

# Large-scale inference and learning algorithm

Jean-Christophe Janodet  
University of Evry, France

# Objectifs

- Revoir la théorie des langages formels :
  - Automates, CFG, transducteurs, HMM, ...
  - Formalismes probabilistes ou non : WFSM / WFST
  - Relations entre ces formalismes
- Etudier les principaux algorithmes exploitant ces machines : parsing, inference, équivalence, apprentissage
- Applications en Traitement de la langue

# Prisme : Inférence grammaticale

- Ce que ça permet :
  - Apprentissage à partir de données structurées :
    - mots, arbres, graphes, etc.
  - Recherche d'une hypothèse *cible* particulière :
    - automate, grammaire, standard ou stochastique
- Ce que ça ne permet pas (directement) :
  - problèmes de classification
  - problèmes de classement

# La communauté en IG

- International Community in Grammatical Inference (ICGI)
- Communauté hétéroclite, avec des chercheurs travaillant en
  - Pattern recognition, Machine learning, Computational linguistics (or not), Bioinformatics
  - Algorithmics, Language theory, Information theory
- Géographie de la communauté :
  - Japan, Spain, Germany, Great Britain, Belgium, The Netherlands, USA, Australia, France

# Principaux événements

- The International *Colloquium* in Grammatical Inference (ICGI) tous les 2 ans depuis 1994
- Workshops + écoles en alternance à ECML, NIPS, IJCAI
- Compétitions d'algorithmes :
  - Abbadingo (1998), Omphalos (2004)
  - Gecco (2004), Tenjinno (2006)
  - Zulu (2010), Stamina (2010)

# *Surveys et livres*

## ■ Surveys:

- D. Angluin & C. Smith. *Inductive inference: Theory and practice*. ACM computing surveys, 15(3):237-269, 1983
- L. Miclet. *Syntactic and structural pattern recognition*. Chap. GI, 237-290. World Scientific, 1990
- Y. Sakakibara. *Recent advances of GI*. Theoretical Computer Science, 185:15-45, 1997

## ■ Book:

- C. de la Higuera. *GI: Learning automata & grammars*. Cambridge University Press, 2010.

# Caractéristiques d'un problème d'Inférence Grammaticale

- Ce que ça permet :
  - Apprentissage à partir de données structurées :
    - mots, arbres, graphes, etc.
  - Recherche d'une hypothèse *cible* particulière :
    - automate, grammaire, standard ou stochastique
- Ce que ça ne permet pas (directement) :
  - problèmes de classification
  - problèmes de classement

# Exemple 1

## Modélisation des systèmes

- Contexte :
  - Réingénierie : reconstruction de la spécification d'un système, « exploration » d'un site web
  - Certification : preuve du bon fonctionnement d'un système critique
  - Débogage, tests logiciels, tests matériels
- Réalisation :
  - Modélisation sous la forme d'automates de Mealy, d'automates IO, d'automates d'interfaces, *etc*

# Exemple 2

## Natural Language Processing

- Contexte :
  - Modèle de langage attribuant une probabilité à toute phrase, correcte ou non
  - Traduction automatique, attribuant une probabilité à toute traduction, correcte ou non
- Réalisation :
  - Automate stochastique, grammaire probabiliste (PCFG), modèle de Markov (HMM), transducteur (stochastique)

# Cas général (1)

- On dispose d'une classe de grammaires :
  - de mots : AFD, AFN, AFER, automates de Moore, de Mealy, GHC, GHC linéaires déterministes, SRM, motifs, boules, expression régulière, ...
  - d'arbres, de graphes, de termes , ...
- Grammaire = tout périphérique permettant de reconnaître / engendrer / décrire un langage
- Deux versions : standard *vs* stochastique

# Cas général (2)

- Une des grammaires est sélectionnée : la grammaire *cible*, celle que cherche l'apprenant
- Le mode d'acquisition des données est fixé par l'application :
  - Elles peuvent être positives seulement, ou positives et négatives
  - Elles peuvent être subies (apprentissage passif) ou choisies (apprentissage actif)

# Cas général (3)

- Ce qu'« apprendre » veut dire est fixé, i.e., un paradigme d'apprentissage est donné :
  - Identification exacte :
    - Données subies : Identification à la limite (Gold 67, 78)
    - Données choisies : Query learning (Angluin 87)
  - Identification approximative :
    - Données subies : PAC-leaning (Valiant 84)
    - Données choisies : active learning

# 1<sup>er</sup> cas : Query learning

- D. Angluin. *Learning regular sets from queries and counterexamples*. Inform. and Computat. 75:87-106, 1987
- Cadre : identification exacte, données choisies
- Problème : deviner un DFA en posant des questions (requêtes) à un professeur
- Applications actuelles :
  - Tests de protocoles de sécurité ou de comm. (Shu & Lee 2007, Aarts et al 2014), détection de failles de sécurité dans les sites web (Hossen et al 2014)
  - Formalisation des langues rares (...)

