arXiv:2007.01028v1 [cs.LG] 2 Jul 2020

QUANTUM ENSEMBLE FOR CLASSIFICATION

A PREPRINT

Antonio Macaluso
Department of Computer Science and Engineering
University of Bologna, Italy
antonio.macaluso2@unibo.it

Luca Clissa
Department of Physics and Astronomy
University of Bologna, Italy
Istituto Nazionale di Fisica Nucleare (INFN), Italy
luca.clissa2@unibo.it

Stefano Lodi
Department of Computer Science and Engineering
University of Bologna, Italy
stefano.lodi@unibo.it

Claudio Sartori
Department of Computer Science and Engineering
University of Bologna, Italy
claudio.sartori@unibo.it

July 3, 2020

ABSTRACT

A powerful way to improve performance in machine learning is to construct an ensemble that combines the predictions of multiple models. Ensemble methods are often much more accurate and lower variance than the individual classifiers that make them up, but have high requirements in terms of memory and computational time. In fact, a large number of alternative algorithms is usually adopted, each requiring to query all available data.

We propose a new quantum algorithm that exploits quantum superposition, entanglement and interference to build an ensemble of classification models. Thanks to the generation of the several quantum trajectories in superposition, we obtain B transformations of the quantum state which encodes the training set in only $log\,(B)$ operations. This implies an exponential speed-up in the size of the ensemble with respect to classical methods. Furthermore, when considering the overall cost of the algorithm, we show that the training of a single weak classifier impacts additively rather than multiplicatively, as it usually happens.

We also present small-scale experiments, using a quantum version of the cosine classifier using IBM qiskit environment to show how the algorithm works.

***K*eywords** Quantum Algorithms · Quantum Machine Learning · Machine Learning · Ensemble methods · Binary classification

1 Introduction

Quantum Computing (QC) can achieve performance orders of magnitude faster than the classical counterparts, with the possibility of tremendous speed-up of complex computational tasks [1, 2, 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 Machine Learning (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].

One of the most studied problems in QML is classification, where an algorithm is trained on data whose category of the target variable is known. According to Dietterich's definition [5], a classifier is a hypothesis about the true function f which allows estimating a target variable y, given a vector of features x. Among the multiple alternatives to build a classifier, a well-known approach is ensemble methods, where a large number of hypotheses is combined by averaging or voting rules to classify new examples. Despite the absence of a unified theory, there are many theoretical reasons for combining multiple learners, e.g. reducing the prediction error by decreasing the uncertainty on the estimates, as well as empirical evidence of the effectiveness of this approach [6, 7].

Recently, the idea of a quantum ensemble has been investigated by Schuld et al. [8]. In this case, the construction of the ensemble corresponds to three different stages: *(i)* a state preparation routine, *(ii)* the evaluation in parallel of the quantum classifiers and *(iii)* the access to the combined decision. This approach is based on Bayesian Model Averaging (BMA) that exploits many models whose parameters are fixed so as to span a large part of parameters domain. The strength of this approach is that the individual classifiers do not have to be trained. However, BMA is not very used in ML because of the limited performance in real-world applications [7]. For this reason, classical ensemble methods are commonly used to generate a collection of complementary hypotheses which are fitted to the data under different training conditions.

2 Background

When trying to predict a target variable using any ML model, the main causes of the difference in actual and predicted values (Expected Prediction Error or *EPE*) are noise, bias and variance [9]. The *noise* component, also known as *irreducible error*, is the variance of the target variable around its true mean. This error is due to the intrinsic uncertainty of the data, so it cannot be avoided no matter how well the model works. The *bias*, instead, is linked to the particular learning technique adopted, and it measures how well the method suits the problem. Finally, the *variance* component measures the variability of the learning method around its expected value. In light of this, in order to improve the performance of any ML technique, one has to try to reduce one or more of these components.

2.1 Ensemble Learning

The idea of ensemble learning is to build a prediction model by combining the strengths of a collection of simpler base models to reduce the *EPE*. A necessary and sufficient condition for an ensemble to outperform any of its members is that the single models are **accurate**, in the sense that they have an error rate better than random guessing, and **diverse**, which means that the individual models make different errors given the same data points [10].

There exist several ways to build ensemble methods, each designed to tackle a specific component of the *EPE*. In *Boosting*, the idea is to exploit a committee of weak learners that evolves over time. In practice, at each iteration a new weak learner is trained with respect to the error of the whole ensemble. This mechanism allows getting closer and closer to the true population values, thus reducing the bias. *Randomisation* methods consist in estimating the single base model with a randomly perturbed training algorithm. This alteration worsens the accuracy of the individual learners, but reduces the ensemble variance thanks to the combination of a large number of randomised models. Unlike the other methods, this approach is applicable also to stable learners, thus enlarging the plethora of methods it applies to. Another approach is *Bagging*. In this case, the same model is fitted to different training sets, thus creating a committee of independent weak learners. The individual votes are then averaged to obtain the ensemble prediction. This approach decreases the *EPE* by reducing the variance component so the more classifiers are included (i.e., the larger the size of the ensemble), the more significant is the reduction.

