Learning probability distributions generated by finite-state machines

Jorge Castro   Ricard Gavaldà

LARCA Research Group
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya, Barcelona

ICGI 2012 tutorial, Sept. 5th

Introduction

- (You know this) Finite-state machines are a great modeling tool

  Speech recognition, speech translation, handwritten text recognition, shallow parsing, computer vision, shape-face-gesture recognition, object tracking, image retrieval, medical image analysis, user modeling, financial returns modeling, DNA analysis, phylogeny construction, . . .

- Learning as an option to expert model construction

- Surveys: [Vidal+ 05] (I & II), [Dupont+ 05], [delaHiguera 10]

Outline

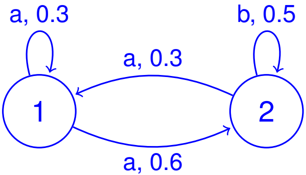
Warnings

- Emphasis on algorithmic aspects
- Unconventional presentation, esp. PDFA learning
- Apologies if you miss many relevant papers!

Finite state models and distributions: PFA

Future event probabilities only depend on current state

PFA: graph + initial, transition, and stopping probabilities



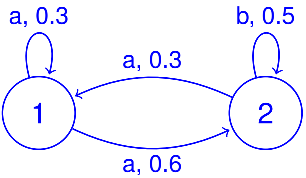
initial probs.: 1, 0
stopping probs.: 0.1, 0.2

$$\begin{aligned}\Pr[abaa] &= 0.6\cdot 0.5\cdot 0.3\cdot 0.3\cdot 0.1 + 0.6\cdot 0.5\cdot 0.3\cdot 0.6\cdot 0.2\\ &= 0.0135\end{aligned}$$

PFA define (semi)distributions on $\Sigma^\star$

Matricial view of PFA

A PFA is a tuple $\langle \alpha_0, \alpha_\infty, \{T_a\}_{a\in\Sigma}\rangle$, each $T_a \in \mathbb{R}^{n\times n}$



a	1	2
1	0.3	0.6
2	0.3	0

b	1	2
1	0	0
2	0	0.5

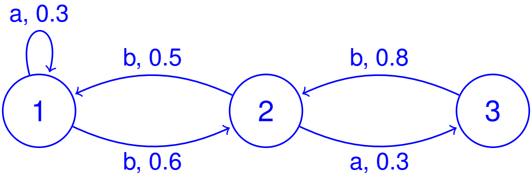
$\alpha_0^T = (1, 0)$ $\qquad$ $\alpha_\infty^T = (0.1, 0.2)$

$$\Pr[abaa] = \alpha_0^T\, T_a\, T_b\, T_a\, T_a\, \alpha_\infty = \alpha_0^T\, T_{abaa}\, \alpha_\infty$$

A particular case: PDFA

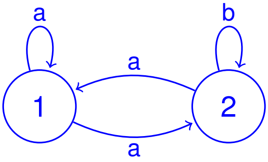
Probabilistic Deterministic Finite Automaton:

- at most one transition per (state,letter), one initial state
- $\therefore \alpha_0^T = (1, 0, \ldots, 0)$ and at most one non-zero per row in each T_a



Finite state models and distributions: HMM

HMM: No stopping probabilities. “Infinite duration” process



a	1	2
1	0.3	0.7
2	0.3	0

b	1	2
1	0	0
2	0	0.7

For each t, it defines a probability distribution on Σ^t
With initial probabilities α_0,

$$\Pr[abaa] = \alpha_0^T \, T_a \, T_b \, T_a \, T_a \, \mathbf{1}$$

Learning: Problem setting

Assumption: An unknown finite state machine generates a target distribution on $\Sigma^\star$, from which we can draw independent samples

Goal: Observing the samples, build a hypothesis generating a distribution that is (close to) the target distribution

Note: The hypothesis may be a finite state machine or some other algorithmic generation device

Learning: Problem setting

Identification in the Limit paradigm (IIL):

- At round t, algorithm gets one more sample x_t and produces a hypothesis h_t
- In the limit h_t generates the target distribution, with probability 1
- $\therefore$ "distance" between h_t and target tends to 0

