

Machine Learning

Inductive and statistical learning of formal grammars

Pierre Dupont

pdupont@info.ucl.ac.be

Goal: to give the *learning ability* to a machine

Design programs the performance of which improves over time

Inductive learning is a particular instance of machine learning

- Goal: to find a *general law* from *examples*
- Subproblem of *theoretical computer science*, *artificial intelligence* or *pattern recognition*

– Typeset by FoilTEX –

Pierre Dupont

2

2002

Grammar Induction

Outline

- Grammar induction definition
- Learning paradigms
- DFA learning from positive and negative examples
- RPNI algorithm
- Probabilistic DFA learning
- Application to a natural language task
- Links with Markov models
- Smoothing issues
- Related problems and future work

Pierre Dupont

1

2002

Grammar Induction

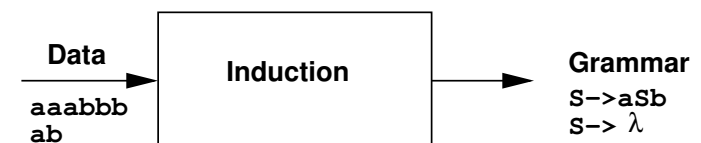
Grammar Induction or Grammatical Inference

Grammar induction is a particular case of inductive learning

The general law is represented by a *formal grammar* or an equivalent machine

The set of examples, known as *positive sample*, is usually made of *strings* or *sequences* over a specific *alphabet*

A *negative sample*, i.e. a set of strings not belonging to the target language, can sometimes help the induction process



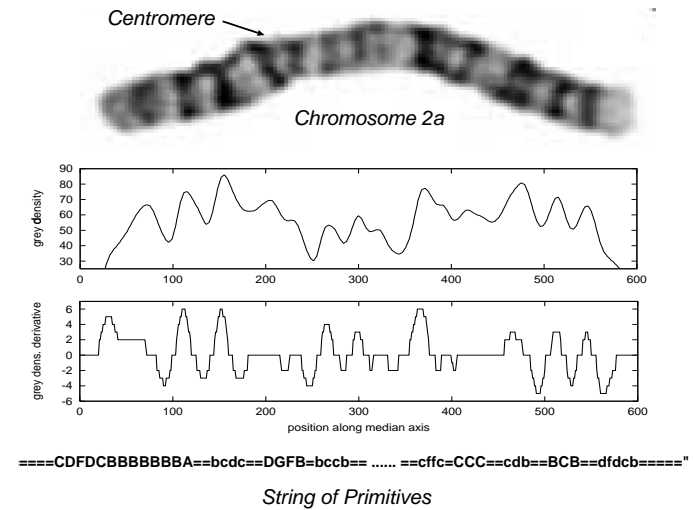
Pierre Dupont

3

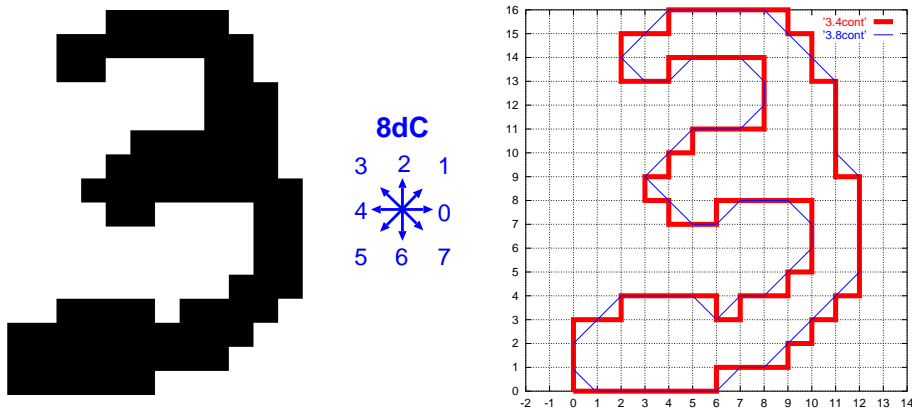
Examples

- Natural language sentence
- Speech
- Chronological series
- Successive actions of a WEB user
- Successive moves during a chess game
- A musical piece
- A program
- A form characterized by a chain code
- A biological sequence (DNA, proteins, ...)

