




# Quantum Algorithms for Shapley Value Calculation

Iain Burge\* , Michel Barbeau\* , Joaquin Garcia-Alfaro† 

\*School of Computer Science, Carleton University, Ottawa, Canada

†SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, Palaiseau, France

**Abstract**—In the classical context, the cooperative game theory concept of the Shapley value has been adapted for post hoc explanations of Machine Learning (ML) models. However, this approach does not easily translate to eXplainable Quantum ML (XQML). Finding Shapley values can be highly computationally complex. We propose quantum algorithms which can extract Shapley values within some confidence interval. Our results perform in polynomial time. We demonstrate the validity of each approach under specific examples of cooperative voting games.

**Index Terms**—Shapley Value, Quantum Computing, Cooperative Game Theory, Explainable Quantum Machine Learning, Machine Learning, Artificial Intelligence, Quantum Machine Learning.

## I. INTRODUCTION

Research in ML over the past decades deals with black box models. Unfortunately, black box models are inherently difficult to interpret. Inherently interpretable models would likely be best [18], as an explanation of an interpretable model is guaranteed to be correct. However, we ideally do not want to discard all of the previous research using black box models. As a result, there has been a substantial effort in implementing and improving post hoc explanation methods. Similarly, eXplainable Quantum ML (XQML) may benefit from adapting post hoc explanation methods to the quantum realm.

One of the most popular methods of generating post hoc explanations involves calculating Shapley values. Yet, classical strategies to approximate Shapley values are unwieldy to apply in the context of quantum computers. As a result, it is necessary to explore a more native quantum solution to Shapley value approximation.

In this paper, we develop a flexible framework for the global evaluation of input factors in quantum circuits that approximates the Shapley values of such factors. Our framework has a one time increased circuit complexity of an additional  $\mathcal{O}(an \log an)$  c-not gates, with a total increase in circuit depth of  $\mathcal{O}(an)$ , where  $n$  is the number of factors, and  $a > 0$  is a real number. The change in space complexity for global evaluations is an additional  $\mathcal{O}(\log an)$  qubits over the evaluated circuit. The circuit of increased complexity must then be repeated  $\mathcal{O}(\epsilon^{-1})$  times. This procedure can achieve an error of  $\mathcal{O}(a^{-1} + \epsilon)$ . This starkly contrasts the  $\mathcal{O}(2^n)$  assessments needed to assess the Shapley values under the general case directly.

The paper is organized as follows. Section II surveys related work. Section III provides background and preliminaries. Sections IV and V present our methods. Sections VI and VII provide some examples. Section VIII concludes the work.

## II. RELATED WORK

Shapley values have been widely used to address multiple engineering problems, including regression, statistical analysis, and machine learning [11]. Finding Shapley values presents a difficult computational combinatorial problem. Our work proposes a novel quantum algorithm that reduces this combinatorial problem to an estimation problem which can leverage the power of quantum computation. Our approach performs in polynomial time. We apply our method to weighted voting games [12].

The deterministic computation of Shapley values in the context of weighted voting games is as difficult as NP-Hard [12], [16]. Since voting games are some of the simplest cooperative games, this result does not bode well for more complex scenarios. In the context of Shapley values for machine learning, it has also been shown that the calculation of Shapley values are not tractable for even regression models [21]. It was also proven that on the empirical distribution, finding a Shapley value takes exponential time [2].

The literature has also addressed the use of Shapley values on Quantum ML (QML). Indeed, XQML aims at adding explainability behind model predictions, e.g., in addition to providing classification [13]. XQML can be considered as an alternative research direction of QML instead of trying to justify quantum advantage [19]. The eventual goal of XQML is to provide, in addition to predictions, humanly understandable interpretations of the predictive models, e.g., for malware detection and classification, under a cybersecurity context [20].

Recent work by Heese et al. extends the notion of feature importance for model predictions in classical ML, to the QML realm [9]. In contrast to classical ML methods, in which Shapley values are applied to evaluate the importance of each feature for model predictions, Heese et al. apply Shapley values to evaluate the relevance of each quantum gate associated with a given parametrized quantum circuit. In their work, Heese et al. compute Shapley values involving stochastic processing. In general, the algorithms presented in this paper improve stochastic computation of Shapley values. Moreover, our work justifies the interest in handling post hoc explanation frameworks such as the one of Heese et al., in a quantum extended manner.

## III. SHAPLEY VALUES

Cooperative game theory is the study of coalitional games. In this article, we are most interested in Shapley values. We now list some definitions and preliminaries,

**Definition 1** (Coalitional game). A coalitional game is described as the pair  $G = (F, V)$ .  $F = \{0, 1, \dots, n\}$  is a set of  $n + 1$  players.  $V : \mathcal{P}(F) \rightarrow \mathbb{R}$  is a value function with  $V(S) \in \mathbb{R}$  representing the value of a given coalition  $S \subseteq F$ , with the restriction that  $V(\emptyset) = 0$ .

**Definition 2** (Payoff vector). Given a game  $G = (F, V)$ , there exists a payoff vector  $\Phi(G)$  of length  $n + 1$ . Each element  $\Phi(G)_i \in \mathbb{R}$  represents the utility of player  $i \in F$ . A payoff vector is determined by the value function. Player  $i$ 's payoff value  $\Phi(G)_i$  is determined by how  $V(S)$ ,  $S \subseteq F$ , is affected by  $i$ 's inclusion or exclusion from  $S$ .

There are a variety of solution concepts for constructing payoffs [1]. We focus on the Shapley solution concept, which returns a payoff vector [23]. Each element of the payoff vector  $\Phi(G)_i$ , is called player  $i$ 's Shapley value. Shapley values have multiple different interpretations depending on the game being analyzed.

Shapley values are derived using one of several sets of axioms. We use the following four [23]. Suppose we have games  $G = (F, V)$  and  $G' = (F, V')$ , and a payoff vector  $\Phi(G)$ , then:

- 1) Efficiency: The sum of all utility is equal to the utility of the grand coalition (the coalition containing all players),

$$\sum_{i=1}^n \Phi(G)_i = V(F)$$

- 2) Equal Treatment: Players  $i, j$  are said to be symmetrical if for all  $S \subseteq F$ , where  $i, j \notin S$  we have that  $V(S \cup \{i\}) = V(S \cup \{j\})$ . If  $i$ , and  $j$  are symmetric in  $G$ , then they are treated equally,  $\Phi(G)_i = \Phi(G)_j$ .
- 3) Null Player: Consider a player  $i \in F$ , if for all  $S \subseteq F$  such that  $i \notin S$ , we have  $V(S) = V(S \cup \{i\})$ , then  $i$  is a null player. If  $i$  is a null player then  $\Phi(G)_i = 0$ .
- 4) Additivity: If a player is in two games  $G$  and  $G'$ , then the Shapley values of the two games is additive

$$\Phi(G + G')_i = \Phi(G)_i + \Phi(G')_i$$

where a game  $G + G'$  is defined as  $(F, V + V')$ , and  $(V + V')(S) = V(S) + V'(S)$ ,  $S \subseteq F$ .