Distance measures

Let p and q be distributions on $\Sigma^\star$

$$
\begin{aligned}
\mathrm{L}_\infty(p,q) &= \max_x |p(x)-q(x)| \\
\mathrm{L}_2(p,q) &= (\sum_x (p(x)-q(x))^2)^{1/2} \\
\mathrm{L}_1(p,q) &= \sum_x |p(x)-q(x)| \\
\mathrm{KL}(p,q) &= \sum_x p(x)\log\frac{p(x)}{q(x)}
\end{aligned}
$$

Theorem (Pinsker 64)

$$\mathrm{L}_\infty \le \mathrm{L}_2 \le \mathrm{L}_1 \le \sqrt{2\mathrm{KL}}$$

Learning: Problem setting

PAC Learning:

- how fast does convergence occur?
- i.e., how close are we to the target given a finite sample?
- and what is the computational complexity of the learner?

Learning: Problem setting

PAC Learning:

- Learner is given a sample of cardinality t
- With probability $1-\delta$, the learned distribution is at distance at most ε from the target distribution
- ... whenever $t \geq \mathrm{poly}(n, |\Sigma|, 1/\delta, 1/\varepsilon, \ldots)$
- running time polynomial in sample size, $|\Sigma|$, $1/\delta$, $1/\varepsilon$, ...

Notation: n = number of target states

Learning: Problem setting

Notes:

- PAC learning implies IIL learning
- Learning PFA and learning HMM are equivalent up to polynomials
- We often assume that n is known to the algorithm
  - Nontrivial assumption!

Learning: Results

- [Baum+ 70] Baum-Welch, practical heuristics for HMM
- [Rudich 85] Information-theoretic, IIL of HMM
- [Carrasco-Oncina 94] ALERGIA, an IIL method for PDFA
  - ... [dlHiguera+ 96], [Carrasco-Oncina 99], [dlHiguera-Thollard 00], [Thollard+ 00], [Kermorvant-Dupont 02], ...
- [Shalizi-Shalizi 04] CSSR, IIL for HMM
- [Denis-Esposito 04] IIL Residual Automata, between PDFA and PFA

Learning: Results

- [Abe-Warmuth 92] PFA are PAC learnable in polynomial space
    - the problem is computation, not information
- [Abe-Warmuth 92] NP-complete to PAC learn PFA in time poly(alphabet size)
- [Kearns+ 94] PAC learning PDFA is at least as hard as the noisy parity problem, even with 2-letter alphabets
- … meaning, possibly hard in time polynomial in n, $|\Sigma|$

Learning: Results

But PAC learning is possible ~~if we cheat~~ if we keep an open mind:

Why take only n and $|\Sigma|$ as the measure of PFA complexity?

Alternatively: we have an algorithm, and analyze its performance

- [Ron+ 96, Clark-Thollard 04] PAC learning PDFA
  - Performance depends on state distinguishability of target
- [Mossel-Roch 05, Hsu+ 09] PAC learning of PFA
  - Performance depends on spectral properties of target
  - Also [Denis+ 06, Bailly+ 09]

A hill-climbing heuristic: Baum-Welch

- Iterative method, instance of Expectation Maximization (EM)
- Idea:

  match predicted # transition visits to computed # transition visits
- Provably improves sample likelihood at each iteration
- No guarantee of convergence to optimum, nor convergence time

HMM parameters

HMM over Σ with n states: $M = \langle \alpha_0, \{T_a\}_{a \in \Sigma} \rangle$

$\alpha_0 \in \mathbb{R}^n$ is the initial distribution over states

$$\alpha_0[i] = \Pr[S(0) = i]$$

$T_a \in \mathbb{R}^{n \times n}$ provides the next state and observation distribution

$$T_a[i,j] = \Pr[S(t+1) = j, O(t+1) = a \mid S(t) = i]$$

Thus,

$$\Pr[x] = \Pr[x_1 \dots x_m] = \alpha_0^T T_{x_1} \cdots T_{x_m} \mathbf{1} = \alpha_0^T T_x \mathbf{1}$$

A posteriori probabilities

Define backward and forward probabilities:

$$\beta_t[j] = \Pr[S(t) = j \wedge x_1 \dots x_t]$$
$$\phi_t[j] = \Pr[x_{t+1} \dots x_m \mid S(t) = j]$$

or equivalently

$$\beta_t^T = \alpha_0^T T_{x_1} \cdots T_{x_t}$$
$$\phi_t = T_{x_{t+1}} \cdots T_{x_m} \mathbf{1}$$

