Received: 24 July 2023 | Revised: 12 December 2023 | Accepted: 8 January 2024

DOI: 10.1049/qtc2.12087

IET Quantum Communication

ORIGINAL RESEARCH

An efficient quantum algorithm for ensemble classification using bagging

Antonio Macaluso[1] | Luca Clissa[2,3] | Stefano Lodi[4] | Claudio Sartori[4]

[1]Agents and Simulated Reality Department, German Research Center for Artificial Intelligence (DFKI), Saarbruecken, Germany

[2]Department of Physics and Astronomy, University of Bologna, Bologna, Italy

[3]Istituto Nazionale di Fisica Nucleare (INFN), Bologna, Italy

[4]Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

Correspondence

Antonio Macaluso, German Research Center for Artificial intelligence (DFKI), Saarbruecken, Germany.
Email: antonio.macaluso@dfki.de

Funding information

German Ministry for Education and Research in the project QAI2-QAICO, Grant/Award Number: 13N15586; European Commission under NextGeneration EU programme, Grant/Award Numbers: PNRR-PE00000013-FAIR-Spoke8-CUP, J33C22002830006

Abstract

Ensemble methods aggregate predictions from multiple models, typically demonstrating improved accuracy and reduced variance compared to individual classifiers. However, they often come with significant memory usage and computational time requirements. A novel quantum algorithm that leverages quantum superposition, entanglement, and interference to construct an ensemble of classification models using bagging as an aggregation strategy is introduced. Through the generation of numerous quantum trajectories in superposition, the authors achieve B transformations of the training set with only $log(B)$ operations, allowing an exponential enlargement of the ensemble size while linearly increasing the depth of the corresponding circuit. Moreover, when assessing the algorithm's overall cost, the authors demonstrate that the training of a single weak classifier contributes additively to the overall time complexity, as opposed to the multiplicative impact commonly observed in classical ensemble methods. To illustrate the efficacy of the authors' approach, experiments on reduced real-world datasets utilising the IBM qiskit environment to demonstrate the functionality and performance of the proposed algorithm are introduced.

KEYWORDS

quantum computing, quantum computing techniques, quantum information

1 | INTRODUCTION

Quantum Computing (QC) can achieve performance orders of magnitude faster than the classical counterpart, with the possibility of tremendous speed-up of complex computational tasks [1–3]. Thanks to the quantum mechanical principles of superposition and entanglement, quantum computers can achieve vast amounts of parallelism without needing the multiple replicas of hardware required in a classical computer. One of the most relevant fields in which QC promises to make an impact in the near future is machine learning (ML). Quantum ML (QML) is a sub-discipline of quantum information processing devoted to developing quantum algorithms that learn from data in order to improve existing methods. However, being an entirely new field, QML comes with many open challenges [4].

In the context of supervised learning, ensemble methods stand out as a well-established approach, involving the combination of numerous simple models through averaging or voting rules to classify new examples. Despite the absence of a unified theory, there are various theoretical justifications for combining multiple learners, such as reducing prediction error by decreasing uncertainty in estimates or empirical evidence supporting the effectiveness of this approach [5, 6].

From an application perspective, ensemble methods play a pivotal role in addressing a wide array of real-world problems [7], spanning diverse domains like finance, healthcare, cybersecurity, and manufacturing. The integration of predictions from multiple models not only enhances accuracy but also fortifies overall robustness. The introduction of more efficient algorithms, including quantum algorithms, holds significant promise for further improving the performance of these applications. Significantly, when dealing with tabular data, ensemble methods demonstrate superiority even over advanced deep learning models [8]. This underscores the importance of ongoing advancements in ensemble methods, as these refinements can have a profound impact on a wide range of applications.

2 | BACKGROUND

When endeavouring to predict a target variable using any ML model, the primary contributors to the discrepancy between actual and predicted values, known as the Expected Prediction Error (EPE) [9], are noise, bias, and variance. The *noise* component, also referred to as *irreducible error*, represents the variance of the target variable around its true mean. This error arises from the inherent uncertainty within the data and remains unavoidable, regardless of the model's performance. In contrast, *bias* is associated with the specific learning technique employed and gauges how well the method aligns with the underlying problem. Finally, the *variance* component quantifies the variability of the learning method with respect to its expected value. Therefore, to enhance the performance of any ML technique, it becomes imperative to mitigate one or more of these components.

Ensemble learning aims to construct a prediction model by leveraging the strengths of a collection of simpler base models to reduce the EPE [10]. A crucial requirement for an ensemble to surpass its individual members is that the constituent models exhibit both accuracy, indicating an error rate better than random guessing, and diversity, implying that the models make different errors on the same data points [11].

Various approaches exist for constructing ensemble methods, each targeting a specific component of the EPE. In Boosting, a committee of weak learners is employed, with each iteration training a new weak learner based on the ensemble's error. This progressive improvement helps minimise bias by approximating the true population values. Randomisation methods involve estimating the individual base models using randomly perturbed training algorithms, sacrificing accuracy but reducing ensemble variance through the combination of numerous randomised models. Notably, this approach is applicable to stable learners as well, expanding its applicability to diverse methods. Another approach, Bagging, involves fitting the same model to different training sets, creating a committee of independent weak learners. The ensemble prediction is then obtained by averaging the individual votes. This approach reduces the EPE by mitigating the variance component, with larger ensembles leading to more significant reductions.

In practice, bagging involves generating multiple predictions $\hat{f}_1(x), \hat{f}_2(x), \ldots, \hat{f}_B(x)$ by training B different models on distinct training sets and subsequently averaging them to obtain a single model with reduced variance:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{T}\hat{f}b(x). \tag{1}$$

While the theoretical formulation lacks practicality due to the unavailability of multiple independent training sets, the bootstrap procedure [12] can be employed to address this issue. By repeatedly sampling from the available data, the bootstrap generates B different bootstrapped training sets on which the learning algorithm is trained to produce B distinct predictions $\hat{f}_b(x)$. Although the generated datasets are not independent, being derived from the same training set, empirical findings suggest that bagging still yields combined models that consistently outperform individual learners [5].

2.1 | Related works

Recently, the concept of quantum ensembles has been explored [13, 14] as the aggregation strategy. This approach involves three stages: (*i*) state preparation routine, (*ii*) parallel evaluation of quantum classifiers, and (*iii*) accessing the combined decision. Bayesian Model Averaging (BMA) is employed utilising a range of models with fixed parameters that span a significant portion of the parameter domain. While this approach eliminates the need to train individual classifiers, it relies on oracles with unspecified quantum gate formulations. Furthermore, BMA is not widely used in ML due to limited performance in real-world applications [6, 15].

Another QML algorithm based on ensemble methods is Quantum Boosting [16]. This approach aims to enhance the performance of a weak learner by simulating an adaptive boosting procedure [17], enabling the conversion of a weak learning algorithm into a strong one. Although quantum boosting allows achieving a quadratic improvement over classical AdaBoost, it assumes efficiently preparing multiple copies of quantum states that encode the training set and executing the Quantum Phase Estimation algorithm [18–20], which requires a fault-tolerant quantum computer.

Lastly, a quantum ensemble based on the bagging strategy has recently been investigated [21]. The authors propose a quantum approach to aggregate multiple diverse functions, mimicking the bagging prediction of classical ensemble methods using a quantum circuit. However, practical tests to assess the effectiveness of this approach through specific quantum routines have not been provided.

A common limitation observed in existing works on quantum ensembles is the absence of experiments conducted on realistic data. Notably, all the referenced works implement the proposed approach solely on synthetic data in just one instance, neglecting an analysis of its performance on non-linearly separable, realistic datasets. Such an analysis could provide insights into whether the performance of the quantum ensemble improves with an increase in ensemble size, as expected from the methodology of classical ensemble methods.

Moreover, with the exception of Quantum Boosting, none of the existing works address whether the proposed quantum ensemble outperforms existing classical ensembles in any way. Importantly, our focus is exclusively on end-to-end quantum approaches that offer advantages over their classical counterparts. We specifically exclude hybrid quantum-classical computation and parameterised quantum circuits from consideration. The evaluation of these approaches in terms of time complexity is not feasible, as they represent heuristic methods that must be assessed on specific problems at hand.

2.2 | Contribution

This work aims to present a quantum algorithm for addressing supervised classification problems through quantum ensembles. Specifically, we extend and refine the formulation of the quantum ensemble based on the bagging strategy [21] and provide a detailed description of the quantum ensemble's implementation.

The key idea is to design a quantum algorithm that propagates an input state to multiple quantum trajectories in superposition, yielding a sum of individual results from each trajectory. Technically, the algorithm generates diverse transformations of the training set in superposition, entangled with a quantum state of a control register. Subsequently, a quantum classifier F is applied to obtain numerous classifications in superposition. Averaging these predictions allows access to the ensemble prediction by measuring a single quantum register.

This architecture offers three major computational advantages. Firstly, the ensemble size (i.e., the number of simple base models) scales exponentially compared to classical methods, while the depth of the corresponding quantum circuit increases linearly. The proposed quantum ensemble requires only d steps to generate 2^d different transformations of the same training set in superposition. Secondly, entangled states provide additive effects from weak classifiers, contrary to the multiplicative burden typically encountered in classical implementations. Thus, the time cost of implementing the ensemble is not dominated by the individual classifier's cost but rather by the data encoding strategy. Thirdly, the number of state preparation routines matches that of a single classifier since the classification routine operates via interference and propagates to all quantum trajectories in superposition with just one execution. Furthermore, the algorithm enables accessing the ensemble prediction by measuring a single register, making the evaluation of large ensembles feasible with relatively small circuits.

Finally, we conduct experiments on both simulated and (reduced) real-world data, employing a straightforward classification routine based on cosine distance as the weak learner. However, it is important to note that the performed experiments have limitations compared to what is achievable with classical ensembles. This limitation arises due to the high computational requirements associated with the adoption of end-to-end quantum ensemble methods, particularly in terms of the number of qubits needed.

3 | QUANTUM ALGORITHM FOR CLASSIFICATION ENSEMBLE

In this section, we introduce the basic idea of our quantum algorithm for ensemble classification using bagging in the context of binary classification. The boosting and randomisation approaches are discussed in Section 3.1.

