

Weighted automata are compact and actively learnable

Artem Kaznatcheev^{a,b}, Prakash Panangaden^c

^a*Department of Biology, University of Pennsylvania, Philadelphia, USA*

^b*Department of Computer Science, University of Oxford, Oxford, UK*

^c*School of Computer Science, McGill University, Montreal, Canada*

Abstract

We show that weighted automata over the field of two elements can be exponentially more compact than non-deterministic finite state automata. To show this, we combine ideas from automata theory and communication complexity. However, weighted automata are also efficiently learnable in Angluin’s minimal adequate teacher model in a number of queries that is polynomial in the size of the minimal weighted automaton. We include an algorithm for learning WAs over any field based on a linear algebraic generalization of the Angluin-Schapire algorithm. Together, this produces a surprising result: weighted automata over fields are structured enough that even though they can be very compact, they are still efficiently learnable.

1. Introduction

Weighted automata (WAs) are an enriched model of finite state machines and define a natural representation of free monoids. They have received a lot of interest in the the learning community because they provide an interesting way to represent and analyze sequence data, such as music [16] or text and speech processing [15]. Mohri [14] provides a nice survey of algorithms related to weighted automata.

In this paper, we aim to expand the theoretical results known about the representational power and learnability of WAs. We show that WAs over \mathbb{Z}_2 (WA2s) – which can be viewed as word recognizers for regular languages in a natural way – can be exponentially more compact than non-deterministic finite state automata (NFAs) and yet learnable in Angluin’s [1] queries and counter-examples (or minimal adequate teacher) model in a number of membership queries and counter-example queries that is polynomial in the size of the minimal weighted automaton.

With Theorem 16, we show that there exists a family of languages where the minimal WA2s are exponentially smaller than the smallest NFAs. Unfortunately, in Theorem 23 we show that there also exists an exponential separation in the other direction. This shows that one can sometimes, but not always, get a significantly more compact representation by using WAs. However, the compactness result is still interesting because there are efficient algorithms for minimizing WAs [5] whereas finding a minimal NFA is PSPACE-complete [13].

This makes weighted automata compact yet – unlike NFAs [2, 8] – structured enough to be actively learnable in Angluin’s minimal-adequate teacher model. In Section 4, we show how to extend the Angluin-Schapire algorithm [1, 17] to weighted automata over any field. As such, we show that although WAs can be exponentially smaller than NFAs (and thus also DFAs), they still have a structure that we can exploit for efficient learning. Since weighted automata correspond more closely to pop-

ular models like POMDPs and probabilistic automata [7], this might open new avenues for learning algorithms of those representations.

2. Formal background

2.1. Finite state automata

Definition 1. Given a fixed alphabet Σ and finite dimensional vector space \mathbb{F}^n , a weighted automaton A over \mathbb{F} of size n is given by:

$$M = \langle \alpha, \omega \in \mathbb{F}^n, \{M^\sigma \in \mathbb{F}^{n \times n} \mid \sigma \in \Sigma\} \rangle \quad (1)$$

where α is the initial state, ω is a final measurement (or final state), and for each $\sigma \in \Sigma$ we have a corresponding transition matrix M^σ . The function recognized by this automaton is given by:

$$f_M(\sigma_1 \dots \sigma_m) = \alpha^T M^{\sigma_1} \dots M^{\sigma_m} \omega \quad (2)$$

When dealing with automata, it is useful to adapt a general matrix representation of the function they recognize:

Definition 2. Given a function $f : \Sigma^* \rightarrow \mathbb{F}$ the *Hankel matrix* $H_f : \Sigma^* \times \Sigma^* \rightarrow \mathbb{F}$ of f is: $H_f(u, v) = f(uv)$.

We will also talk about the *restricted Hankel matrix* $H_f|_n : \Sigma^n \times \Sigma^n \rightarrow \mathbb{F}$ of f to strings of length n .

The Hankel matrix allows us to come to grips with weighted automata and their size:

Proposition 3 ([6, 10]). $\text{rank}_{\mathbb{F}}(H_f) \leq n$ if and only if there exists a weighted automaton A over \mathbb{F} of size n such that $f_A = f$.

