Mathematical Structures in Computer Science (2024), **34**, pp. 128–146
doi:10.1017/S0960129523000373

CAMBRIDGE UNIVERSITY PRESS

PAPER

Learning quantum finite automata with queries

Daowen Qiu[1,2]

[1]Institute of Quantum Computing and Computer Theory, School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China and [2]The Guangdong Key Laboratory of Information Security Technology, Sun Yat-sen University, Guangzhou 510006, China
Email: issqdw@mail.sysu.edu.cn

(Received 28 December 2021; revised 8 August 2023; accepted 28 October 2023; first published online 30 November 2023)

Abstract

Learning finite automata (termed as *model learning*) has become an important field in machine learning and has been useful realistic applications. Quantum finite automata (QFA) are simple models of quantum computers with finite memory. Due to their simplicity, QFA have well physical realizability, but one-way QFA still have essential advantages over classical finite automata with regard to state complexity (two-way QFA are more powerful than classical finite automata in computation ability as well). As a different problem in *quantum learning theory* and *quantum machine learning*, in this paper, our purpose is to initiate the study of *learning QFA with queries* (naturally it may be termed as *quantum model learning*), and the main results are regarding learning two basic one-way QFA (1QFA): (1) we propose a learning algorithm for measure-once 1QFA (MO-1QFA) with query complexity of polynomial time and (2) we propose a learning algorithm for measure-many 1QFA (MM-1QFA) with query complexity of polynomial time, as well.

Keywords: quantum computing; quantum finite automata; learning from queries; quantum model learning; SD oracles

1. Introduction

Learning finite automata has become an important field in machine learning (Kearns and Vazirani, 1994) and has been applied to wide-ranging realistic problems (Higuera, 2005; Vaandrager, 2017), for example, smartcards, network protocols, legacy software, robotics and control systems, pattern recognition, computational linguistics, computational biology, data compression, data mining, etc. In Vaandrager (2017), learning finite automata is termed as *model learning*. In fact, model learning and model checking as well as and model-based testing have intrinsic connections (see the pioneering contribution of Peled et al. (2002) and Higuera (2010)).

Learning finite automata was first considered by Moore (1956) and an exponential-time query algorithm was proposed. In particular, Angluin (1987) proposed the so-called membership queries (MQ) and equivalence queries (EQ), a ground-breaking method for learning the models of finite automata. For learning *deterministic finite automata* (DFA), according to Angluin's algorithm, the learner initially only knows the inputs (i.e. alphabet) of the model to be learned (say $\mathcal{M}$), and the aim of the learner is to learn the model by means of two types of queries, that is, MQ and EQ. MQ means that the learner asks what the result (accepting or rejecting) of output is in response to an input sequence, and the oracle answers with accepting or rejecting, while EQ signifies the learner whether a hypothesized machine model (say $\mathcal{H}$) is the same as the learned machine, and the oracle answers yes if this is the case. Otherwise "no" is replied and an input string is provided as a counterexample to distinguish $\mathcal{H}$ and $\mathcal{M}$.

The complexity of queries of Angluin's algorithm (Angluin, 1987) is polynomial for learning DFA and Mealy machines. Angluin (1988) proved that DFA cannot be learned in polynomial time by MQ (or EQ) only. Since Angluin's algorithm was proposed by Angluin (1987), learning other models of finite automata has been investigated. Tzeng (1992) studied learning *probabilistic finite automata* (PFA) and Markov chains via *SD oracle*, where SD oracle can answer state distribution, i.e., probability distribution of states for each input string, so it is more powerful than MQ. For learning DFA via SD oracle, a state is replied for each input string, and the query complexity of learning DFA via SD oracle is polynomial (Tzeng, 1992).

Then Bergadano and Varricchio (1996) used MQ and EQ to learn appropriately PFA, and a *probably approximately correct* learning algorithm (i.e. PAC algorithm) was presented. Learning *nondeterministic finite automata* (NFA) was studied by Bollig (2009). In recent years, Angluin et al. (2015) initiated the research of learning alternating automata, and Berndt et al. (2017) further solved the learning problem of residual alternating automata.

A natural inquiry is that SD oracle seems too strong. However, it was showed by Tzeng (1992) that SD oracle is actually not too strong for learning DFA and PFA if the query complexity is required to be polynomial, because learning a consistency problem related to DFA and PFA via SD oracle is still NP-complete (Tzeng, 1992). In this paper, we use an *AD oracle* for learning *quantum finite automata* (QFA) in polynomial time, that is, AD oracle can answer a state of superposition for each input string, i.e., amplitude distribution of states. Similarly it follows that using *AD oracle* to learn a consistency problem related to reversible finite automata (RFA) and MO-1QFA is NP-complete.

Quantum machine learning (QML) was early considered by Bshouty and Jackson (1999) with learning from quantum examples, and then *quantum learning theory* (Arunachalam and de Wolf, 2017) as an important theoretical subject of QML has been deeply developed. Quantum learning theory (Arunachalam and de Wolf, 2017) includes models of quantum exact learning, quantum PAC learning, and quantum agnostic learning; these models are combinations of corresponding classical learning models with quantum computing (in a way, quantum query algorithms). We further review quantum learning theory and QML more specifically.

We first recall quantum learning theory, which studies the theoretical aspects of QML. As pointed out above, Bshouty and Jackson (1999) showed that all PAC-learnable function classes are learnable in the quantum models, and notably, Servedio and Gortler (2004) studied quantum versions of Angluin's model of exact learning from MQ and Valiant's PAC model of learning from random examples. Then, Aaronson (2007) investigated learning quantum states, and Zhang (2010) further investigated the quantum PAC learning model. Gavinsky (2012) initialed a new quantum learning model called predictive quantum, which is the quantum analogue of PAC, and afterwards, Belovs (2015) investigated the junta learning problem by designing quantum algorithms. Quantum deep learning was studied by Wiebe (2016), and Cheng et al. (2016) provided a framework to analyze learning matrices in the Schatten class. A detailed survey concerning quantum learning theory was presented by Arunachalam and de Wolf (2017), and Arunachalam and de Wolf (2018) further showed that classical and quantum sample complexities are equal up to constant factors for every concept class.

Now we simply recall the development of quantum machine learning (QML). Harrow et al. (2009) proposed a quantum algorithm for solving systems of linear equations, which may be thought of the start of studying quantum machine learning. Then, Wiebe (2012) proposed a quantum linear regression algorithm by virtue of HHL algorithm. Lloyd et al. (2014) proposed a quantum version of principal component analysis dimension reduction algorithm. Also, quantum matrix inversion was employed in a supervised discriminative learning algorithm (Rebentrost et al., 2014). Schuld et al. (2015) presented a comprehensive perspective on quantum machine learning, and Cong and Duan (2016) proposed a quantum data dimension reduction algorithm. Biamonte et al. (2017) focused on utilizing a quantum computer to analyze classical or quantum data encoded as quantum states, and Kerenidis and Prakash (2017) proposed a quantum

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

algorithm for recommendation systems. Lloyd and Weedbrook (2018) changed a classical generative adversarial network to obtain a quantum generative adversarial network, and Mitarai et al. (2018) constructed a quantum neural network model. Zhao et al. (2019) designed quantum Bayesian neural networks, and Benedetti (2019) provided an overview of these models' components and investigated their application. Recently, an application of machine learning techniques to quantum devices was found by Marquardt (2021). Huang et al. (2022) confirmed that QML can more effectively learn the operating rules of the physical world than any classical machine learning method, and then, prediction of the evolution of quantum systems has been achieved successfully in Rodriguez et al. (2022). More recently, Meyer et al. (2023) explored how symmetries in learning problems can be exploited to create quantum learning models, and Krenn et al. (2023) discussed the application of machine learning and artificial intelligence in analyzing quantum measurements.

However, *learning QFA* is still a pending problem to be studied, and this is the main goal of this paper. QFA can be thought of as a theoretical model of quantum computers in which the memory is finite and described by a finite-dimensional state space (Ambainis and Yakaryilmaz, 2021; Bhatia and Kumar, 2019; Gruska, 1999; Qiu et al., 2012). An excellent and comprehensive survey on QFA was presented by Ambainis and Yakaryilmaz (2021). Moreover, QFA have been studied significantly in physical experiment (Mereghetti et al., 2020; Plachta et al., 2022; Tian, 2019).

One-way QFA (1QFA) were firstly proposed and studied by Moore and Crutchfield (2000), Kondacs and Watrous (1997), and then Ambainis and Freivalds (1998), Brodsky and Pippenger (2002), and other authors (e.g., the references in Ambainis and Yakaryilmaz (2021), Qiu et al. (2012), and Bhatia and Kumar (2019)), where "1" means "one-way," that is, the tape-head moves only from the left side to the right side. The decision problems regarding the equivalence of 1QFA and the minimization of states of 1QFA have been studied in Qiu et al. (2012), Mateus (2012), Qiu et al. (2011), and Li and Qiu (2006, 2008).