The algorithm adopts three quantum registers: data, control, and test. The *data* register encodes the training set, and it is employed together with the d-qubits *control* register to generate 2^d altered copies of the training set in superposition. The *test* register, instead, encodes unseen observations from the test set. Starting from these three registers, the algorithm involves four main steps: *state preparation*, *sampling in superposition*, *learning via interference* and *measurement*.

(Step 1) State Preparation

State preparation consists of the initialisation of the *control* register into a uniform superposition through a Walsh–Hadamard gate and the encoding of the training set (x, y) in the *data* register:

$$\begin{aligned}|\Phi_0\rangle &= \left(W \otimes S_{(x,y)}\right) \bigotimes_{i=1}^{d} |0\rangle \otimes |0\rangle \\ &= \left(H^{\otimes d} \otimes S_{(x,y)}\right) \bigotimes_{i=1}^{d} |0\rangle \otimes |0\rangle = \bigotimes_{i=1}^{d} |c_i\rangle \otimes |x, y\rangle,\end{aligned} \tag{2}$$

where $S_{(x,y)}$ is the state preparation routine for the training set, and it strictly depends on the encoding strategy, W is the Walsh–Hadamard gate and $|c_i\rangle$ is the ith qubit of the control register initialised into a uniform superposition between $|0\rangle$ and $|1\rangle$

(Step 2) Sampling in Superposition

The second step regards the generation of 2^d different transformations of the training set in superposition, each entangled with a state of the *control* register. To this end, d steps are necessary, where each step consists in the entanglement of the ith control qubit with two transformations of $|x, y\rangle$ based on two random unitaries, $U_{(i,1)}$ and $U_{(i,2)}$, for $i = 1, \ldots, d$. The most straightforward way to accomplish this is to apply the $U_{(i,j)}$ gate through controlled operations, using as control state the two basis states of the current control qubit. In particular, the generic ith step involves the following three transformations:

- First, the controlled-unitary $CU_{(i,1)}$ is executed to entangle the transformation $U_{(i,1)}|x, y\rangle$ with the excited state of the ith control qubit:

$$\begin{aligned}|\Phi_{i,1}\rangle &= \left(CU_{(i,1)}\right)|c_i\rangle \otimes |x, y\rangle \\ &= \left(CU_{(i,1)}\right)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |x, y\rangle \\ &= \frac{1}{\sqrt{2}}\left(|0\rangle|x, y\rangle + |1\rangle U_{(i,1)}|x, y\rangle\right)\end{aligned} \tag{3}$$

- Second, the i–th control qubit is transformed based on a Pauli–X gate:

$$|\Phi_{i,2}\rangle = (X \otimes \mathbb{1})|\Phi_{i,1}\rangle$$
$$= \frac{1}{\sqrt{2}}\left(|1\rangle|x,y\rangle + |0\rangle U_{(i,1)}|x,y\rangle\right) \tag{4}$$

- Third, a second controlled-unitary $CU_{(i,2)}$ is executed:

$$|\Phi_i\rangle = \left(CU_{(i,2)}\right)|\Phi_{i,2}\rangle$$
$$= \left(CU_{(i,2)}\right)\frac{1}{\sqrt{2}}\left(|1\rangle|x,y\rangle + |0\rangle U_{(i,1)}|x,y\rangle\right) \tag{5}$$
$$= \frac{1}{\sqrt{2}}\left(|1\rangle U_{(i,2)}|x,y\rangle + |0\rangle U_{(i,1)}|x,y\rangle\right).$$

These three transformations are repeated for each qubit in the control register and, at each iteration, two random $U_{(i,1)}$ and $U_{(i,2)}$ are applied. After d steps, the *control* and *data* registers are fully entangled, and 2^d different quantum trajectories in superposition are generated (more details are provided in the Appendix A). The output of this procedure can be expressed as follows:

$$|\Phi_d\rangle = \frac{1}{\sqrt{2^d}}\sum_{b=1}^{2^d}|b\rangle V_b|x,y\rangle = \frac{1}{\sqrt{2^d}}\sum_{b=1}^{2^d}|b\rangle|x_b,y_b\rangle \tag{6}$$

where V_b results from the product of d matrices $U_{(i,j)}$ and it represents a single quantum trajectory that differs from the others for at least one matrix $U_{(i,j)}$. In general, it is possible to refer to the unitary V_b as a unitary that transforms the original training set to obtain a random sub-sample of it:

$$|x,y\rangle \xrightarrow{V_b} |x_b,y_b\rangle. \tag{7}$$

The composition of V_b strictly depends on the encoding strategy chosen for data. In Section 4.1, we provide an example of $U_{(i,j)}$ based on the qubit encoding strategy, where a single observation is encoded into a qubit. Notice that the only requirement to perform ensemble learning using bagging effectively is that small changes in the product of the unitaries $U_{(i,j)}$ imply significant differences in (x_b, y_b), since the more independent sub-samples are, the better the ensemble works.

(Step 3) Learning via Interference

The third step of the algorithm is *Learning via Interference*. First, the *test* register is initialised to encode the test set, $x^{(\text{test})}$, also considering an additional register to store the final predictions:

$$(S_{x^{(\text{test})}} \otimes \mathbb{1})|0\rangle|0\rangle = |x^{(\text{test})}\rangle|0\rangle. \tag{8}$$

Then, the *data* and *test* registers interact via interference to compute the estimates of the target variable. To this end, we define a quantum classifier F that satisfies the necessary conditions described in Section 2. In particular, F acts on three registers to predict $y^{(\text{test})}$ starting from the training set (x_b, y_b):

$$|x_b,y_b\rangle|x^{(\text{test})}\rangle|0\rangle \xrightarrow{F} |x_b,y_b\rangle|x^{(\text{test})}\rangle|\hat{f}_b\rangle. \tag{9}$$

Thus, F represents the classification function $\hat{f}$ that estimates the value of the target variable of interest. For example, in binary classification problems, the prediction can be encoded into the probability amplitudes of a qubit, where the state $|0\rangle$ encodes one class and the state $|1\rangle$ the other.

The *Learning via Interference* step leads to

$$|\Phi_f\rangle = \left(\mathbb{1}^{\otimes d} \otimes F\right)|\Phi_d\rangle$$
$$= \left(\mathbb{1}^{\otimes d} \otimes F\right)\left[\frac{1}{\sqrt{2^d}}\sum_{b=1}^{2^d}|b\rangle|x_b,y_b\rangle\right] \otimes |x^{(\text{test})}\rangle|0\rangle$$
$$= \frac{1}{\sqrt{2^d}}\sum_{b=1}^{2^d}|b\rangle|x_b,y_b\rangle|x^{(\text{test})}\rangle|\hat{f}_b\rangle \tag{10}$$

where $\hat{f}_b$ represents the prediction for $x^{(\text{test})}$ given the bth training set, and it is implemented via quantum gate F. Notice that expressing the prediction according to Equation (10) implies that it is necessary to execute F only once in order to propagate its use to all the quantum trajectories. Furthermore, as a consequence of Steps 2 and 3, the *bth* state of the *control* register is entangled with the *bth* value of $\hat{f}$.

(Step 4) Measurement

Measuring the *test* register allows retrieving the average of the predictions provided by all the classifiers:

$$\langle\langle M\rangle\rangle = \langle\Phi_f|\mathbb{1}^{\otimes d} \otimes \mathbb{1} \otimes \mathbb{1} \otimes M|\Phi_f\rangle$$
$$= \frac{1}{2^d}\sum_{b=1}^{2^d}\langle *|\hat{f}_b|M\rangle\hat{f}_b = \frac{1}{2^d}\sum_{b=1}^{2^d}\langle M_b\rangle \tag{11}$$
$$= \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b = \hat{f}_{bag}\left(x^{(\text{test})}|x,y\right)$$

where $B = 2^d$ and M is a measurement operator (e.g., the Pauli-Z gate).

The expectation value $\langle M\rangle$ computes the ensemble prediction since it results from the average of the predictions of all the weak learners. Thus, if the two classes of the target variable are encoded in the two basis states of a qubit, it is possible to access the ensemble prediction by a single-qubit measurement:

$$\hat{f}_{bag} = \sqrt{a_0}|0\rangle + \sqrt{a_1}|1\rangle \tag{12}$$

where a_0 and a_1 are the average of the probabilities for $x^{(\text{test})}$ to be classified in class 0 and 1, respectively. The quantum circuit of the quantum ensemble is illustrated in Figure 1.

3.1 | Quantum algorithm for boosting and randomisation

The same framework presented above can be adapted with slight variations to allow for randomisation and boosting. The main principle of the ensemble based on randomisation consists of introducing casual perturbations that decorrelate the predictions of individual classifiers. In this case, it is possible to loosen the constraints imposed on the classifier F, which can be generalised beyond weak learners. The procedure described in Step 2 (*Sampling in Superposition*), indeed, can be employed to introduce a random component in the single learner, so as to decrease the accuracy of each individual model. As a consequence, the predictions are less correlated, and the variance of the final prediction is reduced.

Technically, it is necessary to define a classification routine that can be decomposed into the product of V_b and F. Here, the different trajectories do not simulate the bootstrap procedure as for bagging, but they are part of the classification routine and introduce randomisation in the computation of $\hat{f}$. In practice, we define a unitary G_b that performs the following transformation:

$$|x,y\rangle|x^{(\text{test})}\rangle|0\rangle \xrightarrow{G_b} |x,y\rangle|x^{(\text{test})}\rangle|\hat{f}_b\rangle, \tag{13}$$

where $G_b = V_bF$ is the quantum classifier composed of F—common to all the classifiers—and V_b, which is its random component—different for each quantum trajectory. This formulation allows rewriting Equation (10) asfollows:

$$\begin{aligned} |\Phi_f\rangle &= \frac{1}{\sqrt{2^d}}\sum_{b=1}^{2^d}|b\rangle G_b|x,y\rangle|x^{(\text{test})}\rangle|0\rangle \\ &= \frac{1}{\sqrt{2^d}}\sum_{b=1}^{2^d}|b\rangle|x,y\rangle|x^{(\text{test})}\rangle|\hat{f}_b\rangle. \end{aligned} \tag{14}$$

Similarly, the proposed framework can also be adapted for boosting, where the estimates provided by the single classifiers are weighted so that individual models do not contribute equally to the final prediction. In practice, the only difference is that the amplitudes of the control register need to be updated as the computation evolves. As a result, the output of a quantum ensemble based on boosting can be described as follows:

$$|\Phi_f\rangle = \frac{1}{\sqrt{2^d}}\sum_{b=1}^{2^d}\alpha_b|b\rangle|\hat{f}_b\rangle, \tag{15}$$

where the contribution of $\hat{f}_b$ to the ensemble depends on α_b. However, although in principle this approach fits in the scheme of a boosting ensemble, the difficulty in updating the *control* register is non-trivial.

