Computational Learning of Weighted Automata

Ricard Gavaldà
Universitat Politècnica de Catalunya, Barcelona

7th Intl. Workshop on
Weighted Automata: Theory and Applications (WATA)
Leipzig, May 8th, 2014

With thanks to Borja Balle, Jorge Castro, Franco Luque, Ariadna Quattoni, Xavier Carreras, the WATA 2014 organizers, the BASMATI and SGR2009-1428 projects

The story in this talk

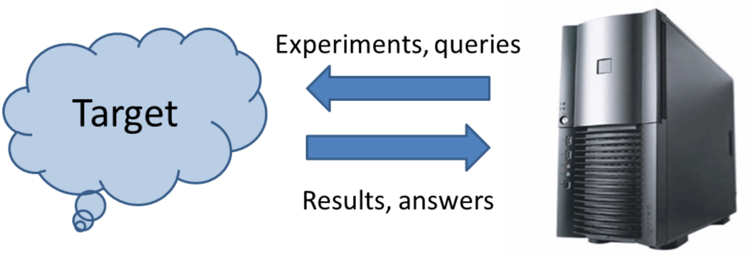
1. Weighted Automata over fields are provably, efficiently learnable in a formal model of function learning
2. Until recently: Probabilistic automata learnable from heuristics or (provably) if deterministic
3. Recent news: Full class of probabilistic automata provably learnable with WA algorithm + Singular Value Decomposition

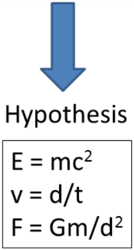
Outline

1. Background: Computational learning
2. Learning WA from queries
3. Learning probabilistic automata as WA
4. Conclusions and further work

Outline

Learning: what and why





Inferring a useful description of a phenomenon from observing and/or interacting with it

Learning: what and why?

- Alternative to explicit modeling by some human expert
- Traditional topic of machine learning, grammatical inference, statistics, pattern recognition, . . .
- Computational Learning Theory (80's): Formal models of learning, study computational resources needed to learn

Representation classes

- Target is supposed to belong to some representation class
- with an associated notion of "complexity" or "size"
- "more complex", "larger" targets require more resources

- Example:
  - "Learning regular languages" not well defined
  - "Learning DFA" well defined
  - Say, size = DFA states $\times$ letters

- Hypotheses in the same or larger representation class

Query Learning [Angluin87-88]

- Goal: Exactly learning the target
- Algorithm produces queries, target returns answers
- Time, memory of algorithm = poly(complexity of the target + length of longest answer)
- Complexity of computing the answers not considered

Query Learning [Angluin87-88]

We focus on the case targets are functions $f : A \to B$

Most common protocol:

- Evaluation queries: "given $x \in A$, return $f(x)$"
- Equivalence queries: "is $f = g$?"
  - YES, or
  - a counterexample $x \in A$ with $f(x) \neq g(x)$

PAC learning [Valiant84]

- Algorithm can ask for random samples $(x, f(x))$
- Samples are drawn independently from an unknown, arbitrary distribution D
- Goal is to approximately learn f, w.r.t. D, most of the times

---

Theorem [Angluin 88]: For every reasonable repr. class,

$$\text{Exact learning with Eval+Equiv queries} \\ \Longrightarrow \\ \text{PAC learning with Eval queries}$$

PAC learning [Valiant84]

Let f and D denote unknown target function and distribution

Let g denote the output of the algorithm upon seeing a sample S of f i.i.d. according to D, and reading parameters $\varepsilon, \delta \in (0,1)$

PAC learning occurs if

$$\Pr_{S \sim D}[\, D(f \triangle g) \leq \varepsilon \,] \geq 1 - \delta$$

Additionally, we require runtime and sample size polynomial in the complexity of target f, $1/\varepsilon$, and $1/\delta$

Outline

Early results

If target complexity = states $\times$ letters,
in the Equivalence + Evaluation query exact model,

Theorem [Angluin 87, Schapire 92, KV 94]

Deterministic Finite Automata are learnable

Theorem [AngluinKharitonov95]

Nondeterministic Finite Automata are not learnable under plausible cryptographic assumptions

Weighted Automata

Let R be a semiring

A WA with n states is a tuple $\langle \alpha_0, \alpha_\infty, \{T_a\}_{a \in \Sigma} \rangle$,