In practice, bagging reduces to computing several predictions $\hat{f}_1(x), \hat{f}_2(x), \ldots, \hat{f}_B(x)$ using B different training sets, which are then averaged to obtain a single model with lower variance:

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

Although this approach guarantees a lower uncertainty in prediction, it is not practical in its theoretical formulation, due to the lack of multiple training sets. To overcome this issue, bootstrap procedure [11] can be employed, that takes repeated samples from the available data and generates B different bootstrapped training sets. The learning algorithm is then trained on the *b-th* bootstrapped observations to get B different predictions $\hat{f}_b(x)$. The difference between the bootstrap and the idealised procedure is the way the training sets are derived. Instead of obtaining independent datasets from the domain, the initial training set is perturbed as many times as the number of weak classifiers to aggregate. The generated datasets are certainly not independent because they are all based on the same training set. Nonetheless, empirical findings suggest that bagging is still able to produce combined models that often significantly outperform individual learners, and that anyway are never substantially worse [6].

2.2 Contribution

The purpose of this work is to provide a general framework to tackle classification problems through quantum ensembles. In particular, we describe in details the implementation of a quantum ensemble using bagging and discuss the possibility to employ the same algorithm for randomisation and boosting.

The high-level idea is to design a quantum framework that propagates an input state to multiple quantum trajectories in superposition, in such a way that a sum of individual results from each trajectory is obtained. From a technical point of view, the algorithm is able to generate different transformations of the training set in superposition, each entangled with a quantum state of a control register. Thus, a quantum classifier F is applied to obtain a large number of classifications in superposition. By averaging those predictions, the ensemble prediction can be accessed by measuring a single register.

As a consequence of this convenient architecture, our method implies three main computational advantages. First, it scales exponentially faster than classical methods with respect to the ensemble size (i.e., number of simple base models), since it requires only d steps to generate 2^d different transformations of the same training set in superposition. Second, having entangled states entails an additive impact of the single weak classifiers, as opposed to the usual multiplicative burden of classical implementations. This means that the time cost of implementing the ensemble is not dominated by the cost of the single classifier, but rather by the data encoding strategy. Third, the number of state preparation routines is equivalent to implement just the single classifier since the classification routine is assumed to work via interference, and its use is propagated to all the quantum trajectories in superposition with just one execution. In addition, the algorithm also allows obtaining the ensemble prediction by measuring a single register and it makes the evaluation of large ensembles feasible with relatively small circuits.

Finally, as a proof of principle we also conduct experiments on simulated data by defining a simple classification routine based on cosine distance to be used for the single weak learners.

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, instead, are discussed in Section 3.1.

The algorithm adopts three quantum registers: data, control, 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 in 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:

$$|\Phi_0\rangle = \big(W \otimes S_{(x,y)}\big) \bigotimes_{j=1}^{d} |0\rangle \otimes |0\rangle = \big(H^{\otimes d} \otimes S_{(x,y)}\big) \bigotimes_{j=1}^{d} |0\rangle \otimes |0\rangle = \bigotimes_{j=1}^{d} |c_j\rangle \otimes |x, y\rangle\,, \tag{2}$$

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 $|c_j\rangle$ is the j-th qubit of the control register.

(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 i-th control qubit with two transformations of $|x, y\rangle$ based on two random unitaries, $U_{(i,1)}$ and $U_{(i,2)}$, for $i = 1, \dots, d$. 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 i-th 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 i-th control qubit:

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

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

$$\begin{aligned}
|\Phi_{i,2}\rangle =& (X \otimes \mathbb{1}) |\Phi_{i,1}\rangle \\
=& \frac{1}{\sqrt{2}} \Big(|1\rangle |x, y\rangle + |0\rangle U_{(i,1)} |x, y\rangle \Big)
\end{aligned} \tag{4}$$

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

$$\begin{aligned}
|\Phi_{i}\rangle =& \Big(CU_{(1,2)}\Big) |\Phi_{i,2}\rangle \\
=& \Big(CU_{(1,2)}\Big) \frac{1}{\sqrt{2}} \Big(|1\rangle |x, y\rangle + |0\rangle U_{(i,1)} |x, y\rangle \Big) \\
=& \frac{1}{\sqrt{2}} \Big(|1\rangle U_{(i,2)} |x, y\rangle + |0\rangle U_{(i,1)} |x, y\rangle \Big).
\end{aligned} \tag{5}$$

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 which differ 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 quantum oracle 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 choosen 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, $x^{(\text{test})}$, considering also 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 2.1. 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:

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

where $\hat{f}_b$ represents the prediction for $x^{(\text{test})}$ given the b-th 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 consequence of Steps 2 and 3, the *b-th* state of the *control* register is entangled with the *b-th* value of $\hat{f}$.