To summarise, the main difference between quantum bagging and the other approaches is the way we define the unitaries $U_{(i,j)}$ and F. However, the exponential growth that comes from the advantage of generating an ensemble of $B = 2^d$ classifiers in only d steps still holds.

3.2 | Aggregation strategy and theoretical performance

When considering classical implementations of ensemble algorithms, it is possible to distinguish two broad families of methods based on the strategy adopted to aggregate the predictions of the individual models. The most popular technique used for ensemble classification is *majority voting*, where each

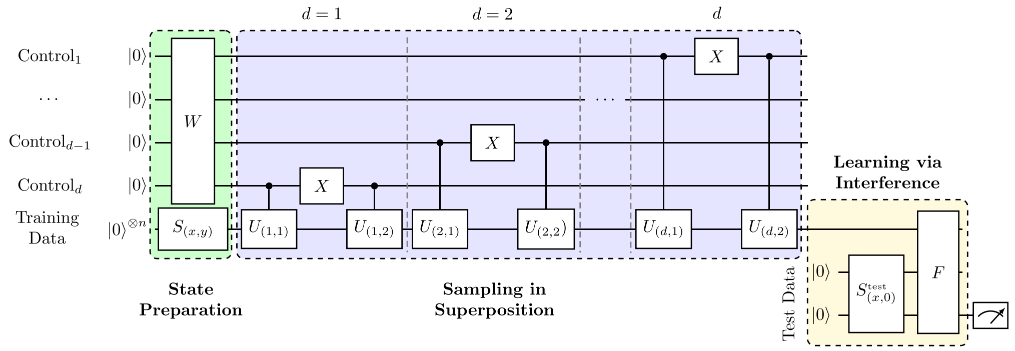


FIGURE 1 Quantum algorithm for ensemble classification. The circuit contains d pairs of unitaries $U_{(i,1)}$, $U_{(i,2)}$ and d control qubits. It produces an ensemble of B classifiers, where $B = 2^d$. The single evaluation of F allows propagating the classification function $\hat{f}$ in all trajectories in superposition. The first d steps allow generating B transformations of the training set (x, y) in superposition, and each transformation is entangled with a quantum state of the *control* register (firsts d qubits). Thus, the test set $x^{(\text{test})}$ is encoded in the *test* register that interferes with all samples in superposition. Finally, the ensemble prediction is obtained as the average of individual results from each trajectory.

classifier votes for a target class, and the most frequent is then selected. An alternative strategy is given by *simple averaging*. In this case, the target probability distribution provided by individual models is considered, and the final prediction is computed as follows:

$$f_{\text{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B} f_b^{(i)}(x), \tag{16}$$

where B is the ensemble size, and $f_b^{(i)}(x)$ is the probability for x to be classified in the ith class provided by the bth classifier. This approach allows a reduction of the variance of the estimate [22] and has shown good performance even for large and complex datasets [23].

In particular, the error E_{ens} of an ensemble obtained by averaging B individual learners can be expressed as [24, 25] follows:

$$E_{\text{ens}} = \frac{1+\rho(B-1)}{B} E_{\text{model}}, \tag{17}$$

where E_{model} is the expected error of the single models, and ρ is the average correlation among their errors.

Hence, the more independent the single classifiers are, the greater the error reduction due to averaging. A graphical illustration of the theoretical performance of an ensemble as a function of B, ρ, and E_{model} is reported in Figure 2.

Regarding our implementation of the quantum ensemble, the prediction of the single classifier is encoded into the probability amplitudes of a quantum state, and the final prediction is computed by averaging the results of all quantum trajectories in superposition. Implicitly, this means that the quantum ensemble fits the simple averaging strategy. Thus, the possibility to generate exponentially larger ensembles at the cost of increasing linearly the number of control qubits d allows the quantum ensemble to significantly improve the performance of the single classifier (Figure 2) using a relatively small circuit ($d \sim 10$).

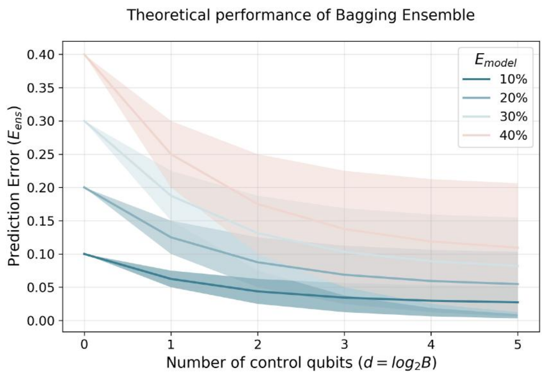


FIGURE 2 Theoretical performance of the quantum ensemble based on the expected prediction error of the base classifiers (E_{model}) and their average correlation (ρ). The ensemble size depends on the number of qubits d in the control register. Each solid line corresponds to an error level, with coloured bands obtained by varying ρ between 0 (lower edge) and 0.5 (upper edge).

3.3 | Computational complexity

Classically, given a number B of base learners and a dataset (x_i, y_i) for $i = 1, \ldots, N$, where x_i is a p-dimensional vector and y_i is the target variable of interest, the overall time complexity for training an ensemble based on randomisation or bagging scales at least linearly with respect to B and polynomially in p and N:

$$\underbrace{\mathcal{O}(BN^{\alpha}p^{\beta})}_{\text{Training}} + \underbrace{\mathcal{O}(Bp)}_{\text{Testing}} \qquad \alpha, \beta \geq 1,$$

where α and β depend on the single base model, and $N^{\alpha}p^{\beta}$ is its training cost. In boosting, instead, the model evolves over time, and the individual classifiers are not independent. This usually implies higher time complexity and less parallelism.

Despite this clear definition of the computational cost, comparing the classical algorithm to its quantum counterpart is not straightforward since they belong to different classes of complexity. For this reason, we benchmarked the two approaches by looking at how they scale in terms of the parameters of the ensemble, that is, the ensemble size B and the cost of each base model. In particular, this results in considering the Boolean circuit model [26] for the classical ensemble and the depth of the corresponding quantum circuit for the quantum algorithm.

In light of this definition, the quantum algorithm described in Section 3 can generate an ensemble of size $B = 2^d$ in only d steps. This means that assuming a unitary cost for each step, we are able to increase exponentially the size of the ensemble while increasing linearly the depth of the corresponding quantum circuit. Furthermore, the cost of the single classifier is additive—instead of multiplicative as in classical ensembles—since it is necessary to execute the quantum classifier F only once to propagate its application to all quantum trajectories in superposition, as shown in Equation (10). In addition, the cost of the state preparation routine is equivalent to any other quantum algorithm for processing the same training and test sets. However, this comparison does not take into account the additional cost due to state preparation, which is not present in classical ensembles. Also, the quantum ensemble comes with an extra cost related to the implementation of the gates $U_{(i,j)}$, which strictly depends on the encoding strategy chosen for the data and needs to be evaluated for any specific implementation.

4 | EXPERIMENTS

To test how our framework for quantum ensemble works in practice, we implemented the circuit illustrated in Figure 1 using IBM Qiskit [27]. Then, we conducted experiments on simulated and real-world datasets to show that (i) one execution of a quantum classifier allows retrieving the ensemble prediction and that (ii) the ensemble outperforms the single model. Importantly, all the experiments in this study replicate the bagging strategies of classical ensemble methods in quantum settings by adopting the methodological formulation described in Section 3. Employing other ensemble strategies based on the proposed quantum approach would necessitate distinct circuit designs.

4.1 | Quantum cosine classifier

In order to implement the quantum ensemble, a classifier that fulfils the conditions in Equation (9) is necessary. For this purpose, we define a simple routine for classification based on the swap test [28] that stores the cosine distance between two vectors into the amplitudes of a quantum state. This metric describes how similar two vectors are depending on the angle that separates them, irrespective of their magnitude. The smaller the angle between two objects, the higher the similarity.

Starting from this, the high-level idea is predicting a similar target class for similar input features. In particular, for any test observation $(x^{(\text{test})}, y^{(\text{test})})$ we take one training point $(x^{(\text{train})}, y^{(\text{train})})$ at random and we express the probability of $y^{(\text{test})}$ and $y^{(\text{train})}$ being equal as a function of the similarity between $x^{(\text{test})}$ and $x^{(\text{train})}$:

$$\Pr\left(y^{(\text{test})} = y^{(\text{train})}\right) = \frac{1}{2} + \frac{\left[d\left(x^{(\text{train})}, x^{(\text{test})}\right)\right]^2}{2} \tag{18}$$

where $d(.,.)$ is the cosine distance between $x^{(\text{train})}$ and $x^{(\text{test})}$. Thus, the final classification rule becomes

$$y^{(\text{test})} = \begin{cases} y^{(\text{train})}, & \text{if } \Pr\left(y^{(\text{test})} = y^{(\text{train})}\right) > \frac{1}{2} \\ 1 - y^{(\text{train})}, & \text{otherwise} \end{cases} \tag{19}$$

Notice that, by definition, $\Pr\left(y^{(\text{test})} = y^{(\text{train})}\right)$ is bounded in $\left[\frac{1}{2}, 1\right]$, which means that Equation (19) estimates the same class as the training point unless $x^{(\text{train})}$ and $x^{(\text{test})}$ are orthogonal. As a consequence, the cosine classifier performs well only if the test and training observations happen to belong to the same target class.

The quantum circuit that implements the cosine classifier is reported in Figure 3.

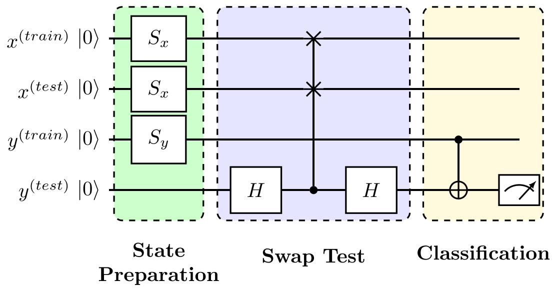


FIGURE 3 Quantum circuit of the cosine classifier using $x^{(\text{train})}$ as a training vector and $x^{(\text{test})}$ as test vector. The training label $y^{(\text{train})}$ is either $|0\rangle$ or $|1\rangle$ based on the binary target value. The measurement of the qubit $y^{(\text{test})}$ provides the prediction for the test observation whose features are encoded in $x^{(\text{test})}$.