These axioms lead to a single unique and intuitive division of utility [23]. The values of the payoff vectors can be interpreted as the responsibility of the respective players for the final outcome [8]. When player  $i$  has a small payoff  $\Phi(G)_i$ , then player  $i$  has a neutral impact on the final outcome. When player  $i$  has a large payoff, then player  $i$  has a large impact on the final outcome.

The Shapley value of  $i$  turns out to be the expected marginal contribution to a random coalition  $S \subseteq F \setminus \{i\}$ , where the marginal contribution is equal to  $V(S \cup \{i\}) - V(S)$  [8].

**Definition 3** (Shapley value). Let  $G = (F, V)$ , for simplicity sake, we write  $\Phi(G)_i$  as  $\Phi_i$ . The Shapley value of the  $i^{\text{th}}$  player  $\Phi_i$  is described as:

$$\Phi_i = \sum_{S \subseteq F \setminus \{i\}} \gamma(|F \setminus \{i\}|, |S|) \cdot (V(S \cup \{i\}) - V(S)) \quad (1)$$

where  $n = |F \setminus \{i\}|$ , and

$$\gamma(n, m) = \frac{1}{\binom{n}{m}(n+1)} = \frac{m!(n-m)!}{(n+1)!}$$

For this paper, we will define algorithms for a special case of games, superadditive games (Definition 4).

**Definition 4** (Superadditive game). A game is superadditive if for all  $S, H \subseteq F$ , such that  $S$  and  $H$  are disjoint ( $S \cap H = \emptyset$ ) we have,  $V(S \cup H) \geq V(S) + V(H)$ .

Note that when a game is superadditive, every summand in Equation (1) is non-negative.

#### IV. A QUANTUM REPRESENTATION OF THE SHAPLEY VALUE CALCULATION

We represent the Shapley value calculation problem in the quantum format. Consider an  $n + 1$  player superadditive game  $G$  represented by the pair  $(F, V)$ , where  $F = \{0, 1, \dots, n\}$  and  $V : \mathcal{P}(F) \rightarrow \mathbb{R}$ , with  $V(\emptyset) = 0$ . Let us define the function,

$$W(S) = V(S \cup \{i\}) - V(S), \quad S \subseteq F \setminus \{i\}. \quad (2)$$

We define  $W_{\max}$  as an upper bound of the absolute maximum change in value when adding player  $i$  to a subset  $S$ .

$$W_{\max} \geq \max_{S \subseteq F \setminus \{i\}} |W(S)|. \quad (3)$$

Let  $i \in F$  be a given player. Consider the selection binary sequence  $x = x_0 x_1 \dots x_{i-1} x_{i+1} \dots x_n$ . Let  $S_x$  be the set of all players  $j \in F$  such that  $x_j = 1$ . Then  $S_x$  could encode every player coalition that excludes the player  $i$ . Next, define,

$$\hat{W}(x) := \frac{W(S_x)}{W_{\max}}. \quad (4)$$

We define the following block diagonal matrix:

$$B = \bigoplus_{k=0}^{2^n - 1} \begin{pmatrix} \sqrt{1 - \phi(k, n)} & \sqrt{\phi(k, n)} \\ \sqrt{\phi(k, n)} & -\sqrt{1 - \phi(k, n)} \end{pmatrix}$$

where

$$\phi(k, n) = \gamma(n, c(k)) \cdot \hat{W}(k)$$

and  $c(k)$  is the number of ones in the binary representation of  $k$ , over  $\log n$  bits.

**Theorem 1.** The block diagonal matrix  $B(n)$  is unitary.

*Proof.* It follows from the fact that each block

$$\begin{pmatrix} \sqrt{1 - \phi(k, n)} & \sqrt{\phi(k, n)} \\ \sqrt{\phi(k, n)} & -\sqrt{1 - \phi(k, n)} \end{pmatrix}$$

of  $B$  is unitary. ■

We construct the quantum system:

$$S = B(H^{\otimes n} \otimes I)|0\rangle^{\otimes n+1} \quad (5)$$

**Theorem 2.** The expected values of the rightmost qubits of  $S$  is  $\frac{\Phi_i}{2^n \cdot W_{\max}}$ .

*Proof.* It follows from the fact that

$$(H^{\otimes n} \otimes I)|0\rangle^{\otimes n+1} = \sum_{k=0}^{2^n-1} \frac{1}{\sqrt{2^n}} |k\rangle |0\rangle$$

and the following sequence of equivalences:

$$\begin{aligned} & \sum_{k=0}^{2^n-1} (B_{k+1,k})^2 \\ &= \sum_{k=0}^{2^n-1} \phi(k, n) = \sum_{k=0}^{2^n-1} \gamma(n, c(k)) \cdot W(k) \\ &= \sum_{S \subseteq F \setminus \{i\}} \gamma(|F \setminus \{i\}|, |S|) \cdot \frac{(V(S \cup \{i\}) - V(S))}{W_{max}} \end{aligned}$$

Hence, the probability of measuring a one in the last qubit of the quantum system  $S$  is

$$\frac{1}{2^n \cdot W_{max}} \sum_{S \subseteq F \setminus \{i\}} \gamma(|F \setminus \{i\}|, |S|) \cdot (V(S \cup \{i\}) - V(S))$$

which is equivalent to  $\frac{\Phi_i}{2^n \cdot W_{max}}$ .

The Shapley value  $\Phi_i$  can be obtained by repeatedly creating the quantum system  $S$ , measuring their last qubit, taking the average, and finally multiplying by  $2^n \cdot W_{max}$ . As will be briefly discussed in Section V, the value can also be extracted with a more efficient strategy. Although, this representation requires the preparation of a quantum state with an exponential number of terms ( $2^n$ ). In the following section, we develop a more efficient solution for Shapley value calculation of superadditive games.

## V. QUANTUM ALGORITHM FOR SUPERADDITIVE GAMES

Consider an  $n + 1$  player game  $G$  represented by the pair  $(F, V)$ , where  $F = \{0, 1, \dots, n\}$  and  $V : \mathcal{P}(F) \rightarrow \mathbb{R}$ , with  $V(\emptyset) = 0$ . A naive approach to finding the  $i^{\text{th}}$  player's Shapley value is through direct calculation using the Shapley Equation (1), completing the task in  $\mathcal{O}(2^n)$  assessments of  $V$ . For structured games, it is occasionally possible to calculate Shapley values more efficiently. Otherwise, the only option is random sampling [5]. In each case, there are substantial trade-offs, in this section we propose a quantum algorithm which has some substantial advantages.