More specifically, *measure-once* one-way QFA (MO-1QFA) were initiated by Moore and Crutchfield (2000) and *measure-many* one-way QFA (MM-1QFA) were studied first by Kondacs and Watrous (1997). In MO-1QFA, there is only a measurement for computing each input string, performing after reading the last symbol; in contrast, in MM-1QFA, measurement is performed after reading each symbol, instead of only the last symbol. Then other 1QFA were also proposed and studied by Ambainis *et al.*, Nayak, Hirvensalo, Yakaryilmaz and Say, Paschen, Ciamarra, Bertoni *et al.*, Qiu and Mateus *et al.* as well other authors (e.g., the references in Ambainis and Yakaryilmaz (2021)), including: Latvian QFA (Ambainis et al., 2006), QFA with control language (Bertoni et al., 2003), 1QFA with ancilla qubits (1QFA-A) (Paschen, 2000), one-way quantum finite automata together with classical states (1QFAC) (Qiu et al., 2015), and other 1QFA such as Nayak-1QFA (Na-1QFA), General-1QFA (G-1QFA), and fully 1QFA (Ci-1QFA), where G-1QFA, 1QFA-A, Ci-1QFA, 1QFA-CL, and 1QFAC can recognize all regular languages with bounded error. For more details, we can refer to Ambainis and Yakaryilmaz (2021), Qiu et al. (2012), and Bhatia and Kumar (2019).

MO-1QFA have advantages over crisp finite automata in state complexity for recognizing some languages (Bhatia and Kumar, 2019; Qiu et al., 2012). Mereghetti et al. (2020) realized an MO-1QFA with optic implementation and the state complexity of this MO-1QFA has exponential advantages over DFA and NFA as well as PFA (Paz, 1971). MM-1QFA have stronger computing power than MO-1QFA (Brodsky and Pippenger, 2002), but both MO-1QFA and MM-1QFA accept with bounded error only proper subsets of regular languages. Indeed, Brodsky and Pippenger (2002) proved that the languages accepted by MO-1QFA with bounded error are exactly *reversible languages* that are accepted by RFA. RFA have three different definitions and were named as group automata, BM-reversible automata, and AF-reversible automata (see Qiu (2007)), respectively. In particular, these three definitions were proved to be equivalent in Qiu (2007).

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

The remainder of this paper is organized as follows. In Section 2, in the interest of readability, we first introduce basics in quantum computing, then one-way QFA are recalled and we focus on reviewing MO-1QFA and MM-1QFA. The main contributions are in Sections 3 and 4. In Section 3, we first show the appropriate oracle to be used, that is, ***AA*** is not strong enough for learning RFA and MO-1QFA with polynomial time, and a more powerful oracle (named as ***AD*** oracle) is thus employed. With ***AD*** oracle, we design an algorithm for learning MO-1QFA with polynomial time, and the correctness and complexity of algorithm are proved and analyzed in detail. Afterwards, in Section 4 we continue to design an algorithm for learning MM-1QFA with polynomial time. Finally, the main results are summarized in Section 5, and further problems are mentioned for studying.

2. Preliminaries on Quantum Computing and QFA

For the sake of readability, in this section we outline basic notations and principles in quantum computing and review the definitions of MO-1QFA, MM-1QFA, and RFA. For more details, we can refer to Nielsen and Chuang (2000) and Qiu et al. (2012), Say and Yakaryılmaz (2014), Ambainis and Yakaryilmaz (2021), and Bhatia and Kumar (2019).

2.1 Basics in quantum computing

Let $\mathbb{C}$ denote the set of all complex numbers, $\mathbb{R}$ the set of all real numbers, and $\mathbb{C}^{n\times m}$ the set of $n \times m$ matrices having entries in $\mathbb{C}$. Given two matrices $A \in \mathbb{C}^{n\times m}$ and $B \in \mathbb{C}^{p\times q}$, their *tensor product* is the $np \times mq$ matrix, defined as

$$A \otimes B = \begin{bmatrix} A_{11}B & \dots & A_{1m}B \\ \vdots & \ddots & \vdots \\ A_{n1}B & \dots & A_{nm}B \end{bmatrix}.$$

$(A \otimes B)(C \otimes D) = AC \otimes BD$ holds if the multiplication of matrices is satisfied.

If $MM^\dagger = M^\dagger M = I$, then matrix $M \in \mathbb{C}^{n\times n}$ is *unitary*, where † denotes conjugate-transpose operation. M is said to be *Hermitian* if $M = M^\dagger$. For n-dimensional row vector $x = (x_1, \dots, x_n)$, its norm $||x||$ is defined as $||x|| = \left(\sum_{i=1}^{n} x_i x_i^*\right)^{\frac{1}{2}}$, where symbol $*$ denotes conjugate operation. Unitary operations preserve the norm, i.e., $||xM|| = ||x||$ for each $x \in \mathbb{C}^{1\times n}$ and any unitary matrix $M \in \mathbb{C}^{n\times n}$.

According to the basic principles of quantum mechanics (Nielsen and Chuang, 2000), a state of quantum system can be described by a unit vector in a Hilbert space. More specifically, let $B = \{q_1, \dots, q_n\}$ associated with a quantum system denote a basic state set, where every basic state q_i can be represented by an n-dimensional row vector $\langle q_i| = (0 \dots 1 \dots 0)$ having only 1 at the ith entry (where $\langle\cdot|$ is Dirac notation, i.e., bra-ket notation). At any time, the state of this system is a *superposition* of these basic states and can be represented by a row vector $\langle\phi| = \sum_{i=1}^{n} c_i\langle q_i|$ with $c_i \in \mathbb{C}$ and $\sum_{i=1}^{n} |c_i|^2 = 1$; $|\phi\rangle$ represents the conjugate-transpose of $\langle\phi|$. So, the quantum system is described by Hilbert space $\mathcal{H}_Q$ spanned by the base $\{|q_i\rangle : i = 1, 2, \dots, n\}$, i.e. $\mathcal{H}_Q = span\{|q_i\rangle : i = 1, 2, \dots, n\}$.

The state evolution of quantum system complies with unitarity. Suppose the current state of system is $|\phi\rangle$. If it is acted on by some unitary matrix (or unitary operator) M_1, then $|\phi\rangle$ is changed to the new current state $M_1|\phi\rangle$; if the second unitary matrix, say M_2, is acted on $M_1|\phi\rangle$, then $M_1|\phi\rangle$ is changed to $M_2M_1|\phi\rangle$. So, after a series of unitary matrices $M_1, M_2, \dots, M_k$ are performed in sequence, the system's state becomes $M_kM_{k-1} \cdots M_1|\phi\rangle$.

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

To get some information from the quantum system, we need to make a measurement on its current state. Here we consider *projective measurement* (i.e. von Neumann measurement). A projective measurement is described by an *observable* that is a Hermitian matrix $\mathcal{O} = c_1P_1 + \cdots + c_sP_s$, where c_i is its eigenvalue and, P_i is the projector onto the eigenspace corresponding to c_i. In this case, the projective measurement of $\mathcal{O}$ has result set $\{c_i\}$ and projector set $\{P_i\}$. For example, given state $|\psi\rangle$ is made by the measurement $\mathcal{O}$, then the probability of obtaining result c_i is $\|P_i|\psi\rangle\|^2$ and the state $|\psi\rangle$ collapses to $\frac{P_i|\psi\rangle}{\|P_i|\psi\rangle\|}$.

2.2 Review of one-way QFA and RFA

For non-empty set Σ, by Σ^* we mean the set of all finite length strings over Σ, and Σ^n denotes the set of all strings over Σ with length n. For $u \in \Sigma^*$, $|u|$ is the length of u; for example, if $u = x_1x_2 \ldots x_m \in \Sigma^*$ where $x_i \in \Sigma$, then $|u| = m$. For set S, $|S|$ denotes the cardinality of S.

2.2.1 MO-1QFA

We recall the definition of MO-1QFA. An MO-1QFA with n states and input alphabet Σ is a five-tuple

$$\mathcal{M} = (Q, |\psi_0\rangle, \{U(\sigma)\}_{\sigma\in\Sigma}, Q_a, Q_r) \tag{1}$$

where

- $Q = \{|q_1\rangle, \ldots, |q_n\rangle\}$ consist of an orthonormal base that spans a Hilbert space $\mathcal{H}_Q$ ($|q_i\rangle$ is identified with a column vector with the ith entry 1 and the others 0); at any time, the state of $\mathcal{M}$ is a superposition of these basic states;
- $|\psi_0\rangle \in \mathcal{H}$ is the initial state;
- for any $\sigma \in \Sigma$, $U(\sigma) \in \mathbb{C}^{n\times n}$ is a unitary matrix;
- $Q_a, Q_r \subseteq Q$ with $Q_a \cup Q_r = Q$ and $Q_a \cap Q_r = \emptyset$ are the accepting and rejecting states, respectively, and it describes an observable by using the projectors $P(a) = \sum_{|q_i\rangle\in Q_a} |q_i\rangle\langle q_i|$ and $P(r) = \sum_{|q_i\rangle\in Q_r} |q_i\rangle\langle q_i|$, with the result set $\{a, r\}$ of which "a" and "r" denote "accepting" and "rejecting", respectively. Here Q consists of accepting and rejecting sets.

Given an MO-1QFA $\mathcal{M}$ and an input word $s = x_1 \ldots x_n \in \Sigma^*$, then starting from $|\psi_0\rangle$, $U(x_1), \ldots, U(x_n)$ are applied in succession, and at the end of the word, a measurement $\{P(a), P(r)\}$ is performed with the result that $\mathcal{M}$ collapses into accepting states or rejecting states with corresponding probability. Hence, the probability $L_{\mathcal{M}}(x_1 \ldots x_n)$ of $\mathcal{M}$ accepting w is defined as:

$$L_{\mathcal{M}}(x_1 \ldots x_n) = \|P(a)U_s|\psi_0\rangle\|^2 \tag{2}$$

where we denote $U_s = U_{x_n}U_{x_{n-1}} \cdots U_{x_1}$.