Chromosome classification



Pattern Recognition



8dC: 000077766676666555545444443211000710112344543311001234454311

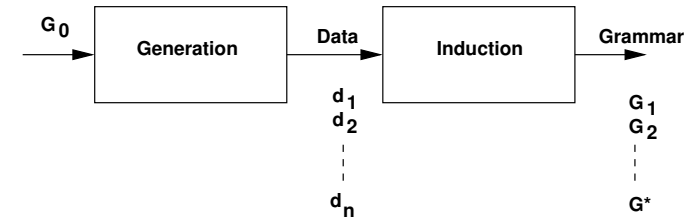
A modeling hypothesis



- Find G as close as possible to G_0
- The induction process does **not** prove the existence of G_0
It is a **modeling hypothesis**

Identification in the limit

- Grammar induction definition
- **Learning paradigms**
- DFA learning from positive and negative examples
- RPNI algorithm
- Probabilistic DFA learning
- Application to a natural language task
- Links with Markov models
- Smoothing issues
- Related problems and future work



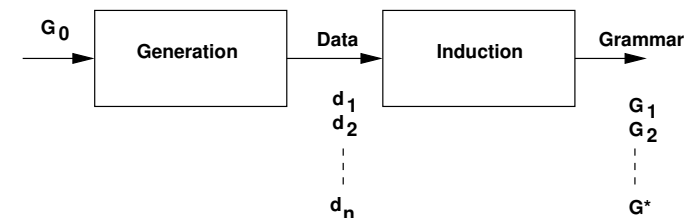
- convergence in **finite time** to G^*
- G^* is a representation of $L(G_0)$ (**exact learning**)

Learning paradigms

How to characterize learning?

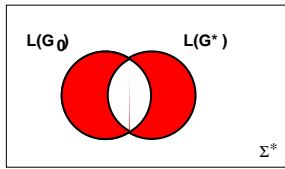
- which **concept classes** can or cannot be learned?
- what is a **good example**?
- is it possible to learn in **polynomial time**?

PAC Learning



- convergence to G^*
- G^* is **close enough** to G_0 **with high probability**
- ⇒ **Probably Approximately Correct** learning
- **polynomial** time complexity

Define a probability distribution D on a set of strings $\Sigma^{\leq n}$



$$P[P_D(L(G^*) \oplus L(G_0)) < \epsilon] > 1 - \delta$$

The same **unknown** distribution D is used to generate the sample and to measure the error

The result must hold for **any distribution** D (distribution free requirement)

The algorithm must return an hypothesis in **polynomial time** with respect to $\frac{1}{\epsilon}, \frac{1}{\delta}, n, |R(L)|$

Identification in the limit: good and bad news

The bad one...

Theorem 1. No **superfinite** class of languages is identifiable in the limit from **positive data only**

The good one...

Theorem 2. Any **admissible** class of languages is identifiable in the limit from **positive and negative data**

Other learnability results

- Identification in the limit in **polynomial time**
 - DFAs cannot be **efficiently** identified in the limit
 - unless we can ask **equivalence and membership queries** to an oracle
- PAC learning
 - DFAs are not PAC learnable (under some cryptographic limitation assumption)
 - unless we can ask membership queries to an oracle

- PAC learning with **simple examples**, i.e. examples drawn according to the conditional Solomonoff-Levin distribution

$$P_c(x) = \lambda_c 2^{-K(x|c)}$$

$K(x|c)$ denotes the Kolmogorov complexity of x given a representation c of the concept to be learned

- **regular languages** are **PACS learnable** with positive examples only
- but Kolmogorov complexity is **not computable!**

Cognitive relevance of learning paradigms

A **largely unsolved** question

Learning paradigms seem irrelevant to model human learning:

- Gold's identification in the limit framework has been criticized as **children** seem to **learn** natural language **without negative examples**
- All learning models assume a **known representation class**
- Some learnability results are based on **enumeration**

However learning models show that:

- an **oracle** can help
- some examples are useless, others are good: **characteristic samples** \Leftrightarrow **typical examples**
- learning well is learning **efficiently**
- example **frequency** matters
- good examples are **simple examples** \Leftrightarrow **cognitive economy**

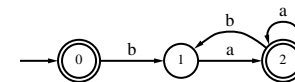
- Grammar induction definition
- Learning paradigms
- **DFA learning from positive and negative examples**
- RPNI algorithm
- Probabilistic DFA learning
- Application to a natural language task
- Links with Markov models
- Smoothing issues
- Related problems and future work

