Quantum Information and Computation, Vol. 24, No. 3&4 (2024) 0181–0209
© Rinton Press

ENSEMBLES OF QUANTUM CLASSIFIERS

EMILIANO TOLOTTI
Department of Information Engineering and Computer Science
University of Trento
via Sommarive 9, 38123 Povo, Trento, Italy
emiliano.tolotti@unitn.it

ENRICO ZARDINI
Department of Information Engineering and Computer Science
University of Trento
via Sommarive 9, 38123 Povo, Trento, Italy

ENRICO BLANZIERI
Department of Information Engineering and Computer Science
University of Trento
via Sommarive 9, 38123 Povo, Trento, Italy
Trento Institute for Fundamental Physics and Applications
via Sommarive 14, 38123 Povo, Trento, Italy

DAVIDE PASTORELLO
Department of Mathematics
Alma Mater Studiorum - Università di Bologna
Piazza di Porta San Donato 5, 40126 Bologna, Italy
Trento Institute for Fundamental Physics and Applications
via Sommarive 14, 38123 Povo, Trento, Italy

Received December 4, 2023
Revised February 27, 2024

In the current era, known as Noisy Intermediate-Scale Quantum (NISQ), encoding large amounts of data in the quantum devices is challenging and the impact of noise significantly affects the quality of the obtained results. A viable approach for the execution of quantum classification algorithms is the introduction of a well-known machine learning paradigm, namely, the ensemble methods. Indeed, the ensembles combine multiple internal classifiers, which are characterized by compact sizes due to the smaller data subsets used for training, to achieve more accurate and robust prediction performance. In this way, it is possible to reduce the qubit requirements with respect to a single larger classifier while achieving comparable or improved performance. In this work, we present an implementation and an extensive empirical evaluation of ensembles of quantum instance-based classifiers for binary classification, with the purpose of providing insights into their effectiveness, limitations, and potential for enhancing the performance of basic quantum models. In particular, three classical ensemble methods and three quantum instance-based classifiers have been taken into account here. Hence, the scheme that has been implemented (in Python) has a hybrid nature. The results (obtained on real-world datasets) have shown an accuracy advantage for the ensemble techniques with respect to the single quantum classifiers, and also an improvement in robustness. In fact, ensembles have proven effective not only in mitigating unsuitable data normalizations but also in reducing the impact of noise on quantum classifiers, enhancing their stability.

Keywords: quantum computing, quantum machine learning, ensemble methods, quantum classifiers, binary classification

1 Introduction

Quantum machine learning (QML) is a recent field of research, which aims at developing quantum algorithms for solving machine learning problems in a more efficient way than the classical counterparts [1]. If a sufficient number of fully connected qubits were available, a quantum advantage (with respect to classical supercomputers) could be achieved on different tasks and practical problems could be tackled effectively. However, the current era, known as Noisy Intermediate-Scale Quantum (NISQ) [2], is characterised by noisy devices with limited numbers of qubits. Additionally, in gate-based quantum devices, the impact of noise increases with the circuit's depth and size. As a consequence, encoding large amounts of data turns out to be challenging, and the quality of the results obtained is significantly affected by noise. To execute quantum algorithms on the current architectures, the size of the circuits must be reduced. In this way, despite the current limitations, practical problems could be addressed, making progress towards solving real-world challenges using quantum computing technologies.

Ensemble methods are a widely used machine learning technique that consists in combining the predictions of multiple models [3]. This approach aims at enhancing prediction accuracy and stability by exploiting model diversity. In the context of quantum computation, classical ensembles represent an effective way to reduce the computational requirements. Indeed, the internal models are typically characterized by compact sizes due to the smaller data subsets used for training. In practice, ensemble methods allow executing quantum algorithms on NISQ devices thanks to a more efficient resource usage with respect to single larger models (the quantum circuits involved are typically smaller).