Baum Welch algorithm

Goal: maximize the likelihood of the training sequence

Idea: A posteriori probabilities may translate to better parameters

Given sample $x_1 \ldots x_m$, update state visit and transition probabilities:

$$
\begin{aligned}
\alpha_0[j] &\leftarrow \frac{1}{m}\sum_t \Pr[S(t)=j \mid x_1 \ldots x_m] = \frac{1}{m}\sum_t \frac{\beta_t[j]\phi_t[j]}{\beta_t^T \phi_t} \\
\xi_t &\leftarrow \Pr[S(t-1)=i, S(t)=j, O(t)=x_t \mid x_1 \ldots x_m] \\
&= \frac{\beta_{t-1}[i]\, T_{x_t}[i,j]\phi_t[j]}{\beta_t^T \phi_t} \\
T_a[i,j] &\leftarrow \frac{\sum_{t:x_t=a} \xi_t}{\sum_t \xi_t}
\end{aligned}
$$

Baum Welch algorithm

get a sample and some initial guess M
repeat
    compute backward and forward probability vectors
    define next model M' updating α_0 and T_a
    $M \leftarrow M'$
until stopping condition

Stopping condition is either parameter convergence or loss of prediction accuracy

Baum Welch convergence

Theorem (Baum+ 70)

$\Pr[x_1 \cdots x_m \mid M'] \geq \Pr[x_1 \cdots x_m \mid M]$, *with equality at local maxima*

Proof idea

Let $x = x_1 \cdots x_m$, let y be a sequence of states and

$$Q(M, M') = \sum_y \Pr[x, y \mid M] \log \Pr[x, y \mid M']$$

- Likelihood function increases when we maximize $Q(M, M')$ as function of M'
- Then, find critical points of $Q(M, M')$ subject to stochastic constraints
- Lagrange multipliers show maximum is achieved on M' if defined as in the Baum-Welch algorithm

Baum-Welch, pros and cons

Pros:

- Intuitive goal: increase sample likelihood
- Hill-climbing guarantee

Cons:

- Strong dependence on initial guess
- May be stuck at local maxima, so neither IIL nor PAC learning

Weighted Automata

Generalization of PFA and DFA (not of NFA as acceptors)
A WA with n states is a tuple $\langle \alpha_0, \alpha_\infty, \{T_a\}_{a \in \Sigma} \rangle$,

- $\alpha_0 \in \mathbb{R}^n$
- $\alpha_\infty \in \mathbb{R}^n$
- $T_a \in \mathbb{R}^{n \times n}$

Defines a real-valued function $f : \Sigma^\star \to \mathbb{R}$:

$$f(x_1 \cdots x_m) = \alpha_0^T \, T_{x_1} \cdots T_{x_m} \, \alpha_\infty = \alpha_0^T \, T_x \, \alpha_\infty$$

Deterministic Weighted Automata (DWA) also make sense

The Hankel matrix

The Hankel matrix of $f : \Sigma^\star \to \mathbb{R}$ is

$$H_f \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$$

$$H_f[x, y] = f(xy)$$

$$\begin{array}{c|cccccc} & \lambda & a & b & aa & ab & \ldots \\ \hline \lambda & f(\lambda) & & & & \vdots & \\ a & & & & & \vdots & \\ b & & & & & \vdots & \\ aa & & & & & \vdots & \\ ab & & & & & \vdots & \\ ba & \ldots & \ldots & \ldots & \ldots & f(baab) & \\ \vdots & & & & & & \end{array}$$

Note: $f(z)$ goes into $|z| + 1$ entries

Weighted Automata and Hankel matrices

Let $f : \Sigma^\star \to \mathbb{R}$.

Theorem (Myhill-Nerode)

if f is 0/1 valued (a language),

$$\# \text{ distinct rows in } H_f = \# \text{ states in smallest DFA for } f$$

For arbitrary f:

$$\# \text{ distinct rows in } H_f \ \underline{\text{up to scalar multiplication}}$$
$$= \# \text{ states in smallest DWA for } f$$

$$\text{rank of } H_f = \# \text{ states in smallest WA for } f$$

Learning PDFA

1. An algebraic, high-level view, for learning DWA
2. Dealing with finite samples, for PDFA
3. Two instances: ALERGIA and Clark-Thollard

Learning DWA

Let $f : \Sigma^\star \to \mathbb{R}$.