# Mots

- Un *alphabet*  $\Sigma$  est un ensemble fini non vide de symboles qu'on appelle *lettres*. E.g.,  $\Sigma = \{0,1\}$ .
- Un *mot*  $w$  sur  $\Sigma$  est une séquence finie de lettres.  
E.g.,  $0, 1, 00, 10, 11010, \lambda, \dots$
- Notations:
  - $|w|$  : longueur du mot  $w$
  - $uv$  : concaténation des mots  $u$  et  $v$
  - $\Sigma^*$  dénote l'ensemble de tous les mots

# Langages

- Un langage est une sous-ensemble de  $\Sigma^*$
- Hiérarchie de Chomsky :
  - Couche 4: Langages finis
  - Couche 3: Langages réguliers
  - Couche 2: Langages hors-contextes
  - Couche 1: Langages sous-contextes
  - Couche 0: Langages récursivement énumérables
- De nombreux langages n'entrent pas dans cette hiérarchie (e.g., *pattern languages, multiple CFL*)

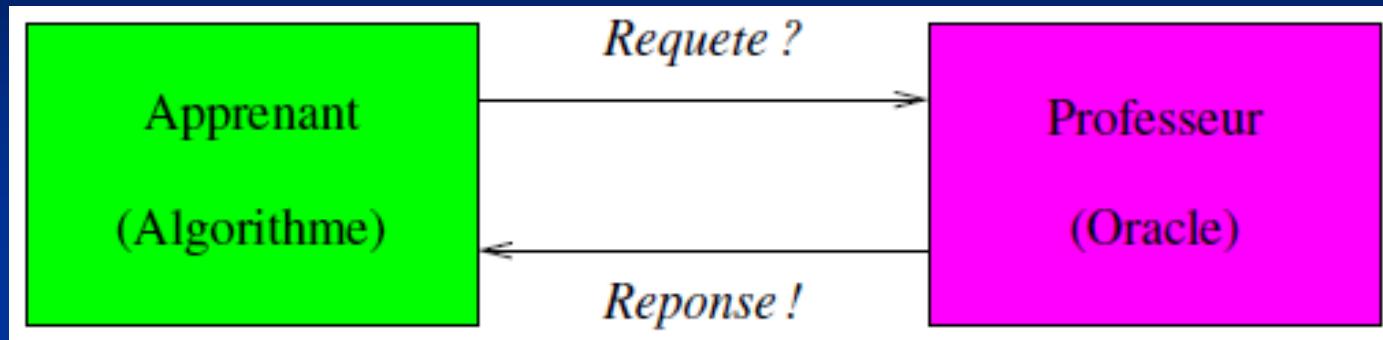
# Langages réguliers (1)

- Réguliers = rationnels = reconnaissables = reconnus par des automates
- Un AFD est un tuple  $A = \langle Q, \Sigma, i, F, \delta \rangle$  t.q.
  - $Q$  : ensemble fini d'états
  - $\Sigma$  : alphabet d'entrée
  - $i$  : état initial
  - $F$  : ensemble des états acceptants
  - $\delta : Q \times \Sigma \rightarrow Q$  : fonction de transition

# Langages réguliers (2)

- Taille de  $A = |Q|$  = nombre d'états
- Parsing : linéaire en  $|w|$
- Equivalence :  $O( n_1 \log n_1 + n_2 \log n_2 )$
- Propriétés « jolies », beaucoup d'algorithmes :
  - Déterminisation des AFN
  - Minimisation des AFD

# Paradigme d'Angluin



- Contexte :
  - L'oracle choisit un DFA, l'apprenant doit le trouver
- Règles :
  - L'apprenant pose des questions (requêtes) à l'Oracle
  - L'Oracle répond sans mentir
  - Le type de requêtes est fixé : appartenance, équivalence, . . .