2.2.2 RFA

Now we recollect RFA. As mentioned above, there are three equivalent definitions for RFA (Qiu, 2007), that is, group automata, BM-reversible automata, and AF-reversible automata. Here we describe group automata. First we review DFA. A DFA $G = (S, s_0, \Sigma, \delta, S_a)$, where S is a finite state set, $s_0 \in S$ is its initial state, $S_a \subseteq S$ is its accepting state set, Σ is an input alphabet, and δ is a transformation function, i.e., a mapping $\delta : S \times \Sigma \to S$.

An RFA (group automaton) $G = (S, s_0, \Sigma, \delta, S_a)$ is DFA and satisfies that for any $q \in S$ and any $\sigma \in \Sigma$, there is unique $p \in S$ such that $\delta(p, \sigma) = q$.

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

The languages accepted by MO-1QFA with bounded error are exactly the languages accepted by RFA (Brodsky and Pippenger, 2002). In fact, RFA are the special cases of MO-1QFA, and this is showed by the following proposition.

Proposition 1. *(1) For any MO-1QFA* $\mathcal{M} = (Q, |\psi_0\rangle, \{U(\sigma)\}_{\sigma\in\Sigma}, Q_a, Q_r)$ *with* $|\psi_0\rangle \in Q$*, if all entries in* $U(\sigma)$ *for each* $\sigma \in \Sigma$ *are either* 0 *or* 1*, then* $\mathcal{M}$ *is actually a group automaton. (2) If* $G = (S, s_0, \Sigma, \delta, S_a)$ *is a group automaton, then G is actually an MO-1QFA.*

Proof. (1) Suppose the base states $Q = \{|q_i\rangle : i = 1, 2, \ldots, n\}$, where $|q_i\rangle$ is an n-dimensional column vector with the ith entry 1 and the others 0. Let $|\psi_0\rangle = |q_{i_0}\rangle$ for some $i_0 \in \{1, 2, \ldots, n\}$. It is clear that $U(\sigma)$ (for each $\sigma \in \Sigma$) is a permutation matrix and therefore $U(\sigma)$ is also a bijective mapping from Q to Q. So, $\mathcal{M}$ is a group automaton.

(2) If $G = (S, s_0, \Sigma, \delta, S_a)$ is a group automaton with $|S| = n$, then we denote $S = \{q_1, q_2, \ldots, q_n\}$ and s_0 is some $q_i \in S$. According to the definition of group automata, for each $\sigma \in \Sigma$, $\delta(\cdot, \sigma)$ is a bijective mapping from S to S. So, we can identify q_i with an n-dimensional column vector with the ith entry 1 and the others 0. Then for each $\sigma \in \Sigma$, $\delta(\cdot, \sigma)$ induces a unitary matrix $U(\sigma)$ acting on the n-dimensional Hilbert space spanned by the base states S. Finally, $S_a \subseteq S$ and $S_r = S \setminus S_a$ are accepting and rejecting sets of states, respectively. As a result, G is actually equivalent to an MO-1QFA.

□

2.2.3 MM-1QFA

We review the definition of MM-1QFA. Formally, given an input alphabet Σ and an endmaker $\$ \notin \Sigma$, an MM-QFA with n states over the *working alphabet* $\Gamma = \Sigma \cup \{\$\}$ is a six-tuple $\mathcal{M} = (Q, |\psi_0\rangle, \{U(\sigma)\}_{\sigma\in\Sigma}, Q_a, Q_r, Q_g)$, where

- Q, $|\psi_0\rangle$, and $U(\sigma)$ ($\sigma \in \Gamma$) are defined as in the case of MO-1QFA; Q_a, Q_r, Q_g are disjoint to each other and represent the accepting, rejecting, and going states, respectively.
- The measurement is described by the projectors $P(a)$, $P(r)$, and $P(g)$, with the results in $\{a, r, g\}$ of which "a," "r," and "g" denote "accepting," "rejecting," and "going on," respectively.

Any input word w to MM-1QFA is in the form: $w \in \Sigma^*\$$, with symbol \$ denoting the end of a word. Given an input word $x_1 \ldots x_n\$$ where $x_1 \ldots x_n \in \Sigma^n$, MM-1QFA $\mathcal{M}$ performs the following computation:

1. Starting from $|\psi_0\rangle$, $U(x_1)$ is applied, and then we get a new state $|\phi_1\rangle = U(x_1)|\psi_0\rangle$. In succession, a measurement of $\mathcal{O}$ is performed on $|\phi_1\rangle$, and then the measurement result i ($i \in \{a, g, r\}$) is yielded as well as a new state $|\phi_1^i\rangle = \frac{P(i)|\phi_1\rangle}{\sqrt{p_1^i}}$ is obtained, with corresponding probability $p_1^i = ||P(i)|\phi_1\rangle||^2$.
2. In the above step, if $|\phi_1^g\rangle$ is obtained, then starting from $|\phi_1^g\rangle$, $U(x_2)$ is applied and a measurement $\{P(a), P(r), P(g)\}$ is performed. The evolution rule is the same as the above step.
3. The process continues as far as the measurement result "g" is yielded. As soon as the measurement result is "a" ("r"), the computation halts and the input word is accepted (rejected).

Thus, the probability $L_{\mathcal{M}}(x_1 \ldots x_n)$ of $\mathcal{M}$ accepting w is defined as:

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

$$L_{\mathcal{M}}(x_1 \ldots x_n) \tag{3}$$

$$= \sum_{k=1}^{n+1} ||P(a)U(x_k) \prod_{i=1}^{k-1} \big(P(g)U(x_i)\big)|\psi_0\rangle||^2, \tag{4}$$

or equivalently,

$$L_{\mathcal{M}}(x_1 \ldots x_n) \tag{5}$$

$$= \sum_{k=0}^{n} ||P(a)U(x_{k+1}) \prod_{i=1}^{k} \big(P(g)U(x_i)\big)|\psi_0\rangle||^2, \tag{6}$$

where, for simplicity, we can denote \$ by x_{n+1} if no confusion results.

3. Learning MO-1QFA

First we recall a definition concerning model learning with an oracle in polynomial time.

Definition 1. (Tzeng, 1992) Let $\mathbf{R}$ be a class to be learned and $O_{\mathbf{R}}$ be an oracle for $\mathbf{R}$. Then $\mathbf{R}$ is said to be polynomially learnable using the oracle $O_{\mathbf{R}}$ if there is a learning algorithm L and a two-variable polynomial p such that for every target $r \in \mathbf{R}$ of size n to be learned, L runs in time $p(n, m)$ at any point and outputs a hypothesis that is equivalent to r, where m is the maximum length of data returned by $O_{\mathbf{R}}$ so far in the run.

In order to learn a model with polynomial time via an oracle, we hope this oracle is as weaker as possible. For learning MO-1QFA, suppose an oracle can only answer the amplitudes of accepting states for each input string, then can we learn MO-1QFA successfully with polynomial time via such an oracle? We name such an oracle as ***AA*** oracle. For clarifying this point, we try to use ***AA*** oracle to learning DFA. In this case, ***AA*** oracle can answer if it is either an accepting state or a rejecting state for each input string. Equivalently, ***AA*** oracle is exactly MQ for learning DFA as the target model. Therefore, learning DFA via ***AA*** oracle is not polynomial by virtue of the following Angluin's result (Angluin, 1988).

Theorem 1. *(Angluin, 1988) DFA are not polynomially learnable using the MQ oracle only.*

In fact, in 2007 a stronger result was proved in Tîrnăucă and Knuutila (2007) that 0-reversible automata (i.e., a 0-reversible automaton is defined as a RFA with only one accepting state (Angluin, 1982)) are not learnable by using MQ only. That can be described by the following theorem.

Theorem 2. *(Tîrnăucă and Knuutila, 2007) Any RFA with only one accepting state is not learnable by using MQ only.*

Therefore, we have the following proposition.

Proposition 2. *DFA and RFA as well as MO-1QFA are not learnable using **AA** oracle only.*

Proof. Due to the above Theorem 2, we know that any RFA is not learnable by using MQ only. Since RFA are special cases of DFA and MO-1QFA, we obtain that neither DFA nor MO-1QFA is learnable by using MQ only.

For learning DFA and RFA, ***AA*** oracle is exactly equal to MQ oracle, so we conclude that DFA and MO-1QFA are not learnable using ***AA*** oracle only.

□

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

So, we consider a stronger oracle, named as ***AD*** oracle that can answer all amplitudes (instead of the amplitudes of accepting states only) of the superposition state for each input string. For example, for quantum state $|\psi\rangle = \sum_{i=1}^{n} \alpha_i |q_i\rangle$ where $\sum_{i=1}^{n} |\alpha_i|^2 = 1$, ***AA*** oracle can only answer the amplitudes of accepting states in $\{|q_1\rangle, |q_2\rangle, \ldots, |q_n\rangle\}$, but ***AD*** oracle can answer the amplitudes for all states in $\{|q_1\rangle, |q_2\rangle, \ldots, |q_n\rangle\}$. Using ***AD*** oracle, we can prove that MO-1QFA and MM-1QFA are polynomially learnable. Therefore, for learning DFA or RFA, ***AD*** oracle can answer a concrete state for each input string, where the concrete state is the output state of the target automaton to be learned.

First we can easily prove that RFA are linearly learnable via ***AD*** oracle, and this is the following proposition.

