Learning probabilistic finite automata

Colin de la Higuera
University of Nantes
Acknowledgements

• Laurent Miclet, Jose Oncina, Tim Oates, Rafael Carrasco, Paco Casacuberta, Rémi Eyraud, Philippe Ezequeil, Henning Fernau, Thierry Murgue, Franck Thollard, Enrique Vidal, Frédéric Tantini,...
• List is necessarily incomplete. Excuses to those that have been forgotten

http://pagesperso.lina.univ-nantes.fr/~cdlh/slides/

Chapters 5 and 16
Outline

1. PFA
2. Distances between distributions
3. FFA
4. Basic elements for learning PFA
5. ALERGIA
6. MDI and DSAI
7. Open questions
1 PFA

Probabilistic finite (state) automata
Practical motivations

(Computational biology, speech recognition, web services, automatic translation, image processing ...)

- A lot of positive data
- Not necessarily any negative data
- No ideal target
- Noise
The grammar induction problem, revisited

- The data consists of positive strings, «generated» following an unknown distribution
- The goal is now to find (learn) this distribution
- Or the grammar/automaton that is used to generate the strings
Success of the probabilistic models

- $n$-grams
- Hidden Markov Models
- Probabilistic grammars
A DPFA (Deterministic Probabilistic Finite Automaton) is a type of automaton where each transition has a probability associated with it. The diagram shows a DPFA with several states and transitions labeled with probabilities. For example, the transition from state 1 to state 2 has a probability of 1/2 for input 'a' and 1/2 for input 'b'. The DPFA has a start state and multiple final states, indicated by the rounded circles with double borders. The diagram illustrates the probabilistic behavior of the automaton for different inputs.
\[ \Pr_A(abab) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{3} \times \frac{2}{3} \times \frac{3}{4} = \frac{1}{24} \]
$PFA$: Probabilistic Finite (state) Automaton
\( \varepsilon \)-PFA: Probabilistic Finite (state) Automaton with \( \varepsilon \)-transitions
How useful are these automata?

- They can define a distribution over $\Sigma^*$
- They do not tell us if a string belongs to a language
- They are good candidates for grammar induction
- There is (was?) not that much written theory
Basic references

- The *HMM* literature
- Azaria Paz 1973: *Introduction to probabilistic automata*
- Chapter 5 of my book
- *Probabilistic Finite-State Machines*, Vidal, Thollard, cdlh, Casacuberta & Carrasco
- *Grammatical Inference papers*
Automata, definitions

Let $\mathcal{D}$ be a distribution over $\Sigma^*$

$$0 \leq \Pr_{\mathcal{D}}(w) \leq 1$$

$$\sum_{w \in \Sigma^*} \Pr_{\mathcal{D}}(w) = 1$$
A Probabilistic Finite (state) Automaton is a \(<Q, \Sigma, I_\rho, F_\rho, \delta_\rho>\)

- \(Q\) set of states
- \(I_\rho: Q \rightarrow [0;1]\)
- \(F_\rho: Q \rightarrow [0;1]\)
- \(\delta_\rho: Q \times \Sigma \times Q \rightarrow [0;1]\)
What does a PFA do?

- It defines the probability of each string \( w \) as the sum (over all paths reading \( w \)) of the products of the probabilities

\[
\Pr_A(w) = \sum_{\pi \in \text{paths}(w)} \Pr(\pi)
\]

- \( \pi = q_0 a_1 q_1 a_2 \ldots a_n q_n \)

- \( \Pr(\pi) = I(\pi_0) \cdot F(\pi_n) \cdot \prod_{a,j} \delta_{\rho}(q_{i-1,j}, a_{i,j}, q_{i,j}) \)

- Note that if \( \lambda \)-transitions are allowed the sum may be infinite
\[
\Pr(aba) = 0.7 \times 0.4 \times 0.1 \times 1 + 0.7 \times 0.4 \times 0.45 \times 0.2 = 0.028 + 0.0252 = 0.0532
\]
• non deterministic PFA: many initial states/only one initial state
• an $\lambda$-PFA: a PFA with $\lambda$-transitions and perhaps many initial states
• DPFA: a deterministic PFA
Consistency

A PFA is consistent if

1. \( \text{Pr}_A(\Sigma^*) = 1 \)
2. \( \forall x \in \Sigma^* \ 0 \leq \text{Pr}_A(x) \leq 1 \)
Consistency theorem

$A$ is consistent if every state is useful (accessible and co-accessible) and

$$\forall q \in Q$$

$$F_P(q) + \sum_{q' \in Q, a \in \Sigma} \delta_P(q, a, q') = 1$$
Equivalence between models

• Equivalence between PFA and HMM...

