bi-directionnelle. Une recherche descendante consiste à partir d'une hypothèse de règle très générale et à spécifier graduellement cette règle. Ce type de recherche est le plus utilisé en apprentissage de règles: la recherche descendante est utilisée entre autres par AQ, CN2 et FOIL. Une recherche ascendante consiste à partir d'une ou de plusieurs

hypothèses spécifiques et à les rendre plus génériques. Ce type de recherche n'est supporté que par très peu de techniques d'apprentissage de règles (un exemple est le système GOLEM). Un problème lié à la recherche ascendante consiste au choix des hypothèses de départ; un grand nombre d'hypothèses initiales est possible à partir des exemples d'apprentissage. À l'inverse, la recherche descendante détermine trivialement l'hypothèse de départ en choisissant généralement l'hypothèse la plus générique (quelque chose est vrai inconditionnellement). La recherche bi-directionnelle combine recherche descendante et recherche ascendante. Il existe également peu de techniques supportant ce type de recherche.

Biais de recherche reliés aux heuristiques d'évaluation

Les heuristiques d'évaluation servent à déterminer la qualité des règles et à guider la fouille dans certaines régions de l'espace de recherche. Ces heuristiques peuvent être implémentées dans la sous-procédure EVALUATE-RULE utilisée par FINDBESTRULE (voir algorithme à la Section 4). Certaines de ces heuristiques telles que l'entropie sont très similaires à celles utilisées pour l'induction d'arbres de décision. Les heuristiques d'évaluation sont utilisées de différentes façons selon qu'il s'agisse d'induction d'arbres de décision ou d'apprentissage de règles: pour l'induction d'arbres de décision, ces heuristiques servent à évaluer la qualité moyenne d'ensembles disjoints contenant toutes les données restantes; pour l'apprentissage de règles, ces heuristiques sont utilisées pour évaluer la qualité de l'ensemble de données couvertes par chaque règle.

Il existe différents types d'heuristique d'évaluation de règle. Un premier type de base vise à mesurer la *couverture relative* de règle. La procédure FINDBESTRULE tente de trouver la règle couvrant le plus d'exemples (ou instances dans le cas de FOIL) positives, tout en couvrant le moins d'exemples négatifs. Ces heuristiques d'évaluation de base tentent d'évaluer les solutions en considérant ces deux objectifs. Parmi ces heuristiques on retrouve entre les suivantes:

- *entropie*: cette métrique correspond au degré d'incertitude de la valeur d'une classe. Dans le contexte des règles, l'entropie correspond au nombre de questions booléennes (vrai/faux) nécessaires afin de déterminer une classe à partir des exemples positifs et négatifs couverts par cette règle. L'entropie est utilisée dans les premières versions de CN2.
- *estimée de Laplace*: cette métrique vise à pénaliser les règles à faible couverture et assume un bruit dans les données; elle favorisera une règle couvrant quelques exemples négatifs mais plusieurs exemples positifs sur une règle couvrant aucun exemple négatif mais peu d'exemples positifs. Cette métrique est utilisée dans de récentes versions de CN2.

Un deuxième type d'heuristique d'évaluation de règle consiste à la mesure de la complexité. Pour des

couvertures relatives similaires les règles les plus simples sont préférées aux plus complexes. Une des heuristiques utilisées pour mesurer la complexité d'une règle est la suivante:

- *longueur de description minimum*: cette métrique tente de trouver le rapport entre la complexité d'une règle et son exactitude. Une variante de cette métrique est utilisée par FOIL.

Un troisième type d'heuristique d'évaluation de règle consiste au gain de couverture qui mesure la différence entre une règle donnée et une version précédente de cette règle. Une des heuristiques utilisées pour mesurer le gain de couverture est la suivante:

- *gain d'information pondéré*: cette heuristique est basée sur la différence des contenus d'information des règles (logarithme négatif du rapport du nombre des exemples positifs sur le nombre de tous les exemples couverts), et sur une pondération de cette différence par le nombre d'exemples positifs. Cette heuristique est utilisée par FOIL.

Un quatrième type d'heuristique d'évaluation de règle s'applique à la détermination des littéraux utilisés dans les expressions de logique d'ordre un. Ces heuristiques portent sur des conditions d'occurrence de variables dans les expressions, visant à assurer l'instantiation des règles. Des heuristiques de ce type sont utilisées par FOIL et GOLEM.