Proposition 3. *Let RFA $G = (S, s_0, \Sigma, \delta, S_a)$ be the target to be learned. Then G is linearly learnable via using **AD** oracle with query complexity at most $|S||\Sigma|$.*

Proof. First, ***AD*** oracle can answer the initial state s_0 via inputting empty string. Then by taking s_0 as a vertex, we use pruning algorithm of decision tree to obtain all states in G while accepting states are marked as well. It is easy to know that the query complexity is $O(|S||\Sigma|)$.

□

Our main concern is whether ***AD*** oracle is too strong, that is to say, whether ***AD*** oracle possesses too much information for our learning tasks. To clarify this point partially, we employ a consistency problem that, in a way, demonstrates ***AD*** oracle is really not too strong for our model learning if the time complexity is polynomial. So, we first recall an outcome from Tzeng (1992).

Theorem 3. *(Tzeng, 1992) For any alphabet Σ and finite set $S = \{q_1, q_2, \ldots, q_n\}$, the following problem is NP-complete: Given a set D with $D \subseteq \Sigma^* \times S$, determine whether there is a DFA $G = (S_1, s_0, \Sigma, \delta, S_a)$ such that for any $(x, q) \in D$, $\delta(s_0, x) = q$, where $p \in S_1$ if and only if $(x, p) \in D$ for some $x \in \Sigma^*$.*

Remark 1. In above theorem, each element in D consists of a string in Σ^* and a state in S, so D can be identified with the information carried by ***AD*** oracle in order to learn a DFA. That is to say, even if an ***AD*** oracle holds so much information like D contained in a DFA in this way, it is still not easy (NP-complete) to learn a consistent DFA. To a certain extent, ***AD*** oracle is not too strong to learn a DFA. Since constructing RFA is not easier than constructing DFA and RFA are special MO-1QFA, we use ***AD*** oracle for learning MO-1QFA and MM-1QFA. However, we still do not know what is the weakest oracle to learn MO-1QFA and MM-1QFA with polynomial-time query complexity.

Let $\mathcal{M} = (Q, |\psi_0\rangle, \{U(\sigma)\}_{\sigma\in\Sigma}, Q_a, Q_r)$ be the target MO-1QFA to be learned, where, as the case of learning PFA (Tzeng, 1992), the learner is supposed to have the information of Q, Q_a, Q_r, but the other parameters are to be learned by mean of querying the oracle for achieving an equivalent MO-1QFA (more concretely, for each $\sigma \in \Sigma$, unitary matrix $V(\sigma)$ corresponding to $U(\sigma)$ needs to be determined, but it is possible that $V(\sigma) \neq U(\sigma)$). For any $x \in \Sigma^*$, ***AD***(x) can answer an amplitude distribution that is exactly equivalent to a state of superposition corresponding to the input string x, more exactly, ***AD***(x) can answer the same state as $U(\sigma_k)U(\sigma_{k-1}) \cdots U(\sigma_1)|\psi_0\rangle$ where $x = \sigma_1\sigma_2 \cdots \sigma_k$. From now on, we denote $U(x) = U(\sigma_k)U(\sigma_{k-1}) \cdots U(\sigma_1)$ for $x = \sigma_1\sigma_2 \cdots \sigma_k$.

We outline the basic idea and method for designing the learning algorithm of MO-1QFA $\mathcal{M}$. First, the initial state can be learned from ***AD*** oracle by querying empty string ε. Then by using ***AD*** oracle we continue to search for a base of the Hilbert space spanned by $\{v^* = U(x)|\psi_0\rangle : x \in \Sigma^*\}$. This procedure will be terminated since the dimension of the space is at most $|Q|$. In fact, we can prove this can be finished in polynomial time. Finally, by virtue of the learned base and solving groups of linear equations we can conclude $V(\sigma)$ for each $\sigma \in \Sigma$. We prove these results in detail following the algorithm and now present Algorithm 1 for learning MO-1QFA as follows.

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

Algorithm 1 Algorithm for learning MO-1QFA $\mathcal{M} = (Q, |\psi_0\rangle, \{U(\sigma)\}_{\sigma\in\Sigma}, Q_a, Q_r)$

$|\psi_0^*\rangle \leftarrow \boldsymbol{AD}(\varepsilon)$;
Set $\mathcal{B}$ to be the empty set;
Set $Nod \leftarrow \{node(\varepsilon)\}$;
while Nod is not empty **do**
    begin Take an element $node(x)$ from Nod;
    $v^*(x) \leftarrow \boldsymbol{AD}(x)$;
    if $v^*(x) \notin span(\mathcal{B})$ **then**
        begin Add $node(x\sigma)$ to Nod for all $\sigma \in \Sigma$;
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{v^*(x)\}$ **end**;
    end if
end while
end;
Let $V(\sigma) = [x_{ij}(\sigma)]$ for any $\sigma \in \Sigma$, $1 \le i, j \le n$;
Define a linear system:
    for any $v^*(x) \in \mathcal{B}$ and any $\sigma \in \Sigma$, $V(\sigma)v^*(x) = v^*(x\sigma) = \boldsymbol{AD}(x\sigma)$,
    for $1 \le i_1, i_2 \le n$, if $i_1 \ne i_2$, then $\sum_{j=1}^{n} x_{i_1 j}(\sigma)\overline{x_{i_2 j}(\sigma)} = 0$; otherwise, it is 1,
Find a suitable solution for $x_{ij}(\sigma)$'s,
if there is a solution **then**
    return $\mathcal{M}^* = (Q, |\psi_0^*\rangle, \{V(\sigma)\}_{\sigma\in\Sigma}, Q_a, Q_r)$
else
    return (not exist);
end if

Next we prove the correctness of Algorithm 1 and then analyze its complexity. First we prove that Step 1 to Step 12 in Algorithm 1 can produce a set of vectors $\mathcal{B}$ consisting of a base of space spanned by $\{v^*(x)|x \in \Sigma^*\}$, where $v^*(x) = \boldsymbol{AD}(x)$ is actually the vector replied by oracle $\boldsymbol{AD}$ for input string x, that is, $v^*(x) = \boldsymbol{AD}(x) = U(\sigma_k)U(\sigma_{k-1})\ldots U(\sigma_1)|\psi_0\rangle$, for $x = \sigma_1\sigma_2\ldots\sigma_k$.

Proposition 4. *In Algorithm 1 for learning MO-1QFA, the final set of vectors $\mathcal{B}$ consists of a base of Hilbert space span$\{v^*(x)|x \in \Sigma^*\}$ that is spanned by $\{v^*(x)|x \in \Sigma^*\}$.*

Proof. From the algorithm procedure we can assume that $\mathcal{B} = \{v^*(x_1), v^*(x_2), \ldots, v^*(x_m)\}$ for some m, where it is clear that some x_i equals to ε, and for any $x \in \Sigma^*$, there are x_j and $y \in \Sigma^*$ such that $x = x_j y$. The rest is to show that $v^*(x)$ can be linearly represented by the vectors in $\mathcal{B}$ for any $x \in \Sigma^*$. Let $x = x_j y$ for some x_j and $y \in \Sigma^*$. By induction on the length $|y|$ of y. If $|y| = 0$, i.e., $y = \varepsilon$, then it is clear for $x = x_j$. If $|y| = 1$, then due to the procedure of algorithm, $v^*(x_j y)$ is linearly dependent on $\mathcal{B}$. Suppose that it holds for $|y| = k \ge 0$. Then we need to verify it holds for $|y| = k + 1$. Denote $y = z\sigma$ with $|z| = k$. Then with induction hypothesis we have $v^*(x_j z) = \sum_k c_k v^*(x_k)$. Therefore, we have

$$\begin{aligned} v^*(x) &= v^*(x_j z\sigma) \\ &= U(\sigma)v^*(x_j z) \\ &= U(\sigma)\sum_k c_k v^*(x_k) \\ &= \sum_k c_k v^*(x_k\sigma). \end{aligned} \tag{7}$$

Since $v^*(x_k\sigma)$ is linearly dependent on $\mathcal{B}$ for $k = 1, 2, \ldots, m$, the proof is completed.

□

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

The purpose of Algorithm 1 is to learn the target MO-1QFA $\mathcal{M} = (Q, |\psi_0\rangle, \{U(\sigma)\}_{\sigma\in\Sigma}, Q_a, Q_r)$, so we need to verify $\mathcal{M}^* = (Q, |\psi_0^*\rangle, \{V(\sigma)\}_{\sigma\in\Sigma}, Q_a, Q_r)$ obtained is equivalent to $\mathcal{M}$. For this, it suffices to check $V(x)|\psi_0^*\rangle = U(x)|\psi_0\rangle$ for any $x \in \Sigma^*$, where $V(x) = V(\sigma_s)V(\sigma_{s-1})\ldots V(\sigma_1)$ and $U(x) = U(\sigma_s)U(\sigma_{s-1})\ldots U(\sigma_1)$ for $x = \sigma_1\sigma_2\ldots\sigma_s$.

Theorem 4. *In Algorithm 1 for learning MO-1QFA, for any* $x \in \Sigma^*$,

$$V(x)|\psi_0^*\rangle = U(x)|\psi_0\rangle. \tag{8}$$

Proof. For $x = \varepsilon$, $V(\varepsilon) = U(\varepsilon) = I$, and $|\psi_0^*\rangle = \boldsymbol{AD}(\varepsilon) = |\psi_0\rangle$, so it holds.

For any $\sigma \in \Sigma$ and for any $v^*(x) \in \mathcal{B}$, according to Algorithm 1, we have $V(\sigma)v^*(x) = v^*(x\sigma) = \boldsymbol{AD}(x\sigma)$. In particular, taking $x = \varepsilon$, then we have $V(\sigma)|\psi_0^*\rangle = v^*(\sigma) = \boldsymbol{AD}(\sigma) = U(x)|\psi_0\rangle$.