• But the HMM usually define distributions over each $\Sigma^n$
A football HMM

win draw lose  win draw lose  win draw lose

\[
\begin{align*}
\frac{1}{2} & \quad \frac{1}{4} & \quad \frac{1}{4} & \quad \frac{1}{4} & \quad \frac{1}{4} & \quad \frac{1}{2} & \quad \frac{1}{4} & \quad \frac{1}{4} & \quad \frac{1}{4} & \quad \frac{1}{2} \\
\frac{3}{4} & \quad \frac{1}{4} & \quad \frac{1}{4} & \quad \frac{1}{4} & \quad \frac{1}{4} & \quad \frac{1}{2} & \quad \frac{3}{4} & \quad \frac{1}{4} & \quad \frac{1}{4} & \quad \frac{1}{4} & \quad \frac{1}{2} \\
\end{align*}
\]
Equivalence between $PFA$ with $\lambda$-transitions and $PFA$ without $\lambda$-transitions

cdlh 2003, Hanneforth & cdlh 2009

- Many initial states can be transformed into one initial state with $\lambda$-transitions;
- $\lambda$-transitions can be removed in polynomial time;
- Strategy:
  - number the states
  - eliminate first $\lambda$-loops, then the transitions with highest ranking arrival state
**PFA are strictly more powerful than DPFA**

*Folk theorem*

(and) You can’t even tell in advance if you are in a good case or not

(see: Denis & Esposito 2004)
This distribution cannot be modelled by a DPFA.
What does a DPFA over $\Sigma = \{a\}$ look like?

And with this architecture you cannot generate the previous one
Parsing issues

- Computation of the probability of a string or of a set of strings
- Deterministic case
  - Simple: apply definitions
  - Technically, rather sum up logs: this is easier, safer and cheaper
\[ \text{Pr}(aba) = 0.7 \times 0.9 \times 0.35 \times 0 = 0 \]
\[ \text{Pr}(abb) = 0.7 \times 0.9 \times 0.65 \times 0.3 = 0.12285 \]
Non-deterministic case

\[
\text{Pr}(aba) = 0.7 \times 0.4 \times 0.1 \times 1 + 0.7 \times 0.4 \times 0.45 \times 0.2 \\
= 0.028 + 0.0252 = 0.0532
\]
In the literature

- The computation of the probability of a string is by dynamic programming: \( O(n^2m) \)
- 2 algorithms: *Backward* and *Forward*
- If we want the most probable derivation to define the probability of a string, then we can use the *Viterbi* algorithm
Forward algorithm

- $A[i,j] = \Pr(q_i|a_1..a_j)$
  (The probability of being in state $q_i$ after having read $a_1..a_j$)
- $A[i,0] = \mathbb{I}_\rho(q_i)$
- $A[i,j+1] = \sum_{k \leq |Q|} A[k,j] \cdot \delta_\rho(q_k,a_{j+1},q_i)$
- $\Pr(a_1..a_n) = \sum_{k \leq |Q|} A[k,n] \cdot F_\rho(q_k)$
2 Distances

What for?
- Estimate the quality of a language model
- Have an indicator of the convergence of learning algorithms
- Construct kernels
2.1 Entropy

- How many bits do we need to correct our model?
- Two distributions over $\Sigma^*$: $\mathcal{D}$ et $\mathcal{D}'$
- Kullback Leibler divergence (or relative entropy) between $\mathcal{D}$ and $\mathcal{D}'$:

$$\sum_{w \in \Sigma^*} \Pr_D(w) \times \left| \log \Pr_D(w) - \log \Pr_{\mathcal{D}'}(w) \right|$$
2.2 Perplexity

- The idea is to allow the computation of the divergence, but relatively to a test set \( (S) \).
- An approximation (sic) is perplexity: inverse of the geometric mean of the probabilities of the elements of the test set.
\[
\prod_{w \in S} \Pr_{\mathcal{D}}(w)^{-1/|S|} = 1
\]

\[
\sqrt[|S|]{\prod_{w \in S} \Pr_{\mathcal{D}}(w)}
\]

Problem if some probability is null...
Why multiply (1)

- We are trying to compute the probability of independently drawing the different strings in set $S$. 
Why multiply? (2)

