Inductive and statistical learning of formal grammars

Pierre Dupont

pdupont@info.ucl.ac.be

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

Machine Learning

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***

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

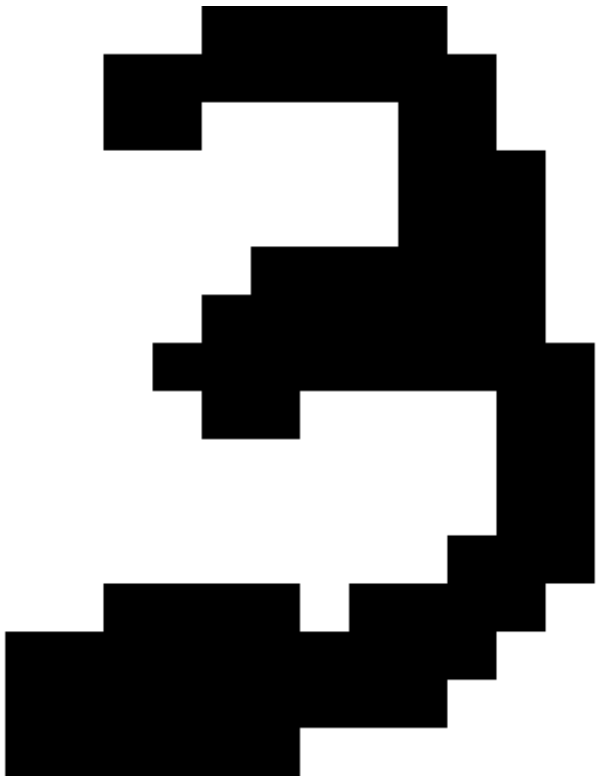
The set of examples, known as ***positive sample***, is usually made of ***strings*** or ***sequences*** over a specific ***alphabet***

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

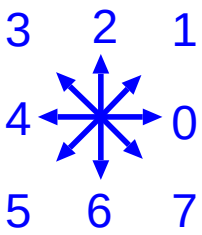
Examples

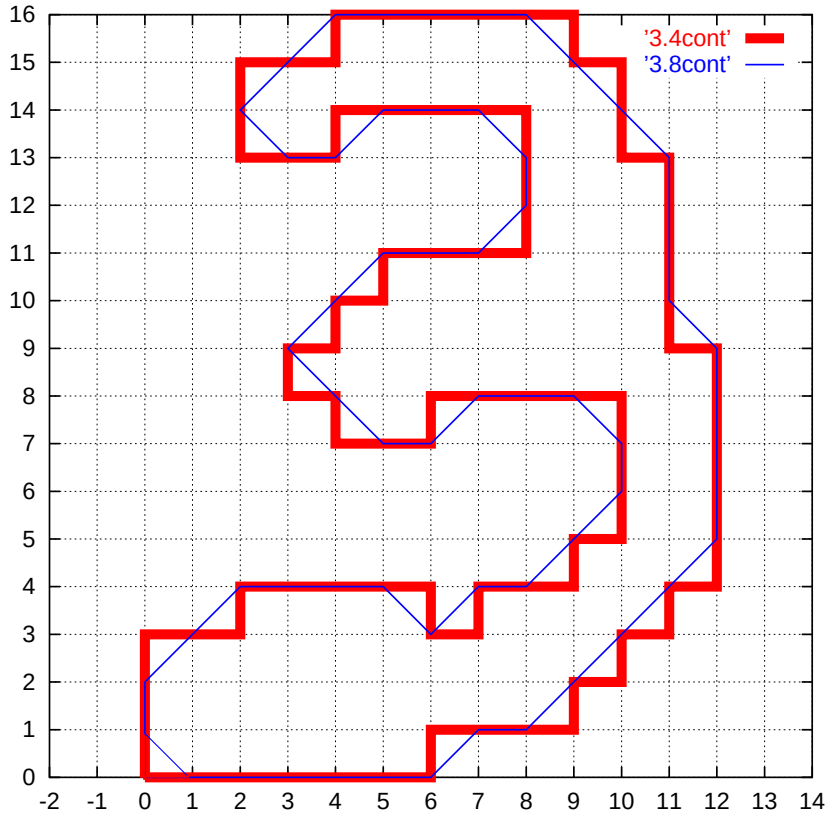
- Natural language sentence
- Speech
- Chronological series
- Successive actions of a WEB user
- Successive moves during a chess game
- A musical piece
- A program
- A form characterized by a chain code
- A biological sequence (DNA, proteins, . . .)