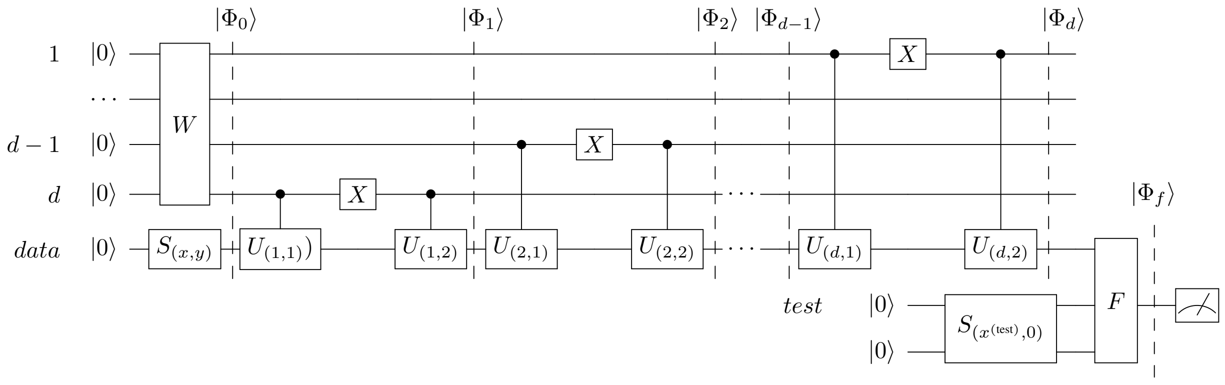


Figure 1: Quantum algorithm for ensemble classification. The circuit contains d pairs of unitaries $U_{(i,1)}$, $U_{(i,2)}$ and d control qubits. It produces an ensemble of B classifiers, where $B = 2^d$. The single evaluation of F allows propagating the classification function $\hat{f}$ in all trajectories in superposition. The firsts d steps allows 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.

(Step 4) Measurement

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

$$
\begin{aligned}
\langle M \rangle &= \langle \Phi_f | \mathbb{1}^{\otimes d} \otimes \mathbb{1} \otimes \mathbb{1} \otimes M | \Phi_f \rangle \\
&= \frac{1}{2^d} \sum_{b=1}^{2^d} \langle b|b\rangle \otimes \langle (x_b, y_b)|(x_b, y_b)\rangle \otimes \langle x^{(\text{test})}|x^{(\text{test})}\rangle \otimes \langle \hat{f}_b|M|\hat{f}_b\rangle \\
&= \frac{1}{2^d} \sum_{b=1}^{2^d} \langle \hat{f}_b|M|\hat{f}_b\rangle = \frac{1}{2^d} \sum_{b=1}^{2^d} \langle M_b \rangle \\
&= \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b = \hat{f}_{bag}(x^{(\text{test})}|x, y)
\end{aligned} \tag{11}
$$

where $B = 2^d$ and M is a measurement operator (e.g. 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 to the ensemble prediction by 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 also randomisation and boosting.

The main principle of the ensemble based on randomisation consists in the introduction of casual perturbations that decorrelate the predictions of individual classifiers as much as possible. 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*), in fact, can be employed to introduce a random component in the single learner, so 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 which can be decomposed in 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 $G_b = V_b F$ is the quantum classifier composed by F – common to all the classifiers – and V_b which is its random component – different for each quantum trajectory. This formulation allows rewriting the quantum state in Equation 10 as:

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

Likewise, 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 now need to be updated as the computation evolves. As a result, the output of a quantum ensemble based on boosting can be described as:

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

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 speed-up 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. On one hand, the most popular technique used in ensemble classification is *majority voting*, where each classifier votes for a target class and the most frequent is then selected. On the other hand, 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:

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

where B is the ensemble size and $f^{(i)}_b(x)$ is the probability for x to be classified in the i-th class provided by the b-th classifier. This approach allows a reduction of the estimates variance [12] and has shown good performance even for large and complex datasets [13]. In particular, the error E_{ens} of an ensemble obtained averaging B individual learners can be expressed as [14, 15]:

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

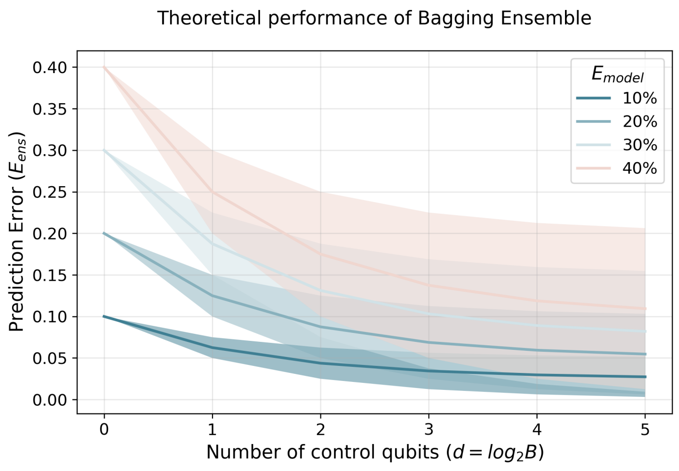


Figure 2: Theoretical performance of the quantum ensemble based on the expected prediction error of the base classifiers (E_{model}) and their average correlation (ρ). The ensemble size depends on the number of qubits d in the control register. Each solid line corresponds to an error level, with coloured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

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 B, ρ and E_{model} is reported in Figure 2.

Coming to 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 in 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 quantum ensemble to improve significantly the performance of the single classifier (Figure 2) using relatively small circuit ($d \sim 10$).

3.3 Computational Complexity

Classically, given a number B of base learners and a dataset (x_i, y_i) for $i = 1, \dots N$, 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 depends 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 benchmark the two approaches by looking at how they scale in terms of the parameters of the ensemble, i.e, the ensemble size B and the cost of each base model. In particular, this resolves in considering the Boolean circuit model [16] 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 is able to generate an ensemble of size $B = 2^d$ in only d steps. This means that we are able to introduce an exponential speed-up with respect to classical ensemble methods in terms of the ensemble size. 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 $U_{(i,j)}$, that strictly depends on the encoding strategy chosen for the data.

4 Experiments

To test how our framework for quantum ensemble works in practice, we implemented the circuit illustrated in Figure 1 using IBM qiskit [17]. Then, we conducted experiments on simulated data to show that (i) one execution of a quantum classifier allows retrieving the ensemble prediction, and that (ii) the ensemble outperforms the single model.

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 [18] that stores the cosine distance into the amplitudes of a quantum state. This metric describes how similar two vectors are depending on the angle that separates them, irrespectively 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_b, y_b) at random and we express the probability of $y^{(\text{test})}$ and y_b being equal as a function of the similarity between $x^{(\text{test})}$ and x_b:

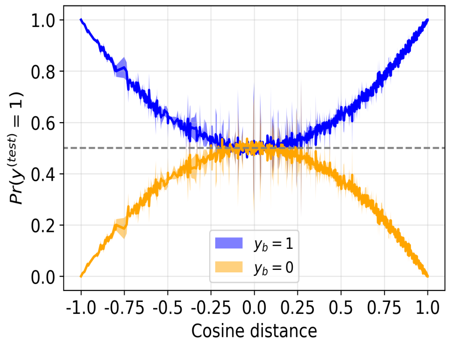
$$Pr\left(y^{(\text{test})} = y_b\right) = \frac{1}{2} + \frac{\left[d\left(x_b, x^{(\text{test})}\right)\right]^2}{2} \tag{18}$$

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

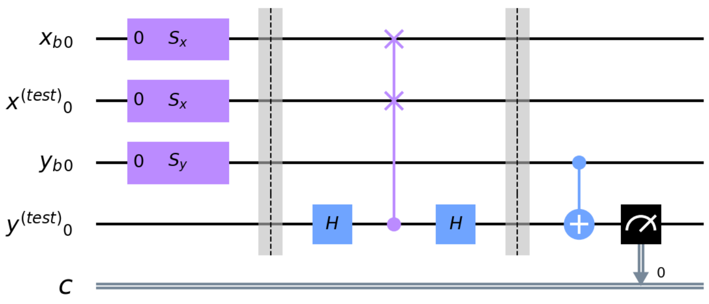
$$y^{(\text{test})} = \begin{cases} y_b, & \text{if } Pr\left(y^{(\text{test})} = y_b\right) > \frac{1}{2} \\ 1 - y_b & \text{otherwise} \end{cases} \tag{19}$$

Notice that, by definition, $Pr\left(y^{(\text{test})} = y_b\right)$ is bounded in $[\frac{1}{2}, 1]$, which means that Equation (19) will always estimate the same class as the training point, unless x_b 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 3b. It encodes data into three different registers: the training vector x_b, the training label y_b and the test point $x^{(\text{test})}$. An additional qubit is then used to store the prediction. The algorithm is made 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 as control qubit the label of the training vector. This implies that $y^{(\text{test})}$ is left untouched if x_b belongs to the 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.



(a) Predictions of the cosine distance classifier based on 10^3 randomly generated datasets per class. The classifier is implemented using the circuit in Figure 3b.



(b) Quantum circuit of the cosine classifier using x_b as training vector and $x^{(\text{test})}$ as test vector. The training label y_b is either $|0\rangle$ or $|1\rangle$ based on the binary target value.