- Suppose we have two predictors for a coin toss
  - Predictor 1: heads 60%, tails 40%
  - Predictor 2: heads 100%
- The tests are H: 6, T: 4
- Arithmetic mean
  - P1: 36% + 16% = 0.52
  - P2: 0.6
- Predictor 2 is the better predictor ;-)
2.3 Distance $d_2$

$$d_2(\mathcal{D}, \mathcal{D}') = \sqrt{\sum_{w \in \Sigma^*} (\Pr_{\mathcal{D}}(w) - \Pr_{\mathcal{D}'}(w))^2}$$

Can be computed in polynomial time if $\mathcal{D}$ and $\mathcal{D}'$ are given by PFA (Carrasco & cdih 2002)

This also means that equivalence of PFA is in $\mathbf{P}$
3 FFA

Frequency Finite (state) Automata
A learning sample

- is a multiset
- Strings appear with a frequency (or multiplicity)
- $S=\lambda (3), \text{aaa (4), aaba (2), ababa (1), bb (3), bbaaa (1)}$
A deterministic frequency finite automaton is a DFA with a frequency function returning a positive integer for every state and every transition, and for entering the initial state such that

- the sum of what enters is equal to what exits and
- the sum of what halts is equal to what starts
Example
From a DFFA to a DPFA

Frequencies become relative frequencies by dividing by sum of exiting frequencies

Zadar, August 2010
From a DFA and a sample to a DFFA

\[ S = \{ \lambda, aaaaa, ab, babb, bbbb, bbbbaa \} \]
Note

- Another sample may lead to the same DFFA
- Doing the same with a NFA is a much harder problem
- Typically what algorithm Baum-Welch (EM) has been invented for...
The frequency prefix tree acceptor

- The data is a multi-set
- The FTA is the smallest tree-like FFA consistent with the data
- Can be transformed into a PFA if needed
From the sample to the FTA

FTA($S$)

$S=\{\lambda\ (3),\ a\ a\ a\ (4),\ a\ a\ b\ a\ (2),\ a\ b\ a\ b\ a\ (1),\ b\ b\ (3),\ b\ b\ a\ a\ a\ (1)\}$
Red, Blue and White states

- **Red** states are confirmed states
- **Blue** states are the (non Red) successors of the Red states
- **White** states are the others

Same as with DFA and what RPNI does
Merge and fold

Suppose we decide to merge with state \( a \)
Merge and fold

First disconnect and reconnect to

Zadar, August 2010
Merge and fold

Then fold
Merge and fold

after folding
State merging algorithm

\[ A = \text{FTA}(S); \quad \text{Blue} = \{ \delta(q_I, a) : a \in \Sigma \}; \]
\[ \text{Red} = \{ q_I \} \]

While Blue \( \neq \emptyset \) do

choose \( q \) from Blue such that Freq(\( q \)) \( \geq t_0 \)

if \( \exists p \in \text{Red}: \ d(A_p, A_q) \) is small

then \( A = \text{merge\_and\_fold}(A, p, q) \)

else \( \text{Red} = \text{Red} \cup \{ q \} \)

Blue = \{ \delta(q, a) : q \in \text{Red} \} - \{ \text{Red} \} \]
The real question

- How do we decide if $d(A_p, A_q)$ is small?
- Use a distance...
- Be able to compute this distance
- If possible update the computation easily
- Have properties related to this distance
Deciding if two distributions are similar

- If the two distributions are known, equality can be tested
- Distance ($L_2$ norm) between distributions can be exactly computed
- But what if the two distributions are unknown?
Taking decisions

Suppose we want to merge with state $a$. 

![Graph Diagram]
Taking decisions

Yes if the two distributions induced are similar

Zadar, August 2010
5 Alergia
Alergia’s test

- $D_1 \approx D_2$ if $\forall x \Pr_{D_1}(x) \approx \Pr_{D_2}(x)$
- Easier to test:
  - $\Pr_{D_1}(\lambda) = \Pr_{D_2}(\lambda)$
  - $\forall a \in \Sigma \Pr_{D_1}(a\Sigma^*) = \Pr_{D_2}(a\Sigma^*)$
- And do this recursively!
- Of course, do it on frequencies
Hoeffding bounds

\[ \gamma \leftarrow \left| \frac{f_1}{n_1} - \frac{f_2}{n_2} \right| \]

\[ \gamma < \left( \sqrt{\frac{1}{n_1}} + \sqrt{\frac{1}{n_2}} \right) \cdot \sqrt{\frac{1}{2} \ln \frac{2}{\alpha}} \]