It encodes data into three different registers: the training vector $x^{(\text{train})}$, the training label $y^{(\text{train})}$, and the test point $x^{(\text{test})}$. An additional qubit is then used to store the prediction. The algorithm consists of three steps. First, data are encoded into three different quantum registers through a routine S. Second, the swap test transforms the amplitudes of the qubit $y^{(\text{test})}$ as a function of the squared cosine distance. In particular, after the execution of the swap test, the probability of getting the basis state $|0\rangle$ is between $1/2$ and 1; hence, the probability of class 0 is never lower than the probability of class 1. Third, a controlled Pauli-X rotation is applied using the label of the training vector as a control qubit. This implies that $y^{(\text{test})}$ is left untouched if $x^{(\text{train})}$ belongs to class 0. Otherwise, the amplitudes of the $y^{(\text{test})}$ qubit are inverted, and $\Pr(y^{(\text{test})} = 1)$ becomes higher as the similarity between the two vectors increases (Figure 4).

Thus, the quantum cosine classifier performs classification via interference and allows calculating the probability of belonging to one of the two classes by single-qubit measurement. A detailed description of the quantum cosine classifier is provided in Appendix B. Furthermore, it is a weak method with high variance, since it is sensitive to the random choice of the training observation. In addition, it requires data to be encoded using qubit encoding, where a dataset with N 2-dimensional observations $x^{(\text{train})}$ is stored into N different qubits. This allows the definition of $U_{(i,j)}$ for the quantum ensemble in terms of random swap gates that move observations from one register to another. All these features make this classifier a good candidate for ensemble methods.

Employing the swap test as a subroutine in the quantum cosine classifier implies a linear scaling of the number of qubits with the number of observations in both the training and test sets. This limitation constrains the feasibility of conducting experiments and comparing the proposed quantum approach against classical methods, which can readily handle large amounts of data. For instance, implementing an ensemble of size 2, with a training set consisting of 2 data points and a single data point for testing, would necessitate a total of 7

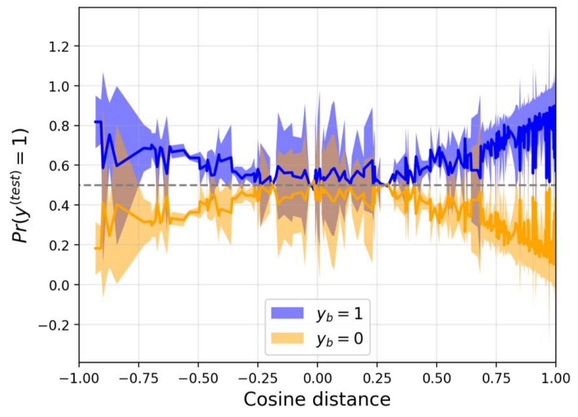


FIGURE 4 Predictions of the cosine distance classifier based on 10^3 randomly generated datasets per class. The classifier is implemented using the circuit in Figure 3 on a 7-qubit quantum device (*ibmq casablanca*). The implementation assuming a perfect quantum device is reported in Appendix B, Figure B1.

qubits (refer to Figure C1). This breakdown includes 1 qubit for the control register, 4 qubits for the training set (two for the features of each training point and two for encoding their respective labels), and an additional two qubits for the test point. A graphical description of different ensembles of the quantum cosine classifier is reported in Appendix C.

4.2 | Quantum ensemble as simple averaging

As a proof of concept for the quantum ensemble based on bagging, we consider different 20 random generated datasets, each containing four training points (2-dim features and label) and one test example. For every simulated dataset, each training point is used as a training observation and fed into a quantum cosine classifier as input so to provide an estimate for a test observation $x^{(\text{test})}$. Thus, the quantum ensemble, which requires only one execution of the quantum cosine classifier is executed. The quantum circuit of the ensemble uses two qubits in the *control* register ($d = 2$) and eight in the *data* register, four for the training vectors x_b and four for training labels y_b. Two additional qubits are then used for the test observation, $x^{(\text{test})}$ and the final prediction. Notice that the four matrices $U_{(i,j)}$ need to be fixed to guarantee that each quantum trajectory V_b described in Section 3 provides the prediction of different and independent training points.

For each training point, the quantum cosine classifier is implemented, and then a prediction for the test point is calculated. This small experiment aims to prove that the quantum ensemble prediction is exactly the average of the values of all trajectories in superposition and it can be obtained with just one execution of the classification routine.

Results are shown in Figure 5. The agreement between the quantum ensemble (orange line) and the average (brown dots) is almost perfect, which confirms the possibility of implementing a quantum ensemble with the advantages described in Section 3 in a fault-tolerant setting. Results considering the real device (light blue line) show slight deterioration, and this may be due to the depth of the quantum circuit which seems to be prohibitive considering the currently available gate-based quantum technology.

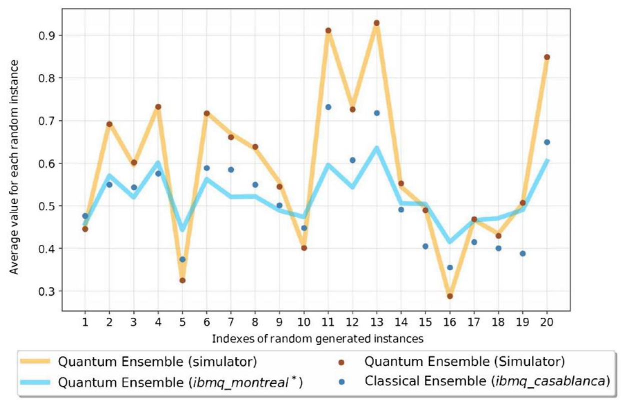


FIGURE 5 Comparison between the Quantum Ensemble and the Classical Ensemble as a result of the classical average of four quantum cosine classifiers executed separately. Both approaches are performed on a simulator (orange line, brown dots) and on a real device (light blue line, blue dots). Importantly, the Quantum Ensemble implementation on a real device is performed using noisy simulations of the specific quantum device (*ibmq_16_montreal**). These simulations are only an approximation of the real errors that occur on actual devices but still allow us to test the effectiveness of the quantum ensemble on near-term devices. The details about the implementation in terms of quantum gates of the quantum ensemble are reported in Appendix A.

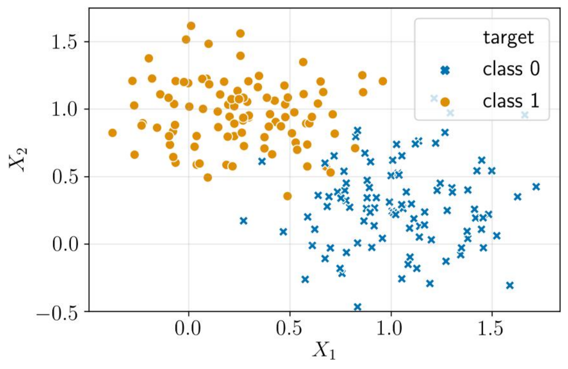


FIGURE 6 Dataset generated by two independent bivariate Gaussian distributions. Mean vectors for the two classes are (1, 0.3) and (0.3, 1). The two distributions have the same diagonal covariance matrix, with a constant value of 0.3.

4.3 | Performance of the quantum ensemble in a fault-tolerant setting

To show that the quantum ensemble outperforms the single classifier, we generated a simulated dataset and compared the performance of the two models. In particular, we drew a random sample of 200 observations (100 per class) from two independent bivariate Gaussian distributions, with different mean vectors and the same covariance matrix (Figure 6). Then, we used the 90% of the data for training and the remaining 10% for testing.

Notice that the definition of the cosine classifier implies executing the classification routine once for each test point (See Appendix A, C for more details). We considered two performance metrics, accuracy and Brier Score (BS). The accuracy is the fraction of labels predicted correctly by the quantum model, and it is evaluated using a set of observations not employed for training (test set). Instead, the BS measures the difference between the probability estimates and the true label in terms of mean squared error:

$$BS = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} [y_i - f(x_i)]^2, \tag{20}$$

where N_{test} is the number of observations in the test set, y_i and $f(x_i)$ are, respectively, the true label and the probability estimates provided by the quantum model for the *ith* observation (for quantum ensemble see Equation (11)). Hence, a low BS score implies a good prediction. Due to the randomness introduced by the choice of training points, we repeated the experiments 10 times and evaluated the classifiers in terms of the mean and standard deviation of both accuracy and BS. The experiments of this section are all run assuming a perfect quantum device. Results are shown in Table 1. The single quantum cosine classifier performed only slightly better than random guessing, with an average accuracy of 55%. Yet, the quantum ensemble managed to achieve definitely better results, with both metrics improving as the ensemble size grows.

4.3.1 | Variance analysis

In addition, we investigated how the quantum ensemble behaves as the generated distributions get closer and less separated. To this end, we drew multiple samples from the two distributions, each time increasing the common standard deviation to force reciprocal contamination. Results are reported in Figure 7.

The accuracy showed a decreasing trend as the overlap of the distributions increased. The opposite behaviour is observed for the BS. Also, the shape of the boxplots is much narrower for greater ensemble sizes (green and red boxplots) than for smaller ones (blue and orange). Hence, this confirms that the variability of the ensemble decreases as the number of weak learners adopted grows as expected.

TABLE 1 Performance comparison between the quantum cosine classifier and quantum ensemble of different sizes $B = 2^d$. The first row indicates the performance of the single quantum cosine classifier. The column N indicates the number of training points used to build the ensemble, which is limited to 8 because of the restricted number of qubits that is possible to simulate.

			Accuracy		Brier score	
D	**B**	***N***	**Mean**	**Std dev**	**Mean**	**Std dev**
0	1	1	0.55	0.09	0.21	0.05
1	2	2	0.92	0.09	0.14	0.09
2	4	4	0.91	0.09	0.15	0.05
3	8	8	0.96	0.04	0.14	0.04
4	16	8	0.98	0.02	0.13	0.02

4.4 | Benchmark on real-world datasets

In this section, we test the performance of the quantum ensemble on real-world datasets that are usually employed to benchmark classical ML algorithms. Importantly, due to the constraints of the proposed algorithm and the restrictions of current quantum simulators in Qiskit, conducting extensive experimental evaluations with large datasets is not feasible. Consequently, introducing a larger dataset could yield different results, particularly in the case of imbalanced data. Nonetheless, the experiments are designed to illustrate that the enhancements affirmed by adopting ensemble methods from classical ML literature apply to the proposed quantum ensemble. It is important to note that a direct comparison with classical ensembles would necessitate larger quantum simulations.