Figure 3: Quantum Cosine Classifier

To summarise, the quantum cosine classifier performs classification via interference and it allows calculating the probability of belonging to one of the two classes by single-qubit measurement. 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_b is stored into N different qubits. This allows the definition of $U_{(i,j)}$ 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.

4.2 Quantum Ensemble as Simple Averaging

As a proof-of-concept for the quantum ensemble based on bagging, we consider a dataset with four training points and one test example. In particular, we show experimentally that the quantum ensemble prediction is exactly the average of the values of all trajectories in superposition and it can be obtain with just one execution of the classification routine.

The toy dataset used here is reported in Table 1. Each training point is fed into a quantum cosine classifier as input so to provide an estimate for a test observation $x^{(\text{test})}$. In practice, 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 [1].

	Dataset				
	X_1	X_2	y	$d(\cdot,\cdot)$	$Pr(y^{(\text{test})}=1)$
x_1	1	3	0	0.89	0.10
x_2	-2	2	1	0	0.50
x_3	3	0	0	0.71	0.25
x_4	3	1	1	0.89	0.90
$x^{(\text{test})}$	2	2	?	1.0	/

Table 1: Each row of the table corresponds to a possible training observation. X_1 and X_2 are the features, $d(\cdot,\cdot)$ is the cosine distance of the training point from $x^{(\text{test})}$ and $Pr(y^{(\text{test})}=1)$ is the predicted probability computed classically (see Equation 18).

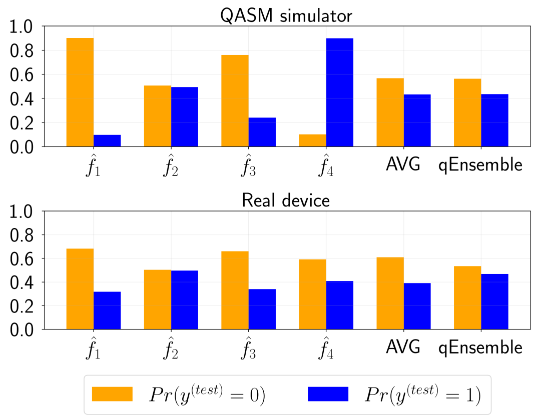


Figure 4: Quantum results based on data in Table 1. The labels $\hat{f}_{b b=1,\cdots,4}$ indicate the estimated probabilities for $x^{(\text{test})}$ given the b-th observation as training set. The AVG bars are obtained by averaging the individual classifiers, while qEnsemble represents the prediction of the quantum ensemble.

The results of the quantum implementation are shown in Figure 4. The value $\hat{f}_b$ indicates the output of the quantum cosine classifier using (x_b, y_b) as training set. The experiments using the QASM simulator (top plot) show an equivalence between the probability of $x^{(\text{test})}$ to be classified in class 1 (blue bar) and the same probability computed classically (column $Pr(y^{(\text{test})}=1)$ of Table 1). Also, the quantum estimate (qEnsemble) matches perfectly the classical ensemble prediction computed by averaging the four classifications (AVG), as expected. The agreement, however, deteriorates when running on a real quantum device (bottom plot).

In order to generalise the results of the quantum ensemble beyond the dataset in Table 1, we performed the same experiment on 20 randomly generated datasets, and we compared the average of the quantum cosine classifiers with the quantum ensemble prediction. Results are shown in Figure 5. In this case, the agreement between the quantum ensemble (orange line) and the average (brown dots) is almost perfect, which confirms the possibility to perform quantum ensemble with the advantages described in Section 3. Results considering the real device (light blue line) show significant deterioration, this may be due to the depth of the quantum circuit which seems to be prohibitive considering current available quantum technology.

[1]For more details about the implementation see https://github.com/amacaluso/Quantum-Ensemble-for-Classification

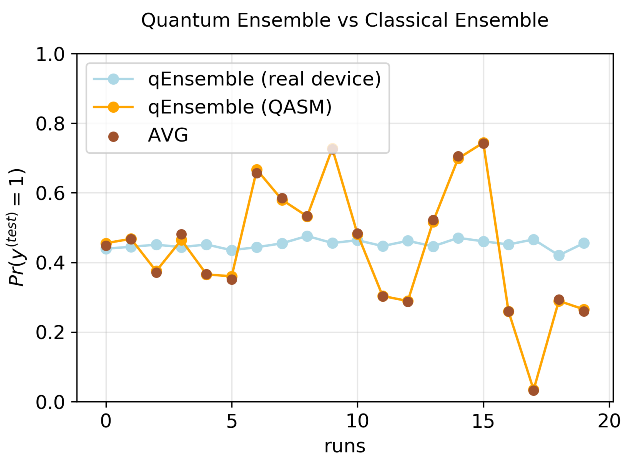


Figure 5: Comparison between the quantum ensemble (qEnsemble) and the average of the four quantum cosine classifiers executed separately (AVG, brown dots), which is computed classically. The simulation of the circuit on the QASM simulator is illustrated in orange, while the light blue line depicts the behaviour on a real device (ibmq_16_melbourne).