Pattern Recognition



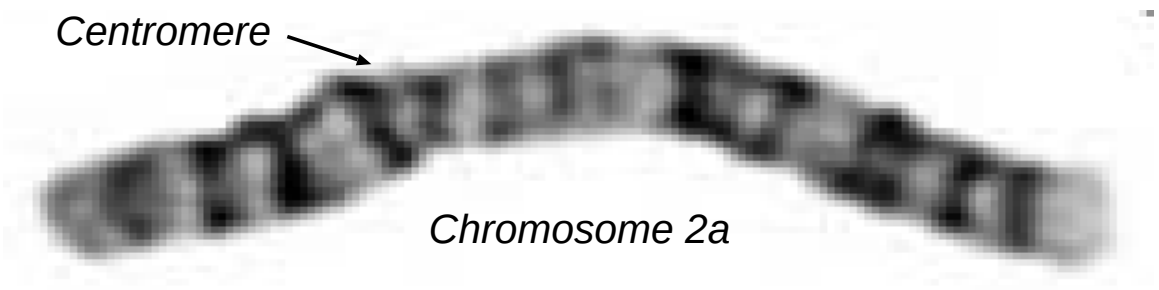
8dC

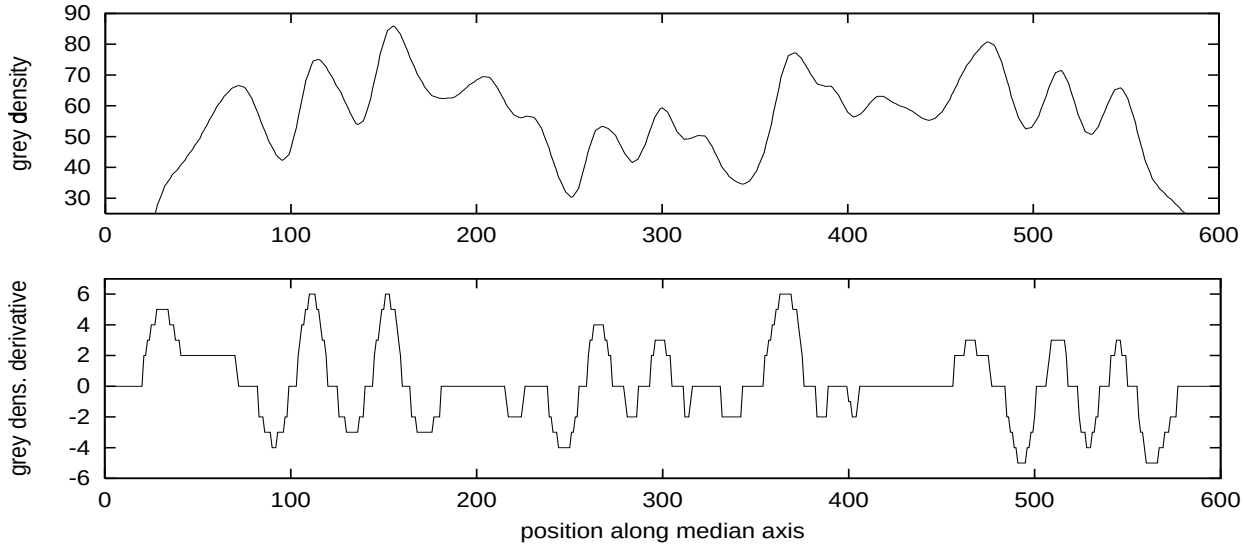




8dC: 0000777666766665555454444432110007101123445433110012344543 11

Chromosome classification

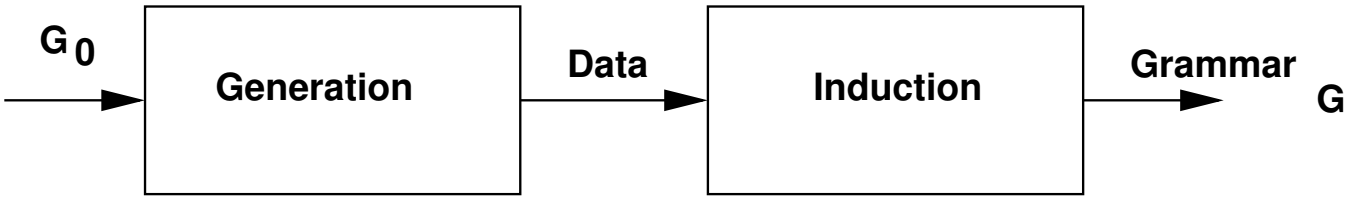




====CDFDCBBBBBBBA==bcdc==DGFB=bccb== ..... ==cffc=CCC==cdb==BCB==dfdcb====="

String of Primitives

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***

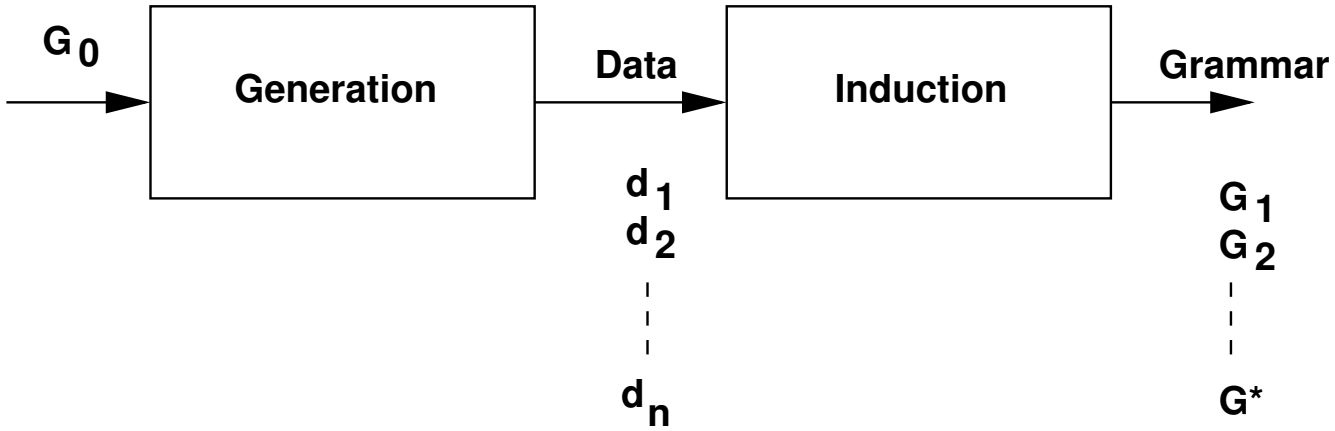
- Grammar induction definition