Regular Inference from Positive and Negative Data

Additional hypothesis: the underlying theory is a **regular grammar** or, equivalently, a **finite state automaton**

Property 1. Any regular language has a **canonical automaton** $A(L)$ which is **deterministic** and **minimal** (*minimal DFA*)

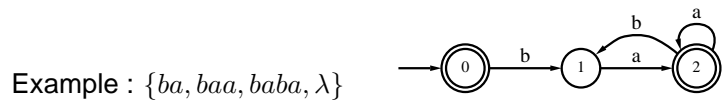
Example : $L = (ba^*a)^*$



A few definitions

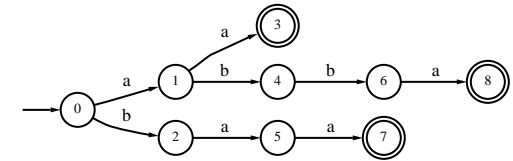
Definition 1. A positive sample S_+ is **structurally complete** with respect to an automaton A if, when generating S_+ from A :

- every transition of A is used at least one
- every final state is used as accepting state of at least one string



A theorem

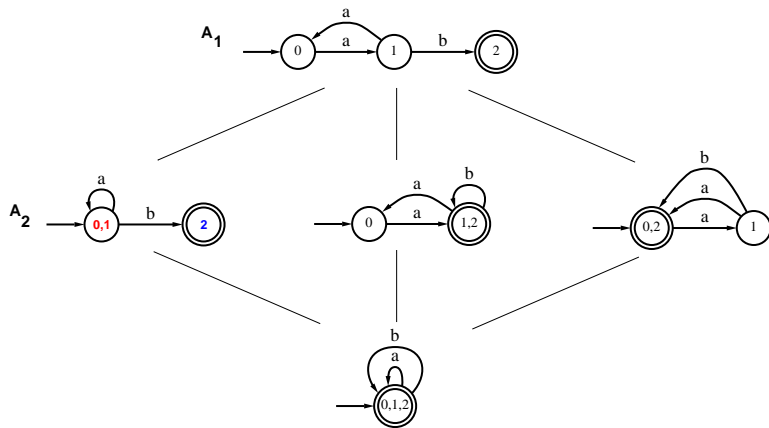
The positive data can be represented by a **prefix tree acceptor** (PTA)



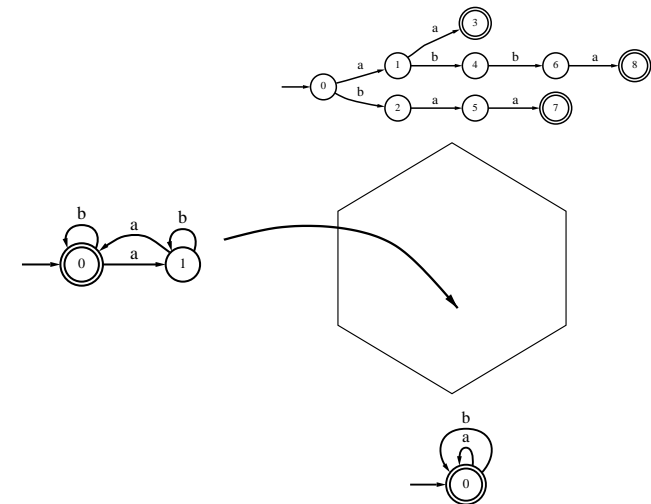
Example : $\{aa, abba, baa\}$

Theorem 3. If the positive sample is structurally complete with respect to a canonical automaton $A(L_0)$ then there exists a partition π of the state set of PTA such that $PTA/\pi = A(L_0)$

Merging is fun



- Merging \Leftrightarrow definition of a partition π on the set of states
Example : $\{\{0,1\}, \{2\}\}$
- If $A_2 = A_1/\pi$ then $L(A_1) \subseteq L(A_2)$: merging states \Leftrightarrow generalize language



How are we going to find the right partition? **Use negative data!**

Summary



We observe some positive and negative data

The positive sample S_+ comes from a regular language L_0

The positive sample is assumed to be structurally complete with respect to the canonical automaton $A(L_0)$ of the target language L_0 (Not an additional hypothesis but a way to restrict the search to reasonable generalizations!)

We build the Prefix Tree Acceptor of S_+ . By construction $L(PTA) = S_+$

Merging states \Leftrightarrow generalize S_+

The negative sample S_- helps to control over-generalization

Note: finding the minimal DFA consistent with S_+, S_- is NP-complete!