4.4.1 | Datasets description

The simulation of a quantum system on a classical device is a challenging task, even for systems of moderate size. For this reason, experiments consider only datasets with a relatively small number of observations (100–150) that will be split in training (90%) and test (10%) set. Furthermore, to restrict the total number of qubits, Principal Components Analysis is conducted to reduce the feature dimensionality to 2. Each training feature is then normalised between 0 and 1 to be encoded in the amplitudes of a quantum state, as mandated by the quantum cosine classifier.

For all the datasets, a given subset of training points (N) is encoded in the *data* register, and then the prediction is retrieved by measuring the label qubit of the test register. This is performed for each test point and finally, the test error in

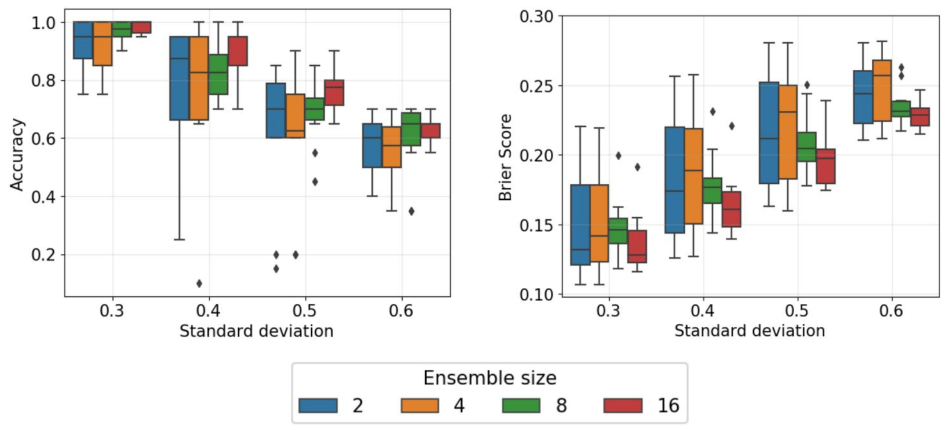


FIGURE 7 Distribution of the performance metrics as a function of the ensemble size (legend colours) and the separation between the two classes (x axis).

terms of Accuracy and BS is considered to evaluate the generalisation error of the quantum model.

MNIST

It is a large dataset of handwritten digits that is commonly used to benchmark various image processing systems. In particular, it is usually employed for training and testing different algorithms in the field of ML. The original dataset contains 60.000 black and white images, each represented by 28 × 28 pixels. Thus, the single image can be described as a vector of binary 784 features (0 if the pixel is white and 1 if it is black). Also, each image belongs to a class of 10 possible that represents the digit depicted in the image. The current implementation of the cosine classifier is arranged to solve a binary classification problem. Hence, only two different classes will be considered, the digits 0 and 9.

Iris

The Iris flower data set collects the data to quantify the morphologic variation of Iris flowers of three related species [29]. The data set consists of 50 examples from each of three species of Iris (Iris *setosa*, Iris *virginica* and Iris *versicolor*). Four features describe each observation: the length and the width of the sepals and petals in centimetres. The dataset is often used in statistical learning theory as classification and clustering examples to test algorithms. Since the current implementation of the quantum ensemble solves a binary classification problem, only two species in the dataset will be considered (three different datasets in total).

4.4.2 | Results

The results of the quantum ensemble on the real-world datasets are reported in Table 2. For each dataset, the quantum ensemble is implemented simulating a perfect quantum device.

Comparing the results in terms of the ensemble size ($B = 2^d$), it is possible to observe a decreasing trend of the BS and an increasing trend for accuracy. This confirms the ability of the quantum ensemble to improve the performance of the single quantum classifier. However, the quantum ensemble does not achieve good performance in the case of the dataset *Iris (1* vs. *2)*. Importantly, all the experiments have been conducted by considering perfectly balanced datasets since the goal was to show that building a quantum ensemble improves the performance in terms of the accuracy of a given weak classifier (as it happens for a classical ensemble). However, to make the proposed quantum algorithm effective for real-world problems, it is necessary to design a better quantum routine for classification that behaves as a weak classifier and is not dependent on the training set at hand.

5 | CONCLUSION AND OUTLOOK

In this paper, we propose a quantum framework for binary classification using ensemble learning. The correspondent algorithm allows generating a large number of trajectories in superposition, performing just one state preparation routine. Each trajectory is entangled with a quantum state of the control register and represents a single classifier. This convenient design allows scaling exponentially the number of base models with respect to the available qubits in the control register ($B = 2^d$). As a consequence, we can obtain an exponentially large number of classifications while increasing only linearly the depth of the correspondent quantum circuit with respect to the size of the control register. Furthermore, when considering the overall time complexity of the algorithm, the cost of the weak classifier is additive, instead of multiplicative as it usually happens.

In addition, we present a practical implementation of the quantum ensemble using bagging where the quantum cosine classifier is adopted as a base model. In particular, we show experimentally that the ensemble prediction corresponds to the average of all the probabilities estimated by the single classifiers. Moreover, we test our algorithm on synthetic and real-world (reduced) datasets and demonstrate that the quantum ensemble systematically outperforms the single classifier. Also, the variability of the predictions decreases as we add more base models to the ensemble.

However, the current proposed implementation requires the execution of the classifier for just one test point at the time, which is a big limitation for real-world applications. In this respect, the main challenge to tackle in order to make the framework effective in the near future is the design of a quantum classifier based on interference that guarantees a more efficient data encoding strategy (e.g., amplitude encoding) and that is able to process larger datasets. Nevertheless, these upgrades would imply a different definition of $U_{(i,j)}$ for the generation of multiple and diverse training sets in

TABLE 2 Performance of the quantum ensemble on real-world datasets.

		Iris (0 vs. 1)		**Iris (0 vs. 2)**		**Iris (1 vs. 2)**		**MNIST**	
D	**B/No**	**Accuracy**	**BS**	**Accuracy**	**BS**	**Accuracy**	**BS**	**Accuracy**	**BS**
0	1	0.49	0.284	0.49	0.284	0.49	0.445	0.50	0.337
1	2	1.0	0.137	1.0	0.276	0.51	0.240	0.79	0.209
2	4	1.0	0.138	1.0	0.139	0.52	0.240	0.78	0.208
3	8	1.0	0.136	1.0	0.138	0.61	0.241	0.84	0.197

superposition. Further, the design of a more accurate base quantum model is necessary.

Another natural follow-up is the implementation of quantum algorithms for randomisation and boosting. In this work, we only referred to an ensemble based on the bagging because the learning step was performed independently in each quantum trajectory and the weak classifiers were assumed to be sensitive to perturbations of the training set. However, with appropriate amendments and the loosening of these constraints, we believe that it is possible to design other types of ensemble techniques.

Finally, it is important to notice that the idea of model aggregation has already been adopted in the context of variational quantum algorithms to build the quantum Single Layer Perceptron [30, 31]. Thus, it is natural to consider the provided quantum architecture for the quantum ensemble as a natural extension of it if adopted adequately in hybrid quantum-classical computation. Although some challenges still remain, we believe this work is a good practical example of how ML, in particular ensemble classification, could benefit from QC.

AUTHOR CONTRIBUTIONS

Antonio Macaluso: Conceptualization; formal analysis; investigation; methodology; writing—original draft; writing–review & editing. **Luca Clissa**: Validation; visualization; writing–original draft. **Stefano Lodi**: Supervision. **Claudio Sartori**: Supervision.

ACKNOWLEDGEMENTS

We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Quantum team. We acknowledge the access to advanced services provided by the IBM Quantum Researchers Programme. This work has been partially funded by the German Ministry for Education and Research (BMB + F) in the project QAI2-QAICO under grant 13N15586. Also, this research has been partly funded by PNRR- M4C2 Investimento 1.3, Partenariato Esteso PE00000013 - “FAIR - Future Artificial Intelligence Research” - Spoke 8 “Pervasive AI”, funded by the European Commission under the NextGeneration EU programme.

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no conflict of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available at github.com/amacaluso/Quantum-Ensemble-for-Classification.