- **Learning paradigms**
- DFA learning from positive and negative examples
- RPNI algorithm
- Probabilistic DFA learning
- Application to a natural language task
- Links with Markov models
- Smoothing issues
- Related problems and future work

Learning paradigms

How to characterize learning?

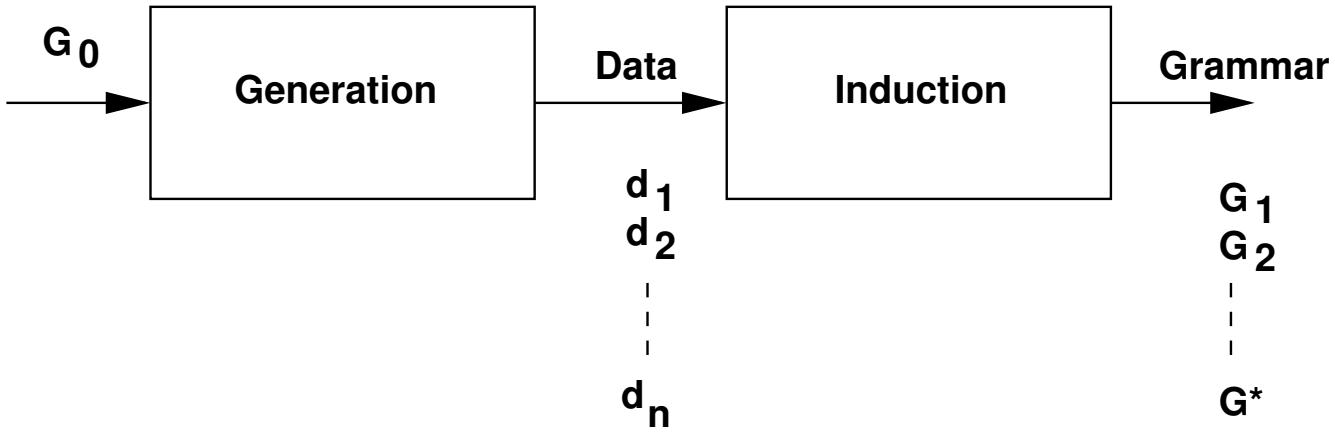
- which ***concept classes*** can or cannot be learned?
- what is a ***good example***?
- is it possible to learn in ***polynomial time***?