Biais de contrôle de la surspécialisation des hypothèses (*overfitting avoidance*)

Les biais de contrôle de surspécialisation des hypothèses (*overfitting avoidance*) visent à permettre l'apprentissage de règles avec relativement peu de données ou avec des données bruitées tout en permettant d'obtenir des règles suffisamment générales. Pour se faire, ces biais favorisent la simplicité des règles parfois au détriment de la couverture ou de l'exactitude des règles.

Parmi les biais de contrôle de la surspécialisation des hypothèses, on retrouve des heuristiques de pré-élagages de règles. Celles-ci peuvent être implémentées dans la procédure STOPPINGCRITERION de l'algorithme présenté à la Figure 2. L'idée générale de ces biais est de cesser le raffinement de règles malgré que ces règles puissent être encore trop générales: les règles peuvent alors couvrir quelques exemples négatifs si le coût de les exclure est considéré comme trop important. Parmi les heuristiques utilisées pour le pré-élagage on retrouve les suivantes:

- *critère de pureté minimum*: cette heuristique utilisée par certaines techniques stochastiques requière qu'un certain pourcentage d'exemples couverts par une règle soit positif.
- *restriction sur la longueur d'encodage*: cette heuristique utilisée par FOIL est basée sur la

longueur de description minimum. Elle tente d'éviter des règles compliquées couvrant trop peu d'exemples.

- *test de signifiante*: cette heuristique utilisée par CN2 teste pour des différences significatives entre la distribution des exemples positifs et négatifs couverts par la règle et la distribution de tous les exemples positifs et négatifs.

Parmi les biais de contrôle de la surspécialisation des hypothèses, on retrouve également des heuristiques de post-élagages de règles. Celles-ci peuvent être implémentées dans la procédure POSTPROCESS de l'algorithme présenté à la Figure 1. Une heuristique commune de post-élagage vise à éliminer les conditions redondantes des règles et à éliminer les règles inutiles: l'approche consiste à procéder à l'élimination et à tester sur un ensemble d'entraînement distinct de celui utilisé pour la construction des règles afin de déterminer si il y a diminution significative de l'exactitude de la classification. Cette heuristique est employée par AQ.

6. Apprentissage d'arbres de décision versus apprentissage de règles

Les arbres de décisions (Lounis, 2006c; Lavoie, 2006) peuvent être interprétés comme des règles. L'interprétation d'un arbre de décision correspond à celui d'une forme normale disjonctive: chaque chemin menant de la racine de l'arbre vers une feuille peut s'interpréter comme une conjonction de valeurs d'attributs; les branches menant vers des feuilles dont la classe est similaire peuvent s'interpréter comme une disjonction. Des règles en forme normale disjonctive peuvent être extraites directement à partir des arbres de décision. Ces règles sont non-ambigües et insensibles à l'ordre mais peuvent être complexes et peuvent généralement être simplifiées automatiquement par des heuristiques.

Les techniques d'apprentissage de règles se distinguent des techniques d'apprentissage d'arbres de décision à différents niveaux dont les deux suivants (Mitchell, 1997; Fürnkranz, 1999):

- Plusieurs techniques d'apprentissage de règles permettent de produire des règles de logique d'ordre un alors que la majorité des techniques d'apprentissage d'arbres de décisions produisent des représentations correspondant à des règles propositionnelles (moins expressives). Il existe quelques techniques d'arbre de décisions qui permettent de traiter des règles logiques d'ordre un mais les règles extraites à partir des arbres sont généralement plus complexes que celles résultant des techniques d'apprentissage de règles (Fürnkranz, 1999, p.7).
- Les techniques d'apprentissage de règles permettent un chevauchement de règles alors que les techniques d'apprentissage d'arbres de décision ne le permettent pas (Mitchell, 1997). Pour les

techniques d'apprentissage de règles, à chaque itération d'apprentissage tous les attributs des exemples restants peuvent être considérés et être combinés indépendamment des combinaisons précédentes (en autant que la règle candidate permet de distinguer de nouveaux exemples positifs). Pour les techniques d'apprentissage d'arbres de décision, les tests des nœuds d'un arbre de décision visent à couvrir les valeurs des attributs des exemples d'entrées de façon exhaustive: les attributs dont les valeurs sont finies (e.g. attributs symboliques) ne sont ainsi généralement sélectionnés qu'au plus qu'une fois pour une branche donnée dans un arbre de décision.