Suppose we have a quantum representation of the function  $W(S)$ , defined in Equation (2), which comprises two registers, a player register, and a utility register. In the player register, the computational basis states represent different subsets of players, where the  $j^{\text{th}}$  qubit represents the  $j^{\text{th}}$  player. In this encoding, the  $j^{\text{th}}$  qubit being  $|1\rangle$  represents  $j$ 's inclusion in the subset, and  $|0\rangle$  represents its exclusion. Thus, every possible subset of players has a corresponding basis state. Representing every subset of players simultaneously in a quantum superposition is possible. The one-qubit utility register encodes the output of  $W$ .

Computing the Shapley value method consists of the following steps:

- 1) Represent every possible subset of players in a quantum superposition, excluding player  $i$ , with amplitudes corresponding to weights  $(\gamma(n, m))$  in the Shapley Equation (1).
- 2) Perform the quantum version of  $W$  controlled by the player register, and any auxiliary registers, on the utility register.

From this point, it is possible to approximate the Shapley value  $\Phi_i$  if one has the expected value of measuring the utility register.

- 3) To approximate the expected value of measuring the utility register, one can either:

- Use  $\mathcal{O}(\epsilon^{-2})$  repetitions of steps 1 and 2 followed by a measurement of the utility register.
- Perform amplitude estimation with  $\mathcal{O}(\epsilon^{-1})$  iterations, where step 1 is  $\mathcal{A}$  and step 2 is  $W$  as defined in [14]. In opposition to the previous choice, this process increases circuit depth.

In either case, the error in approximating the expected value for measuring the utility register is within  $\pm\epsilon$  with some predetermined likelihood, where  $\epsilon$  is a small positive real number.

The details of the method will now be described. For the sake of simplicity, we assume superadditivity; however, this algorithm could be easily extended to include general cooperative games [3]. The goal is to efficiently approximate the Shapley value  $\Phi_i$  of a given player  $i \in F$ . Suppose quantum representation of function of function  $\hat{W}(x)$ , defined in Equation (4), is given as:

$$U_W |x\rangle |0\rangle := |x\rangle \left( \sqrt{1 - \hat{W}(x)} |0\rangle + \sqrt{\hat{W}(x)} |1\rangle \right),$$

where  $|x\rangle$  is a vector in the computational basis (i.e.,  $x \in \{0, 1\}^n$ ).

A critical step for the algorithm is to approximate the weights of the Shapley value function. These weights correspond perfectly to a slightly modified beta function.

**Definition 5** (Beta function). *We define the beta function as:*

$$\beta_{n,m} = \int_0^1 x^m (1-x)^{n-m} dx, \quad 0 \leq m \leq n, \quad n, m \in \mathbb{N}.$$

**Lemma 1.** *We have the following recurrence relationship:*

$$\beta_{n,0} = \beta_{n,n} = \frac{1}{n+1} \text{ and } \beta_{n,m} = \frac{m}{n - (m-1)} \beta_{n,m-1}.$$

*Proof.* There are two cases.

Case 1 ( $m$  is equal to zero, or  $n$ ). We have the following integration.

$$\beta_{n,0} = \int_0^1 (1-x)^n dx = -\frac{(1-x)^{n+1}}{n+1} \Big|_0^1 = \frac{1}{n+1}$$

A nearly identical calculation can be used to show  $\beta_{n,n}$  is equal to  $\frac{1}{n+1}$ .

Case 2 ( $0 < m < n$ ). We have the following partial integration.

$$\begin{aligned}
\beta_{n,m} &= \int_0^1 x^m (1-x)^{n-m} dx \\
&= \frac{x^m (1-x)^{n-(m-1)}}{n-(m-1)} \Big|_0^1 \\
&= \int_0^1 \frac{-m}{n-(m-1)} x^{m-1} (1-x)^{n-(m-1)} dx \\
&= 0 + \frac{m}{n-(m-1)} \int_0^1 x^{m-1} (1-x)^{n-(m-1)} dx \\
&= \frac{m}{n-(m-1)} \beta_{n,m-1}
\end{aligned}$$

**Theorem 3.** *The beta function  $\beta_{n,m}$  is equal to the Shapley weight function  $\gamma(n,m)$ , with  $0 \leq m \leq n$  and  $m, n \in \mathbb{N}$ .*

*Proof.* The proof is by induction on  $m$ .

Base case ( $m = 0$ ). According to Case 1 of Lemma 1, we have that  $\beta_{n,0}$  is equal to  $\frac{1}{n+1}$ , which is equal to  $\gamma(n,0)$ .

Inductive step ( $m > 0$ ). Suppose  $\beta_{n,k}$  is equal to  $\gamma(n,k)$ ,  $k \in \mathbb{N}$ , we need to show  $\beta_{n,k+1} = \gamma(n,k+1)$ ,  $0 \leq k < n$ . According to Case 2 of Lemma 1,  $\beta_{n,k+1}$  is equal to  $\frac{k+1}{n-k} \beta_{n,k}$ . Using the inductive hypothesis, the latter is equivalent to

$$\frac{k+1}{n-k} \gamma(n,k) = \frac{k+1}{n-k} \cdot \frac{k!(n-k)!}{(n+1)!}$$

which matches the definition of  $\gamma(n,k+1)$ .

The beta function, and by extension the Shapley weights, are approximated with Riemann sums which roughly represent the area under the function  $x^m(1-x)^{n-m}$  using partitions of the interval  $[0,1]$  with respect to  $x$ . Function  $x^m(1-x)^{n-m}$  can be implemented efficiently on a quantum computer.

**Definition 6** (Riemann sum). *A Riemann sum [17, page 276] of a function  $f$  with respect to a partition  $P = (t_0, \dots, t_s)$  of the interval  $[a,b]$  is an approximation of the integral of  $f$  from  $a$  to  $b$  of the form:*

$$\sum_{k=0}^{s-1} (t_{k+1} - t_k) \cdot f(x_k)$$

Where  $t_{k+1} - t_k$  is the width of the subinterval, and  $f(x_k)$ ,  $x \in [t_k, t_{k+1}]$ , is the height.

The following is the initial quantum state:

$$|\psi_0\rangle = |0\rangle_{\text{Pt}} \otimes |0\rangle_{\text{Pl}} \otimes |0\rangle_{\text{Ut}}, \quad (6)$$

where Pt, Pl, and Ut respectively denote the partition, player, and utility registers. Pt is used to prepare the  $\gamma(n,m)$  weights. Suppose the number of qubits  $\ell \in \mathbb{N}$  in Pt is  $\ell = \lceil \log_2 an \rceil$ , for some fixed  $a > 0$ , thus  $\ell = \mathcal{O}(\log an)$ . Then the partition register can be loaded with an arbitrary

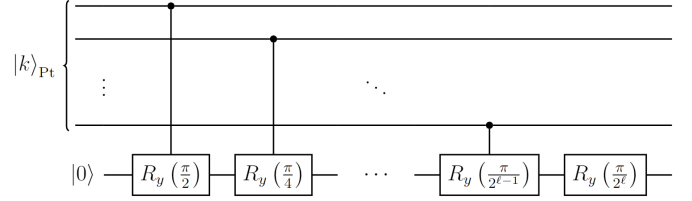


Fig. 1. This circuit  $R$  is a controlled rotation, where  $R_y(\theta) = (\cos(\theta/2), -\sin(\theta/2); \sin(\theta/2), \cos(\theta/2))$ . (Note: Library used for visualizing circuits can be found here: [10])

quantum state in  $\mathcal{O}(an)$  time [15]. Let the number of qubits in the player register be  $n$ , one qubit per player excluding player  $i$ . Let the number of qubits in the utility register be one.