Concerning hybrid schemes, Incudini et al. [11] have proposed and evaluated the performance of classical ensembles (bagging and boosting) of quantum neural networks for regression and classification tasks. In their work, they have shown the ensemble methods' ability of enhancing the performance of the base models while limiting the circuit size via an attribute bagging technique (indeed, for quantum neural networks, the circuit size is not affected by the number of training samples). However, their focus lies on trainable models. In order to investigate the behaviour of the ensemble methods in a cleaner context, where the variability is given only by the data instance sampling procedure, instance-based classification models and no attribute bagging technique are considered here. In this way, it is possible to better understand the effect of ensemble methods on the performance of the base models, and also their capability of mitigating noise and measurement sampling variance.

Quantum ensemble methods have been also developed. For instance, Schuld and Petruccione [4] and Abbas et al. [5] have proposed quantum ensemble classifiers based on Bayesian Model Averaging (BMA). The ensembles in question exploit non-trainable classifiers, under the assumption that a large ensemble of weak classifiers can achieve good performance. Instead, Araujo and da Silva have presented a quantum ensemble of trainable classifiers [6]. In particular, they have considered a superposition of quantum classifiers, with the classifiers being quantum neural networks. Macalauso et al. have proposed a quantum ensemble framework [7] that is based on bagging and is characterised by an exponential growth of the ensemble size at the price of a linear increase in the circuit depth. Regarding the work by Windridge and Nagarajan [8], a quantum-SVM-based attribute bootstrap aggregation is presented. In practice, a superposition of quantum decision hyperplanes is used to perform attribute selection. Eventually, Qin et al. [9] and Zhang and Wang [10] have proposed hybrid techniques to efficiently combine quantum classification algorithms, showing how the parallel combination of multiple variational quantum classifiers can outperform state-of-the-art classification methods.

In this work, we present an implementation and the empirical evaluation of ensembles of quantum instance-based classifiers. In detail, the scheme is hybrid, with classical ensemble methods and quantum classification algorithms. Regarding the ensemble methods, bootstrap [12], boosting [13], and stacking [14] have been considered. Concerning the quantum classifiers, three non-trainable classifiers have been considered, namely, a quantum cosine classifier [15], a quantum distance classifier [16], and a quantum k-nearest neighbors (quantum k-NN) classifier [17, 18]. The quantum algorithms employed in this work, like many other quantum machine learning algorithms, require the existence of a quantum random access memory (QRAM) [19] to achieve a speedup with respect to classical computation. Working prototypes of QRAMs have not been developed yet. However, this work does not focus on the potential quantum time advantage, but on the empirical evaluation of the considered scheme in terms of accuracy. To this end, the methods have been evaluated on a binary classification task on real-world datasets. The results have shown an accuracy advantage with respect to single classifiers and also an improvement in robustness. Indeed, the ensembles have shown the capability of mitigating both unsuitable data normalizations and circuit noise.

The article is structured as follows: Section 2 provides some background information; Section 3 introduces the hybrid scheme and presents the implementation details; Section 4 deals with the experiments performed and the results obtained; Section 5 concludes the work.

2 Background

In this section, some background information about quantum information and quantum machine learning is provided. Then, the algorithms considered in this work, which include classical ensemble methods and quantum classifiers, are introduced.

2.1 Quantum machine learning

Quantum computing leverages quantum mechanical principles like superposition and entanglement for computation. In 2013, exploiting quantum computations, Lloyd, Mohseni, and Rebentrost demonstrated an exponential speedup over classical clustering algorithms [20], sparking the interest in Quantum Machine Learning (QML). In quantum computing, qubits serve as the fundamental unit of information, analogously to classical bits. More precisely, a qubit is any quantum system that can be described in a 2-dimensional Hilbert space. The quantum states are in bijective correspondence with the projective rays in the Hilbert space. Moreover, a qubit's state, represented by a unit vector in $\mathbb{C}^2$, can be in a superposition of its basis states $|0\rangle$ and $|1\rangle$. In the Dirac notation, this is written as