7. Aperçu de trois systèmes et d'une technique d'apprentissage de règles

Cette section décrit quelques caractéristiques de trois systèmes d'apprentissage de règles (CN2, FOIL et GOLEM) qui ont été introduits dans les sections précédentes et qui sont basés sur l'algorithme général de couverture séquentielle (voir Section 4). Cette section présente également une technique d'apprentissage de règles de logique d'ordre un basée sur la résolution inversée et utilisée par GOLEM.

Systeme CN2

CN2 (Clark & Niblett, 1989; Clark & Boswell, 1991; Mitchell, 1999; Lounis, 2006b) est un système d'apprentissage de règles propositionnelles. CN2 intègre des fonctionnalités de deux types de systèmes: d'une part, il intègre des fonctionnalités du système AQ (voir Clark & Niblett (1989)) pour l'induction de règles propositionnelles de façon descendante (exploration *hill-climbing* ou *beam-search*); d'autre part il intègre des fonctionnalités du système d'induction d'arbres de décision ID3 (voir Clark & Niblett (1989)) dans l'utilisation de méthodes statistiques pour l'évaluation de règles, offrant ainsi une certaine robustesse au bruit dans les données. Les premières versions de CN2 (Clark & Niblett, 1989) utilisent l'entropie dans l'évaluation des règles alors que les versions plus récentes (Clark & Boswell, 1991) utilisent l'estimée de Laplace qui permet de meilleurs résultats lorsque les données sont bruitées (voir description de ces métriques à la Section 5). Les premières versions de CN2 produisent des règles ordonnées alors que les versions plus récentes permettent de produire des règles qui ne sont pas ordonnées. CN2 utilise un test de signifiante comme heuristique de pré-élagage (voir Section 5). L'algorithme de couverture séquentielle présentée par Mitchell (1999) s'applique bien au système CN2; à l'intérieur de la procédure de recherche de la meilleure règle, des attributs sont ajoutés itérativement aux hypothèses de règles propositionnelles courantes de façon à les spécialiser jusqu'à ce qu'elle atteigne un niveau de couverture appropriée. CN2 induit des règles permettant de prédire quand des concepts sont vrais ou faux; des règles par défaut sont utilisées pour prédire quand des concepts sont faux.

Système FOIL

FOIL (Quinlan, 1990; Quinlan & Cameron-Jones, 1993; Mitchell, 1999; Lounis, 2006b) est un système d'apprentissage de clauses de Horn. Comme CN2, FOIL induit les règles de façon descendante et utilise des métriques statistiques pour évaluer et pré-élaguer les règles. Toutefois les méthodes statistiques utilisées par FOIL sont différentes de celles utilisées par CN2: pour l'évaluation de règles FOIL utilise une métrique basée sur le gain pondéré d'information (voir Section 5) mais adaptée aux littéraux et variables; pour le pré-élagage FOIL utilise une métrique basée sur la longueur de description minimum (voir également Section 5). FOIL ne supporte que l'exploration *hill-climbing* (alors que CN2 supporte à la fois l'exploration *hill-climbing* et l'exploration *beam-search*). FOIL ne recherche également que des règles prédisant quand les concepts cibles sont vrais (alors que CN2 supporte également les règles prédisant quand les concepts sont faux). L'apprentissage de règles se fait à partir d'une règle dont la conclusion est formée par un prédicat. Lors du raffinement de règles, FOIL ajoute des prémisses aux règles sous forme de littéraux tel que chaque littéral est formé d'un prédicat existant, de la fonction d'égalité, ou de la négation de ce prédicat ou de cette fonction d'égalité. Une heuristique permet d'introduire une nouvelle variable à la fois qui n'est pas liée à la conclusion ou aux littéraux des prémisses courantes.

Système GOLEM

GOLEM (Muggleton & Feng, 1990) est un système d'apprentissage de clauses de Horn. Contrairement à CN2 et FOIL, GOLEM applique une recherche de règles façon ascendante. GOLEM initialise la recherche à partir d'une paire d'exemples positifs choisis aléatoirement dont il généralise des règles par exploration de type *hill-climbing*. La technique d'exploration ascendante utilisée par GOLEM est basée sur la résolution inversée (voir prochaine sous-section) et sur la procédure de *généralisation relative du moins général* (*relative least general generalization*): cette procédure combine deux ou plusieurs clauses en une seule et peut appliquer des opérations de substitutions sur les variables.