If we are going to study weighted automata over \mathbb{Z}_2 (WA2) and non-deterministic finite state automata (NFA) together then it is best to express them in a common framework. To do this, we will define a generic finite state automaton (Definition 4) and then see how augmenting this model with different acceptance criteria can produce NFAs (Definition 8) or WA2s (Definition 9), or restricting the kinds of transitions can produce deterministic finite-state automata (DFA; Definition 11).

Definition 4. A *finite state automaton* (FSA) is a tuple $A = \langle Q, \Sigma, \delta : Q \times \Sigma \rightarrow 2^Q, S \subseteq Q, F \subseteq Q \rangle$ where Q is a finite set of states, Σ is a finite alphabet, δ is the transition function, S is a set of starting states, and F is a set of final states. The size $|A|$ of the automaton is the number of states $|Q|$.

Definition 5. The dynamics of an FSA A are defined by looking at $\text{paths}_A : \Sigma^* \rightarrow 2^Q$ where for $p \in Q^*$, $w \in \Sigma^*$, $q, q' \in Q$, and $a \in \Sigma$ we have the recursive definition:

- $\text{paths}_A(\epsilon) = S$; and
- $pqq' \in \text{paths}_A(wa)$ if $pq \in \text{paths}_A(w)$ and $q' \in \delta(q, a)$.

We say that a *path is accepting* if it ends in F , or formally: $\text{apaths}_A(w) \subseteq \text{paths}_A(w)$ where $pq \in \text{apaths}_A(w)$ if $q \in F$.

It will also be useful to have the following two refinements of paths:

Definition 6. Given an FSA A and a state $q \in Q$ we say that a word $w \in \text{past}(q)$ if $pq \in \text{paths}(w)$ for some $p \in Q^*$.

In other words, $\text{past}(q)$ is the set of all words that lead to q . In a similar vein, we can define:

Definition 7. Given an FSA A and a state $q \in Q$ we say that $w \in \text{future}(q)$ if $\exists v \in \text{past}(q)$ $p, r \in Q^*$ s.t $pqr \in \text{apaths}(vw)$.

In other words, $\text{future}(q)$ is the set of all words that lead from a reachable state q to a state in F .

Together, Definitions 4 and 5 specify a generic finite state automaton and how it runs. What remains is how the automaton produces its corresponding recognized language. This requires giving an acceptance criterion or membership criterion for the corresponding language. Given an FSA A , we can get the traditional language $L^{\text{NFA}}(A)$ recognized by a non-deterministic finite automaton (NFA) as:

Definition 8. An FSA A is said to *NFA-recognize* a language $L^{\text{NFA}}(A)$ if

$$w \in L^{\text{NFA}}(A) \iff |\text{apaths}_A(w)| \geq 1. \quad (3)$$

For shorthand – and consistency with traditional nomenclature – we say that an NFA A recognizes language L_A to means that an FSA A NFA-recognizes a language L . The reason for the unwieldy term “NFA-recognize” is because we want to provide a similar definition for weighted automata over \mathbb{Z}_2 :

Definition 9. An FSA A is said to *WA2-recognize* a language $L^{\text{WA2}}(A)$ if

$$w \in L^{\text{WA2}}(A) \iff |\text{apaths}_A(w)| = 1 \pmod{2}. \quad (4)$$

This allows us to give a machine view of the matrix- and function-based Definition 1 of weighted automata:

Proposition 10. Given a weighted automaton M over \mathbb{Z}_2 of size n computing the function f_M :

$$M = \langle \alpha, \omega, \{M^\sigma \mid \sigma \in \Sigma\} \rangle \quad (5)$$

Let:

$$Q = \{1, \dots, n\} \quad (6)$$

$$\delta(q, \sigma) = \{r \mid M_{q,r}^\sigma = 1\} \quad (7)$$

$$S = \{r \mid \alpha_r = 1\} \quad (8)$$

$$F = \{r \mid \omega_r = 1\} \quad (9)$$

then the FSA $A = \langle Q, \Sigma, \delta, S, F \rangle$ WA2-recognizes L if

$$w \in L \iff f(w) = 1 \quad (10)$$

This transition between linear algebraic and machine views of weighted automata is standard [9], but we include a proof for convenience:

Proof. Note that it doesn’t matter when we switch to mod 2: the matrix multiplication in Definition 1 can be done over \mathbb{R} until we multiply by the final measurement vector. Since the transition function $\delta(\cdot, \sigma)$ is given by the matrix M^σ , we can just use matrix multiplication. Multiplying α by transition matrices against is the same thing as counting the number of paths from S . Multiplication by ω adds up the paths that lead to final states F and so computes $|\text{apaths}(w)|$. Finally taking the mod 2 that we deferred completes our computation. \square

As with NFAs, Proposition 10 allows us to shorten the unwieldy language of “FSA A WA2-recognizes the language L ” by the shorter and more traditional “WA2 A recognizes the language L ”.