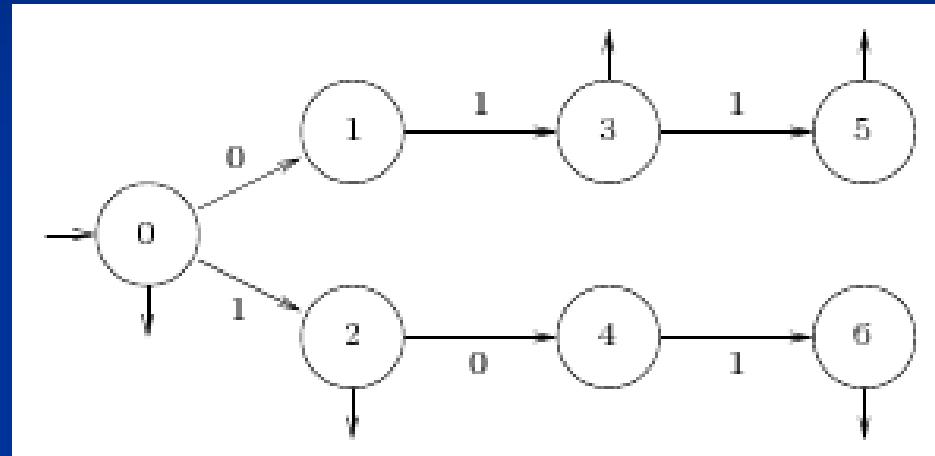
# Résultats

- Les AFD sont identifiables en temps polynomial à partir de requêtes d'appartenance et d'équivalence
- ... « polynomial » en :
  - la taille de l'AFD cible
  - la longueur du plus long contre-exemple
- L\* : algorithme de référence aujourd'hui
- Extensions théoriques aux GHC

## 2<sup>nd</sup> cas : appr. passif des AFD

- Exemple : On considère  $E_+ = \{ \lambda, 1, 01, 011, 101 \}$  et  $E_- = \{ 00, 10 \}$ .
- Problème : apprendre un AFD compatible avec les données, i.e., qui accepte tous les mots de  $E_+$  et rejette tous ceux de  $E_-$

# Problème mal formulé



- On calcule le PTA de  $E_+ = \{ \lambda, 1, 01, 011, 101 \}$
- Problème pas très intéressant ...

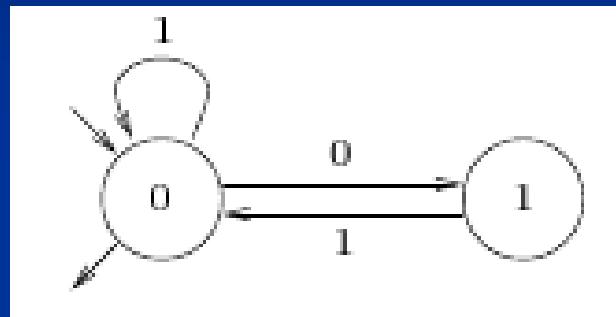
# Problème trop difficile

- Problème : on cherche le *plus petit* AFD compatible avec les données
- Théorème [Gold 78] :  
Le problème suivant est NP-complet :
  - Données : 1 entier  $N$ , 2 ensembles  $E_+$  et  $E_-$  de mots
  - Problème : Existe-t'il un AFD de moins de  $N$  états reconnaissant tous les mots de  $E_+$  et aucun de  $E_-$  ?

# Biais inductif

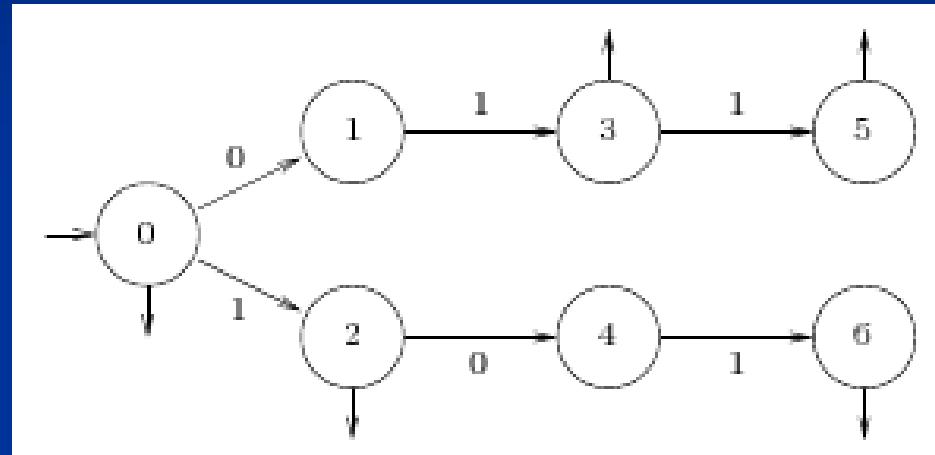
- On suppose que les données *caractérisent* la cible :
  - Les mots de  $E_+$  exercent tous les états, tous les états finaux, et toutes les transitions de l'AFD cible
  - Les mots de  $E_+$  et  $E_-$  permettent de distinguer deux états qui sont différents
- Déf. formelle : cf (de la Higuera '10)