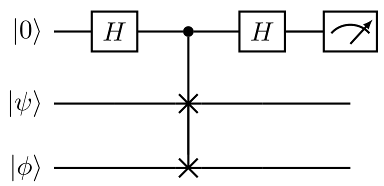
$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{1}$$

where $\alpha, \beta \in \mathbb{C}$ are called amplitudes and must satisfy the constraint $|\alpha|^2 + |\beta|^2 = 1$. Actually, the amplitudes values cannot be directly observed. In fact, when a qubit is measured, one of the basis states ($|0\rangle$ or $|1\rangle$) is obtained, with probabilities $|\alpha|^2$ and $|\beta|^2$, respectively. In practice, by operating on composite systems of qubits (*registers*), characterised by state superposition, it is possible to perform parallel operations, and the presence of quantum entanglement is the key enabling quantum advantages with respect to classical computations.

In quantum circuits, which represent the most common quantum computation model, the computation is performed by means of quantum gates, which implement unitary operations. In particular, there exist universal sets of quantum gates, in the sense that any n-qubit quantum gate can be implemented up to arbitrary precision by using gates taken from these sets. For example, the set $\{H,\ P_\phi,\ CNOT\}$, where H is the Hadamard gate, P_ϕ is the phase shift gate, and $CNOT$ is the controlled-NOT gate, is universal [21].

A non-trivial problem in quantum computing (still unsolved) is how to encode data into quantum states. In particular, there are two main strategies: basis encoding and amplitude encoding. In basis encoding, data is encoded into the computational basis as classical bits. Hence, given a binary string $x = (b_1, ..., b_n)$, with $b_i \in \{0, 1\}$, x is encoded as $|x\rangle = \bigotimes_{i=1}^{n} |b_i\rangle$. In addition, by exploiting superposition, it is possible to represent a dataset $X = \{x_1, ..., x_m\}$ as $|X\rangle = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} |x_i\rangle$. Instead, in amplitude encoding, data is stored into the amplitudes of quantum states. Given a normalized data vector x, i.e., a vector $x \in \mathbb{C}^d$ such that $||x|| = 1$, it is encoded as $|x\rangle = \sum_{i=1}^{d} x_i\,|i\rangle$. On the one hand, within the amplitude encoding framework, d-dimensional data vectors can be encoded in $log(d)$ qubits. On the other hand, amplitudes are not directly observable; therefore, a repeated sampling of the qubits state is necessary in order to estimate the amplitudes values.

As stated previously, in QML, quantum computing procedures are exploited to improve machine learning algorithms. An important procedure is the so-called SWAP test [22], which allows estimating the dot product of two data vectors. Specifically, the corresponding quantum circuit, which is also the basis of two quantum classifiers employed in this work, is the following:



Given the quantum states $|\psi\rangle$ and $|\phi\rangle$ (that can be also n-qubit states), the probability of measuring 0 on the ancillary qubit after performing the SWAP test is $\mathbb{P}(0) = \frac{1}{2} + \frac{1}{2}|\langle\psi|\phi\rangle|^2$. Hence, by executing the circuit multiple times, it is possible to estimate the squared inner product of the quantum states, which corresponds to the squared dot product of the data vectors encoded in the amplitudes of $|\psi\rangle$ and $|\phi\rangle$. In detail, in order to obtain an estimate up to an error ϵ, the number of circuit repetitions required is $O(\frac{1}{\epsilon^2})$.

Eventually, many QML algorithms, including one classifier used in this work, assume the existence of a quantum random access memory [19] for an efficient state preparation. The idea is to query a superposition of addresses to retrieve a superposition of $N = 2^n$ memory cells in time $O(n)$. Some physical proposals have been suggested, but there is no working implementation yet and there are still doubts on its actual feasibility.

2.2 Ensemble techniques

Ensemble learning is a machine learning paradigm based on the intuition that combining multiple models is more effective than using a single model [3]. Indeed, weak base models can suffer from high bias or high variance, but their combination can produce a strong and more robust learner with good performance. The ensemble techniques taken into account in this work are bootstrap, boosting and stacking.