Consider the following function,

$$t_\ell(k) = \sin^2\left(\frac{\pi k}{2^{\ell+1}}\right) \text{ with } k = 0, 1, \dots, 2^\ell$$

with which the partition  $P_\ell = (t_\ell(k))_{k=0}^{2^\ell}$  of interval  $[0,1]$  is constructed. This partition is computationally simple to implement. Finally, let us define  $w_\ell(k)$  to be the width of the  $k^{\text{th}}$  subinterval of  $P_\ell$ ,  $w_\ell(k) = t_\ell(k+1) - t_\ell(k)$ ,  $k = 0, 1, \dots, 2^\ell - 1$ .

Let us prepare the partition register to be,

$$\sum_{k=0}^{2^\ell-1} \sqrt{w_\ell(k)} |k\rangle.$$

the new state of the quantum system becomes,

$$|\psi_1\rangle = \sum_{k=0}^{2^\ell-1} \sqrt{w_\ell(k)} |k\rangle_{\text{Pt}} |0\rangle_{\text{Pl}} |0\rangle_{\text{Ut}}.$$

Next, perform a series of controlled rotations  $R$  (circuit in Figure 1) of the form,

$$R |k\rangle |0\rangle := |k\rangle \left( \sqrt{1-t'_\ell(k)} |0\rangle + \sqrt{t'_\ell(k)} |1\rangle \right),$$

where  $t'_\ell(k) = t_{\ell+1}(2k+1)$  is used to sample the height of the  $k^{\text{th}}$  subinterval in the Riemann sum. Note that the value of  $t'_\ell(k)$  is always somewhere in the range  $[t_\ell(k), t_\ell(k+1)]$ .

Unitary  $R$  is performed on each qubit in the player register, controlled by the partition register. The entirety of the applications of  $R$  can be performed with  $\mathcal{O}(an \log an)$  gates in  $\mathcal{O}(an)$  layers and yields the quantum state:

$$|\psi_2\rangle = \sum_{k=0}^{2^\ell-1} \sqrt{w_\ell(k)} |k\rangle_{\text{Pt}} \left( \sqrt{1-t'_\ell(k)} |0\rangle + \sqrt{t'_\ell(k)} |1\rangle \right)^{\otimes n} |0\rangle_{\text{Ut}}.$$

Note that the player register is of size  $n$  qubits. Let  $H_m$  be the set of binary numbers of hamming distance  $m$  from 0 in  $n$  bits, then we can rewrite  $|\psi_2\rangle$  as:

$$|\psi_2\rangle = \sum_{k=0}^{2^\ell-1} \sqrt{w_\ell(k)} |k\rangle_{\text{Pt}} \cdot \sum_{m=0}^n \sqrt{t'_\ell(k)^m (1-t'_\ell(k))^{n-m}} \cdot \sum_{h \in H_m} |h\rangle_{\text{Pl}} |0\rangle_{\text{Ut}}$$



Note that, with this encoding style for  $S_x$ ,  $x$ 's hamming distance from 0 in  $n$  bits is equal to the size of  $S_x$ . In other words, if  $h \in H_m$ , then  $S_h$  contains  $m$  players.

**Example 1.** Let us consider an example where the number of players is three ( $n$  is two). We have,

$$\begin{aligned} & \left( \sqrt{1-t'_\ell(k)} |0\rangle + \sqrt{t'_\ell(k)} |1\rangle \right)^{\otimes 2} = \\ & \sqrt{(1-t'_\ell(k))^2} |00\rangle + \sqrt{t'_\ell(k)(1-t'_\ell(k))} |01\rangle \\ & + \sqrt{t'_\ell(k)(1-t'_\ell(k))} |10\rangle + \sqrt{t'_\ell(k)^2} |11\rangle. \end{aligned}$$

Note that  $|00\rangle$  is hamming distance 0 from  $|00\rangle$ ,  $|01\rangle$  and  $|10\rangle$  are hamming distance 1 from  $|00\rangle$ , and  $|11\rangle$  is hamming distance 2 from  $|00\rangle$ . With this knowledge in hand, we can rewrite the quantum state of Example 1 as,

$$\sqrt{(1-t'_\ell(k))^2} \sum_{h \in H_0} |h\rangle + \sqrt{t'_\ell(k)(1-t'_\ell(k))} \sum_{h \in H_1} |h\rangle + \sqrt{t'_\ell(k)^2} \sum_{h \in H_2} |h\rangle$$

This can now be arranged into the general form,

$$\sum_{m=0}^n \sqrt{t'_\ell(k)^m (1-t'_\ell(k))^{n-m}} \sum_{h \in H_m} |h\rangle, \quad n=2.$$

This completes the example.

Rearranging the quantum state  $|\psi_2\rangle$  gives,

$$|\psi_2\rangle = \sum_{m=0}^n \sum_{h \in H_m} \sum_{k=0}^{2^\ell-1} \sqrt{w_\ell(k) t'_\ell(k)^m (1-t'_\ell(k))^{n-m}} |k\rangle_{\text{Pt}} |h\rangle_{\text{Pl}} |0\rangle_{\text{Ut}}.$$

Next, we perform  $U_W$  on the utility register controlled by the player register. For convenience, let us for the moment write  $U_W |h\rangle |0\rangle = |h\rangle |W(h)\rangle$ , where,

$$|W(h)\rangle = \sqrt{1-\hat{W}(h)} |0\rangle + \sqrt{\hat{W}(h)} |1\rangle$$

Applying  $U_W |h\rangle_{\text{Pl}} |0\rangle_{\text{Ut}}$  gives  $|\psi_3\rangle$ , which is equal to,

$$\sum_{m=0}^n \sum_{h \in H_m} \sum_{k=0}^{2^\ell-1} \sqrt{w_\ell(k) t'_\ell(k)^m (1-t'_\ell(k))^{n-m}} |k\rangle_{\text{Pt}} |h\rangle_{\text{Pl}} |W(h)\rangle_{\text{Ut}}.$$

This operation is wholly dependent on the game being analyzed and its complexity. Assuming the algorithm is being implemented with a look-up table, one could likely use qRAM [6]. This approach has a time complexity of  $\mathcal{O}(n)$  at the cost  $\mathcal{O}(2^n)$  qubits for storage. However, depending on the problem, there are often far less resource-intensive methods of implementing  $U_W$ , as will be seen with the implementation of weighted voting games (Section VII).