Note also that by the same argument as Proposition 10, we could view the NFA from Definition 8 as a weighted automaton that uses the boolean semiring (‘or’ for addition, and ‘and’ for multiplication) instead of over a field as in Definition 1. In other words, NFAs can also be thought of as weighted automata over the boolean semiring. This is why when we discuss weighted automata in this article, we focus only on WAs over fields (and do not consider the more general setting of WAs over rings).

Finally, let us make the familiar definition of deterministic finite state automata by putting restrictions on δ and S :

Definition 11. An FSA A is a *deterministic finite automaton* (DFA) if it respects the restriction of a single start state ($|S| = 1$) and deterministic transitions:

$$\forall q \in Q, a \in \Sigma |\delta(q, a)| = 1; \quad (11)$$

The DFA A is said to recognize a language L_A if

$$w \in L_A \subseteq \Sigma^* \iff |\text{paths}(w)| = 1. \quad (12)$$

Note that the DFA restrictions of a single start state and determinism (Equation 11) imply that given a DFA A , any word w defines only one path (i.e., $\forall w \in \Sigma^* |\text{paths}_A(w)| = 1$) and this path is either accepting or not. This means that a DFA is also an NFA, and WA2.

2.2. Tools from communication complexity

It will be useful to observe a link between the Hankel matrix and a concept from communication complexity:

Definition 12. The *1-monochromatic rectangle covering* of a function $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ is the smallest number $\chi_1(f)$ of pairs of sets (called rectangles) $A_i, B_i \subseteq \{0, 1\}^n$ for $1 \leq i \leq \chi_1(f)$ such that:

1. for every $(x, y) \in A_i \times B_i$ we have $f(x, y) = 1$ (i.e., $A_i \times B_i$ is 1-monochromatic), and
2. for every $(x, y) \in f^{-1}(1)$ we have at least one index $i \in \{1, \dots, \chi_1(f)\}$ such that $(x, y) \in A_i \times B_i$.

Based on formalizing the argument in Hromkovič and Schnitger [11] that views NFAs as a non-deterministic one-way communication protocol where the message sent by the first computer to the second corresponds to the state of the NFA, we can show that the 1-monochromatic rectangle covering (which is a kind of non-deterministic one-way communication protocol) lower bounds the size of NFAs:

Proposition 13. $|\text{NFA}(f)| \geq \chi_1(H_{f|_n})$ for any $n \in \mathbb{N}$.

Proof. Let A be a minimal NFA recognizing f . For each state $q \in Q$ define $A_q = \text{past}(q)$ and $B_q = \text{future}(q)$, by the definition of $\text{future}(q)$ for any $u \in A_q$ and $v \in B_q$ we have $f(uv) = 1$. Therefore, the $\{A_q, B_q\}_{q \in Q}$ are 1-monochromatic rectangles. Now, consider any $uv \in f^{-1}(1)$, say that $q \in Q_u$ if $\exists p \in Q^*$ such that $pq \in \text{paths}(u)$. Since $f(uv) = 1$, there must be at least one $q \in Q_u$ such that $v \in \text{future}(q) = B_q$. Therefore, the $\{A_q, B_q\}_{q \in Q}$ are a cover of the whole Hankel matrix, and hence any restricted submatrix is also covered. \square

Another useful tool for proving lower bounds in communication complexity is:

Definition 14. The *discrepancy* of a function $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ is:

$$\text{disc}(f) = \max_{A, B \subseteq \{0, 1\}^n} \frac{1}{2^{2n}} \left| \sum_{x \in A, y \in B} (-1)^{f(x, y)} \right| \quad (13)$$

Definitions 12 and 14 relate nicely to each other by an extension of Lemma 13.13 from Arora and Barak [3]:

Lemma 15. $\chi_1(f) \geq \frac{|f^{-1}(1)|}{2^{2n} \text{disc}(f)}$

Proof. Since all the ones in our function can be covered by $\chi_1(f)$ squares, and a total of $|f^{-1}(1)|$ ones need to be covered, there must be at least one monochromatic rectangle $A \times B$ that covers the average number of ones or more. This means that $|A||B| \geq |f^{-1}(1)|/\chi_1(f)$. Now, since the discrepancy is a max over rectangles, we can pick $A \times B$ to lower bound it:

$$\text{disc}(f) \geq \frac{1}{2^{2n}} \left| \sum_{x \in A, y \in B} (-1)^{f(x, y)} \right| \quad (14)$$

$$\geq \frac{1}{2^{2n}} \left| \sum_{x \in A, y \in B} -1 \right| \quad (15)$$

$$\geq \frac{|A||B|}{2^{2n}} \quad (16)$$

$$\geq \frac{|f^{-1}(1)|}{2^{2n} \chi_1(f)} \quad (17)$$

where the second line follows from the first because the rectangle is 1-monochromatic. The last line can be rearranged to complete the proof. \square

3. Size of NFAs and WA2s

We are interested in the following question: given a regular language L , what is the size of the smallest automaton A with $L_A = L$? In particular, we will define $|\text{NFA}(L)|$ to be the largest integer such that for any NFA A , if $L_A = L$ then $|A| \geq |\text{NFA}(L)|$ and similarly for $|\text{DFA}(L)|$, and $|\text{WA2}(L)|$.

3.1. WA2s can be exponentially smaller than NFAs

The gap between $|\text{NFA}(L)|$ and $|\text{WA2}(L)|$ can be exponentially large. Technically, this means that:

Theorem 16. *There exists a family of regular languages $\{L_n\}$ such that $|\text{NFA}(L_n)| \in 2^{\Omega(|\text{WA2}(L_n)|)}$*

To find our separating family of languages, we will look at the inner-product function:

Definition 17. The *n -bit inner product* is a function $\wedge^{\otimes n} : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ acting on two bit strings $x = x_1 \dots x_n \in \{0, 1\}^n$ and $y = y_1 \dots y_n \in \{0, 1\}^n$ as $x \wedge^{\otimes n} y = \sum_{i=1}^n x_i \cdot y_i \pmod 2$

Sometimes, when the size of x and y is obvious, we will omit the $\otimes n$. Note that the number of zeros and ones in \wedge is well balanced.

Proposition 18. $|(\wedge^{\otimes n})^{-1}(1)| = 2^{n-1}(2^n - 1)$

Proof. Let $D = \{(x, y) | \exists i \in [n] \text{ s.t. } x_i = y_i\}$ be the set of pairs of strings that overlap in at least one place. Now, consider a function h defined on D that given (x, y) take the smallest index of overlap i (i.e. for all $j < i, x_j \neq y_j$) and sends $x_i \rightarrow \bar{x}_i$ and $y_i \rightarrow \bar{y}_i$ this function is a bijection on D . However, note that if

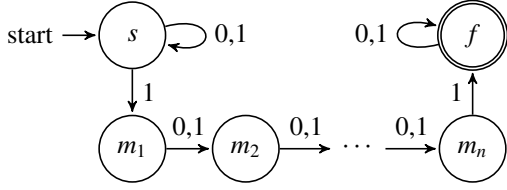


Figure 1: A picture of the weighted automaton used to prove Theorem 16.

$\wedge^{\otimes n}(x, y) = b$ then $\wedge^{\otimes n}h(x, y) = \bar{b}$. Thus, \wedge has the same number of zeros and ones in D .

The only pairs missing from D are the ones of the form (x, \bar{x}) and there are 2^n such strings, so $|D| = 2^{2n} - 2^n$. Finally, note that $x \wedge^{\otimes n} \bar{x} = 0$ thus $|(\wedge^{\otimes n})^{-1}(1)| = |D|/2$. \square

Lemma 19. $\chi_1(\wedge^{\otimes n}) \geq 2^{n/2-2}$

Proof. Example 13.16 in [3] shows that $\text{disc}(\wedge^{\otimes n}) \leq 2^{-n/2}$ which combined with Lemma 15 and Proposition 18 gives us $\chi_1(\wedge^{\otimes n}) \geq \frac{2^{n-1}(2^n-1)2^{n/2}}{2^{2n}} \geq 2^{n/2-2}$. \square

The inner-product allows us to define a special class of language families with an important property:

Definition 20. A language family $\{L_n\}$ is called an *inner-product kernel family* if:

$$\forall n \forall x, y \in \{0, 1\}^n \quad L_n(xy) = x \wedge^{\otimes n} y \quad (18)$$

Note that the above definition places no restriction on how L_n behaves on words of length other than $2n$, so there are many inner-product kernel families based on the many ways languages can behave outside the kernels.