4.3 Performance of Quantum Ensemble

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, given the limitations of the present quantum technology and the definition of the cosine classifier, we need to execute the classification routine once for each test point. We considered the accuracy and brier score as performance metrics. The latter measures the difference between the estimates and the true probabilities in terms of mean squared error, hence the lower the score, the better the predictions. Because of the random component in the models due to the choice of training points, we repeated the experiments 10 times and evaluated the classifiers in terms of mean and standard deviation of both accuracy and brier score.

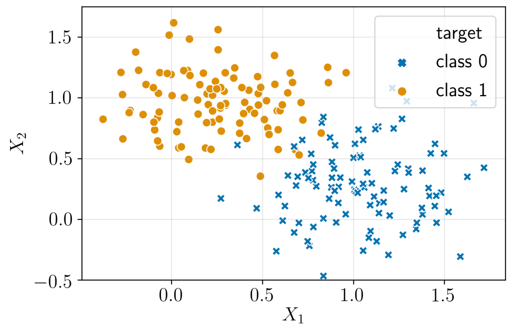


Figure 6: Dataset generated by two independent bivariate Gaussian distributions. Mean vectors for the two classes are $(1, 0.3)$ and $(0.3, 1)$. The two distributions have the same diagonal covariance matrix, with constant value of 0.3.

	Assessment metrics (QASM simulator)	
B	Accuracy	Brier Score
1	0.55 ± 0.09	0.21 ± 0.05
2	0.92 ± 0.09	0.14 ± 0.09
4	0.91 ± 0.09	0.15 ± 0.09
8	0.96 ± 0.04	0.14 ± 0.04
16	0.98 ± 0.02	0.13 ± 0.02

Table 2: Performance comparison between quantum cosine classifier and quantum ensemble of different sizes B. The first row indicates the performance of a single classifier.

Results are shown in Table 2. 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.

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

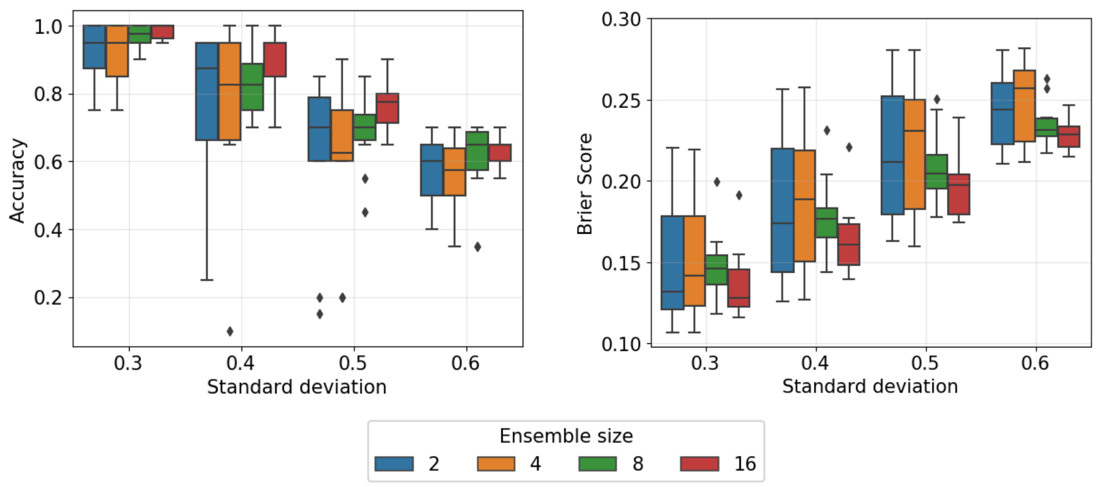


Figure 7: Distribution of the performance metrics as a function of the ensemble size (legend colors) and the separation between the two classes (x axis).

deviation so to force reciprocal contamination. Results are reported in Figure 7. The accuracy showed a decreasing trend as the overlap of the distributions increased. The oppsosite behaviour is observed for the brier score. 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.

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 introduce an exponential speed-up in the ensemble size with respect to the classical counterpart. 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 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 data and demonstrate that the quantum ensemble systematically outperform of the single classifier. Also, the variability decreases as the 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. However, these upgrades would imply a different definition of $U_{(i,j)}$ for the generation of multiple and diverse training sets in superposition.

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 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 loosening these constraints, we believe that it is possible to design other types of ensemble techniques.

Although some challenges still remain, we believe this work is the first practical example of how Machine Learning, in particular ensemble classification, could benefit from Quantum Computing.

References

[1] Lov K Grover. A fast quantum mechanical algorithm for database search. *arXiv preprint quant-ph/9605043*, 1996.

[2] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[3] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009.

[4] Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291, 2015.

[5] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[6] Kagan Tumer and Joydeep Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3-4):385–404, 1996.