This is the final quantum state. Let us now analyze this state through the lens of density matrices. Taking the partial trace with respect to the partition and player registers yields,

$$\text{tr}_{\text{Pt,Pl}}(|\psi_3\rangle\langle\psi_3|) = \sum_{m=0}^n \sum_{h \in H_m} \left( \sum_{k=0}^{2^\ell-1} w_\ell(k) t'_\ell(k)^m (1-t'_\ell(k))^{n-m} \right) \cdot |W(h)\rangle_{\text{Ut}} \langle W(h)|_{\text{Ut}}.$$

**Theorem 4.** The Riemann sum using partition  $P_\ell$  to approximate area under  $x^m(1-x)^{n-m}$  for  $x \in [0, 1]$  asymptotically approaches  $\gamma(n, m)$ . Formally,

$$\lim_{\ell \rightarrow \infty} \sum_{k=0}^{2^\ell-1} w_\ell(k) t'_\ell(k)^m (1-t'_\ell(k))^{n-m} = \gamma(n, m) \quad (7)$$

*Proof.* Let  $f(x) = x^m(1-x)^{n-m}$ , and recall our definition for partition of  $[0, 1]$ ,  $P_\ell = (t'_\ell(k))_{k=0}^{2^\ell}$ .

Define mesh( $P_\ell$ ) = sup $\{w_\ell(k) : k = 0, 1, \dots, 2^\ell - 1\}$  [17, page 275]. Since,

$$w_\ell(k) = \sin^2\left(\frac{\pi(k+1)}{2^{\ell+1}}\right) - \sin^2\left(\frac{\pi k}{2^{\ell+1}}\right),$$

we can bound  $w_\ell(k)$ ,

$$w_\ell(k) \leq \frac{\pi}{2} \left( \left( \frac{\pi(k+1)}{2^{\ell+1}} \right) - \left( \frac{\pi k}{2^{\ell+1}} \right) \right) = \frac{\pi^2}{2^{\ell+2}}.$$

This is valid because  $d/dx \sin^2((\pi/2)x) \leq d/dx(\pi/2)x$  for all  $x \in [0, 1]$ . It is then clear that as  $\ell \rightarrow \infty$ , mesh( $P_\ell$ )  $\rightarrow 0$ . Additionally,  $f$  is integrable over  $[a, b]$ , as it is a polynomial and hence continuous [17, page 278].

By [17, page 278, Corollary 32.10], as  $\ell$  increases, any Riemann sum of  $f$  using partition  $P_\ell$  converges to:

$$\int_0^1 x^m (1-x)^{n-m},$$

which is equal to  $\gamma(n, m)$ . ■

The Riemann sum approximation of the modified beta function,  $\beta_{n,m} = \gamma(n, m)$ , is visualized in Figure 2. Applying a Riemann sum approximation for  $\gamma(n, m)$ , we have,

$$\text{tr}_{\text{Pt,Pl}}(|\psi_3\rangle\langle\psi_3|) \approx \sum_{m=0}^n \sum_{h \in H_m} \gamma(n, m) |W(h)\rangle_{\text{Ut}} \langle W(h)|_{\text{Ut}}.$$

Finally, suppose we measuring the utility register in the computational basis. This yields the following expected value with empirical error less than  $\mathcal{O}(a^{-1})$  (Section VIII):

$$\sum_{m=0}^n \sum_{h \in H_m} \gamma(n, m) \hat{W}(h)$$

Plugging in the definition for  $\hat{W}$ , we have

$$\frac{1}{W_{\max}} \sum_{m=0}^n \sum_{h \in H_m} \gamma(n, m) (V(S_h \cup \{i\}) - V(S_h))$$

Notice that, in the  $S_x$  encoding,  $H_m$  represents each subset of  $F \setminus \{i\}$  of size  $m$ . As a result, the equation is, in effect, summing over each subset of  $F \setminus \{i\}$ . As a final step, multiply by  $W_{\max}$ :

$$\sum_{S \subseteq F \setminus \{i\}} \gamma(|F \setminus \{i\}|, |S|) \cdot (V(S \cup \{i\}) - V(S)).$$

This expected value is precisely the Shapley value  $\Phi_i$  to some error which is shown empirically in Section VII.

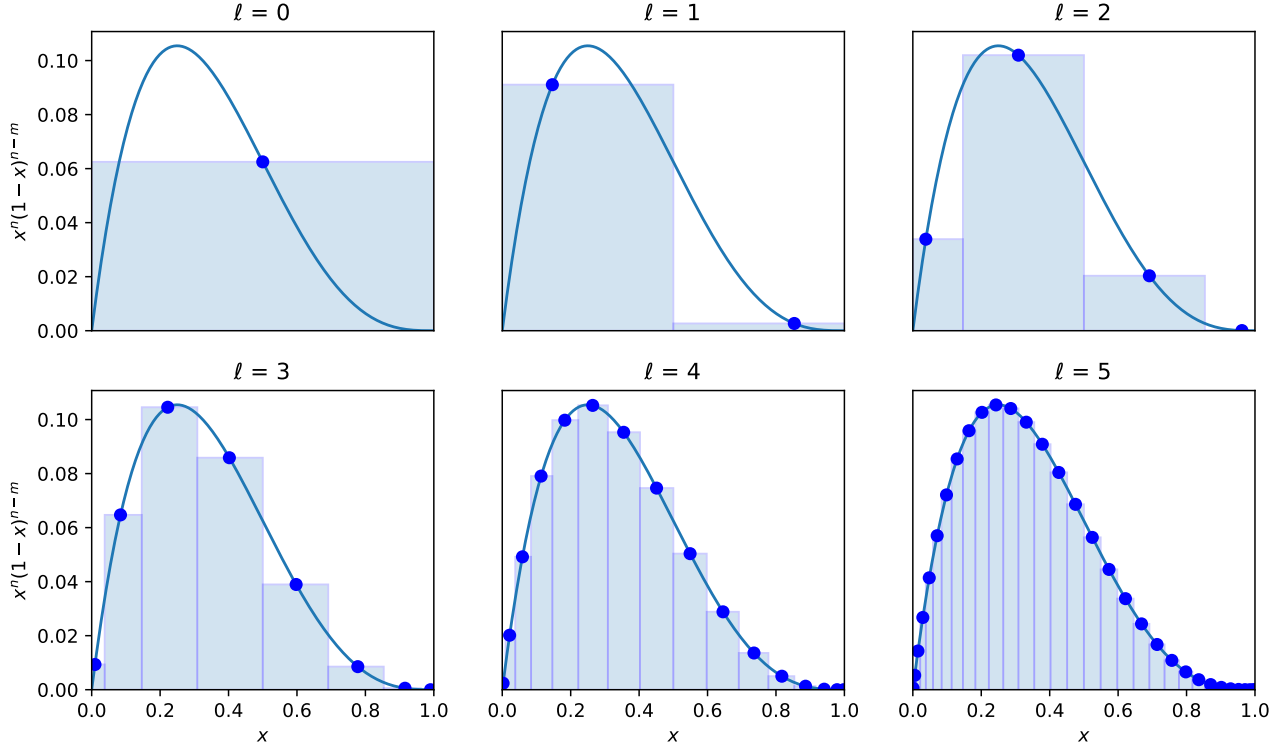


Fig. 2. Visual representation of  $\beta_{n,m}$  being approximated using Riemann sums with partition  $P_\ell$  over function  $x^m(1-x)^{n-m}$ ,  $t \in [0, 1]$ ,  $n = 4$ ,  $m = 1$ . The  $k^{\text{th}}$  rectangle's height is  $(t'_\ell(k))^m(1 - t'_\ell(k))^{n-m}$ , and its width is  $w_\ell(k)$ .