Proposition 21. *If $\{L_n\}$ is an inner-product kernel family then $|\text{NFA}(L_n)| \geq 2^{n/2-2}$*

Proof. We use the communication complexity techniques from Proposition 13. We can use any finite submatrix of H_L to lower-bound $|\text{NFA}(L)|$. In particular, if for L_n we look at the submatrix of H_L with rows and columns indexed by strings of length n then this submatrix is the same as the matrix for $\wedge^{\otimes n}$. Thus, $|\text{NFA}(L_n)| \geq \chi_1(\wedge^{\otimes n}) \geq 2^{n/2-2}$ where the first inequality is an application of the Proposition 13 lowerbound technique and the second inequality is from Lemma 19. \square

We finish the proof of Theorem 16 by noticing that the family of weighted automata in Figure 1 recognize languages in an inner-product kernel family but only have $n + 2$ states. More formally:

Proposition 22. *Let $\text{WA}_n^{\text{prod}}$ be the weighted automaton in Figure 1. Given any $x, y \in \{0, 1\}^n$:*

$$xy \in L_{\text{WA}_n^{\text{prod}}} \iff \sum_{i=1}^n x_i y_i \pmod{2} = 1. \quad (19)$$

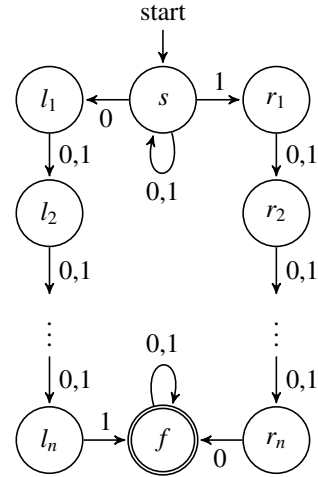


Figure 2: A picture of the NFA used in the proof of Theorem 23

Proof. Any accepting path in $\text{WA}_n^{\text{prod}}$ must have the form $p \in s^* m_1 m_2 \dots m_n f^*$. A path p is caused by transitions corresponding to a word of the pattern:

$$\{0, 1\}^* 1 \{0, 1\}^{n-1} 1 \{0, 1\}^* \quad (20)$$

i.e., by a word that has two 1s that are exactly n letters apart.

Now, let us count the number of accepting paths for any xy . The word xy matches the pattern in Equation 20 for each $1 \leq i \leq n$ such that $x_i = y_i = 1$ and for no other: i.e., only for the partition $\{0, 1\}^{i-1} 1 \{0, 1\}^{n-1-i}$. Each of these partitions of xy corresponds to a unique path, so the total number of accepting paths is $\sum_{i=1}^n x_i y_i$ and Equation 19 follows from the acceptance criteria of WAs in Definition 9. \square

3.2. NFAs can be exponentially smaller than WA2s

Unfortunately, there are also cases where the opposite happens and we do not have a small WA2 while a small NFA exists:

Theorem 23. *There exists a family of regular languages $\{L_n\}$ such that $|\text{WA2}(L_n)| \in 2^{\Omega(|\text{NFA}(L_n)|)}$*

Proof. For this, consider a language family where for $u, v \in \{0, 1\}^n$ $uv \in L_n$ if and only if $u \neq v$. If we look at the Hankel matrix of L_n restricted to columns and rows of length n then it is a matrix of all ones except with zeros on the diagonal. Clearly, this matrix has full rank, so by Theorem 3 $|\text{WA2}(L_n)| \geq 2^n$.