Identification in the limit



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

PAC Learning



- convergence to G^*
- G^* is ***close enough*** to G_0 ***with high probability***

  $\Rightarrow$ ***P***robably ***A***pproximately ***C***orrect learning
- ***polynomial*** time complexity

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***

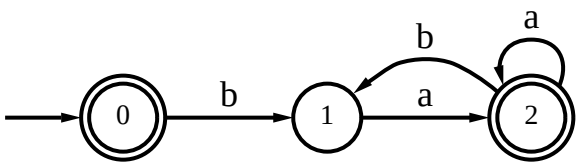
- Grammar induction definition
- Learning paradigms
- **DFA learning from positive and negative examples**
- RPNI algorithm
- Probabilistic DFA learning
- Application to a natural language task
- Links with Markov models
- Smoothing issues
- Related problems and future work

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

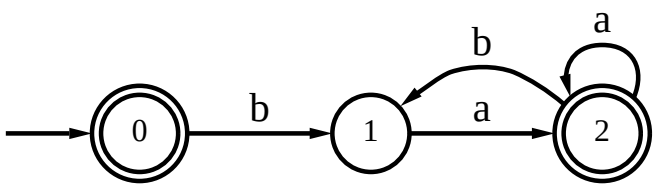


A few definitions

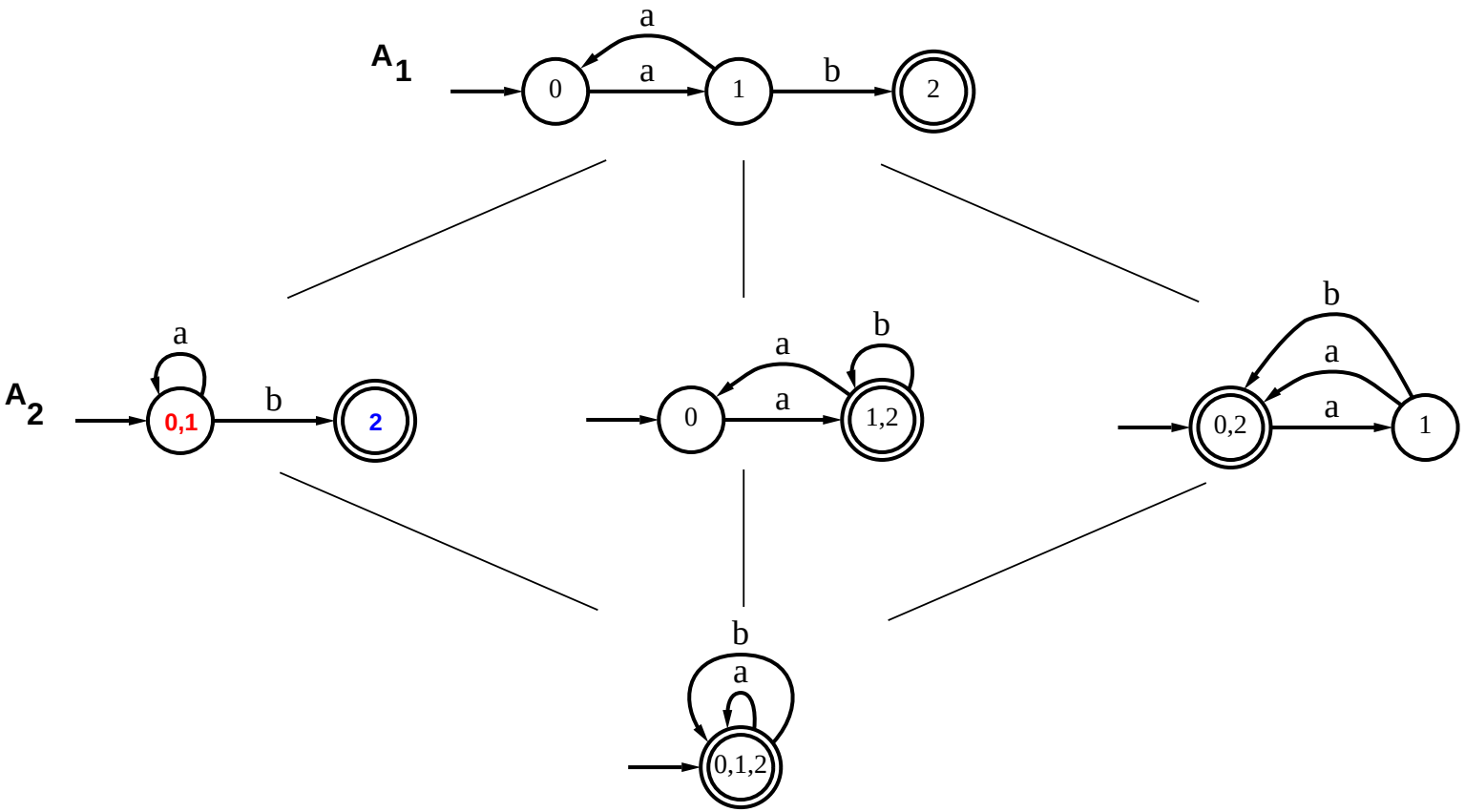
Definition 1. A positive sample S_+ is ***structurally complete*** with respect to an automaton A if, when generating S_+ from A:

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

Example : $\{ba, baa, baba, \lambda\}$



Merging is fun

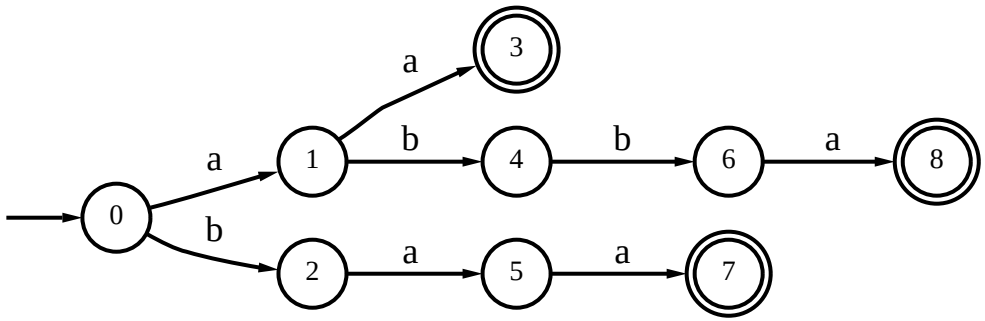


- Merging $\Leftrightarrow$ definition of a partition π on the set of states
  Example : $\{\{0,1\}, \{2\}\}$

- If $A_2 = A_1/\pi$ then $L(A_1) \subseteq L(A_2)$: merging states $\Leftrightarrow$ generalize language

A theorem

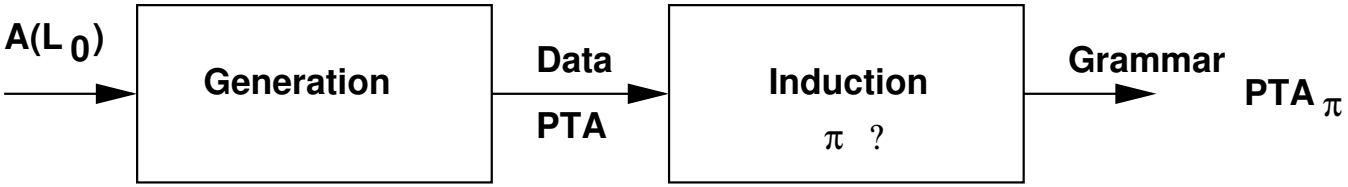
The positive data can be represented by a ***prefix tree acceptor*** (PTA)



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/π = $A(L_0)$

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!