Suppose it holds for $|x| = k$. The rest is to prove that it holds for $|x| = k+1$. Denote $y = x\sigma$ where $|x| = k$ and $\sigma \in \Sigma$. Due to Proposition 4, $v^*(x)$ can be linearly represented by $\mathcal{B} = \{v^*(x_1), v^*(x_2), \ldots, v^*(x_m)\}$, i.e., $v^*(x) = \sum_k c_k v^*(x_k)$ for some $c_k \in \mathbb{C}$. With the induction hypothesis, $V(x)|\psi_0^*\rangle = U(x)|\psi_0\rangle$ holds. Then by means of Algorithm 1, we have

$$\begin{aligned} V(y)|\psi_0^*\rangle &= V(x\sigma)|\psi_0^*\rangle \\ &= V(\sigma)V(x)|\psi_0^*\rangle \\ &= V(\sigma)U(x)|\psi_0\rangle \\ &= V(\sigma)v^*(x) \\ &= V(\sigma)\sum_k c_k v^*(x_k) \\ &= \sum_k c_k V(\sigma)v^*(x_k) \\ &= \sum_k c_k v^*(x_k\sigma). \end{aligned} \tag{9}$$

On the other hand, since $v^*(z) = \boldsymbol{AD}(z) = U(z)|\psi_0\rangle$ for any $z \in \Sigma^*$, we have

$$\begin{aligned} \sum_k c_k v^*(x_k\sigma) &= \sum_k c_k U(x_k\sigma)|\psi_0\rangle \\ &= \sum_k c_k U(\sigma)U(x_k)|\psi_0\rangle \\ &= \sum_k c_k U(\sigma)v^*(x_k) \\ &= U(\sigma)\sum_k c_k v^*(x_k) \\ &= U(\sigma)v^*(x) \\ &= U(x\sigma)|\psi_0\rangle \\ &= U(y)|\psi_0\rangle. \end{aligned} \tag{10}$$

So, the proof is completed.

□

From Theorem 4, it follows that Algorithm 1 returns an equivalent MO-1QFA to the target MO-1QFA $\mathcal{M} = (Q, |\psi_0\rangle, \{U(\sigma)\}_{\sigma\in\Sigma}, Q_a, Q_r)$ to be learned. Next we analyze the computational complexity of Algorithm 1.

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

Proposition 5. *Let the target MO-1QFA to be learned have n's bases states. The the computational complexity of Algorithm 1 is* $O(n^5|\Sigma|)$.

Proof. We consider it from two parts.

(I) The first part of Algorithm 1 to get $\mathcal{B}$: The complexity to determine the linear independence of some n-dimensional vectors is $O(n^3)$ (Faddeev and Faddeev, 1963), and there are at most n time to check this, so the first part of Algorithm 1 to get $\mathcal{B}$ needs time $O(n^4)$.

(II) The second part of finding the feasible solutions for $V(\sigma)$ for each $\sigma \in \Sigma$: For any $\sigma \in \Sigma$, Step 15 defines $|\mathcal{B}|$'s matrix equations and these equations are clearly equivalent to a group of linear equations, but are subject to the restriction conditions in Step 16. So, this part is a problem of linear programming and we can refer to Boyd and Vandenberghe (2004) and Karmarkar (1984) to get the time complexity is $O(n^5|\Sigma|)$.

Therefore, by combining (I) and (II) we have the complexity of Algorithm 1 is $O(n^5|\Sigma|)$. □

To illustrate Algorithm 1 for learning MO-1QFA, we give an example as follows.

Example 1. Suppose that $\mathcal{M} = (Q, |\psi_0\rangle, \{U(\sigma)\}_{\sigma\in\Sigma}, Q_a, Q_r)$ is an MO-1QFA to be learned by Algorithm 1, where $Q = \{q_0, q_1\}$, $Q_a = \{q_1\}$, $Q_r = \{q_0\}$, $\Sigma = \{a\}$, $U(a) = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$, states q_0 and q_1 correspond to the two quantum basis states $|q_0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|q_1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, and $|\psi_0\rangle = |q_0\rangle$. Denote $\mathcal{M}^* = (Q, |\psi_0\rangle^*, \{V(\sigma)\}_{\sigma\in\Sigma}, Q_a, Q_r)$ as the MO-1QFA learned from Algorithm 1, and the procedure for obtaining $\mathcal{M}^*$ from Algorithm 1 is given below.

Step 1 of Algorithm 1 yields

$$\begin{aligned} |\psi_0\rangle^* =& \mathbf{AD}(\epsilon) \\ =& U(\epsilon)|\psi_0\rangle \\ =& I|\psi_0\rangle \\ =& |q_0\rangle. \end{aligned} \tag{11}$$

The 1st iteration run of the while loop body in Algorithm 1 is given below, with the computation of each set.

Step 6 of Algorithm 1 yields

$$\begin{aligned} v^*(\epsilon) =& \mathbf{AD}(\epsilon) \\ =& U(\epsilon)|\psi_0\rangle \\ =& I|\psi_0\rangle \\ =& |q_0\rangle. \end{aligned} \tag{12}$$

Step 8 of Algorithm 1 yields

$$Nod = \{node(a)\}. \tag{13}$$

Step 9 of Algorithm 1 yields

$$\mathcal{B} = \{|q_0\rangle\}. \tag{14}$$

The 2nd iteration run of the while loop body in Algorithm 1 is given below, with the computation of each set.

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

Step 6 of Algorithm 1 yields

$$\begin{aligned} v^*(a) =& \mathbf{AD}(a) \\ =& U(a)|\psi_0\rangle \\ =& U(a)|q_0\rangle \\ =& \frac{|q_0\rangle + |q_1\rangle}{\sqrt{2}}. \end{aligned} \tag{15}$$

Step 8 of Algorithm 1 yields

$$Nod = \{node(aa)\}. \tag{16}$$

Step 9 of Algorithm 1 yields

$$\mathcal{B} = \left\{ |q_0\rangle, \frac{|q_0\rangle + |q_1\rangle}{\sqrt{2}} \right\}. \tag{17}$$

The 3rd iteration run of the while loop body in Algorithm 1 is given below, with the computation of each set.

Step 6 of Algorithm 1 yields

$$\begin{aligned} v^*(aa) =& \mathbf{AD}(aa) \\ =& U(aa)|\psi_0\rangle \\ =& U(a)U(a)|q_0\rangle \\ =& |q_0\rangle. \end{aligned} \tag{18}$$

Since $v^*(x)$ belongs to $span(\mathcal{B})$, the statements in the branch statement are not executed at this point. The set Nod is the empty set at this point, so Algorithm 1 exits from the while loop body.

Finally, let $V(a) = \begin{bmatrix} x_{11}(a) & x_{12}(a) \\ x_{21}(a) & x_{22}(a) \end{bmatrix}$, and according to steps 15 and 16 of Algorithm 1, we get

$$\begin{aligned} V(a)v^*(\epsilon) =& V(a)|q_0\rangle \\ =& v^*(a) \\ =& \mathbf{AD}(a) \\ =& \frac{|q_0\rangle + |q_1\rangle}{\sqrt{2}}. \end{aligned} \tag{19}$$

$$\begin{aligned} V(a)v^*(a) =& V(a)\left(\frac{|q_0\rangle + |q_1\rangle}{\sqrt{2}}\right) \\ =& v^*(aa) \\ =& \mathbf{AD}(aa) \\ =& |q_0\rangle. \end{aligned} \tag{20}$$

From Eqs. (19) and (20), the following system of equations is obtained

$$\begin{cases} \begin{bmatrix} x_{11}(a) & x_{12}(a) \\ x_{21}(a) & x_{22}(a) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \\ \begin{bmatrix} x_{11}(a) & x_{12}(a) \\ x_{21}(a) & x_{22}(a) \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \end{cases} \tag{21}$$

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

Solving the system of Eq. (21) gives

$$V(a) = \begin{bmatrix} x_{11}(a) & x_{12}(a) \\ x_{21}(a) & x_{22}(a) \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}. \tag{22}$$

As a result, we can obtain $\mathcal{M}^* = (Q, |\psi_0\rangle^*, \{V(\sigma)\}_{\sigma\in\Sigma}, Q_a, Q_r)$, where $Q = \{q_0, q_1\}$, $|\psi_0\rangle^* = |q_0\rangle$, $Q_a = \{q_1\}$, $Q_r = \{q_0\}$, $\Sigma = \{a\}$, and $V(a) = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$.

Therefore, the MO-1QFA $\mathcal{M}^*$, which is equivalent to MO-1QFA $\mathcal{M}$, can be obtained from Algorithm 1.

4. Learning MM-1QFA

In this section, we study learning MM-1QFA via ***AD*** oracle. Let $\mathcal{M} = (Q, |\psi_0\rangle, \{U(\sigma)\}_{\sigma\in\Gamma}, Q_a, Q_r, Q_g)$ be the target QFA to be learned, where $\Gamma = \Sigma \cup \{\$\}$, and $\$ \notin \Sigma$ is an end-maker. As usual, Q, Q_a, Q_r, Q_g are supposed to be known, and the goal is to achieve unitary matrices $V(\sigma)$ for each $\sigma \in \Gamma$ in order to get an equivalent MM-1QFA $\mathcal{M}^* = (Q, |\psi_0^*\rangle, \{V(\sigma)\}_{\sigma\in\Gamma}, Q_a, Q_r, Q_g)$. ***AD*** oracle can answer an amplitude distribution $\boldsymbol{AD}(x)$ for any $x \in \Gamma^*$. MM-1QFA performs measuring after reading each input symbol, and then only the non-halting (i.e. going on) states continue to implement computing for next step, and the amplitude distribution for the superposition state after performing each unitary matrix needs to be learned from oracle.