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

Defines a function $f : \Sigma^\star \to R$

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

Deterministic Weighted Automata (DWA) also make sense

Learning Weighted Automata

Theorem [BergadanoVarricchio94, Beimel+97]

Weighted automata over any field are learnable from Evaluation and Equivalence queries, assuming constant time for field operations

- Extends to commutative Artinian rings [Bshouty+ 98]
    - with hypotheses that are decision trees of WA
- Unlikely for the boolean semiring: implies learning NFA
- Not systematically studied for other semirings

Learning Weighted Automata

Theorem [BergadanoVarricchio94, Beimel+97]

Weighted automata over any field are learnable from Evaluation and Equivalence queries, assuming constant time for field operations

- WA define, in a certain algebraic setting, the largest class of Boolean functions learnable without learning DNF formulas [G-Thérien 09]

Importance in learning theory

Unifies and subsumes many learning results
at the expense of the larger hypothesis class, WA

- Unambiguous NFA
- Polynomials over finite fields
- Bounded degree polynomials over infinite fields
- Boolean decision trees
- Certain geometric boxes
- Certain subclasses of boolean DNF formulae
- ...

The Hankel matrix

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

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

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

	λ	a	b	aa	ab	$\dots$
λ	$f(\lambda)$				$\vdots$	
a					$\vdots$	
b					$\vdots$	
aa					$\vdots$	
ab					$\vdots$	
ba	$\dots$	$\dots$	$\dots$	$\dots$	$f(baab)$	
$\vdots$						

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

Weighted Automata and Hankel matrices

Hankel matrices provide information on WA size

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

Theorem (Myhill-Nerode)

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

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

Theorem (Castro-G13, probably known before)

For fields, # distinct rows in H_f up to scalar multiplication = # states in smallest DWA for f

Weighted Automata and Hankel matrix rank

Let f be $f : \Sigma^\star \to \mathbb{F}$, for $\mathbb{F}$ a field

Theorem (Schützenberger61,Carlyle+71,Fliess74,Beimel+97)

f has a WA of size $\leq n$ iff $rank(H_f) \leq n$.

Weighted Automata and Hankel matrix rank

Theorem (Schützenberger61,Carlyle+71,Fliess74,Beimel+97)

f has a WA of size $\leq n$ iff $rank(H_f) \leq n$.

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

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

$rank(H_f) \leq rank(B) \leq n$

Weighted Automata and Hankel matrix rank

Theorem (Schützenberger61,Carlyle+71,Fliess74,Beimel+97)

f has a WA of size $\leq n$ iff $rank(H_f) \leq n$.

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 $\alpha_0^T = (1, 0, \ldots, 0) \in \mathbb{F}^n$, $\alpha_\infty^T = (f(x^1), \ldots, f(x^n)) \in \mathbb{F}^n$, and $T_a \in \mathbb{F}^{n \times n}$ 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,

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

The algorithm

Grow sets $X, Y \subseteq \Sigma^\star$, initially empty

- Build WA:
  - fill $H = f(XY)$, $H_a = f(XaY)$ using Evaluation queries
  - $\alpha_0^T = (1, 0, \ldots, 0) = (H H^{-1})[\lambda, :] = H[\lambda, :]H^{-1}$
  - $\alpha_\infty = H[:, \lambda]$
  - $T_a = H_a H^{-1}$
- Ask $\langle \alpha_0, \alpha_\infty, \{T_a\}_a \rangle$ as Equivalence query
- If answer is YES, we are done
- else, use the counterexample to expand X and Y, increasing $rank(H_f[X, Y])$

The algorithm must stop when $rank(H_f[X, Y]) = rank(H_f)$

Outline

Probabilistic automata (PA) as WA

Setting:
Target $f : \Sigma^\star \to \mathbb{R}$ is a probability distribution computed by a PA

(Ref.: Colin de la Higuera's tutorial)

- Evaluation: "Give me the exact probability of x"
- Equivalence: "Does automaton h exactly compute the target distribution?"