With the ability to craft these states, we can now extract the required information to find a close approximation to the Shapley value. Assuming we could get the expected value instantly, we would (empirically; see Section VII) have an error of  $\mathcal{O}(a^{-1})$ . However, it takes some work to approximate an expected value. This can be achieved with accuracy  $\epsilon$  with some chosen confidence using amplitude amplification as is shown in [14], with circuit depth  $\mathcal{O}(\epsilon^{-1})$  times our algorithm. However, for simplicity, we can also repeatedly construct state  $|\psi_3\rangle$  and measure the utility register roughly  $\mathcal{O}(\epsilon^{-2})$  times. The resulting measurements  $|0\rangle$  and  $|1\rangle$  can be analyzed using binomial distributions. We can estimate the underlying probability, from which we can extract the Shapley value, within some confidence interval [22]. This estimation takes up the bulk of the runtime for the algorithm. Thus the total error can be as low as  $\mathcal{O}(a^{-1} + \epsilon)$ . This error will be multiplied by  $W_{\max}$ .

## VI. WEIGHTED VOTING GAMES

Let us consider a scenario where Shapley values correspond to voting power. The voting power of a player represents how many instances for which that player has the deciding vote. Three friends sit around a table. They are deliberating a grave matter. Should they get Chinese food for the second weekend in a row? They decide they should take a vote. Alice just got a promotion at work. To celebrate this, her friends agreed to give her three votes. Bob, who is the youngest of the group, also

had good news, an incredible mark on his latest assignment! So, everyone decided Bob should get two votes. Charley, who had nothing to celebrate, gets one vote. The group decides to go out for Chinese food if there are four *yes* votes. In the end, all of the friends vote for Chinese food. At the restaurant, they run into their friend David, who is a mathematician. David is intrigued when he hears about their vote. He begins to wonder how much power each friend had in the vote. The intuitive answer is that Alice had the most voting power, Bob had the second most, and Charley had the least. However, David notices something strange. Bob does not seem to have a more meaningful influence than Charley's. There are no cases where Bob's two votes would do more than Charley's single vote. So, David concludes, there must be a more nuanced answer. Lucky for David, he has heard of cooperative game theory and Shapley values. So, he might be able to answer his question.

How can Definition 3 be applied to David's problem? Let us define the game  $G = (F, V)$ . The players are Alice (0), Bob (1), and Charley (2) represented as  $F = \{0, 1, 2\}$ . Denote each player's voting weight as  $w_0 = 3$ ,  $w_1 = 2$ , and  $w_2 = 1$ . Recall that the vote threshold was  $q = 4$ . Then, for any subset  $S \subseteq F$ , we can define  $V$  as:

$$V(S) = \begin{cases} 1 & \text{if } \sum_{j \in S} w_j \geq q, \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

So,  $V(S)$  is one if the sum of players' votes in  $S$  reaches the

threshold of four. Otherwise, the vote fails, and so  $V(S)$  is zero.

In general, this is called a weighted voting game [12]. One could easily add more players with arbitrary non-negative  $w_j$  and  $q$ . Note that weighted voting games fall into the family of superadditive games (Definition 4).

In this context, the terms in the Shapley value equation have an intuitive meaning. Take player  $i$ , and consider a set  $S \subseteq F \setminus \{i\}$ . If  $V(S \cup \{i\}) - V(S) = 1$ , then the  $i^{\text{th}}$  player is a deciding vote for the set of players  $S$ . Otherwise, player  $i$  is not a deciding vote.

Thus, for weighted voting games, player  $i$ 's Shapley value represents a weighted count of how many times  $i$  is a deciding vote. We can work out the Shapley values by hand:

$$\begin{aligned} V(\emptyset) &= 0 & V(\{0, 1\}) &= 1 \\ V(\{0\}) &= 0 & V(\{0, 2\}) &= 1 \\ V(\{1\}) &= 0 & V(\{1, 2\}) &= 0 \\ V(\{2\}) &= 0 & V(\{0, 1, 2\}) &= 1 \end{aligned}$$

From this, we have:

$$\begin{aligned} \Phi_0 &= \sum_{S \subseteq F \setminus \{i\}} \gamma(|F \setminus \{i\}|, |S|) \cdot (V(S \cup \{i\}) - V(S)) \\ &= \gamma(2, 0) \cdot (V(\{0\}) - V(\emptyset)) + \gamma(2, 1) \cdot (V(\{0, 1\}) - V(\{1\})) \\ &\quad + \gamma(2, 1) \cdot (V(\{0, 2\}) - V(\{2\})) + \gamma(2, 2) \cdot (V(\{0, 1, 2\}) \\ &\quad - V(\{1, 2\})) = \gamma(2, 0) \cdot (0 - 0) + \gamma(2, 1) \cdot (1 - 0) \\ &\quad + \gamma(2, 1) \cdot (1 - 0) + \gamma(2, 2) \cdot (1 - 0) \\ &= 2 \cdot \gamma(2, 1) + \gamma(2, 2) = 2 \cdot \frac{1!(2-1)!}{(2+1)!} + \frac{2!(2-2)!}{(2+1)!} \\ &= 2 \cdot \frac{1}{6} + \frac{1}{3} = \frac{2}{3} \end{aligned}$$

This can be repeated to get,

$$\Phi_1, \Phi_2 = \frac{1}{6}$$

In the case of Alice, Bob, and Charley's voting game, it is trivial to calculate their respective Shapley values. However, what if one-hundred colleagues were choosing a venue for a party, all with different numbers of votes? In that case, a direct calculation would take  $2^{100}$  assessments of  $V!$  for this more general case, we need to be clever.