Therefore, for any $x = \sigma_1\sigma_2\ldots\sigma_k \in \Gamma^*$, since MM-1QFA $\mathcal{M}$ outputs the following state (unnormalized form) as the current state:

$$U(\sigma_k)P_nU(\sigma_{k-1})P_n\ldots U(\sigma_1)P_n|\psi_0\rangle, \tag{23}$$

we require ***AD*** oracle can answer $\boldsymbol{AD}(x) = U(\sigma_k)P_nU(\sigma_{k-1})P_n\ldots U(\sigma_1)P_n|\psi_0\rangle$. In particular, $\boldsymbol{AD}(\varepsilon) = |\psi_0\rangle$.

Before presenting the algorithm of learning MM-1QFA, we describe the main ideas and procedure.

First the initial state can be learned from ***AD*** oracle via querying empty string ε.

Then by using ***AD*** oracle we are going to search for a base $\mathcal{B}$ of the Hilbert space spanned by $\{v^*(x) : x \in \Sigma^*\}$ where for any $x = \sigma_1\sigma_2\ldots\sigma_k \in \Sigma^*$,

$$v^*(x) = \boldsymbol{AD}(x) = U(\sigma_k)P_nU(\sigma_{k-1})P_n\ldots U(\sigma_1)P_n|\psi_0\rangle. \tag{24}$$

This procedure will be terminated due to the finite dimension of the space (at most $|Q|$), and this can be completed with polynomial time.

Finally, by combining the base $\mathcal{B}$ and with groups of linear equations we can obtain $V(\sigma)$ for each $\sigma \in \Sigma$. These results can be verified in detail after Algorithm 2, and we now present Algorithm 2 for learning MM-1QFA in the following.

Next we first demonstrate that the algorithm can find out a base $\mathcal{B}$ for Hilbert space $span\{v^*(x)|x \in \Sigma^*\}$.

Proposition 6. *In Algorithm 2 for learning MM-1QFA, the final set of vectors $\mathcal{B}$ consists of a base of Hilbert space $span\{v^*(x)|x \in \Sigma^*\}$, where $v^*(x)$ is actually the vector replied by oracle **AD** for input string x, that is $v^*(x) = \boldsymbol{AD}(x) = U(\sigma_k)P_nU(\sigma_{k-1})P_n\ldots U(\sigma_1)P_n|\psi_0\rangle$, for $x = \sigma_1\sigma_2\ldots\sigma_k \in \Sigma^*$.*

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

Algorithm 2 Algorithm for learning MM-1QFA $\mathcal{M} = (Q, |\psi_0\rangle, \{U(\sigma)\}_{\sigma\in\Gamma}, Q_a, Q_r, Q_g)$

$|\psi_0^*\rangle \leftarrow \boldsymbol{AD}(\varepsilon)$; $v^*(\$) \leftarrow \boldsymbol{AD}(\$)$;
Set $\mathcal{B}$ to be the empty sets;
Set $Nod \leftarrow \{node(\varepsilon)\}$;
while *Nod*s not empty **do**
    begin Take an element $node(x)$ from *Nod*;
    $v^*(x) \leftarrow \boldsymbol{AD}(x)$;
    if $v^*(x) \notin span(\mathcal{B})$ **then**
        begin Add $node(x\sigma)$ to *Nod* for all $\sigma \in \Sigma$;
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{v^*(x)\}$ **end**;
    end if
end while
end;
Let $V(\sigma) = [x_{ij}(\sigma)]$ for any $\sigma \in \Sigma \cup \{\$\}$, $1 \le i, j \le n$;
Define linear systems:
    for any $v^*(x) \in \mathcal{B}$ and any $\sigma \in \Sigma \cup \{\$\}$, $V(\sigma)P_n v^*(x) = v^*(x\sigma) = \boldsymbol{AD}(x\sigma)$;
    for $1 \le i_1, i_2 \le n$, if $i_1 \ne i_2$, then $\sum_{j=1}^{n} x_{i_1 j}(\sigma)\overline{x_{i_2 j}(\sigma)} = 0$; otherwise, it is 1,
Find a suitable solution for $x_{ij}(\sigma)$'s, and denote $\mathbb{V} = \{V(\sigma) : \sigma \in \Sigma \cup \{\$\}\}$;
if there is a solution **then**
    return $\mathcal{M}^* = (Q, |\psi_0^*\rangle, \{V(\sigma)\}_{\sigma\in\Sigma\cup\{\$\}}, Q_a, Q_r, Q_g)$
else
    return (not exist);
end if

Proof. Suppose that $\mathcal{B} = \{v^*(x_1), v^*(x_2), \ldots, v^*(x_m)\}$, where it is clear that some x_i equals to ε. So, for any $x \in \Sigma^*$, there are x_j and $y \in \Sigma^*$ such that $x = x_j y$. It suffices to show that $v^*(x)$ can be linearly represented by the vectors in $\mathcal{B}$. By induction on the length $|y|$ of y. If $|y| = 0$, i.e., $y = \varepsilon$, then it is obvious for $x = x_j$. In addition, for $|y| = 1$, $v^*(x_j y)$ is linearly dependent on $\mathcal{B}$ in terms of the algorithm's operation. Suppose that it holds for $|y| = k \ge 0$. Then we need to verify it holds for $|y| = k + 1$. Denote $y = z\sigma$ with $|z| = k$. Then by induction hypothesis $v^*(x_j z) = \sum_k c_k v^*(x_k)$ for some $c_k \in \mathbb{C}$ with $k = 1, 2, \ldots, m$. Therefore, we have

$$\begin{aligned} v^*(x) &= v^*(x_j z\sigma) \\ &= U(\sigma)P_n v^*(x_j z) \\ &= U(\sigma)P_n \sum_k c_k v^*(x_k) \\ &= \sum_k c_k U(\sigma)P_n v^*(x_k) \\ &= \sum_k c_k v^*(x_k\sigma). \end{aligned} \tag{25}$$

Since $v^*(x_k\sigma)$ is linearly dependent on $\mathcal{B}$ for $k = 1, 2, \ldots, m$, $v^*(x)$ can be linearly represented by the vectors in $\mathcal{B}$ and the proof is completed.

□

Then we need to verify that the MM-1QFA $\mathcal{M}^*$ obtained in Algorithm 2 is equivalent to the target MM-1QFA $\mathcal{M}$. This can be achieved by checking $V(\$)P_n|\psi_0^*\rangle = U(\$)P_n|\psi_0\rangle$ and for any $x = \sigma_1\sigma_2\ldots\sigma_k \in \Sigma^*$,

$$V(\sigma_k)P_n V(\sigma_{k-1})P_n \ldots V(\sigma_1)P_n|\psi_0^*\rangle = U(\sigma_k)P_n U(\sigma_{k-1})P_n \ldots U(\sigma_1)P_n|\psi_0\rangle. \tag{26}$$

So we are going to prove the following theorem.

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

Theorem 5. *In Algorithm 2 for learning MM-1QFA, we have*

$$V(\$)P_n|\psi_0^*\rangle = U(\$)P_n|\psi_0\rangle; \tag{27}$$

and for any $x = \sigma_1\sigma_2\ldots\sigma_k \in \Sigma^*$, *Eq. (26) holds, where* $|x| \geq 1$.

Proof. Note $v^*(\varepsilon) \in \mathcal{B}$, by means of Step 15 in Algorithm 2 and taking $\sigma = \$$, we have

$$V(\$)P_n v^*(\varepsilon) = v^*(\$) = \boldsymbol{AD}(\$) = U(\$)P_n|\psi_0\rangle. \tag{28}$$

Since $v^*(\varepsilon) = \boldsymbol{AD}(\varepsilon) = |\psi_0\rangle$, and from Algorithm 2 we know $\boldsymbol{AD}(\varepsilon) = |\psi_0^*\rangle$, Eq. (27) holds.

Next we prove that Eq. (26) holds for any $x = \sigma_1\sigma_2\ldots\sigma_k \in \Sigma^*$. We do it by induction method on the length of $|x|$.

If $|x| = 1$, say $x = \sigma \in \Sigma$, then with Step 15 in Algorithm 2 and taking $v^*(\varepsilon)$, we have $V(\sigma)P_n v^*(\varepsilon) = v^*(\sigma) = \boldsymbol{AD}(\sigma) = U(\sigma)P_n|\psi_0\rangle$, so, Eq. (26) holds for $|x| = 1$ due to $v^*(\varepsilon) = |\psi_0\rangle$.