Technique d'apprentissage de règles de logique d'ordre un basée sur la résolution inversée

L'induction peut être vue comme un processus de déduction inversée. Dans ce contexte, il est possible d'inverser le processus de résolution utilisé pour la déduction automatique afin de l'appliquer pour l'induction automatique de règles (Mitchell, 1999; Lounis, 2006b). Si une base de connaissances contient deux clauses logiques (C1 et C2) contenant une disjonction de littéraux dont un littéral (L) apparaît dans les deux clauses mais avec des signes opposés, la règle de résolution permet de combiner ces clauses en une seule (C) tout en éliminant ce littéral. La résolution inversée consiste à procéder de la manière inverse et à introduire la clause C2 (par exemple) à partir de la clause C et C1. Dans l'application de la

résolution inversée, un littéral apparaissant dans C1 et non dans C est ajouté sous forme positive dans C2, et un littéral apparaissant dans C1 et non dans C est ajouté sous forme négative dans C2 (Lounis, 2006b).

8. Avantages et inconvénients des techniques d'apprentissage de règles

Les techniques d'apprentissage de règles ont certains avantages dont les suivants:

- L'apprentissage de règles permet de produire des représentations qui sont parmi les plus expressives en apprentissage automatique (Mitchell, 1997); par exemple, les règles de logique d'ordre un produites par certaines techniques d'apprentissage de règles ont un formalisme plus puissant que celui des règles propositionnelles qui peuvent généralement (il y a des exceptions) être obtenues à partir d'arbres de décision.
- Les représentations à base de règles sont parmi les représentations les plus compréhensibles pour l'humain (Mitchell, 1997): cela peut faciliter la validation ou l'utilisation de ces règles par des experts.

Les techniques d'apprentissage de règles ont certains désavantages dont les suivants:

- Les techniques d'apprentissage de règles ne produisent généralement pas de règles optimales (Fürnkranz, 1999); la recherche de type *hill-climbing* fréquemment utilisée par les techniques est efficace mais est limitée par les minimums locaux. La recherche de type *best-first search* qui pourrait garantir des règles optimales est généralement trop coûteuse à appliquer. La recherche de type *beam search* est une solution intermédiaire (en terme de qualité de règle et performance) parfois utilisée sans toutefois garantir de trouver règles optimales.
- La puissance d'expression des règles d'ordre un qui peuvent être inférées demeure limitée; la restriction aux clauses de Horn limite le pouvoir d'expression des règles.

9. Références

Clark, Peter; Tim Niblett (1989). The CN2 Induction Algorithm. *Machine Learning Journal*, 3(4), pp. 261–283.

Clark, Peter; Robin Boswell (1991). Rule Induction with CN2: Some Recent Improvement. *Proceedings of the Fifth European Working Session on Learning – Lecture Notes in Artificial Intelligence*, Springer-Verlag, volume 482, pp. 151–163.

- Fürnkranz, Johannes (1999). Separate-and-Conquer Rule Learning. *Artificial Intelligence Review*, volume 13, pp. 3–54.
- Lavoie, Benoit (2006). *Arbres de décisions*. Synthèse de lectures (Séminaire sur l'apprentissage automatique), Programme de Doctorat en Informatique Cognitive, Université du Québec à Montréal.
- Lounis, Hakim (2006a). *Apprentissage automatique*. Notes de cours (Séminaire sur l'apprentissage automatique), Programme de Doctorat en Informatique Cognitive, Université du Québec à Montréal.
- Lounis, Hakim (2006b). *Apprentissage de règles*. Notes de cours (Séminaire sur l'apprentissage automatique), Programme de Doctorat en Informatique Cognitive, Université du Québec à Montréal.
- Lounis, Hakim (2006c). *Arbres de décision*. Notes de cours (Séminaire sur l'apprentissage automatique), Programme de Doctorat en Informatique Cognitive, Université du Québec à Montréal.
- Mitchell, Tom (1997). Learning set of rules. Chapter 10 of *Machine Learning*, McGraw-Hill, pp. 274–306.
- Muggleton, Stephen; Cao Feng (1990). Efficient Induction Of Logic Programs. *Proceedings of the 1st Conference on Algorithmic Learning Theory*. Tokyo, pp.368–381.
- Quinlan, John Ross (1990). Learning logical definitions from relations. *Machine Learning*, 5(3), pp. 239–266.
- Quinlan, John Ross; R. Mike Cameron-Jones (1993). FOIL: A midterm report. *Proceedings of the 6th European Conference on Machine Learning – Lecture notes in Computer Science*, volume 667, Springer-Verlag, pp. 3–20.