Particular case of WA over $\mathbb{R}$, so exactly learnable

But this scenario is not very realistic

Stochastic setting

More realistic model:

- We sample independent runs of a target PA computing D
- We obtain a multiset of m strings = sample of D^m
- We want to compute a distribution D' "close to" D

PAC learning distributions [after Valiant84]

Let D be a probability distribution over $\Sigma^\star$

An algorithm PAC-learns D if upon seeing a sample from D^m and reading parameter $\varepsilon, \delta \in (0,1)$ it outputs a representation a distribution D' such that

$$\Pr[\, dist(D, D') \leq \varepsilon \,] \geq 1 - \delta$$

where, e.g.

$$dist(D, D') = L_1(D, D') = \sum_{x \in \Sigma^\star} |D(x) - D'(x)|$$

Additionally, we require the running time and m to be polynomial in the complexity of the target D, $1/\varepsilon$, and $1/\delta$

Good and bad news

Say “complexity of target PFA” = states $\times$ letters

- [AbeWarmuth92] There is an algorithm using polynomial sample size, but exponential time (pspace, actually)
- [AbeWarmuth92,Kearns+94] With plausible complexity-theoretic assumptions, poly-time learning is not possible, even for PDFA
  - “RP $\neq$ NP” for unbounded alphabet size
  - “noisy parity learning is hard” for binary alphabet
- Many heuristics proposed and used
  - EM (Baum-Welch), state merge - split (ALERGIA), Gibbs sampling

Changing perspective

Maybe states $\times$ alphabet is not the right "complexity measure"

Theorem [Clark-Thollard 04,Ron+96]

PDFA are PAC learnable in time polynomial in #states, alphabet size, $1/\varepsilon$, and a certain distinguishability parameter of the target PDFA

Theorem [Denis+06,Hsu+09,Bailly+09,Balle+11]

PFA are PAC learnable as WA in time polynomial in #states, alphabet size, $1/\varepsilon$, and e.g. some spectral value of the target distribution

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

It can be shown $\|H - \hat{H}\|_F = O(1/\sqrt{|S|})$

So, can we apply the WA algorithm on $\hat{H}$?

Problem: $\hat{H}$ probably has maximal rank, even if $|X|, |Y| \gg n$

Learning PFA from a sample

Central idea of spectral method: how to clean up $\hat{H}$

Find H_n s.t.

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

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

$\therefore rank(A) = rank(D) =$ number of non-zero singular values

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

Singular Value Decomposition

Let $H = UDV^T$ be the SVD of H

D is diagonal with nonnegative entries

For each n, let D_n keep only the largest n diagonal values of D

Fact

$H_n = UD_nV^T$ has rank n and minimizes $\|H - G\|_F$ among all rank-n matrices G

Frobenius norm: $\|A\|_F = \sqrt{\sum_{i,j} A_{i,j}^2} = \sqrt{\sum_i D_i^2}$

We now want to replace H with H_n in our algorithm

Problem: the algorithm uses H^{-1}, which now may not exist

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

Convergence & Generalization

The following PAC result holds for every D computed by PFA

Run the algorithm above on a sample S of D, get D'

Theorem (Hsu+ 09, Balle+ 12)

Let σ_n be the nth largest singular value of H_D. If $|S| \geq poly(n, |\Sigma|, 1/\sigma_n, 1/\varepsilon)$, then for each t with high probability

$$\sum_{|x|=t} |D[x] - D'[x]| < \varepsilon$$

Observation: $\sigma_n \neq 0$ iff $rank(H_D) \geq n$

Pros and cons

- It actually works. Faster than EM
- Can be rephrased / relaxed as the minimization of a convex loss function [Balle+12]
  - Mainstream in current Machine Learning today
- Con: Hypothesized WA need not be a probabilistic automaton
  - Weights and values not in $[0, 1]$, not summing to 1
  - Bad in some applications

Extensions

- Structured output. E.g. transducers, parsing [Balle+13,...]
- Some functions $\Sigma^\star \to \mathbb{R}$ that are not probability distributions [BalleMohri12]
    - but uses “low rank matrix completion” instead of SVD
- To non-string case, as “moment of methods” or “unmixing”

Outline

1. Background: Computational learning
2. Learning WA from queries
3. Learning probabilistic automata as WA
4. Conclusions and further work

Conclusions

- WA + SVD at the heart of new methods for learning probabilistic automata
- Efficient and with rigorous PAC guarantees
- Extensible to more complex tasks (transduction, parsing)

Some suggestions for further work

- Which semirings give learnable WA?
- Extensions to valuation monoids?
- Timed weighted automata?