ORCID

Antonio Macaluso https://orcid.org/0000-0002-1348-250X

REFERENCES

1. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, pp. 212–219 (1996)
2. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. 41(2), 303–332 (1999). https://doi.org/10.1137/s0036144598347011
3. Harrow, A.W., Hassidim, A., Lloyd, S.: Quantum algorithm for linear systems of equations. Phys. Rev. Lett. 103(15), 150502 (2009). https://doi.org/10.1103/PhysRevLett.103.150502
4. Aaronson, S.: Read the fine print. Nat. Phys. 11(4), 291–293 (2015). https://doi.org/10.1038/nphys3272
5. Tumer, K., Ghosh, J.: Error correlation and error reduction in ensemble classifiers. Connect. Sci. 8(3-4), 385–404 (1996). https://doi.org/10.1080/095400996116839
6. Domingos, P.: Bayesian averaging of classifiers and the overfitting problem. In: ICML, vol. 2000, pp. 223–230 (2000)
7. Okun, O., Valentini, G., Re, M.: Ensembles in Machine Learning Applications, vol. 373. Springer Science & Business Media (2011)
8. Grinsztajn, L., Oyallon, E., Varoquaux, G.: Why do tree-based models still outperform deep learning on typical tabular data? Adv. Neural Inf. Process. Syst. 35, 507–520 (2022)
9. Hastie, T., et al.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, vol. 2. Springer (2009)
10. Hastie, T., et al.: The elements of statistical learning: data mining, inference and prediction. Math. Intel. 27(2), 223–224 (2005)
11. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Trans. Pattern Anal. Mach. Intell. 12(10), 993–1001 (1990). https://doi.org/10.1109/34.58871
12. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. CRC press (1994)
13. Schuld, M., Petruccione, F.: Quantum ensembles of quantum classifiers. Sci. Rep. 8(1), 2772 (2018). https://doi.org/10.1038/s41598-018-20403-3
14. Abbas, A., Schuld, M., Petruccione, F.: On quantum ensembles of quantum classifiers. Quan. Mach. Intell. 2(1), 1–8 (2020). https://doi.org/10.1007/s42484-020-00018-6
15. Domingos, P.M.: Why does bagging work? A Bayesian account and its Implications. In: KDD Citeseer, pp. 155–158 (1997)
16. Arunachalam, S., Maity, R.: Quantum boosting. arXiv preprint (2020). arXiv:200205056
17. Freund, Y., Schapire, R., Abe, N.: A short introduction to boosting. J. Jpn. Soc. Artif. Intell. 14(771-780), 1612 (1999)
18. Luis, A., Peřina, J.: Optimum phase-shift estimation and the quantum description of the phase difference. Phys. Rev. A 54(5), 4564–4570 (1996). https://doi.org/10.1103/physreva.54.4564
19. Cleve, R., et al.: Quantum algorithms revisited. Proc. R. Soc. Lond. Ser. A 454(1969), 339–354 (1998). https://doi.org/10.1098/rspa.1998.0164
20. Bužek, V., Derka, R., Massar, S.: Optimal quantum clocks. Phys. Rev. Lett. 82(10), 2207–2210 (1999). https://doi.org/10.1103/physrevlett.82.2207
21. Macaluso, A., Lodi, S., Sartori, C.: Quantum algorithm for ensemble learning. In: ICTCS, pp. 149–154 (2020)
22. Tumer, K., Ghosh, J.: Analysis of decision boundaries in linearly combined neural classifiers. Pattern Recogn. 29(2), 341–348 (1996). https://doi.org/10.1016/0031-3203(95)00085-2
23. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Trans Syst Man Cybern. 22(3), 418–435 (1992). https://doi.org/10.1109/21.155943
24. Jacobs, R.A.: Methods for combining experts' probability assessments. Neural Comput. 7(5), 867–888 (1995). https://doi.org/10.1162/neco.1995.7.5.867
25. Oza, N.C., Tumer, K.: Classifier ensembles: select real-world applications. Inf. Fusion 9(1), 4–20 (2008). https://doi.org/10.1016/j.inffus.2007.07.002
26. Arora, S., Barak, B.: Computational Complexity: A Modern Approach. Cambridge University Press (2009)
27. Abraham, H., et al.: Qiskit: An Open-Source Framework for Quantum Computing (2019)
28. Buhrman, H., et al.: Quantum fingerprinting. Phys. Rev. Lett. 87(16), 167902 (2001). https://doi.org/10.1103/PhysRevLett.87.167902

29. Anderson, E.: The species problem in Iris. Ann. Mo. Bot. Gard. 23(3), 457–509 (1936). https://doi.org/10.2307/2394164
30. Macaluso, A., et al.: A variational algorithm for quantum neural networks. In: International Conference on Computational Science, pp. 591–604. Springer (2020)
31. Macaluso, A., et al.: A variational algorithm for quantum single layer Perceptron. In: International Conference on Machine Learning, Optimization, and Data Science, pp. 341–356. Springer (2022)

How to cite this article: Macaluso, A., et al.: An efficient quantum algorithm for ensemble classification using bagging. IET Quant. Comm. 5(3), 253–268 (2024). https://doi.org/10.1049/qtc2.12087

Appendix A

QUANTUM ENSEMBLE AS SIMPLE AVERAGING

Here, we describe the quantum circuit to obtain four *independent* quantum trajectories in superposition considering a quantum ensemble of cosine classifiers (Section 4.2).

(Step 1) State Preparation

For a 2-qubit *control* register ($d = 2$), we can build an ensemble of $B = 4$ classifiers. The *data* encodes a single observation using a single qubit. In particular, given a dataset made up of N observations $\{x_i, y_i\}_{i=1,\dots,N}$, where $x_i = (x_{i,1}, x_{i,2})$ is a 2-dimensional vector and $y_i \in \{0, 1\}$ is the binary target variable, the *data* register encodes N training points $2 \times N$ qubits:

$$\underbrace{\left(\bigotimes_{i=1}^{4}|x_i\rangle\right)}_{features} \otimes \underbrace{\left(\bigotimes_{i=1}^{4}|y_i\rangle\right)}_{labels} \tag{A1}$$

where the values $x_{i,1}$ and $x_{i,2}$ are encoded into the amplitudes of a single qubit:

$$|x_i\rangle = x_{i,1}|0\rangle + x_{i,2}|1\rangle, \tag{A2}$$

and the two classes of the target variable are represented by the two basis states of a single qubit. Thus, if $|y_i\rangle = |0\rangle$ the ith observation belongs to the class 0. Otherwise, if $|y_i\rangle = |0\rangle$ the ith observation belongs to the class 1.

Qubit encoding strategy allows to store a training set of 4 observations using an 8-qubit *data* register. In formulas, the state preparation step leads to

$$\begin{aligned}|\Phi_0\rangle &= \left(H^{\otimes 2} \otimes S_{(x,y)}\right)|0\rangle \otimes |0\rangle \otimes |0\rangle \\ &= |c_1\rangle \otimes |c_2\rangle \otimes |x\rangle|y\rangle \\ &= |+\rangle \otimes |+\rangle \otimes |x_0, x_1, x_2, x_3\rangle|y_0, y_1, y_2, y_3\rangle,\end{aligned} \tag{A3}$$

where S_x is the routine which encodes in the amplitudes of a qubit of a real vector x and H is the Hadamard transformation matrix.

(Step 2) Sampling in Superposition

The second step regards the generation of 2^d different transformations of the training set in superposition, each entangled with a state of the control register. To this end, d steps are necessary, where each step consists of the entanglement of the ith control qubit with two transformations of $|x, y\rangle$ based on two random unitaries, $U_{(i,1)}$ and $U_{(i,2)}$, for $i = 1, 2$. The sampling in superposition step leads to the following quantum state:

$$\begin{aligned}|\Phi_1\rangle = \frac{1}{2}\big[&|00\rangle U_{(2,1)}U_{(1,1)}|x_0, x_1, x_2, x_3\rangle|y_0, y_1, y_2, y_3\rangle \\ &+ |01\rangle U_{(2,1)}U_{(1,2)}|x_0, x_1, x_2, x_3\rangle|y_0, y_1, y_2, y_3\rangle \\ &+ |10\rangle U_{(2,2)}U_{(1,1)}|x_0, x_1, x_2, x_3\rangle|y_0, y_1, y_2, y_3\rangle \\ &+ |11\rangle U_{(2,2)}U_{(1,2)}|x_0, x_1, x_2, x_3\rangle|y_0, y_1, y_2, y_3\rangle\big]. \\ = \frac{1}{\sqrt{4}}&\sum_{b=1}^{4}|b\rangle V_b|x_0, x_1, x_2, x_3; y_0, y_1, y_2, y_3\rangle\end{aligned} \tag{A4}$$

In order to obtain independent quantum trajectories, we provide the following definition for $U_{(i,j)}$:

$$U_{(1,1)} = \mathrm{SWAP}(x_0, x_2) \times \mathrm{SWAP}(y_0, y_2); \tag{A5}$$

$$U_{(1,2)} = \mathrm{SWAP}(x_1, x_3) \times \mathrm{SWAP}(y_1, y_3); \tag{A6}$$

$$U_{(2,1)} = \boldsymbol{I}; \tag{A7}$$

$$U_{(2,2)} = \mathrm{SWAP}(x_2, x_3) \times \mathrm{SWAP}(y_2, y_3); \tag{A8}$$

where $\mathbb{1}$ is the identity matrix. Thus, we get

$$\begin{aligned}|\Phi_2\rangle = \frac{1}{2}\big[&|11\rangle|x_0, x_3, x_1, x_2\rangle|y_0, y_3, y_1, y_2\rangle \\ &+ |10\rangle|x_2, x_1, x_3, x_0\rangle|y_2, y_1, y_3, y_0\rangle \\ &+ |01\rangle|x_0, x_3, x_2, x_1\rangle|y_0, y_3, y_2, y_1\rangle \\ &+ |00\rangle|x_2, x_1, x_0, x_3\rangle|y_2, y_1, y_0, y_3\rangle\big].\end{aligned} \tag{A9}$$

We can see that the swap operation allows to entangle a different observation stored in the *data* register to a different state of the *control* register. In particular, when considering the last qubit of the *features* and *labels* (sub-)registers, the above choices for $U_{(i,j)}$ guarantee that each quantum state of the control register is entangled with a different training observation when considering the last qubit of the *data* register. Using a compact representation

$$|\Phi_{2'}\rangle = \frac{1}{2}[|11\rangle|\ldots\rangle|x_2\rangle|y_2\rangle + |10\rangle|\ldots\rangle|x_0\rangle|y_0\rangle + |10\rangle|\ldots\rangle|x_1\rangle|y_1\rangle + |10\rangle|\ldots\rangle|x_3\rangle|y_3\rangle] = \frac{1}{\sqrt{4}}\sum_{i=0}^{3}|i\rangle|\ldots\rangle|x_i,y_i\rangle. \tag{A10}$$

Note that in this case, the ith basis state does not correspond to the integer representation of the binary state.

Importantly, when considering random swap operations as unitaries $U_{(i,j)}$ instead of the fixed ones (Equations A5–A8)), we have no guarantees that the four quantum trajectories will be independent, but this randomness is necessary to generate the typical scenario of ensemble methods where the different training sets are randomly sampled from the original one.

(Step 3) Learning via interference

The *test* register is initialised to encode the test set, $\tilde{x}$, considering also an additional qubit to store the final prediction.

$$\left(S_{\tilde{x}}\otimes\mathbb{1}\right)|0\rangle|0\rangle = |x^{(test)}\rangle\otimes|0\rangle = \left(x_{\text{test},1}|0\rangle + x_{\text{test},2}|1\rangle\right)\otimes|0\rangle. \tag{A11}$$

Then, the *data* and *test* registers interact via interference using the quantum version of the cosine classifier (gate F) to compute the estimates of the target variable:

$$|\Phi_f\rangle = \left(\mathbb{1}^{\otimes 2}\otimes F\right)|\Phi_d\rangle = \frac{1}{\sqrt{4}}\sum_{b=1}^{4}|b\rangle|x_b,y_b\rangle|x^{(test)}\rangle|\hat{f}_b\rangle. \tag{A12}$$

Since the 4 points of the training set are in superposition, the application of the quantum cosine classifier allows computing 4 different predictions for the test point, $\left\{\hat{f}_b\right\}_{b=1,\ldots4}$, executing the classifier only once.