[7] Pedro Domingos. Bayesian averaging of classifiers and the overfitting problem. In *ICML*, volume 2000, pages 223–230, 2000.

[8] Maria Schuld and Francesco Petruccione. Quantum ensembles of quantum classifiers. *Scientific reports*, 8(1):2772, 2018.

[9] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):223–224, 2005.

[10] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, Oct 1990.

[11] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[12] Kagan Tumer and Joydeep Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, 1996.

[13] Lei Xu, Adam Krzyzak, and Ching Y Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE transactions on systems, man, and cybernetics*, 22(3):418–435, 1992.

[14] Robert A Jacobs. Methods for combining experts' probability assessments. *Neural computation*, 7(5):867–888, 1995.

[15] Nikunj C Oza and Kagan Tumer. Classifier ensembles: Select real-world applications. *Information Fusion*, 9(1):4–20, 2008.

[16] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[17] Héctor Abraham and et al. Qiskit: An open-source framework for quantum computing, 2019.

[18] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. Quantum fingerprinting. *Phys. Rev. Lett.*, 87:167902, Sep 2001.

Appendices

A Sampling in Superposition for a 2-qubits control register

In this section we describe the procedure of *Sampling in Superposition* in the case of $d = 2$. According to Equation 2, the *State Preparation* step leads to:

$$\begin{aligned}|\Phi_0\rangle &= \big(H^{\otimes 2} \otimes S_{(x,y)}\big) |0\rangle \otimes |0\rangle \otimes |0\rangle = |c_1\rangle \otimes |c_2\rangle \otimes |x,y\rangle \\ &= \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle \big) \otimes \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle \big) \otimes |x,y\rangle \end{aligned} \tag{20}$$

Thus, the two steps of *Sampling in Superposition* regard the entanglement of the two control qubits with four different transformations of the training set. The steps are the following:

Step 2.1

- First, the controlled-unitary $CU_{(1,1)}$ is executed to entangle the transformation $U_{(1,1)}|x,y\rangle$ with the excited state of $|c_2\rangle$:

$$\begin{aligned}|\Phi_{1,1}\rangle &= \Big(\mathbb{1} \otimes CU_{(1,1)}\Big) |\Phi_0\rangle \\ &= \Big(\mathbb{1} \otimes CU_{(1,1)}\Big) |c_1\rangle \otimes \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle \big) \otimes |x,y\rangle \\ &= |c_1\rangle \otimes \frac{1}{\sqrt{2}}\Big(|0\rangle |x,y\rangle + |1\rangle U_{(1,1)} |x,y\rangle \Big)\end{aligned} \tag{21}$$

- Second, $|c_2\rangle$ is transformed based on Pauli–X gate, so that the two basis states are inverted:

$$\begin{aligned}|\Phi_{1,2}\rangle &= (\mathbb{1} \otimes X \otimes \mathbb{1}) |\Phi_{1,1}\rangle \\ &= |c_1\rangle \otimes \frac{1}{\sqrt{2}}\Big(|1\rangle |x,y\rangle + |0\rangle U_{(1,1)} |x,y\rangle \Big)\end{aligned} \tag{22}$$

- Third, a second controlled-unitary $CU_{(1,2)}$ is executed:

$$\begin{aligned}|\Phi_1\rangle &= (\mathbb{1} \otimes CU_{(1,2)}) |c_1\rangle \otimes \frac{1}{\sqrt{2}}\Big(|1\rangle |x,y\rangle + |0\rangle U_{(1,1)} |x,y\rangle \Big) \\ &= \bigotimes_{j=1}^{d-1} |c_j\rangle \otimes \frac{1}{\sqrt{2}}\Big(|1\rangle U_{(1,2)} |x,y\rangle + |0\rangle U_{(1,1)} |x,y\rangle \Big)\end{aligned} \tag{23}$$

At this point, two different transformations, $U_{(1,1)}$ and $U_{(1,2)}$, of the initial state $|x,y\rangle$ are generated in superposition and they are entangled with the two basis states of the control qubit $|c_2\rangle$.

Step 2.2

The same operations are applied using $|c_1\rangle$ as control qubit and different random matrices, $U_{(2,1)}$ and $U_{(2,2)}$.

- First, the controlled-unitary $U_{(2,1)}$ is applied to entangle a transformation of $|x,y\rangle$ with the excited state of $|c_1\rangle$:

$$\begin{aligned}|\Phi_{2,1}\rangle &= (C \otimes \mathbb{1} \otimes U_{(2,1)}) |\Phi_1\rangle \\ &= \frac{1}{2}\Big[|0\rangle \Big(|1\rangle U_{(1,2)} |x,y\rangle + |0\rangle U_{(1,1)} |x,y\rangle \Big) + \\ &\quad + |1\rangle \Big(|1\rangle U_{(2,1)} U_{(1,2)} |x,y\rangle + |0\rangle U_{(2,1)} U_{(1,1)} |x,y\rangle \Big)\Big]\end{aligned} \tag{24}$$

where the position of the gate C indicates the control qubit used to apply $U_{(2,1)}$.