## VII. NUMERICAL EXAMPLES

Perhaps we can help David solve his problem (cf. Section VI) using our quantum approach. We intend to apply the method presented in Section V for weighted voting games. Additional results, together with the simulation code, are available in a [companion github repository](#) [4]. Let us approximate each player's Shapley value,  $\Phi_i$ . We have a game  $G = (F, V)$ , where  $F = \{0, 1, 2\}$ ,  $n = 2$ , and  $V$  is defined in Equation 8. In this case,  $W(S)$ ,  $S \subseteq F \setminus \{i\}$ , represents whether or not player  $i$  has the deciding vote assuming those in  $S$  are voting for Chinese food. For example, Alice (0) is a deciding vote for  $S = \{1\}$  ( $S$  contains Bob). Let the voting weights be  $w_0 = 3$ ,

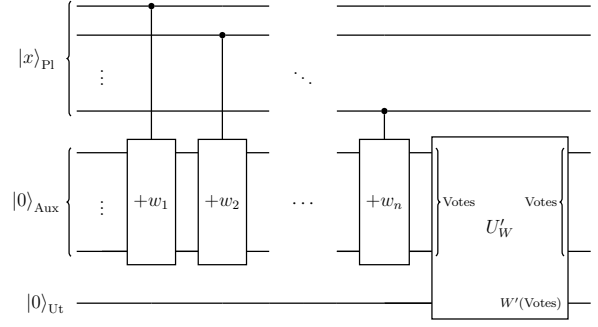


Fig. 3. Circuit representation of  $U'_W$  for a weighted voting game. This circuit takes an input  $x$  and outputs  $W(S_x)$  in the utility register (Recall,  $S_x$  is defined in Section V). The auxiliary register stores the total vote count. Just before  $U'_W$ , the Aux register is in a basis state corresponding to the vote count of  $S_x$ .  $U'_W$  uses the auxiliary register as an input, and outputs whether the vote count is in the correct range for player  $i$  to be a deciding vote. Note that there is no  $+w_i$  gate activation, as  $S_x$  does not include player  $i$  by definition. Additional results, together with the simulation code, are available in a [companion GitHub repository](#) [4].

$w_1 = 2$ , and  $w_2 = 1$ . Thus, we can define  $W(S)$ ,  $S \in F \setminus \{i\}$ , for player  $i$  to be,

$$W(S) = W'(V(S))$$

where,

$$W'(Votes) = \begin{cases} 1 & \text{if } q - w_i \leq Votes < q, \\ 0 & \text{otherwise.} \end{cases}$$

Note that,  $W(S)$  is either 0 or 1. Thus, we can take  $W_{\max} = 1$ , so  $\hat{W}(x) = W(S_x)$ . Thus we can define  $U_W$  to be:

$$U_W |x\rangle |0\rangle = |x\rangle \otimes \left[ \left(1 - \hat{W}(x)\right) |0\rangle + \hat{W}(x) |1\rangle \right]$$

The quantum circuit for the scenario is shown in Figure 3.

With  $U_W$  defined, all other steps are entirely agnostic to voting games. Let  $\ell = 2$ , and suppose for simplicity's sake  $\epsilon = 0$ . This is not a realistic scenario, but it will demonstrate how quickly the expected value of the utility register converges. With these parameters, we get the following approximations for the Shapley values:

$$\tilde{\Phi}_0 \approx 0.6617, \quad \tilde{\Phi}_1, \tilde{\Phi}_2 \approx 0.1616$$

The direct calculation for Alice can be seen in the appendix.

To rigorously demonstrate efficacy, we performed many trials on random weighted voting games (Figure 4). Visual inspection confirms that, in most cases, when  $\ell$  is incremented, the error is divided by more than a factor of two (so the reciprocal more than doubles). As incrementing  $\ell$  implies doubling the work to prepare  $|\psi_1\rangle$ , we have empirically shown a linear or better relationship between complexity and error for Step 1 (Section V). So, the empirical error for this step is  $\mathcal{O}(a^{-1})$ .

Step 2 depends entirely on the game. However, if it is possible to implement on a classical computer, much like with Grover's algorithm, then it can be implemented in a quantum setting [7].

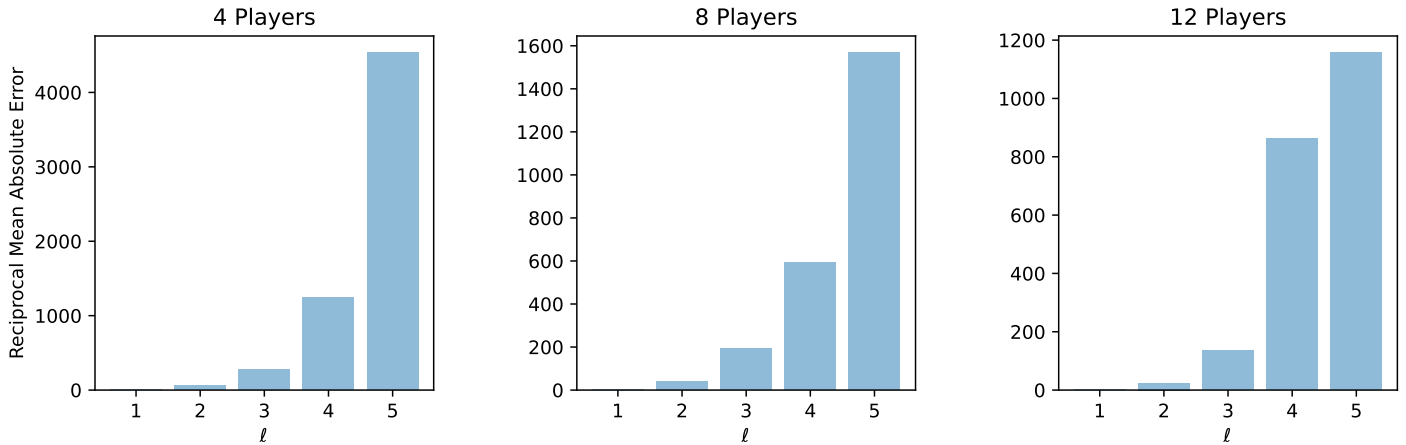


Fig. 4. We generated 32 random weighted voting games for each condition. We generated random weights  $w_j \in \mathbb{N}$  for each case such that  $q \leq \sum_j w_j < 2q$ . There were three primary scenarios: (1) Four players, voting threshold  $q = 8$ ; (2) Eight players, voting threshold  $q = 16$ ; and (3) 12 players, voting threshold  $q = 32$ . We approximated every player’s Shapley value for each scenario with our quantum algorithm using various  $\ell$ ’s, assuming  $\epsilon = 0$ . Next, we found the absolute error of our approximation by comparing each approximated Shapley value to its true value. Finally, we took the reciprocal of the mean for all Shapley value errors in each random game for each condition.

Step 3 is well studied, and there are two valid approaches to finding the expected value of measuring the utility register with error  $\epsilon$ . Either (i), repeat steps one and two of Section V and measure the utility register  $\mathcal{O}(\epsilon^{-2})$  times. Otherwise, proceed with (ii), use amplitude estimation as described in Section V with the techniques described in [14]. Note that (ii) results in an increase to the circuit depth by  $\mathcal{O}(\epsilon^{-1})$  times, but only needs to be repeated  $\mathcal{O}(1)$  times.