Assume that Eq. (26) holds for any $|x| = k \geq 1$. The rest is to prove that Eq. (26) holds for any $|x| = k+1$. Let $x = y\sigma$ with $y = \sigma_1\sigma_2\ldots\sigma_k$. Suppose $v^*(y) = \sum_i c_i v^*(x_i)$ for some $c_k \in \mathbb{C}$. For each i, by means of Step 15 in Algorithm 2, we have

$$V(\sigma)P_n v^*(x_i) = v^*(x_i\sigma) = \boldsymbol{AD}(x_i\sigma) = U(\sigma)P_n\boldsymbol{AD}(x_i), \tag{29}$$

and therefore

$$V(\sigma)P_n \sum_i c_i v^*(x_i) = U(\sigma)P_n \sum_i c_i \boldsymbol{AD}(x_i). \tag{30}$$

Since $v^*(x_i) = \boldsymbol{AD}(x_i)$, we further have

$$V(\sigma)P_n v^*(y) = U(\sigma)P_n v^*(y). \tag{31}$$

By using $v^*(y) = U(\sigma_k)P_n U(\sigma_{k-1})P_n\ldots U(\sigma_1)P_n|\psi_0\rangle$, and the above induction hypothesis (i.e., Eq. (26) holds), we have

$$V(\sigma_{k+1})P_n V(\sigma_k)P_n V(\sigma_{k-1})P_n\ldots V(\sigma_1)P_n|\psi_0^*\rangle = U(\sigma_{k+1})P_n U(\sigma_{k-1})P_n\ldots U(\sigma_1)P_n|\psi_0\rangle. \tag{32}$$

Consequently, the proof is completed. □

To conclude the section, we give the computational complexity of Algorithm 2.

Proposition 7. *Let the target MM-1QFA to be learned have n's bases states. Then the computational complexity of Algorithm 2 is* $O(n^5|\Sigma|)$.

Proof. It is actually similar to the proof of Proposition 5. □

Remark 2. Weighted finite automata (WFA) (e.g., see Balle and Mohri (2015)) are finite automata whose transitions and states are augmented with some weights, elements of a semiring, and a WFA also induces a function over strings. Learning WFA has been significantly studied and the details can be referred to Balle and Mohri (2015) and the references therein. The algorithms for learning WFA are closely related to Hankel matrices. More specifically, for a field S and a finite alphabet Σ, then the rank of the Hankel matrix H_f associated with a function $f : \Sigma^* \to S$ is finite if and only if there exists a WFA $\mathcal{A}$ representing f with rank (H_f) states and no WFA representing f admits fewer states. Though MQ and EQ are used in in the algorithms of learning WFA, the way to induce functions by WFA is different from the definitions of accepting probabilities in QFA, and particularly it is not known whether the Hankel matrices can be used to study QFA (as we are aware, there are no results concerning the Hankel matrices associated with QFA). Therefore, it is still an open problem of whether the algorithms of learning WFA can be used to study learning QFA.

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

5. Concluding Remarks

QFA are simple models of quantum computing with finite memory, but QFA have significant advantages over classical finite automata concerning state complexity (Ambainis and Yakaryilmaz, 2021; Bhatia and Kumar, 2019; Say and Yakaryılmaz, 2014), and QFA can be realized physically to a considerable extent (Mereghetti et al., 2020). As a new topic in quantum learning theory and quantum machine learning, learning QFA via queries has been studied in this paper. As classical model learning (Vaandrager, 2017), we can term it as *quantum model learning*.

The main results we have obtained are that we have proposed two polynomial-query learning algorithms for measure-once one-way QFA (MO-1QFA) and measure-many one-way QFA (MM-1QFA), respectively. The oracle to be used is an ***AD*** oracle that can answer an amplitude distribution, and we have analyzed that a weaker oracle being only able to answer accepting or rejecting for any inputting string may be not enough for learning QFA with polynomial time.

Here a question is how to compare ***AD*** oracle with ***MQ*** oracle and ***EQ*** oracle? In general, ***MQ*** oracle and ***EQ*** oracle are together used in classical models learning with deterministic or nondeterministic transformation of states; ***AD*** oracle can return a superposition state for an input string in QFA, as ***SD*** oracle in Tzeng (1992) can return a distribution of state for an input string in PFA, so for learning DFA, ***AD*** oracle and ***SD*** oracle can return a state for each input string, not only accepting state as ***MQ*** oracle can do. Of course, the problem of whether both ***MQ*** oracle and ***EQ*** oracle together can be used to study QFA learning is still not clear. Furthermore, if ***AD*** oracle can return the weight of general weighted automata for each input string, then the problem of whether ***AD*** oracle can be used to study general weighted automata learning is worthy of consideration carefully.

However, we still do not know whether there is a weaker oracle $\mathcal{Q}$ than ***AD*** oracle but by using $\mathcal{Q}$ one can learn MO-1QFA or MM-1QFA with polynomial time. Furthermore, what is the weakest oracle to learn MO-1QFA and MM-1QFA with polynomial query complexity? These are interesting and challenging problems to be solved. Of course, for learning RFA, similar to learning DFA (Angluin, 1987), we can get an algorithm of polynomial time by using MQ together with EQ.

Another interesting problem is how to realize these query oracles physically, including ***SD*** oracle, ***AD*** oracle, and even ***MQ*** as well as ***EQ***. In quantum query algorithms, for any given Boolean function f, it is supposed that a quantum query operator called an oracle, denoted by O_f, can output the value of $f(x)$ with any input x. As for the construction of quantum circuits for O_f, there are two cases: (1) If a Boolean function f is in the form of disjunctive normal form and suppose that the truth table of f is known, then an algorithm for constructing quantum circuit to realize O_f was proposed in Avron et al. (2021). However, this method relies on the truth table of the function, which means that it is difficult to apply, since the truth table of the function is likely not known in practice. (2) If a Boolean function f is a conjunctive normal form, then a polynomial-time algorithm was designed in (Qiu et al., 2022) for constructing a quantum circuit to realize O_f, without any further condition on f.

As mentioned above, besides MO-1QFA and MM-1QFA, there are other one-way QFA, including Latvian QFA (Ambainis et al., 2006), QFA with control language (Bertoni et al., 2003), 1QFA with ancilla qubits (1QFA-A) (Paschen, 2000), one-way quantum finite automata together with classical states (1QFAC) (Qiu et al., 2015), and other 1QFA such as Nayak-1QFA (Na-1QFA), General-1QFA (G-1QFA), and fully 1QFA (Ci-1QFA). So, one of the further problems worthy of consideration is to investigate learning these QFA via queries.

Finally, we would like to analyze partial possible methods for considering these problems. In the present paper, we have used ***AD*** oracle as queries and quantum algorithms for determining the equivalence between 1QFA to be learned for learning both MO-1QFA and MM-1QFA. So, an algorithm for determining the equivalence between 1QFA to be learned is necessary in our method. In general, as we studied in Li and Qiu (2008), for designing an algorithm to determine the equivalence between 1QFA, we first transfer the 1QFA to a classical linear mathematical model, and then

https://doi.org/10.1017/S0960129523000373 Published online by Cambridge University Press

obtain the result by using the known algorithm for determining the equivalence between classical linear mathematical models. As pointed out in Ambainis and Yakaryilmaz (2021), the equivalence between 1QFA can be determined, though the equivalence problems for some of 1QFA still have not been studied carefully. Of course, some 1QFA also involve more parameters to be learned, for example, 1QFAC have classical states to be determined.

Acknowledgements. The authors are grateful to the two anonymous referees for invaluable suggestions and comments that greatly helped us improve the quality of this paper. This work was supported in part by the National Natural Science Foundation of China (Nos. 61876195, 61572532) and the Natural Science Foundation of Guangdong Province of China (No. 2017B030311011).

References

Aaronson, S. (2007). The learnability of quantum states. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **463** (2088) 3089–3114.

Angluin, D. (1982). Queries and concept learning. *Journal of the ACM* **29** (3) 741–765.

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation* **75** (2) 87–106.

Angluin, D. (1988). Queries and concept learning. *Machine Learning* **2** (4) 319–342.

Ambainis, A., Beaudry, M., Golovkins, M., Kikusts, A., Mercer, M. and Therien, D. (2006). Algebraic results on quantum automata. *Theory of Computing Systems* **39** (1) 165–188.

Avron, J., Casper, O. and Rozen, I. (2021). Quantum advantage and noise reduction in distribute quantum computing. *Physical Review A* **104** 052404.

Angluin, D., Eisenstat, S. and Fisman, D. 2015. Learning regular languages via alternating automata. In: *Proceeding of the 24th International Joint Conference on Artificial Intelligence, IJCAI 2015*, AAAI Press, 3308–3314.

Ambainis, A. and Freivalds, R. (1998). One-way quantum finite automata: Strengths, weaknesses and generalizations. In: *Proceeding of the 39th Foundations of Computer Science, FOCS 1998*, IEEE Computer Society Press, 332–341.

Arunachalam, S. and de Wolf, R. (2017). A survey of quantum learning theory. *ACM SIGACT News* **48** (2) 41–67. Also, arXiv:1701.06806v3.

Arunachalam, S. and de Wolf, R. (2018). Optimal quantum sample complexity of learning algorithms. *Journal of Machine Learning Research* **19** 1–36.

Ambainis, A. and Yakaryilmaz, A. (2021). Automata and quantum computing. In: J.-E. Pin (ed.) *Handbook of Automata Theory (II)*, 1457–1493. arXiv:1507.01988.

Belovs, A. (2015). Quantum algorithms for learning symmetric juntas via the adversary bound. *Computational Complexity* **24** (2) 255–293.

Bollig, B., Habermehl, P., Kern, C. and Leucker, M. (2009). Angluin-style learning of NFA. In: *Proceeding of the 21st International Joint Conference on Artificial Intelligence, IJCAI-09*, AAAI Press, 1004–1009.

Bshouty, N. H. and Jackson, J. C. (1999). Learning DNF over the uniform distribution using a quantum example oracle. *SIAM Journal on Computing* **28** (3) 1136–1153. Earlier version in COLT'95.