- Second, $|c_1\rangle$ is transformed based on Pauli–X gate:

$$\begin{aligned}
|\Phi_{2,2}\rangle &= (X \otimes \mathbb{1} \otimes \mathbb{1})\,|\Phi_{2,1}\rangle \\
&= \frac{1}{2}\Big[\,|1\rangle\,\Big(\,|1\rangle\,U_{(1,2)}\,|x,y\rangle + |0\rangle\,U_{(1,1)}\,|x,y\rangle\,\Big) + \\
&\quad + |0\rangle\,\big(\,|1\rangle\,U_{(2,1)}U_{(1,2)}\,|x,y\rangle + |0\rangle\,U_{(2,1)}U_{(1,1)}\,|x,y\rangle\,\big)\Big]
\end{aligned} \tag{25}$$

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

$$\begin{aligned}
|\Phi_2\rangle &= (C \otimes \mathbb{1} \otimes U_{(2,2)})\,|\Phi_{2,2}\rangle \\
&= \frac{1}{2}\Big[\,|1\rangle\,\Big(\,|1\rangle\,U_{(2,2)}U_{(1,2)}\,|x,y\rangle + |0\rangle\,U_{(2,2)}U_{(1,1)}\,|x,y\rangle\,\Big) + \\
&\quad + |0\rangle\,\Big(\,|1\rangle\,U_{(2,1)}U_{(1,2)}\,|x,y\rangle + |0\rangle\,U_{(2,1)}U_{(1,1)}\,|x,y\rangle\,\Big)\Big]
\end{aligned} \tag{26}$$

Notice that the entanglement performed in **Step 2.1** influences the entanglement in **Step 2.2**, and each trajectory describes a different transformation of $|x,y\rangle$. Equation 26 can be rewritten expressing the four basis states of the control register using natural numbers:

$$\begin{aligned}
|\Phi_2\rangle = \frac{1}{2}\Big[\; & |00\rangle\,U_{(2,1)}U_{(1,1)}\,|x,y\rangle \\
& + |01\rangle\,U_{(2,1)}U_{(1,2)}\,|x,y\rangle \\
& + |10\rangle\,U_{(2,2)}U_{(1,1)}\,|x,y\rangle \\
& + |11\rangle\,U_{(2,2)}U_{(1,2)}\,|x,y\rangle\,\Big] \\
= \frac{1}{\sqrt{4}} & \sum_{b=1}^{4} |b\rangle\,V_b\,|x,y\rangle
\end{aligned} \tag{27}$$

where V_b is the product of $d = 2$ unitaries $U_{(i,j)}$ for $i, j = 1, 2$. We can see that using 2 control qubits we generated 4 different quantum trajectories that correspond to 4 different transformations of data $|x,y\rangle$.

For instance, when $d = 3$ we have:

$$\begin{aligned}
|\Phi_3\rangle = \frac{1}{2}\Big[\, & |000\rangle\,U_{(3,1)}U_{(2,1)}U_{(1,1)}\,|x,y\rangle + |001\rangle\,U_{(3,1)}U_{(2,1)}U_{(1,2)}\,|x,y\rangle \\
& + |010\rangle\,U_{(3,1)}U_{(2,2)}U_{(1,1)}\,|x,y\rangle + |011\rangle\,U_{(3,1)}U_{(2,2)}U_{(1,2)}\,|x,y\rangle \\
& + |100\rangle\,U_{(3,2)}U_{(2,1)}U_{(1,1)}\,|x,y\rangle + |101\rangle\,U_{(3,2)}U_{(2,1)}U_{(1,2)}\,|x,y\rangle \\
& + |110\rangle\,U_{(3,2)}U_{(2,2)}U_{(1,1)}\,|x,y\rangle + |111\rangle\,U_{(3,2)}U_{(2,2)}U_{(1,2)}\,|x,y\rangle\,\Big] \\
= \frac{1}{\sqrt{8}} & \sum_{b=1}^{8} |b\rangle\,V_b\,|x,y\rangle
\end{aligned} \tag{28}$$

where each V_b is the product of 3 unitaries $U_{(i,j)}$.

Repeating this procedure d times with different control qubits results in the following quantum state:

$$|\Phi_d\rangle = \frac{1}{\sqrt{2^d}} \sum_{b=1}^{2^d} |b\rangle\,V_b\,|x,y\rangle = \frac{1}{\sqrt{2^d}} \sum_{b=1}^{2^d} |b\rangle\,|x_b, y_b\rangle \tag{29}$$

where each V_b is the product of d unitaries $U_{(i,j)}$ for $i = 1, \cdots, d$ and $j = 1, 2$.