2.2.1 Bootstrap aggregating

The bootstrap aggregating algorithm, also known as bagging, is a simple ensemble scheme based on the bootstrap sampling procedure proposed by Breiman [12]. In practice, homogeneous internal models are independently trained on sets obtained from the training set by random sampling with replacement. This means that an element can be sampled multiple times, and subsequent samplings are independent (i.i.d. samples). For a regression task, the predicted value is the average of the models outputs. Instead, for a classification task, a majority voting scheme is used. In particular, in the case of a binary classification task with labels in $\{-1, +1\}$, the majority voting can be expressed as

$$y(x) = \operatorname{sign}\left(\sum_{i=1}^{M} m_i(x)\right), \tag{2}$$

where M is the number of internal models, and m_i is the i-th internal model. By introducing diversity in the data through the bootstrapping process, bagging decreases the variance, improving the performance and the robustness. In addition, it is a scalable and parallelizable algorithm. Indeed, both the ensemble building (the sampling of the training instances for the classifiers is independent) and the prediction step can be parallelized.

2.2.2 Boosting

Boosting is a homogeneous ensemble model based on an iterative training procedure. In detail, at each iteration, a weak classifier is trained on data sampled from the training set according to a distribution that is influenced by the performance of the previous iteration classifiers. The boosting algorithm used in this work is one of the most famous ones, namely, AdaBoost [13]. In AdaBoost, which stands for adaptive boosting, a weak classifier is trained on a subset of the training set and used to predict the class (let us restrict to classification) of all training samples. The classification errors are then used to increase the weight of the misclassified instances for the next iteration sampling and to compute a weight for the classifier inside the final aggregated model. Indeed, the aggregation strategy is a weighted average of the internal models. In particular, for a binary classification task with labels in $\{-1, +1\}$, the ensemble prediction can be written as

$$y(x) = \text{sign}\left(\sum_{i=1}^{M} \alpha_i m_i(x)\right), \tag{3}$$

where α_i is the weight of the i-th internal model m_i. Boosting decreases the bias, while also reducing the variance. Nevertheless, the training procedure, which is iterative, cannot be parallelized. Instead, at prediction time, the output of the internal classifiers can be computed in parallel.

2.2.3 Stacking

Stacking is a heterogeneous ensemble technique [14]. In particular, different internal models are trained and evaluated on the training set using a k-fold cross validation technique, obtaining a prediction for every internal classifier - training point pair. These predictions are then used as the training set of a meta-classifier that combines the output of the internal classifiers into a final prediction. Specifically, the internal classifiers are trained on the full training set, and the stacking classifier prediction can be expressed as

$$y(x) = m_{\text{meta}}(m_1(x), ..., m_M(x)), \tag{4}$$

where m_{meta} is the meta-classifier model. The main advantage of stacking is that, by combining diverse classifiers based on different assumptions, the performance with respect to the single classifiers improve. In addition, the internal classifiers can be trained and executed in parallel.

2.3 Quantum classifiers

The quantum classification algorithms considered are a quantum cosine classifier, a quantum distance classifier, and a quantum k-nearest neighbors classifier. Their details are provided below.

2.3.1 Quantum cosine classifier

The quantum cosine classifier proposed by Pastorello and Blanzieri [15] is an algorithm for binary classification based on the cosine similarity of data vectors. In particular, the classification function implemented by the classifier is the following:

$$y(x) = \operatorname{sign}\left(\sum_{i=0}^{N-1} y_i \cos(x_i, x)\right), \tag{5}$$

where N is the number of training samples, x_i is the feature vector of the i-th sample, $y_i \in \{-1, +1\}$ is the corresponding label, and

$$\cos(x, y) = \frac{x \cdot y}{\|x\| \|y\|}. \tag{6}$$

Concerning the data encoding scheme, the classifier uses the amplitude encoding for the feature vectors x_i, which must be unit-norm normalized, and the basis encoding for the binary labels y_i, which are mapped to the domain $\{1, 0\}$ according to

$$l_i = \frac{1 - y_i}{2}. \tag{7}$$

The initial state is defined as

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|\psi_x\rangle |0\rangle + |\psi\rangle |1\rangle) \in \mathcal{H}_n \otimes \mathcal{H}_d \otimes \mathcal{H}_l \otimes \mathcal{H}_a, \tag{8}$$

where

$$|\psi_x\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle |x_i\rangle |l_i\rangle \in \mathcal{H}_n \otimes \mathcal{H}_d \otimes \mathcal{H}_l, \text{ and}$$
$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle |x\rangle |-\rangle \in \mathcal{H}_n \otimes \mathcal{H}_d \otimes \mathcal{H}_l.$$