On the other hand, an NFA of size $2(n + 1)$ is given that recognizes a language consistent with L_n in Figure 2. Notice that any accepting path in this NFA can only have been caused by a word of the pattern $\{0, 1\}^* 0 \{0, 1\}^{n-1} 1 \{0, 1\}^*$ (left branch) or $\{0, 1\}^* 1 \{0, 1\}^{n-1} 0 \{0, 1\}^*$ (right branch). When we restrict this to words xy with $x, y \in \{0, 1\}^n$, we see that one of the patterns is realized only if there is some $1 \leq i \leq n$ such that $x_i \neq y_i$. \square

4. Efficient active learning algorithm for weighted automata

Deterministic finite state automata (DFAs) are not passive learnable: i.e., DFAs are known to be difficult to PAC-learn

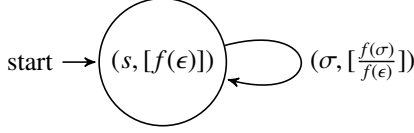


Figure 3: Initial weighted automaton. There is a single state that is initial and outputs its weight times $f(\epsilon)$. There is a self-loop for each letter $\sigma \in \Sigma^*$ weighted by $\frac{f(\sigma)}{f(\epsilon)}$. Note that we are using the WLOG assumption that $f(\epsilon) \neq 0$.

from randomly drawn labeled examples in any representation [12]. However, we can instead consider a model with active learning that instead of random labeled examples has the following two types of queries:

1. for any string $x \in \Sigma^*$ we can do a membership query to get $f(x)$. This is the active learning component, since the algorithm generates the query to ask, and
2. given a candidate weighted automaton A , we can ask if it is correct with a counter-example query. If A computes f (i.e. $f_A = f$) then the teacher will say ‘‘CORRECT’’, otherwise the teacher will return a counter-example z such that $f_A(z) \neq f(z)$. If a teacher is unavailable then this can alternatively be replaced by random sampling if we want a PAC-like model, and would correspond to the non-active part of learning.

This is Angluin’s queries and counter-examples or ‘minimal adequate teacher’ (MAT) model [1]. Angluin [1] famously showed that – in the MAT model – regular languages are efficiently learnable in the size of their minimal DFA representation. Later, Schapire [17] improved the efficiency of Angluin’s algorithm for learning DFAs. In this section, we show how to adapt the Angluin-Schapire algorithm from learning DFAs to learning WAs over any field \mathbb{F} .

For the rest of the section, suppose we are trying to learn an unknown function $f : \Sigma^* \rightarrow \mathbb{F}$ with Hankel matrix $H : \Sigma^* \times \Sigma^* \rightarrow \mathbb{F}$.

4.1. Initialization

At all times, our algorithm will keep track of two finite sets $S, E \subseteq \Sigma^*$ of equal size ($|S| = |E|$). S will be prefix closed and we will call its elements states.

For convenience, we will define a function $F : S \rightarrow \mathbb{F}^E$. If we view F as a matrix, then it is a restriction of H to S and E , i.e. $F = H(S, E)$ or more explicitly for $s \in S$ and $e \in E$, $F(s, e) = f(se)$. Our algorithm will ensure that F is full rank, i.e. $\text{rank}_{\mathbb{F}}(F) = |S|$.

We will start with $S = E = \{\epsilon\}$ and without loss of generality assume that $f(\epsilon) \neq 0$ (if it is equal zero then just replace f by $f + 1$, learn that, and then subtract 1 from each value in the final/measurement state). See Figure 3 for the initial automaton. This initialization requires one membership query to learn $f(\epsilon)$.

4.2. Automaton corresponding to matrix F

For each $\sigma \in \Sigma$, consider $F^\sigma : S \rightarrow \mathbb{F}^E$ where $F^\sigma(s, e) = f(s\sigma e)$. Since F has full rank, we know that its columns form

a basis for \mathbb{F}^E . Thus, every other vector $F^\sigma(s) \in \mathbb{F}^E$ can be expressed as some linear combination of the $F^\sigma(s')$ for $s' \in S$. Define $T^\sigma : S \times S \rightarrow \mathbb{F}$ as the matrix that stores the coefficients of these linear combinations: i.e., define T^σ such that for every $s \in S$ we have $F^\sigma(s) = \sum_{s' \in S} T_{s,s'}^\sigma F^\sigma(s')$.

This allows us to define the corresponding weighted automaton over \mathbb{F} (see Definition 1) on state space \mathbb{F}^S . Let the weighted automaton T have...