Bayesian learning

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

$$\hat{M} = \underset{M}{\operatorname{argmax}}\ P(X|M).P(M)$$

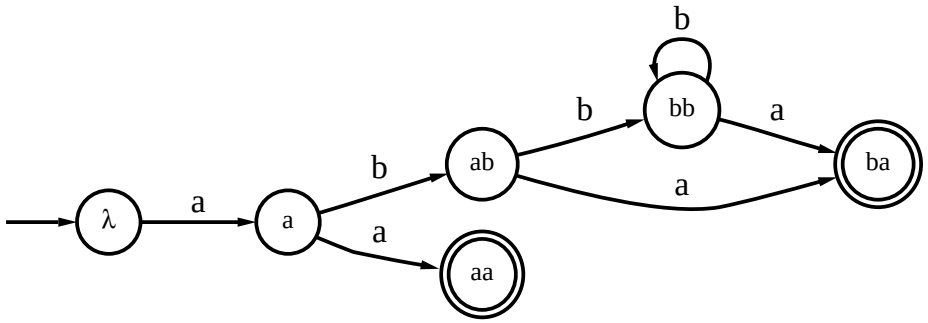
PPTA maximizes the data likelihood

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

Links with Markov chains

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

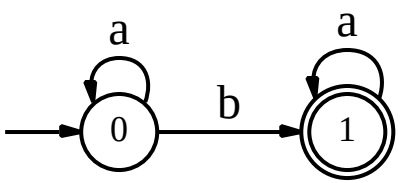
A k-TSS language is generated by an automaton such that all subsequences sharing the same last $k-1$ symbols lead to the same state



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

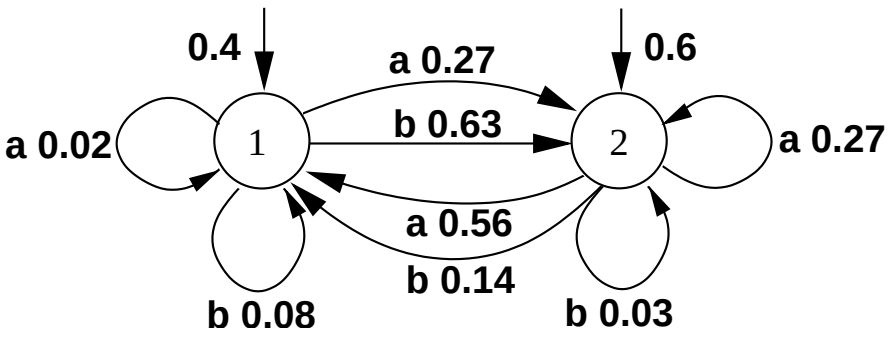
A probabilistic k-TSS language is equivalent to a $k-1$ order Markov chain

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

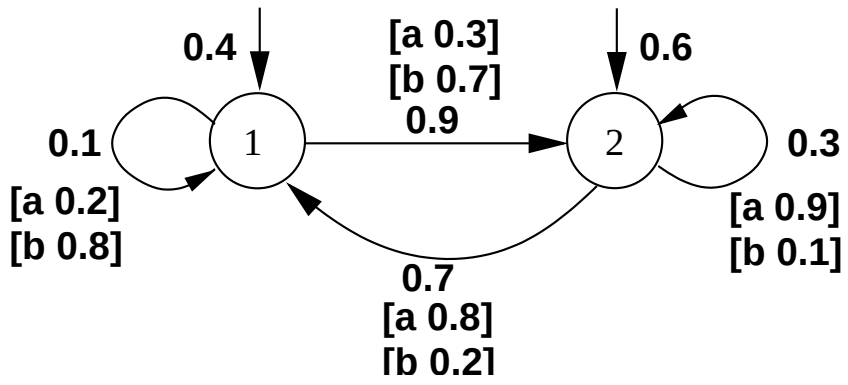


Equivalence between PNFA and HMM

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



PNFA



HMM with emission on transitions

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 $\Rightarrow$ 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?***

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***