Pierre Dupont

24

- Grammar induction definition
- Learning paradigms
- DFA learning from positive and negative examples
- **RPNI algorithm**
- Probabilistic DFA learning
- Application to a natural language task
- Links with Markov models
- Smoothing issues
- Related problems and future work

Pierre Dupont

26

An automaton induction algorithm

Algorithm Automaton Induction

input

S_+ // positive sample
 S_- // negative sample
 $A \leftarrow PTA(S_+)$ // PTA

```

while (i, j) ← choose_states () do // Choose a state pair
  if compatible (i, j, S_-) then // Check for compatibility of merging i and j
    A ← A / πij
  end if
end while
return A

```

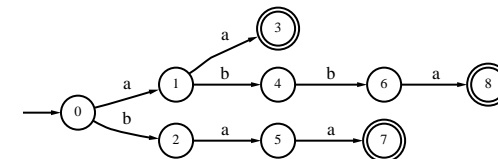
Pierre Dupont

25

RPNI algorithm

RPNI is a particular instance of the “**generalization as search**” paradigm

RPNI follows the **prefix order** in PTA



Polynomial time complexity with respect to sample size (S_+, S_-)

RPNI **identifies in the limit** the class of regular languages

A **characteristic sample**, i.e. a sample such that RPNI is guaranteed to produce the correct solution, has a **quadratic size** with respect to $|A(L_0)|$

Additional heuristics exist to improve performance when such a sample is not provided

Pierre Dupont

27

RPN algorithm: pseudo-code

```

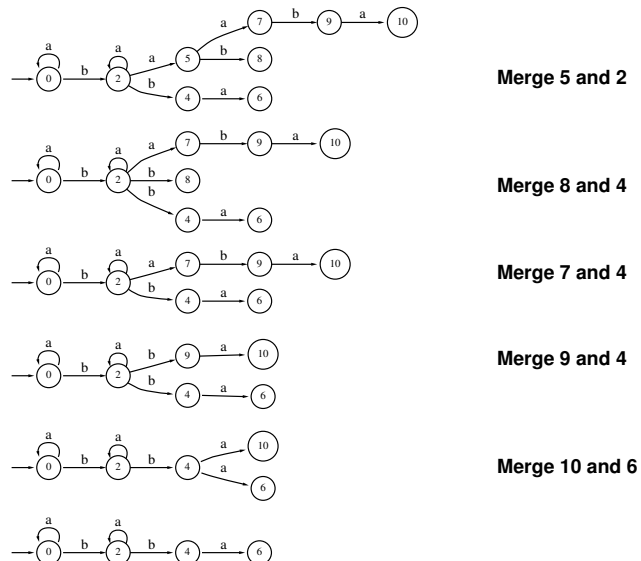
input  $S_+, S_-$ 
output A DFA consistent with  $S_+, S_-$ 
begin
 $A \leftarrow PTA(S_+)$  //  $N$  denotes the number of states of  $PTA(S_+)$ 
 $\pi \leftarrow \{\{0\}, \{1\}, \dots, \{N-1\}\}$  // One state for each prefix according to standard order <
for  $i = 1$  to  $|\pi| - 1$  // Loop over partition subsets  $\pi$ 
  for  $j = 0$  to  $i - 1$  // Loop over subsets of lower rank
     $\pi' \leftarrow \pi \setminus \{B_j, B_i\} \cup \{B_i \cup B_j\}$  // Merging  $B_i$  and  $B_j$ 
     $A/\pi' \leftarrow \text{derive}(A, \pi')$ 
     $\pi'' \leftarrow \text{determ\_merging}(A/\pi')$ 
    if compatible( $A/\pi'', S_-$ ) then // Deterministic parsing of  $S_-$ 
       $\pi \leftarrow \pi''$ 
    break // Break  $j$  loop
  end if
end for // End  $j$  loop
end for // End  $i$  loop
return  $A/\pi$ 

```

Pierre Dupont

28

An execution step of RPN



Pierre Dupont

29

Search space characterization

Conditions on the learning sample to guarantee the **existence of a solution**

DFA and NFA in the lattice

Characterization of the set of maximal generalizations \Rightarrow similar to the G set from Version Space

Efficient **Incremental** lattice construction is possible \Rightarrow RPN12 algorithm