(Step 4) Measurement

Due to the entanglement between the predictions for $\tilde{x}$ and the *control* register, the expectation measurement allows retrieving the average of all the predictions, which correspond to the ensemble prediction that uses bagging strategy aggregation:

$$\langle M\rangle = \langle *|\hat{f}_b\rangle M|\hat{f}_b = \frac{1}{4}\sum_{b=1}^{4}\hat{f}_b = \hat{f}_{bag}(\tilde{x}|x,y) \tag{A13}$$

The implementations of the quantum ensemble to perform simple averaging are depicted in Figure A1.

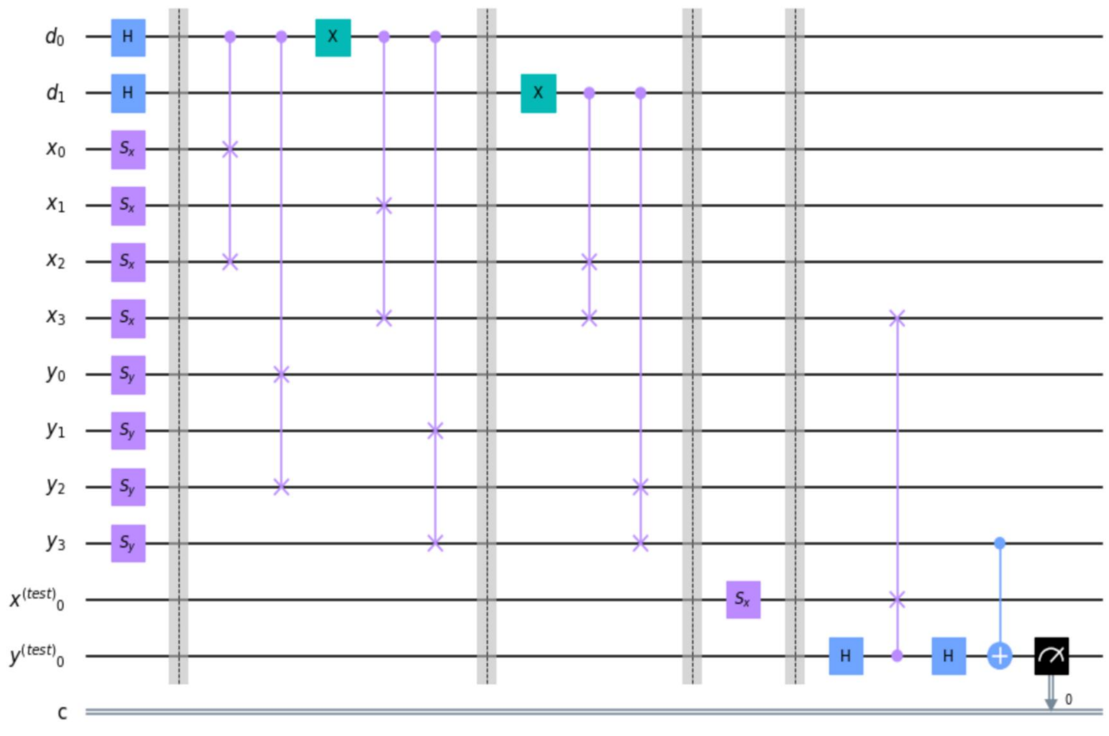


FIGURE A1 Qiskit implementation of the quantum ensemble for four independent quantum trajectories.

Appendix B

QUANTUM COSINE CLASSIFIER

Classically, cosine classifier is defined as follows:

$$\Pr\left(y^{(test)} = y^{(train)}\right) = \frac{1}{2} + \frac{\left[d\left(x^{(train)}, x^{(test)}\right)\right]^2}{2} \qquad \text{(B1)}$$

where $(x^{(train)}, y^{(train)})$ is a random training example, $x^{(test)}$ the test point and $d(\cdot,\cdot)$ the cosine distance between $x^{(train)}$ and $x^{(test)}$. Since the probability of belonging to a class depends on the squared cosine distance between the two vectors, the maximum dissimilarity occurs when training and test observations are orthogonal. In this case, the cosine classifier assigns a uniform probability distribution in the two classes for $y^{(test)}$. This means that the cosine classifier performs well only if the test point belongs to the same class of the training point.

The quantum circuit that implements the cosine classifier (Figure 3) encodes data into three different registers: the training vector $x^{(train)}$, the training label $y^{(train)}$ and the test point $x^{(test)}$. One last qubit is used to store the prediction. The algorithm is made of the following three steps.

Step 1: State Preparation

The state preparation routine can be performed independently for each qubit:

$$\begin{aligned}|\Phi_1\rangle &= \left(S_{x^{(train)}} \otimes S_{x^{(test)}} \otimes S_{y^{(train)}} \otimes \mathbb{1}\right)|0\rangle^{\otimes 4} \\ &= |x^{(train)}\rangle|x^{(test)}\rangle|y^{(train)}\rangle|0\rangle,\end{aligned} \qquad \text{(B2)}$$

where S_x is the routine which encodes in the amplitudes of a qubit a 2-dimensional, normalised real vector x.

Step 2: Execution of the swap test

In the second step, the swap test [28] transforms the amplitudes of the qubit $y^{(test)}$ as a function of the squared cosine distance. After the execution of the swap test, the probability to readout the basis state $|0\rangle$, that is, the probability for the test observation to be classified in class 0 is

$$P\left(y^{(test)} = |0\rangle\right) = \frac{1}{2} + \frac{|\langle x^{(train)}|x^{(test)}\rangle|^2}{2}. \qquad \text{(B3)}$$

Step 3: Controlled Pauli-X gate

The third step consists of applying a controlled-Pauli-X gate using as control qubit the label of the training vector. This implies that $y^{(test)}$ is left untouched if $x^{(train)}$ belongs to the class 0. Otherwise, the amplitudes of the $y^{(test)}$ qubit are exchanged, and the probability $P(y^{(test)} = 1)$ is higher as the similarity between the two vectors increases. At this point, the expectation measurement on the last qubit provides the prediction of interest. The result predictions of the quantum cosine classifier assuming a perfect quantum device are depicted in Figure B1.

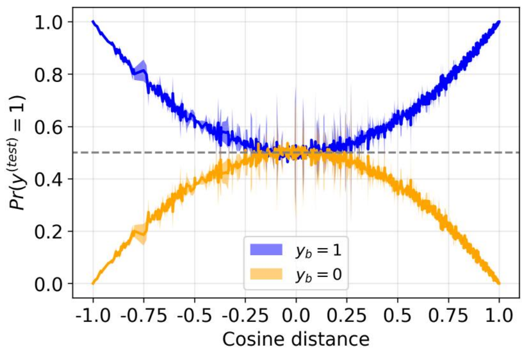


FIGURE B1 Predictions of the cosine distance classifier based on 10^3 randomly generated datasets per class. The classifier is implemented using the circuit in Figure 2.

Appendix C

QUANTUM ENSEMBLE OF COSINE CLASSIFIER

Examples of quantum circuits for implementing a quantum ensemble of quantum cosine classifiers for different ensemble sizes (2, 4, 8, 16).

see Figure C2.

see Figure C3.

see Figure C4.

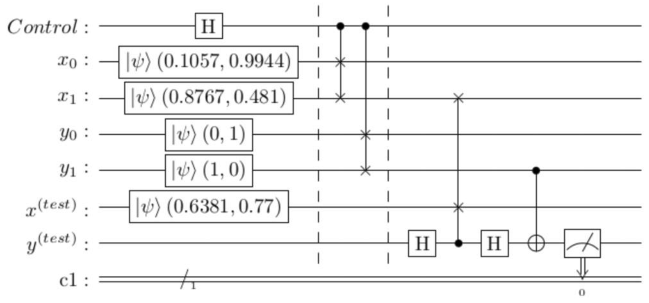
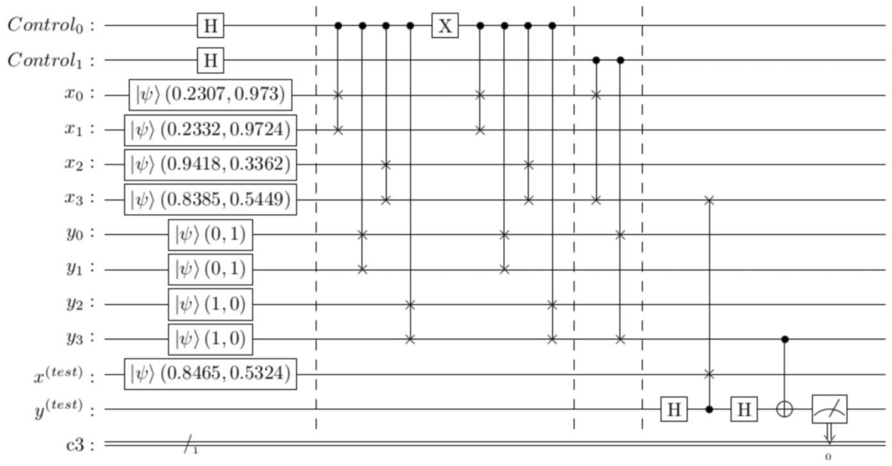
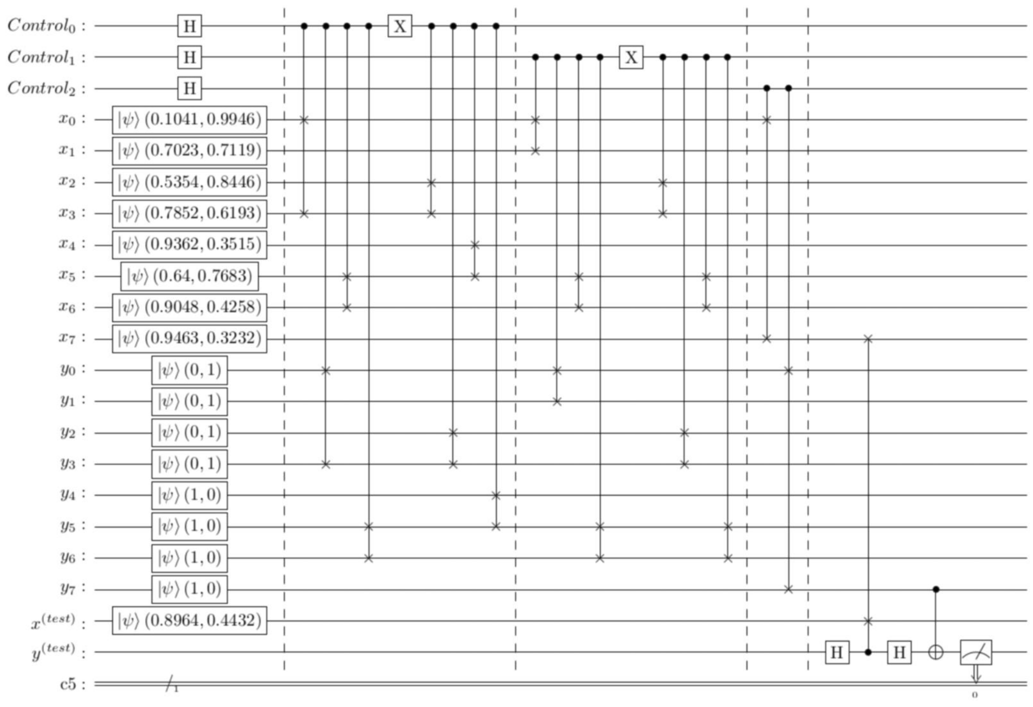