If we could query the exact value of $f(x)$, for every x,

then we could do as follows ...

Learning DWA

1. Choose sets $X, Y \subseteq \Sigma^\star$, $\lambda \in X \cap Y$
2. Build $H = H_f[X \cup X\Sigma, Y]$
3. Find a minimal $X' \subseteq X$ such that, for every $z \in X$, $H[z,:]$ is a multiple of a row $H[x,:]$ for some $x \in X'$
4. Build a DWA from H, X, Y, X'

Building a DWA

Say $X' = \{x^1 = \lambda, x^2, \ldots, x^n\}$

- $\alpha_0^T = [1, 0, \ldots, 0]$
- $\alpha_\infty^T = [f(x^1), \ldots, f(x^n)]$
- $T_a[i,j] = \begin{cases} v & \text{if } H[x^i a, :] = v\, H[x^j, :] \\ 0 & \text{otherwise} \end{cases}$

Theorem: If $|X'| \geq$ # states in smallest DWA for f then this n-state DWA computes f

...but pigs don't fly

... back to learning PDFA from samples ...

- We get a finite sample S
- $X = \textit{prefixes}(S)$, $Y = \textit{suffixes}(S)$
- $\hat{H}[x, y]$ = empirical probability of xy in S

The question

"is $\hat{H}[x, :]$ a multiple of $\hat{H}[z, :]$?"

1. tends to unambiguous answer as sample size grows
2. for finite sample size, requires a statistical test

State merge - state split algorithms

- Many algorithms in the literature eventually rely on this construction
- But they identify states not in one shot from H, but by iteratively splitting or merging simpler states
- They often build PDFA, not WA
- Statistical tests and smoothing for unseen mass make a big difference

Example: ALERGIA [Carrasco-Oncina 94]

- Starts from a trivial prefix tree automaton for the sample
- Recursively merges "compatible" nodes, that seem to generate same distribution
- Hoeffding-based test for compatibility

Can be shown to learn in the IIL sense

Note: merging states increases the information about one state, w.r.t. considering a single row of $\hat{H}$

Example: Clark & Thollard's algorithm (2004)

- Grows state graph sequentially from a single-state graph
- Assumes a lower bound μ on L_∞-distinguishability:

$$\min\{\, L_\infty(\Pr[\colon|q],\Pr[\colon|q']) \mid q,\, q' \text{ distinct target states} \,\} \geq \mu$$

- Applies a test for L_∞ distance
- Smooths empirical transition probabilities

Example: Clark & Thollard's algorithm (2004)

Theorem

The [Clark-Thollard 04] algorithm is a PAC learner for PDFA with respect to the KL-distance, with sample size and running time

$$poly(n, |\Sigma|, 1/\mu, L, 1/\varepsilon, \log(1/\delta))$$

where L is the expected length of the distribution, and μ a lower bound on the L_∞-distinguishability of the target

Variants of [Clark-Thollard 04]

- [Gutman+ 05] Requires only L_2-distinguishability
- [Palmer-Goldberg 07] Learns w.r.t. L_1 distance – and no dependence on L
- [Castro-G 08] Online, adaptive method experimentally more efficient while still PAC-correct
- [Balle-Castro-G 12] PAC-learns in the data stream model: sublinear memory, sublinear time

Learning PFA

- [Hsu+ 09] PAC learns PFA as WA, spectral method
- [Denis+06] PAC learns PFA as WA, morally

Complexity depends on singular values of the Hankel matrix, besides n

Weighted Automata and rank

Theorem (Schützenberger 61, Carlyle+ 71, Fliess74, Beimel+ 00)

$f : \Sigma^\star \to \mathbb{R}$ is a finite WA iff $\mathrm{rank}(H_f)$ is finite.