Possible search by **genetic optimization**

Pierre Dupont

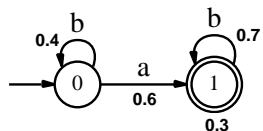
30

- Grammar induction definition
- Learning paradigms
- DFA learning from positive and negative examples
- RPN algorithm
- Probabilistic DFA learning
- Application to a natural language task
- Links with Markov models
- Smoothing issues
- Related problems and future work

Pierre Dupont

31

Probabilistic DFA



$$P(ab) = 0.6 * 0.7 * 0.3$$

A **structural** and **probabilistic** model \Rightarrow an **explicit** and **noise tolerant** theory

A combined **inductive learning** and **statistical estimation** problem

Learning from **positive examples only** and **frequency** information

Outside of the scope of the previous learning paradigms

A probabilistic automaton induction algorithm

Algorithm Probabilistic Automaton Induction

input

S_+

α

$A \leftarrow PPTA(S_+)$

// positive sample

// precision parameter

// Probabilistic PTA

while $(i, j) \leftarrow \text{choose_states}()$ **do**

if $\text{compatible}(i, j, \alpha)$ **then**

$A \leftarrow A / \pi_{ij}$

end if

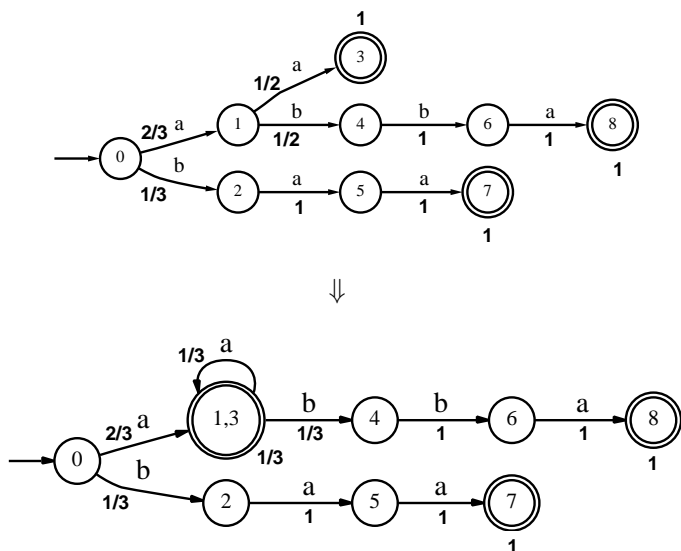
end while

return A

// Choose a pair of states

// Check for compatibility of merging i and j

Probabilistic prefix tree acceptor (PPTA)



Compatibility criterion

ALERGIA, RLIPS

Two states are compatible (can be merged) if their **suffix distributions** are close enough

MDI

Two states are compatible if **prior probability gain** of the merged model compensates for the **likelihood loss** of the data:

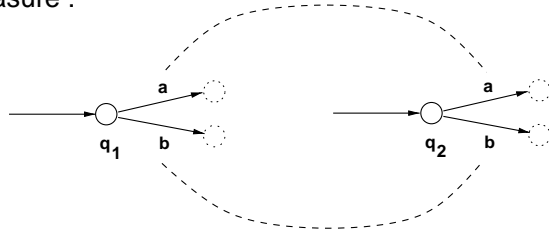
\Rightarrow Bayesian learning (not strictly in this case)

\Rightarrow based on Kullback-Leibler divergence

ALERGIA

RPNI state merging order

Compatibility measure :



- $\left| \frac{C(q_1, a)}{C(q_1)} - \frac{C(q_2, a)}{C(q_2)} \right| < \sqrt{\frac{1}{2} \ln \frac{2}{\alpha_A}} \left(\frac{1}{\sqrt{C(q_1)}} + \frac{1}{\sqrt{C(q_2)}} \right), \forall a \in \Sigma \cup \{\#\}$
- $\delta(q_1, a)$ and $\delta(q_2, a)$ are α_A -compatible, $\forall a \in \Sigma$

Remarks:

It is a recursive measure of suffix proximity

This measure does not depend on the prefixes of q_1 and $q_2 \Rightarrow$ local criterion