FIGURE C1 Quantum circuit for implementing a quantum ensemble of size 2 ($d = 1$) of cosine classifiers. The total number of qubits is 7: four for encoding the training set (x_0, y_0, x_1, y_1), two for the test set $(x^{(test)}, y^{(test)})$ and one for control qubit.



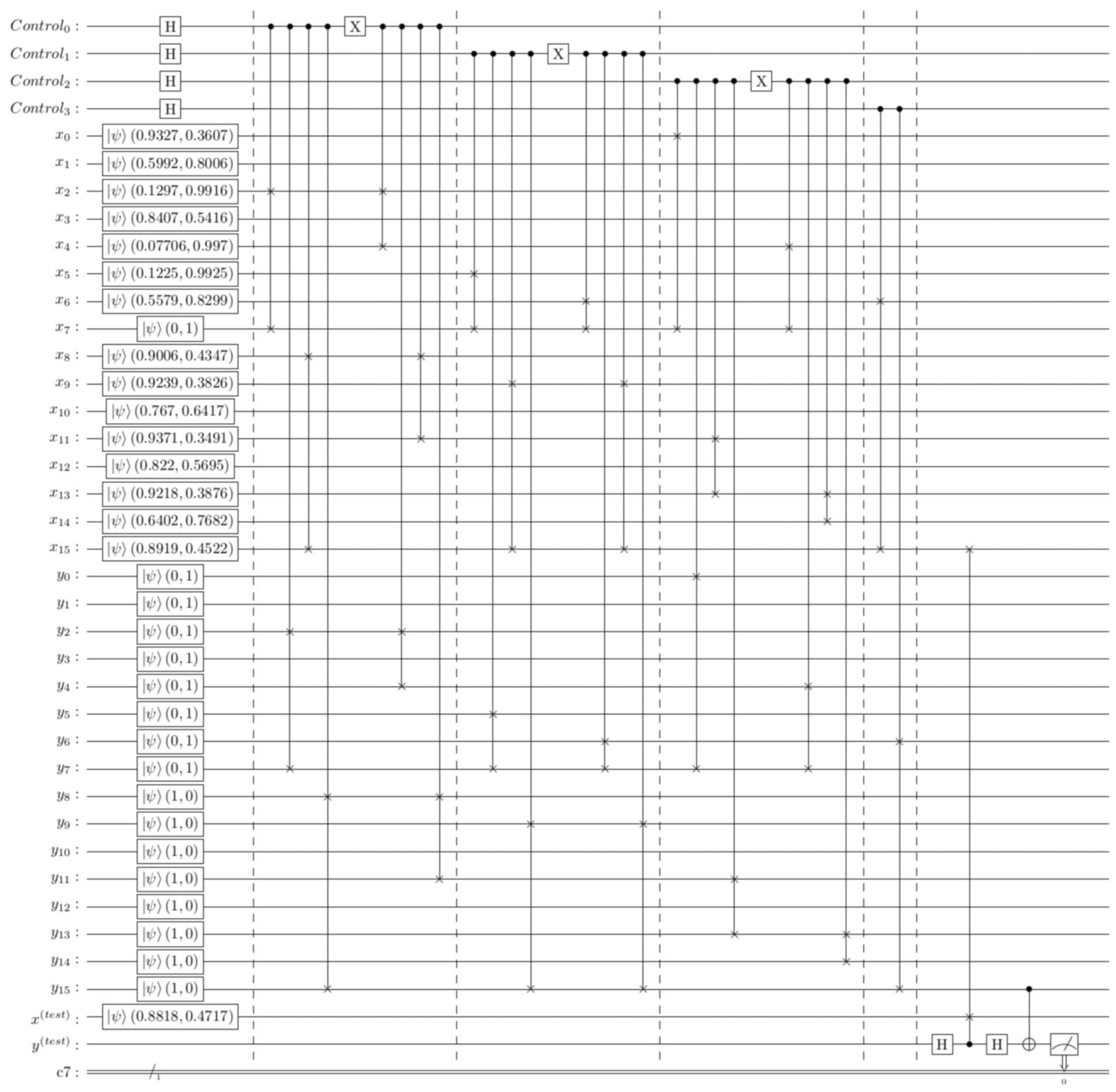
$Control_0$:
$Control_1$:
x_0 : $|\psi\rangle\,(0.2307, 0.973)$
x_1 : $|\psi\rangle\,(0.2332, 0.9724)$
x_2 : $|\psi\rangle\,(0.9418, 0.3362)$
x_3 : $|\psi\rangle\,(0.8385, 0.5449)$
y_0 : $|\psi\rangle\,(0, 1)$
y_1 : $|\psi\rangle\,(0, 1)$
y_2 : $|\psi\rangle\,(1, 0)$
y_3 : $|\psi\rangle\,(1, 0)$
$x^{(test)}$: $|\psi\rangle\,(0.8465, 0.5324)$
$y^{(test)}$:
c3 :
H
X

FIGURE C2 Quantum circuit for implementing a quantum ensemble of size 4 ($d = 2$) of cosine classifiers. The total number of qubits is 12: eight for encoding the training set $(x_i, y_i)_{i=0,\ldots3}$, two for the test set ($x^{(test)}$, $y^{(test)}$) and two for control qubits.



$Control_0$:
$Control_1$:
$Control_2$:
x_0 : $|\psi\rangle\,(0.1041, 0.9946)$
x_1 : $|\psi\rangle\,(0.7023, 0.7119)$
x_2 : $|\psi\rangle\,(0.5354, 0.8446)$
x_3 : $|\psi\rangle\,(0.7852, 0.6193)$
x_4 : $|\psi\rangle\,(0.9362, 0.3515)$
x_5 : $|\psi\rangle\,(0.64, 0.7683)$
x_6 : $|\psi\rangle\,(0.9048, 0.4258)$
x_7 : $|\psi\rangle\,(0.9463, 0.3232)$
y_0 : $|\psi\rangle\,(0, 1)$
y_1 : $|\psi\rangle\,(0, 1)$
y_2 : $|\psi\rangle\,(0, 1)$
y_3 : $|\psi\rangle\,(0, 1)$
y_4 : $|\psi\rangle\,(1, 0)$
y_5 : $|\psi\rangle\,(1, 0)$
y_6 : $|\psi\rangle\,(1, 0)$
y_7 : $|\psi\rangle\,(1, 0)$
$x^{(test)}$: $|\psi\rangle\,(0.8964, 0.4432)$
$y^{(test)}$:
c5 :
H
X

FIGURE C3 Quantum circuit for implementing a quantum ensemble of size 8 ($d = 3$) of cosine classifiers. The total number of qubits is 12: 16 for encoding the training set $(x_i, y_i)_{i=0,\ldots7}$, two for the test set ($x^{(test)}$, $y^{(test)}$) and three control qubits.



Control_0 :
Control_1 :
Control_2 :
Control_3 :
H
X
x_0 : |ψ⟩ (0.9327, 0.3607)
x_1 : |ψ⟩ (0.5992, 0.8006)
x_2 : |ψ⟩ (0.1297, 0.9916)
x_3 : |ψ⟩ (0.8407, 0.5416)
x_4 : |ψ⟩ (0.07706, 0.997)
x_5 : |ψ⟩ (0.1225, 0.9925)
x_6 : |ψ⟩ (0.5579, 0.8299)
x_7 : |ψ⟩ (0, 1)
x_8 : |ψ⟩ (0.9006, 0.4347)
x_9 : |ψ⟩ (0.9239, 0.3826)
x_10 : |ψ⟩ (0.767, 0.6417)
x_11 : |ψ⟩ (0.9371, 0.3491)
x_12 : |ψ⟩ (0.822, 0.5695)
x_13 : |ψ⟩ (0.9218, 0.3876)
x_14 : |ψ⟩ (0.6402, 0.7682)
x_15 : |ψ⟩ (0.8919, 0.4522)
y_0 : |ψ⟩ (0, 1)
y_1 : |ψ⟩ (0, 1)
y_2 : |ψ⟩ (0, 1)
y_3 : |ψ⟩ (0, 1)
y_4 : |ψ⟩ (0, 1)
y_5 : |ψ⟩ (0, 1)
y_6 : |ψ⟩ (0, 1)
y_7 : |ψ⟩ (0, 1)
y_8 : |ψ⟩ (1, 0)
y_9 : |ψ⟩ (1, 0)
y_10 : |ψ⟩ (1, 0)
y_11 : |ψ⟩ (1, 0)
y_12 : |ψ⟩ (1, 0)
y_13 : |ψ⟩ (1, 0)
y_14 : |ψ⟩ (1, 0)
y_15 : |ψ⟩ (1, 0)
x^(test) : |ψ⟩ (0.8818, 0.4717)
y^(test) :
c7 :
1
0

FIGURE C4 Quantum circuit for implementing a quantum ensemble of size 16 ($d = 4$) of cosine classifiers. The total number of qubits is 22: 16 for encoding the training set $(x_i, y_i)_{i=0,\dots 15}$, two for the test set ($x^{(test)}$, $y^{(test)}$) and four control qubits.