# RPNI (Oncina '92)



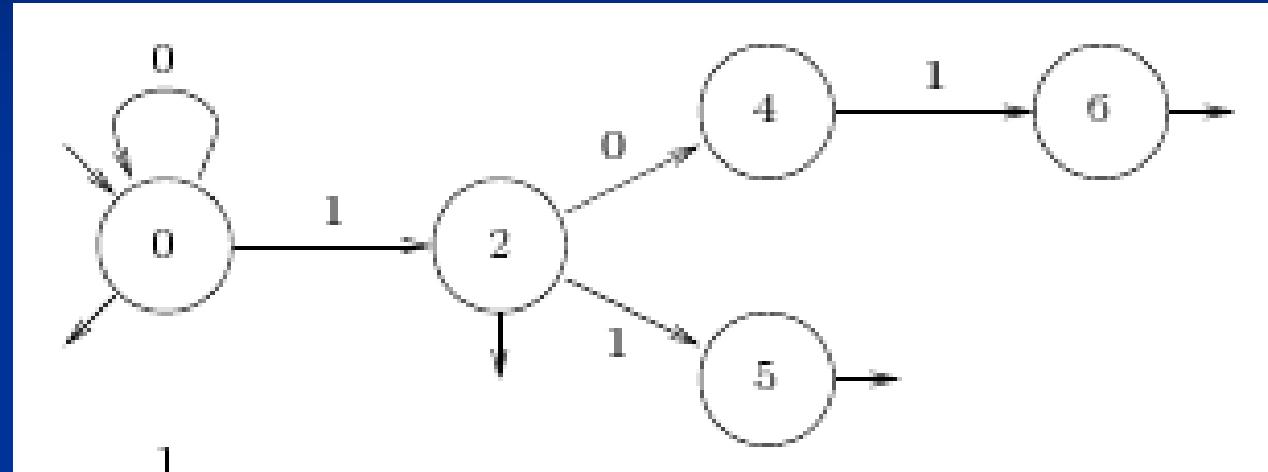
- On considère la cible précédente et l'échantillon  $E_+ = \{ \lambda, 1, 01, 011, 101 \}$  et  $E_- = \{ 00, 10 \}$ .

# Prefix Tree Acceptor



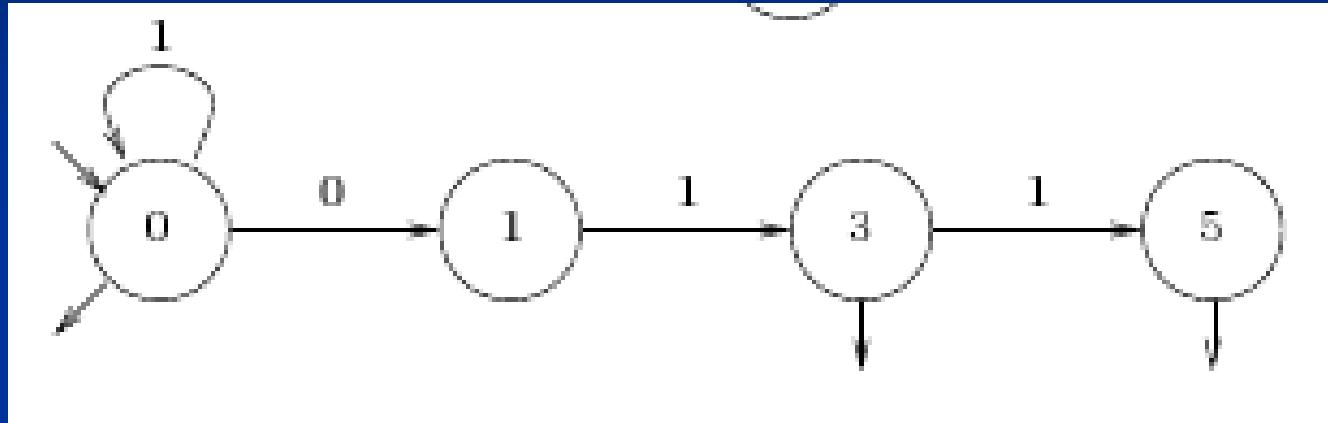
- On calcule le PTA de  $E_+ = \{ \lambda, 1, 01, 011, 101 \}$