Bhatia, A. S. and Kumar, A. (2019). *Quantum Finite Automata: Survey, Status and Research Directions*, arXiv:1901.07992v1.

Berndt, S., Liśkiewicz, M., Lutter, M. and Reischuk, R. (2017). Learning residual alternating automata. In: *Proceeding of the 31st AAAI Conference on Artificial Intelligence, AAAI-17*, AAAI Press, 1749–1755.

Benedetti, M., Lloyd, E., Sack, S. and Fiorentini, M. (2019). Parameterized quantum circuits as machine learning models. *Quantum Science and Technology* **4** (4), 043001.

Balle, B. and Mohri, M. (2015). Learning weighted automata. In: A. Maletti (ed.) *Algebraic Informatics. CAI 2015*, LNCS, vol. 9270, Springer, 1–21.

Bertoni, A., Mereghetti, C. and Palano, B. (2003). Quantum computing: 1-way quantum automata. In: *Proceeding of the 7th International Conference on Developments in Language Theory, DLT 2003*, Springer, 1–20.

Brodsky, A. and Pippenger, N. (2002). Characterizations of 1-way quantum finite automata. *SIAM Journal on Computing* **31** (5) 1456–1478.

Bergadano, F. and Varricchio, S. (1996). Learning behaviors of automata from multiplicity and equivalence queries. *SIAM Journal on Computing* **25** (6) 1268–1280.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*, Cambridge University Press.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N. and Lloyd, S. (2017). Quantum machine learning. *Nature* **549** (7671) 195–202.

Cong, I. and Duan, L. (2016). Quantum discriminant analysis for dimensionality reduction and classification. *New Journal of Physics* **18** (7) 073011.

Cheng, H.C., Hsieh, M. H. and Yeh, P. C. (2016). The learnability of unknown quantum measurements. *Quantum Information & Computation* **16** (7&8) 615–656.

Faddeev, D. K. and Faddeev, V. N. (1963). *Computational Methods of Linear Algebra*, Freeman.
Gavinsky, D. ((2012)). Quantum predictive learning and communication complexity with single input. *Quantum Information & Computation* **12** (7&8) 575–588.
Gruska, J. (1999). *Quantum Computing*, McGraw-Hill.
de la Higuera, C. (2005). A bibliographical study of grammatical inference. *Pattern Recognition,* **38** (9) 1332–1348.
de la Higuera, C. (2010). *Grammatical Inference: Learning Automata and Grammars*, Cambridge University Press.
Huang, H. Y., Broughton, M. and Cotler, J. (2022). Quantum advantage in learning from experiments. *Science* **376** (6598) 1182–1186.
Harrow, A. W., Hassidim, A. and Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical Review Letters* 103 (5) 150502.
Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica* **4** 373–395.
Kerenidis, I. and Prakash, A. (2017). Quantum recommendation systems. *Innovations in Theoretical Computer Science Conference* **49** 1–21.
Krenn, M., Landgraf, J., Foesel, T. and Marquardt, F. (2023). Artificial intelligence and machine learning for quantum technologies. *Physical Review A* **107** (1) 010101.
Kearns, M. J. and Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory*, MIT Press.
Kondacs, A. and Watrous, J. (1997). On the power of quantum finite state automata. In: *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, FOCS 97*, IEEE Computer Society Press, 66–75.
Lloyd, S., Mohseni, M. and Rebentrost, P. (2014). Quantum principal component analysis. *Nature Physics* **10** (9) 108–113.
Li, L. and Qiu, D. W. (2006). Determination of equivalence between quantum sequential machines. *Theoretical Computer Science* **358** 65–74.
Li, L. and Qiu, D. W. (2008). Determining the equivalence for one-way quantum finite automata. *Theoretical Computer Science* **403** 42–51.
Lloyd, S. and Weedbrook, C. (2018). Quantum generative adversarial learning. *Physical Review Letters* **121** (4) 040502.
Marquardt, F. (2021). Machine learning and quantum devices. *SciPost Physics Lecture Notes* **029**.
Moore, E. (1956). Gedanken-experiments on sequential machines. *Automata Studies, Annals of Mathematics Studies* **34** 129–153.
Moore, C. and Crutchfield, J. P. (2000). Quantum automata and quantum grammars. *Theoretical Computer Science* **237** 275–230.
Meyer, J. J., Mularski, M., Gil-Fuster, E., Mele, A. A., Arzani, F., Wilms, A. and Eisert, J. (2023). Exploiting symmetry in variational quantum machine learning. *PRX Quantum* **4** (1) 010328.
Mitarai, K., Negoro, M., Kitagawa, M. and Fujii, K. (2018). Quantum circuit learning. *Physical Review A* **98** (3) 032309.
Mereghetti, C., Palano, B., Cialdi, S. et al. (2020). Photonic realization of a quantum finite automaton. *Physical Review Research* **2** Art. no. 013089.
Mereghetti, C., Palano, B., Cialdi, S., Vento, V., Paris, M. G. A. and Olivares, S. (2020). Photonic realization of a quantum finite automaton. *Physical Review Research* **2** 013089 (15 pages).
Mateus, P., Qiu, D. W. and Li, L. (2012). On the complexity of minimizing probabilistic and quantum automata. *Information and Computation* **218** 36–53.
Nielsen, M. and Chuang, I. (2000). *Quantum Computation and Quantum Information*, Cambridge University Press.
Paschen, K. (2000). *Quantum Finite Automata using Ancilla Qubits*, University of Karlsruhe, Technical report.
Paz, A. (1971). *Introduction to Probabilistic Automata*, Academic Press.
Plachta, S. Z. D., Hiekkamäki, M., Yakaryilmaz, A. and Fickler, R. (2022). Quantum advantage using high-dimensional twisted photons as quantum finite automata. *Quantum-Austria* **6** 752.
Peled, D., Vardi, M. and Yannakakis, M. (2002). Black box checking. *Journal of Automata Languages & Combinatorics* **7** (2) 225–246.
Qiu, D. W. (2007). Automata theory based on quantum logic: Reversibilities and pushdown automata. *Theoretical Computer Science* **386** (1–2) 38–56.
Qiu, D. W., Li, L., Mateus, P. and Gruska, J. (2012). Quantum Finite Automata. In: J. Wang (ed.) *Finite State-Based Models and Applications*, CRC Handbook, 113–144.
Qiu, D. W., Li, L., Mateus, P. and Sernadas, A. (2015). Exponentially more concise quantum recognition of non-RMM languages. *Journal of Computer and System Sciences* **81** (2) 359–375.
Qiu, D. W., Luo, L. and Xiao, L. (2022). *Distributed Grover's Algorithm*, arXiv:2204.10487v4.
Qiu, D. W., Li, L., Zou, X., Mateus, P. and Gruska, J. (2011). Multi-letter quantum finite automata: Decidability of the equivalence and minimization of states. *Acta Informatica* **48** 271–290.
Qiu, D. W. and Yu, S. (2009). Hierarchy and equivalence of multi-letter quantum finite automata. *Theoretical Computer Science* **410** 3006–3017.
Rebentrost, P., Mohseni, M. and Lloyd, S. (2014). Quantum support vector machine for big data classification. *Physical Review Letters* **113** (13) 130503.
Rodriguez, L. E. H., Ullah, A., Espinosa, K. J. R., Dral, P. O. and Kananenka, A. A. (2022). A comparative study of different machine learning methods for dissipative quantum dynamics. *Machine Learning: Science and Technology* **3** (4) 045016.

Schuld, M., Sinayskiy, I. and Petruccione, F. (2015). An introduction to quantum machine learning. *Contemporary Physics* **56** (2) 172–185.
Say, A. C. and Yakaryılmaz, A. (2014). Quantum finite automata: A modern introduction. In: *Computing with New Resources*, Springer, 208–222.
Servedio, R. and Gortler, S. (2004). Equivalences and separations between quantum and classical learnability. *SIAM Journal on Computing* **33** (5) 1067–1092.
Tzeng, W. G. (1992). Learning probabilistic automata and Markov chains via queries. *Machine Learning* **8** (2) 151–166.
Tian, Y., Feng, T., Luo, M., Zheng, S. and Zhou, X. (2019). Experimental demonstration of quantum finite automaton. *NPJ Quantum Information* **5** (1) 1–5.
Tîrnăucă, C. and Knuutila, T. (2007). Polynomial time algorithms for learning k-reversible languages and pattern languages with correction queries. In: M. Hutter, R. Servedio, and E. Takimoto (eds.) *Algorithmic Learning Theory: 18th International Conference, ALT' 2007*, Lecture Notes in Artificial Intelligence, vol. 4754, Springer-Verlag, 272–284.
Vaandrager, F. (2017). Model learning. *Communications of the ACM* **60** (2) 86–95.
Wiebe, N., Braun, D. and Lloyd, S. (2012). Quantum algorithm for data fitting. *Physical Review Letters* **109** (5) 050505.
Wiebe, N., Kapoor, A. and Svore, K. (2016). Quantum deep learning. *Quantum Information & Computation* **16** (7) 541–587.
Zhang, C. (2010). An improved lower bound on query complexity for quantum PAC learning. *Information Processing Letters* **111** (1) 40–45.
Zhao, Z. Pozas-Kerstjens, A., Rebentrost, P. and Wittek, P. (2019). Bayesian deep learning on a quantum computer. *Quantum Machine Intelligence* **1** 41–51.

Cite this article: Qiu D (2024). Learning quantum finite automata with queries. *Mathematical Structures in Computer Science* **34**, 128–146. https://doi.org/10.1017/S0960129523000373