- initial state α such that $\alpha(\epsilon) = 1$ and $\alpha(s) = 0$ if $s \neq \epsilon$,
- final/measurement state $\omega = F(\cdot, \epsilon)$ (i.e., the row of F corresponding to $\epsilon \in E$, and
- transition matrices T^σ .

4.3. Learning from counter-example query

Now, suppose we tried this automaton T and our teacher returned a counter-example z . We will use this counter-example to find strings to extend S and E and thus increase the rank of our matrix F . Now for each $1 \leq i \leq |z|+1$ consider the partitions $z = z_{<i}\sigma_i z_{>i}$. For each $z_{<i}$ define $Z_i : S \rightarrow \mathbb{F}$ to be the state of our candidate automaton when we run it on $z_{<i}$:

$$Z_i = \alpha^T T^{z_1} T^{z_2} \dots T^{z_{i-1}}. \quad (21)$$

Let $f_i = \sum_{s \in S} Z_i(s) f(s\sigma_i z_{>i})$. From our definition, we know that $f_1 = f(z) \neq f_T(z) = f_{|z|+1}$, thus as we increase i there must be some point k where $f_k \neq f_{k+1}$. Find this point by using binary search on i . This requires at most $|S| \lceil \log(|z|) \rceil$ membership queries to f .

Let us write out f_{k+1} :

$$f_{k+1} = \sum_{s' \in S} Z_{k+1}(s') f(s' z_{>k}) \quad (22)$$

$$= \sum_{s, s' \in S} T_{s,s'}^{\sigma_k} Z_k(s) f(s' z_{>k}) \quad (23)$$

Now, proceed by contradiction: if $\forall s \in S$ we have $f(s\sigma_k z_{>k}) = \sum_{s' \in S} T_{s,s'}^{\sigma_k} f(s' z_{>k})$ then

$$f_k = \sum_{s \in S} Z_k(s) f(s\sigma_k z_{>k}) \quad (24)$$

$$= \sum_{s \in S} Z_k(s) \sum_{s' \in S} T_{s,s'}^{\sigma_k} f(s' z_{>k}) = f_{k+1} \quad (25)$$

where the last equality follows from Equation 23 and contradicts $f_k \neq f_{k+1}$. Thus, there must be some $s^* \in S$ such that $f(s^* \sigma_k z_{>k}) \neq \sum_{s' \in S} T_{s^*,s'}^{\sigma_k} f(s' z_{>k})$.

Now, consider an $s\sigma \in S$ then

$$F(s\sigma) = F_\sigma(s) = \sum_{s'} T_{s,s'}^\sigma F(s') \quad (26)$$

but since the $F(s)$ are linearly independent, we must have that $T_{s,s\sigma}^\sigma = 1$ and for $s' \neq s\sigma$ we must have $T_{s,s'}^\sigma = 0$. Plugging this into our contradiction assumption, we see that for $s\sigma_k \in S$ we have $\sum_{s' \in S} T_{s\sigma_k, s'}^{\sigma_k} f(s' z_{>k}) = f(s\sigma_k z_{>k})$. Therefore, our $s^* \sigma_k \notin S$. Now, we can add $s^* \sigma_k$ to S and $z_{>k}$ to E to get a new linearly independent row and column and increase the rank of our matrix by 1.

4.4. Termination

Since our candidate automaton agrees with f on every value in F , it must be that the real weighted automaton corresponding to f must have more states than $\text{rank}(F)$. At every counter-example query, we increase our rank by one, so if our world f is represented by a minimum weighted automaton with n states then after $n - 1$ counter-example queries we must have $\text{rank}(F) = \text{rank}(H_f)$. Since our automata agrees with f on every value in F , the n th counter-example query gets it “CORRECT”. If m is the length of the longest-counterexample then the total number of membership queries is less than $1 + \binom{n}{2} \lceil \log m \rceil$.

5. Discussion and Conclusion

As far as we know, this is the first time it has been show that weighted automata (WAs) can be exponentially smaller than NFAs. Together with the learning algorithm, this produces a somewhat surprising result: weighted automata are structured enough that even though they are compact, they are still efficiently learnable. This also means that some languages where the minimal DFAs and NFAs are exponentially bigger than the minimal WAs can be learned much faster using the WA representation.