Pierre Dupont

36

Bayesian learning

Find a model \hat{M} which maximizes the likelihood of the data $P(X|M)$ and the *prior* probability of the model $P(M)$:

$$\hat{M} = \operatorname{argmax}_M P(X|M) \cdot P(M)$$

PPTA maximizes the data likelihood

A smaller model (number of states) is *a priori* assumed more likely

Kullback-Leibler divergence

$$\begin{aligned} D(P_{A_0} \parallel P_{A_1}) &\stackrel{\text{notation}}{=} D(A_0 \parallel A_1) \\ &= \sum_{x \in \Sigma^*} P_{A_0}(x) \log \frac{P_{A_0}(x)}{P_{A_1}(x)} \\ &= - \sum_{x \in \Sigma^*} P_{A_0}(x) \log P_{A_1}(x) - H(A_0) \end{aligned}$$

Likelihood of x given model A_1 :

$$P_{A_1}(x) = P(x|A_1)$$

Cross entropy between A_0 and A_1 :

$$- \sum P_{A_0}(x) \log P_{A_1}(x)$$

When A_0 is a maximum likelihood estimate, e.g. the PPTA, cross entropy measure the **likelihood loss** while going from A_0 to A_1

Pierre Dupont

38

MDI algorithm

RPNI state merging order

Compatibility measure: small **divergence increase** (= small likelihood loss) with respect to **size reduction** (= *prior* probability increase)

\Rightarrow a global criterion

$$\frac{\Delta(A_1, A_2)}{|A_1| - |A_2|} < \alpha_M$$

Efficient computation of divergence increase

$$\begin{aligned} D(A_0 \parallel A_2) &= D(A_0 \parallel A_1) + \Delta(A_1, A_2) \\ \Delta(A_1, A_2) &= \sum_{q_i \in Q_{012}} \sum_{a \in \Sigma \cup \{\#\}} c_i \gamma_0(q_i, a) \log \frac{\gamma_1(q_i, a)}{\gamma_2(q_i, a)} \end{aligned}$$

$Q_{012} = \{q_i \in Q_0 \mid B_{\pi_{01}}(q_i) \neq B_{\pi_{02}}(q_i)\}$ denotes the set of states of A_0 which have been merged to get A_2 from A_1

- Grammar induction definition
- Learning paradigms
- DFA learning from positive and negative examples
- RPNI algorithm
- Probabilistic DFA learning
- **Application to a natural language task**
- Links with Markov models
- Smoothing issues
- Related problems and future work

Natural language application: the ATIS task

Air travel information system, “spontaneous” American English

“Uh, I'd like to go from, uh, Pittsburgh to Boston next Tuesday, no wait, Wednesday”.

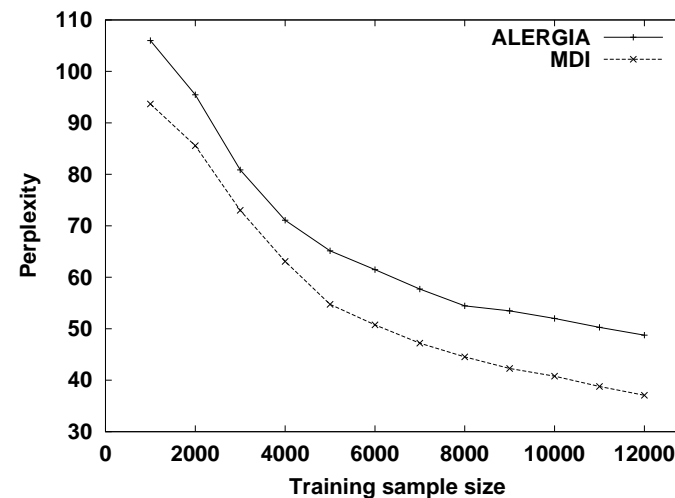
Lexicon (alphabet): 1294 words

Learning sample: 13044 sentences, 130773 words

Validation set: 974 sentences, 10636 words

Test set: 1001 sentences, 11703 words

Comparative results



Perplexity measure the prediction power of the model: the smaller, the better

Perplexity

$P(x_i^j | q^i)$: probability of generating x_i^j , the i -th symbol of the j -th string from state q^i

$$LL = \left(-\frac{1}{\|S\|} \sum_{j=1}^{|S|} \sum_{i=1}^{|x|} \log P(x_i^j | q^i) \right)$$

$$PP = 2^{LL}$$

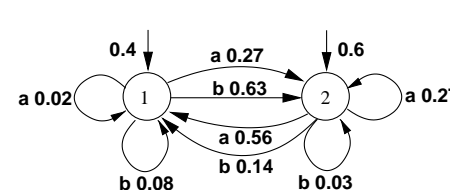
$PP = 1 \Rightarrow$ a perfectly predictive model