# 1ère fusion (de 1 et 0)



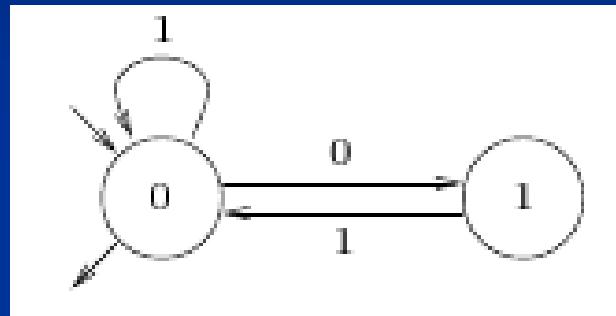
- On a  $E_- = \{ 00, 10 \}$  et 00 est reconnu
- La fusion est rejetée

# 2<sup>nde</sup> fusion (de 2 et 0)



- On a  $E_- = \{ 00, 10 \}$ , pas d'inconsistance
- La fusion est acceptée

# 3ème fusion (de 3 et 0)



- On a  $E_- = \{ 00, 10 \}$ , pas d'inconsistance
- La fusion est acceptée
- Miracle : c'est la cible

# Propriétés de RPNI

## ■ Théorème :

- L'AFD retourné par RPNI est toujours consistant avec l'échantillon initial
- Chaque AFD admet au moins un échantillon caractéristique de taille polynomial qui, s'il est inclus dans l'échantillon d'apprentissage, fait converger RPNI vers l'AFD minimal équivalent.

# Extensions de RPNI

- Approches *data-driven* : EDSM :
  - L'ordre des fusions d'états dépend de statistiques sur les données (*i.e.*, on favorise les fusions les plus sûres)
  - Arbitrairement mauvais sur un plan théorique, mais pas sur un plan pratique ...
- Extensions aux transducteurs de mots, aux automates d'arbres, *etc.*

# 3<sup>ème</sup> cas : appr. passif des GHC

- Pb. beaucoup plus difficile que pour le AFD
  - Dans le cas des AFD, relation profonde entre les données (mots) et les états de l'AFD (notion de résiduel), mais rien de ce genre pour les règles d'une grammaire : les données ne nous apportent pas d'info. sur la structure de la GHC cible

# Langages hors-contextes (1)

- Engendrés par des GHC
- Une GHC est un tuple  $G = \langle \Sigma, V, P, S \rangle$  t.q.
  - $\Sigma$  : alphabet d'entrée
  - $V$  : alphabet non terminal
  - $P$  : ensemble des productions de la forme  $X \rightarrow w$ , où  $w$  est un mot de  $(\Sigma \cup V)^*$
  - $S$  : symbole *start*

# Langages hors-contextes (2)

- Parsing :  $O(\|G\| |w|^3)$  :
  - Passage à la forme normale de Chomsky
  - Algo. CYK par programmation dynamique
- Equivalence : indécidable
- Nombreuses « mauvaises » propriétés :
  - problèmes d'ambiguité, de déterminisme
  - problèmes de non linéarité
  - problèmes de clôture

# Lien données / GHC (Clark '07)

- Congruence syntaxique : soit  $L$  un langage ;  
 $u \approx v$  si pour tous mots  $l, r$ , ( $lur \in L$  ssi  $lvr \in L$ )
- Intuition : deux mots équivalents sont générés par le même non terminal (Harris '57)
- « Congruence » faible : soit  $L$  un langage ;  
 $u \sim v$  si il existe  $l, r$ , ( $lur \in L$  ssi  $lvr \in L$ )
- $L$  est substituable si pour tout mot  $u, v$   
 $si u \sim v \text{ alors } u \approx v$

# Algorithme SGL (Clark '07) (1)

- On suppose que le langage cible est substituable
- On a un échantillon positif de mots  $E_+$
- On construit un graphe :
  - Nœuds : facteurs des mots de  $E_+$
  - Arêtes :  $u$  et  $v$  sont liés si  $u \sim v$ , ie, ils apparaissent dans le même contexte

# Algorithme SGL (2)

- Construction de la grammaire :
  - Non terminaux : composante connexe du graphe
  - Starter : composante contenant tous les mots de  $E_+$
  - Règles :
    - $[[\alpha]] \rightarrow \alpha$  pour toute lettre  $\alpha$
    - $[[uv]] \rightarrow [[u]] [[v]]$  pour tout facteur  $uv$  de longueur  $\geq 2$
    - $[[u]] \rightarrow [[v]]$  pour tout  $u \sim v$