This is not the case for NFAs. Although several algorithms have been developed for learning NFAs in the minimum adequate teacher model [18, 4], the results for NFAs differ from the case of WAs in two fundamental ways:

1. The algorithms for learning NFAs are not guaranteed to return a minimal NFA that recognizes the language. In fact, they return a special kind of NFAs called residual finite state automata (RFSAs) [8, 4]. These RFSAs are always the same size or larger than NFAs and in some cases are exponentially larger than the minimal NFA that recognizes a language [8].
2. The number of queries required for learning these RFSAs is not polynomial in the size of the minimal NFA nor the minimal RFSA, but only polynomial in the size of the minimal DFA. So although RFSAs can be exponentially more compact than DFAs, this does not necessarily provide a speed-up for learning those RFSAs. In fact, there are hardness results suggesting that one cannot learn NFAs or RFSAs in a number of queries that is polynomial in the size of smallest NFA recognized the language [2] nor polynomial in the size of smallest RFSA recognizing the language [8].

In contrast, we show that a minimal WA can be learned in a number of queries that is polynomial in the size of the minimal WA corresponding to that unknown function. Since WAs are always smaller than DFAs and sometimes exponentially smaller, that means that learning WA2s replaces the standard Angluin-Schapire algorithm [1, 17] for learning regular languages. In the cases where WAs are the same size as DFAs, we can achieve the same performance, and in the cases in which WAs are more compact, we provide an exponential savings in terms of queries used.

Acknowledgements

We are indebted to helpful discussion with Borja Balle and Doina Precup. The paper also benefited from the feedback of several anonymous reviewers. The work began when AK was at the School of Computer Science, McGill University and completed thanks to the generous support of a James S. McDonnell Foundation Postdoctoral Fellowship for Understanding Dynamic & Multi-scale Systems.

References

- [1] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and computation*, 75(2):87–106, 1987.
- [2] Dana Angluin and Michael Kharitonov. When won’t membership queries help? *Journal of Computer and System Sciences*, 50(2):336–355, 1995.
- [3] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*, volume 1. Cambridge University Press Cambridge, 2009.
- [4] Benedikt Bollig, Peter Habermehl, Carsten Kern, and Martin Leucker. Angluin-style learning of nfa. In *IJCAI*, volume 9, pages 1004–1009, 2009.
- [5] Filippo Bonchi, Marcello Bonsangue, Helle Hvid Hansen, Prakash Panangaden, Jan Rutten, and Alexandra Silva. Algebra-coalgebra duality in Brzozowski’s minimization algorithm. *ACM Transactions of Computational Logic*, 2013.
- [6] J.W. Carlyle and A. Paz. Realizations by stochastic finite automata. *J. Comput. Syst. Sci.*, 5:26–40, 1971.
- [7] Alexander Clark and Franck Thollard. PAC-learnability of probabilistic deterministic finite state automata. *Journal of Machine Learning Research*, 5:473–497, 2004.
- [8] François Denis, Aurélien Lemay, and Alain Terlutte. Learning regular languages using RFSAs. *Theoretical Computer Science*, 313(2):267–294, 2004.
- [9] M. Droste, W. Kuich, and H. Vogler. *Handbook of Weighted Automata*. Springer, 2009.
- [10] M. Fliess. Matrices de Hankel. *Journal de Mathématiques Pures et Appliquées*, 53:187–222, 1974.
- [11] Juraj Hromkovič and Georg Schnitger. On the hardness of determining small NFAs and of proving lower bounds on their sizes. In *Developments in Language Theory*, pages 34–55. Springer, 2008.
- [12] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.
- [13] Albert Meyer and Larry Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th Annual IEEE Symposium on Switching and Automata Theory*, pages 125–129, 1972.
- [14] Mehryar Mohri. Weighted automata algorithms. In *Handbook of Weighted Automata*, pages 213–254. Springer, 2009.
- [15] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In *Handbook on Speech Processing and Speech Communication, Part E: Speech recognition*. Springer, 2008.
- [16] Mehryar Mohri, Pedro Moreno, and Eugene Weinstein. Efficient and robust music identification with weighted finite-state transducers. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(1):197–207, 2009.
- [17] Robert E Schapire. The design and analysis of efficient learning algorithms. Technical report, DTIC Document, 1991.
- [18] Takashi Yokomori. Learning non-deterministic finite automata from queries and counterexamples. *Machine Intelligence*, 13:169–189, 1994.