Only if: take an n-state WA for f. Then $H_f = BF$, where $B \in \mathbb{R}^{\infty \times n}$ and $F \in \mathbb{R}^{n \times \infty}$

$$B[x,:] = \alpha_0^T T_x$$
$$F[:,y] = T_y \alpha_\infty$$

$\mathrm{rank}(H_f) \leq rank(B) \leq n$

Rank characterization

Theorem (Schützenberger 61, Carlyle+ 71, Fliess74, Beimel+ 00)

$f : \Sigma^\star \to \mathbb{R}$ is a finite WA iff $\mathrm{rank}(H_f)$ is finite.

If: Choose $X = \{x^1, \ldots, x^n\}$ and $Y = \{y^1, \ldots, y^n\}$ a rank basis of H_f with $x^1 = y^1 = \lambda$. Define $\alpha_0^T = (1, 0, \ldots, 0) \in \mathbb{R}^n$, $\alpha_\infty^T = (f(x^1), \ldots, f(x^n)) \in \mathbb{R}^n$, and $T_a \in \mathbb{R}^{n \times n}$ as $T_a[i,j] = a_j^i$ satisfying:

$$H_f[x^i a, :] = a_1^i H_f[x^1, :] + \cdots + a_n^i H_f[x^n, :].$$

By induction on $|w|$, it can be proved

$$f(x^i w) = T_w[i, :]\alpha_\infty$$

Thus,

$$f(z) = f(x^1 z) = T_z[1, :]\alpha_\infty = \alpha_0^T T_z \alpha_\infty$$

$H = f(XY)$, $H_a = f(XaY)$. We've defined

- $\alpha_0^T = (1,0,\dots,0) = (HH^{-1})[\lambda,:] = H[\lambda,:]H^{-1}$
- $\alpha_\infty = H[:,\lambda]$
- $T_a = H_a H^{-1}$

so that or $H_a = T_a H$, or $H[x^i a,:] = a_1^i H[x^1,:] + \cdots + a_n^i H[x^n,:]$, and

$$f(abc) = H[\lambda,:]H^{-1}H_a H^{-1}H_b H^{-1}H_c H^{-1}H[:,\lambda]$$

Therefore, if $H = QR$, $Q, R \in \mathbb{R}^{n\times n}$ the following also works:

- $\alpha_0^T = H[\lambda,:]R^{-1}$
- $\alpha_\infty = Q^{-1}H[:,\lambda]$
- $T_a = Q^{-1}H_a R^{-1}$

Learning PFA from a sample

- We get a finite sample S
- $X = \mathit{prefixes}(S)$, $Y = \mathit{suffixes}(S)$
- $\hat{H}[x, y]$ = empirical probability of xy in S = approximation to $f(xy)$

Problem: $\hat{H}$ will probably have maximal rank, even if $|X|, |Y| > n$

Learning PFA from a sample

Workaround 1:

Find $X' \subseteq X$ s.t.

1. X' easy to compute
2. $|X'| = n$
3. every row of $\hat{H}[X, :]$ is "almost" a linear combination of rows indexed by X'

Approach by [Denis+ 06]

Learning PFA from a sample

Workaround 2:

Find H' s.t.

1. H' easy to compute
2. H' same dimensions as $\hat{H}$, but rank n
3. H' "as close as possible" to $\hat{H}$ under some metric

Approach by [Hsu+ 09], spectral method

Singular Value Decomposition

Let $A \in \mathbb{R}^{m \times n}$. There are matrices $U \in \mathbb{R}^{m \times m}$, $D \in \mathbb{R}^{m \times n}$ and $V \in \mathbb{R}^{n \times n}$ such that:

- $A = UDV^T$
- U and V are orthonormal: $U^T U = I \in \mathbb{R}^{m \times m}$ and $V^T V = I \in \mathbb{R}^{n \times n}$
- D is a diagonal matrix of non-negative real numbers. Diagonal values are the singular values
- Column vectors of U are the left singular vectors

It follows $\mathrm{rank}(A) = \mathrm{rank}(D)$ is the number of non-zero singular values

W.l.o.g., diagonal values are nondecreasing, $\sigma_1 \geq \sigma_2 \geq \dots$

Singular Value Decomposition

For $H = UDV^T$, with $U, D, V \in \mathbb{R}^{m\times m}$ and $m \geq n$,

$$H'_n = \left(\begin{array}{c|c|c|c} u_1 & \cdots & u_n & \mathbf{0} \end{array}\right) \left(\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_n & \\ \hline & & & \mathbf{0} \end{array}\right) \left(\begin{array}{c} v_1 \\ \hline \vdots \\ \hline v_n \\ \hline \mathbf{0} \end{array}\right)$$