\( \gamma \) indicates if the relative frequencies \( \frac{f_1}{n_1} \) and \( \frac{f_2}{n_2} \) are sufficiently close.
A run of Alergia
Our learning multisample

\[ S = \{ \lambda(490), a(128), b(170), aa(31), ab(42), ba(38), bb(14), aaa(8), aab(10), aba(10), abb(4), baa(9), bab(4), bba(3), bbb(6), aaaa(2), aaab(2), aaba(3), aabb(2), abaa(2), abab(2), abba(2), abbb(1), baaa(2), baab(2), baba(1), babb(1), bbab(1), bbba(1), aaaa(1), aaaab(1), aaaba(1), aabaa(1), aabab(1), aabba(1), abbaa(1), abbab(1), aabab(1), aabba(1), abbaa(1), abbab(1) \} \]
Parameter $\alpha$ is arbitrarily set to 0.05. We choose 30 as a value for threshold $t_0$.

Note that for the blue state who have a frequency less than the threshold, a special merging operation takes place.
Can we merge $\lambda$ and $a$?

- Compare $\lambda$ and $a$, $a\Sigma^*$ and $aa\Sigma^*$, $b\Sigma^*$ and $ab\Sigma^*$
- $490/1000$ with $128/257$, $257/1000$ with $64/257$, $253/1000$ with $65/257$, . . .

- All tests return true
And fold
Next merge? 
$\lambda$ with b?
Can we merge $\lambda$ and $b$?

- Compare $\lambda$ and $b$, $a\Sigma^*$ and $ba\Sigma^*$, $b\Sigma^*$ and $bb\Sigma^*$
- $660/1341$ and $225/340$ are different (giving $\gamma = 0.162$)
- On the other hand

$$\left(\sqrt{\frac{1}{n_1}} + \sqrt{\frac{1}{n_2}}\right)\cdot\sqrt{\frac{1}{2\ln\frac{2}{\alpha}}} = 0.111$$
Merge
And fold

Zadar, August 2010
Merge

Diagram showing two connected nodes labeled 660 and 225 with arrows indicating flow. Numbers associated with arcs include 'a: 341' and 'b: 340' for the 660 node, and 'a: 95' for the 225 node. Another diagram shows a tree structure with nodes labeled 29, 7, 8, and 1, with arcs labeled 'a: 11', 'b: 49', 'a: 2', 'b: 2', 'a: 1', and 'b: 9'.
And fold

As a PFA
Conclusion and logic

- Alergia builds a DFFA in polynomial time
- Alergia can identify DPFA in the limit with probability 1
- No good definition of Alergia’s properties
6 DSAI and MDI

Why not change the criterion?
Criterion for DSAI

- Using a distinguishable string
- Use norm $L_\infty$
- Two distributions are different if there is a string with a very different probability
- Such a string is called $\mu$-distinguishable
- Question becomes:

  Is there a string $x$ such that
  \[ |\Pr_{A,q}(x) - \Pr_{A,q'}(x)| > \mu \]
(much more to DSAI)


- PAC learnability results, in the case where targets are acyclic graphs
Criterion for MDI

- MDL inspired heuristic
- Criterion is: does the reduction of the size of the automaton compensate for the increase in preplexity?
7 Conclusion and open questions
A good candidate to learn NFA is DEES

Never has been a challenge, so state of the art is still unclear

Lots of room for improvement towards probabilistic transducers and probabilistic context-free grammars
Appendix

Stern Brocot trees
Identification of probabilities

*If we were able to discover the structure, how do we identify the probabilities?*
By estimation: the edge is used 1501 times out of 3000 passages through the state.
Stern-Brocot trees: (Stern 1858, Brocot 1860)

Can be constructed from two simple adjacent fractions by the «mean» operation

\[
\frac{a}{b} \quad m \quad \frac{c}{d} = \frac{a+c}{b+d}
\]
Idea:

- Instead of returning $c(x)/n$, search the Stern-Brocot tree to find a good simple approximation of this value.
Iterated Logarithm:
With probability 1, for a co-finite number of values of $n$ we have:

$$\left| \frac{c(x)}{n} - \frac{a}{b} \right| < \sqrt{\frac{\lambda \log \log n}{n}}$$

$\forall \lambda > 1$