### VIII. CONCLUSION

We have addressed one of the main problems of generating post hoc explanations of ML models in the quantum context. More precisely, we have addressed the problem of efficiently generating post hoc explanations using the concept of the Shapley value. Under the XQML context, the challenge of explainability is amplified since measuring a quantum system destroys the information. Using the classical concept of Shapley values for post hoc explanations in quantum computing does not translate trivially. We have proposed novel algorithms to reduce the problem of accurately estimating the Shapley values of a quantum algorithm into the solved problem of estimating the expected value of a quantum algorithm. Finally, we have determined the efficacy of the algorithms by using empirical and simulation methods.

**Acknowledgments** — We thank Prof. Yuly Billig (Carleton University) for his notational suggestion regarding hamming distances. We would also like to thank a peer of Iain, Michael Ripa, for many productive conversations regarding this research. The research was supported by Natural Sciences and Engineering Research Council (NSERC) of Canada.

### REFERENCES

- [1] R. J. Aumann. Some non-superadditive games, and their Shapley values, in the Talmud. *International Journal of Game Theory*, 39:1–10, 2010.
- [2] L. Bertossi, J. Li, M. Schleich, D. Suci, and Z. Vagena. Causality-based explanation of classification outcomes. In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, pages 1–10, 2020.
- [3] I. Burge, M. Barbeau, and J. Garcia-Alfaro. A Quantum Algorithm for Shapley Value Estimation, arXiv:2301.04727, <https://doi.org/10.48550/arXiv.2301.04727>, March 2023.
- [4] I. Burge, M. Barbeau, and J. Garcia-Alfaro. Quantum Algorithms for Shapley Value Calculation [extended simulation github repository], github, <https://github.com/iain-burge/QuantumShapleyValueAlgorithm>, May 2023.
- [5] J. Castro, D. Gómez, and J. Tejada. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.
- [6] V. Giovannetti, S. Lloyd, and L. Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.
- [7] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.
- [8] S. Hart. Shapley value. In *Game theory*, pages 210–216. The New Palgrave. Palgrave Macmillan, London, 1989.
- [9] R. Heese, T. Gerlach, S. Mücke, S. Müller, M. Jakobs, and N. Pitkowski. Explaining quantum circuits with shapley values: Towards explainable quantum machine learning, arXiv: 2301.09138, <https://doi.org/10.48550/arXiv.2301.09138>, March 2023.
- [10] A. Kay. Tutorial on the quantikz package, arXiv: 1809.03842, <https://arxiv.org/abs/1809.03842>, March 2023.
- [11] S. Lipovetsky. Quantum-like data modeling in applied sciences. *Stats*, 6(1):345–353, 2023.
- [12] Y. Matsui and T. Matsui. NP-completeness for calculating power indices of weighted majority games. *Theoretical Computer Science*, 263(1-2):305–310, 2001.
- [13] F. Mercaldo, G. Ciarabella, G. Iadarola, M. Storto, F. Martinelli, and A. Santone. Towards explainable quantum machine learning for mobile malware detection and classification. *Applied Sciences*, 12(23):12025, 2022.
- [14] A. Montanaro. Quantum speedup of monte carlo methods. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2181):20150301, 2015.
- [15] M. Plesch and Č. Brukner. Quantum-state preparation with universal gate decompositions. *Physical Review A*, 83(3):032302, 2011.
- [16] K. Prasad and J. S. Kelly. NP-completeness of some problems concerning voting games. *International Journal of Game Theory*, 19(1):1–9, 1990.
- [17] K. A. Ross. *Elementary analysis*. Springer, 2013.



- [18] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.
- [19] M. Schuld and N. Killoran. Is quantum advantage the right goal for quantum machine learning? *PRX Quantum*, 3(3):030101, jul 2022.
- [20] P. Steinmüller, T. Schulz, F. Graf, and D. Herr. explainable ai for quantum machine learning. *arXiv preprint arXiv:2211.01441*, 2022.
- [21] G. Van den Broeck, A. Lykov, M. Schleich, and D. Suciú. On the tractability of SHAP explanations. *Journal of Artificial Intelligence Research*, 74:851–886, 2022.
- [22] S. Wallis. Binomial confidence intervals and contingency tests: mathematical fundamentals and the evaluation of alternative methods. *Journal of Quantitative Linguistics*, 20(3):178–208, 2013.
- [23] E. Winter. The Shapley value. *Handbook of game theory with economic applications*, 3:2025–2054, 2002.

#### APPENDIX

Quantum estimation of Alice’s Shapley value by hand. Let  $\ell = 2$ . Note that the Auxiliary register stores vote count. We begin with the state:

$$|00\rangle_{\text{Pt}} |00\rangle_{\text{Pl}} |000\rangle_{\text{Aux}} |0\rangle_{\text{Ut}}$$

We perform the first step of the algorithm from Section V, yielding:

$$\sum_{k=0}^3 w_2(k) |k\rangle_{\text{Pt}} \left[ (1 - t'_2(k)) |00\rangle_{\text{Pl}} + \sqrt{t'_2(k)(1 - t'_2(k))} |01\rangle_{\text{Pl}} + \sqrt{t'_2(k)(1 - t'_2(k))} |10\rangle_{\text{Pl}} + t'_2(k) |11\rangle_{\text{Pl}} \right] |000\rangle_{\text{Aux}} |0\rangle_{\text{Ut}}$$

Next, we tally the votes. This step is the first half of the circuit in Figure 3, up to and without including  $U'_W$ . We get,

$$\sum_{k=0}^3 w_2(k) |k\rangle_{\text{Pt}} \left[ (1 - t'_2(k)) |00\rangle_{\text{Pl}} |000\rangle_{\text{Aux}} + \sqrt{t'_2(k)(1 - t'_2(k))} |01\rangle_{\text{Pl}} |001\rangle_{\text{Aux}} + \sqrt{t'_2(k)(1 - t'_2(k))} |10\rangle_{\text{Pl}} |010\rangle_{\text{Aux}} + t'_2(k) |11\rangle_{\text{Pl}} |011\rangle_{\text{Aux}} \right] |0\rangle_{\text{Ut}}$$

Performing the remainder of  $U_W$  gives,

$$\sum_{k=0}^3 w_2(k) |k\rangle_{\text{Pt}} \left[ (1 - t'_2(k)) |00\rangle_{\text{Pl}} |000\rangle_{\text{Aux}} |0\rangle_{\text{Ut}} + \sqrt{t'_2(k)(1 - t'_2(k))} |01\rangle_{\text{Pl}} |001\rangle_{\text{Aux}} |1\rangle_{\text{Ut}} + \sqrt{t'_2(k)(1 - t'_2(k))} |10\rangle_{\text{Pl}} |010\rangle_{\text{Aux}} |1\rangle_{\text{Ut}} + t'_2(k) |11\rangle_{\text{Pl}} |011\rangle_{\text{Aux}} |1\rangle_{\text{Ut}} \right]$$

The expected value when measuring the utility register is  $\approx 0.6617$ , a close approximation for  $\Phi_0 = 2/3 = 0.6666\dots$ .