$PP = |\Sigma| \Rightarrow$ uniform random guessing over Σ

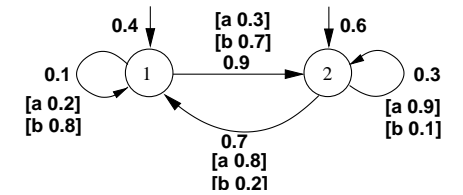
- Grammar induction definition
- Learning paradigms
- DFA learning from positive and negative examples
- RPNI algorithm
- Probabilistic DFA learning
- Application to a natural language task
- Links with Markov models
- Smoothing issues
- Related problems and future work

Equivalence between PNFA and HMM

Probabilistic non-deterministic automata (PNFA), with **no end-of-string probabilities**, are equivalent to Hidden Markov Models (HMMs)



PNFA

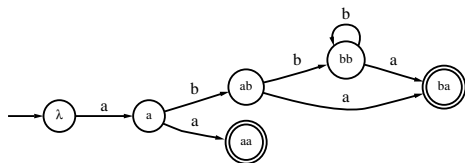


HMM with emission on transitions

Links with Markov chains

A subclass of regular languages: the **k-testable languages in the strict sense**

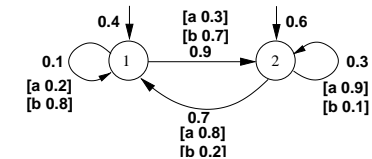
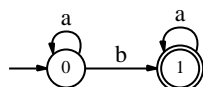
A k-TSS language is generated by an automaton such that all subsequences sharing the same last $k - 1$ symbols lead to the same state



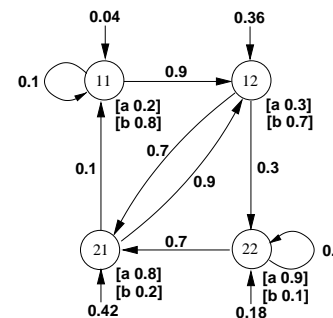
$$\hat{p}(a|bb) = \frac{C(bba)}{C(bb)}$$

A probabilistic k-TSS language is equivalent to a $k - 1$ order Markov chain

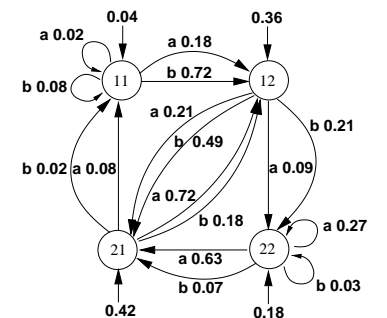
There exists probabilistic regular languages **not reducible** to Markov chains of **any finite order**



HMM with emission on transitions



HMM with emission on states



PNFA

- Grammar induction definition
- Learning paradigms
- DFA learning from positive and negative examples
- RPNI algorithm
- Probabilistic DFA learning
- Application to a natural language task
- Links with Markov models
- **Smoothing issues**
- Related problems and future work

The smoothing problem

A probabilistic DFA defines a **probability distribution** over a set of strings

Some strings are not observed on the training sample but they could be observed
 ⇒ their probability should be strictly positive

The smoothing problem: how to assign a **reasonable probability** to (yet) unseen random events ?

Highly optimized smoothing techniques exist for **Markov chains**

How to **adapt** these techniques **to** more general **probabilistic automata**?

- Grammar induction definition
- Learning paradigms
- DFA learning from positive and negative examples
- RPNI algorithm
- Probabilistic DFA learning
- Application to a natural language task
- Links with Markov models
- Smoothing issues
- **Related problems and future work**

Related problems and approaches

I did not talk about

- other induction problems (**NFA**, **CFG**, tree grammars, ...)
- **heuristic approaches** as neural nets or genetic algorithms
- how to use **prior knowledge**
- **smoothing techniques**
- how to parse natural language without a grammars (**decision trees**)
- how to learn **transducers**
- **benchmarks, applications**

Ongoing and future work

- Definition of a *theoretical framework* for *inductive and statistical* learning
- *Links with HMMs*: parameter estimation, structural induction
- *Smoothing techniques* improvement \Rightarrow a key issue for practical applications
- Applications to probabilistic modeling of *proteins*
- *Automatic translation*
- Applications to *Text categorization* or *Text mining*