Fact

H'_n has rank n and minimizes $\|H - G\|_F$ among all rank-n matrices G

Singular Value Decomposition

We now want to replace H with H'_n in our algorithm

Problem: in the algorithm we used H^{-1}, or alternatively $H = QR$, then Q^{-1} and R^{-1}. They do not exist now

Luckily, we do not need the true inverses. One notion of pseudoinverse satisfies what we need for the proof, and is easily computable from the SVD decomposition

Moore-Penrose pseudoinverse

If $A \in \mathbb{R}^{m \times n}$ is a diagonal matrix, $A^+ \in \mathbb{R}^{n \times m}$ is formed by transposing A, and the taking the inverse of each non-zero element.

In the general case, if $A = UDV^T$ then $A^+ = VD^+U^T$

Some properties:

- in general, $AA^+ \neq I$ and $A^+A \neq I$
- but if A is invertible, $A^+ = A^{-1}$
- if columns of A are independent, $A^+A = I \in \mathbb{R}^{n \times n}$
- if rows of A are independent, $AA^+ = I \in \mathbb{R}^{m \times m}$

Spectral learning of PFA

$f : \Sigma^\star \to \mathbb{R}$ has finite rank n. Goal: find WA computing f.

1. Get n and sample S; $X = preffixes(S)$; $Y = suffixes(S)$
2. Def. $H[X, Y] \in \mathbb{R}^{p\times q}$ and set $H[x, y] =$ empirical probability of xy
3. Def. $H_a[X, Y] \in \mathbb{R}^{p\times q}$ and set $H_a[x, y] =$ empirical probability of xay
4. Let QR be a rank n factorization of H:
   - $Q \in \mathbb{R}^{p\times n}$ and $R \in \mathbb{R}^{n\times q}$ have rank n
   - $H = QR$

   For instance, take Q to be the first n left singular vectors of H
5. Output the WA M such that

$$\alpha_0^T = H[\lambda, :]R^+, \quad \alpha_\infty = Q^+ H[:, \lambda], \quad T_a = Q^+ H_a R^+$$

Convergence

Run the algorithm above (approximately) from a sample S.
The following PAC result holds for probability distributions P:

Theorem (Hsu+ 09, Balle+ 12)

Let σ_n be the nth largest singular value of H_P. If $|S| \geq \mathrm{poly}(n, |\Sigma|, 1/\sigma_n, 1/\varepsilon, \log(1/\delta))$, with probablity at least $1-\delta$:

$$\sum_{|x|=t} |P[x] - M[x]| < \varepsilon$$

Observation: $\sigma_n \neq 0$ iff $\mathrm{rank}(H_P) \geq n$

Practical implementation

Learn from substring statistics [Luque+ 12]

For $a, b, c \in \Sigma$, define

- $\widehat{P}[a,b] \sim$ expected number of times *ab* appears in a random string
- $\widehat{P}_c[a,b] \sim$ expected number of times *acb* appears in a random string
- $\widehat{p}_0[a] \sim$ probability of strings starting by *a*
- $\widehat{p}_\infty[a] \sim$ probability of strings ending by *a*
- $\widehat{U} =$ top *n* left singular vectors of $\widehat{P}$

Define WA *M* such that

$$\alpha_0^T = \widehat{p}_0^T (\widehat{U}^T \widehat{P})^+, \ \alpha_\infty = \widehat{U}^T \widehat{p}_\infty, \text{ and } T_c = \widehat{U}^T \widehat{P}_c (\widehat{U}^T \widehat{P})^+$$

Conclusions

- PFA turn out to be PAC learnable, with “the right” notion of PFA complexity
    - “polynomial in # states” may be the wrong question (?)
- Spectral method seems to be competitive with EM both in runtime, and learning curve [Balle+ 11, Luque+ 12, Bailly 12, Balle+ 12]

Conclusions / questions

- Current research: Spectral methods for more general WA's, e.g. transductors

- PDFA cannot exactly simulate all PFA
- But how well can PDFA approximate PFA?