In detail, the classifier assumes the existence of a QRAM in order to have an efficient preparation of the initial state. Then, a SWAP test on the states $|+\rangle$ and $|\Psi\rangle$ is performed, and the probability of measuring 1 on the SWAP test ancillary qubit turns out to be

$$\mathbb{P}(1) = \frac{1}{4}(1 - \langle\psi_x|\psi\rangle), \tag{9}$$

with

$$\langle\psi_x|\psi\rangle = \frac{1}{\sqrt{2}N} \sum_{i=0}^{N-1} l_i \cos(x_i, x). \tag{10}$$

Eventually, the predicted label is given by

$$y(x) = \operatorname{sign}(1 - 4\mathbb{P}(1)). \tag{11}$$

In practice, the quantum circuit requires 3 qubits for the SWAP test, $\lceil \log_2 N \rceil$ qubits for the index register ($\mathcal{H}_n$), $\lceil \log_2 D \rceil$ qubits for the feature register ($\mathcal{H}_d$, with D being the number of features) and 1 qubit for the binary labels ($\mathcal{H}_l$). If a QRAM is available, the algorithm has a time complexity of $O(\log(ND))$ and a space complexity of $O(\log(ND))$ qubits.

2.3.2 Quantum distance classifier

The quantum distance classifier proposed by Schuld et al. [16] is a binary classification algorithm based on the squared euclidean distance of feature vectors. In this work, a slightly modified version is considered. Specifically, the classification function implemented by the classifier is defined as

$$y(x) = \text{sign}\left(\sum_{i=0}^{N-1} y_i(1 - \frac{1}{4}||x_i - x||^2)\right), \tag{12}$$

where N is the number of training samples, x_i is the feature vector of the i-th sample, and $y_i \in \{-1, +1\}$ is the corresponding label.

As in the quantum cosine classifier, the amplitude encoding is used for the features vector, the basis encoding is used for the binary labels (the mapping is given by Eq. (7)), and the presence of a QRAM is assumed. The initial state of the circuit is defined as

$$|\Psi\rangle = \frac{1}{\sqrt{2N}} \sum_{i=0}^{N-1} |i\rangle \left(|0\rangle |x\rangle + |1\rangle |x_i\rangle\right) |l_i\rangle \in \mathcal{H}_n \otimes \mathcal{H}_a \otimes \mathcal{H}_d \otimes \mathcal{H}_l. \tag{13}$$

In particular, the quantum circuit consists of a Hadamard gate and two qubits measurements, with the second one being a conditional measurement; thus, the circuit complexity is constant. The probability of obtaining $k \in \{0, 1\}$ in the second measurement, which is performed on the label qubit, is equal to

$$\mathbb{P}(k) = \frac{1}{4Np_0} \sum_{i:l_i=k} ||x + x_i||^2, \tag{14}$$

where p_0 is the probability of obtaining 0 in the first measurement. Since the data vectors are characterised by unit norm, the following relationship holds:

$$\frac{1}{4N} \sum_i ||x + x_i||^2 = \frac{1}{N} \sum_i (1 - \frac{1}{4}||x - x_i||^2). \tag{15}$$

Hence, the predicted label is given by

$$y(x) = \text{sign}\left(\mathbb{P}(0|0) - \frac{1}{2}\right). \tag{16}$$

Basically, the quantum circuit needs 1 ancillary qubit ($\mathcal{H}_a$), $\lceil \log_2 N \rceil$ qubits for the index register ($\mathcal{H}_n$), $\lceil \log_2 D \rceil$ qubits for the feature register ($\mathcal{H}_d$, with D being the number of features), and 1 qubit to encode the binary labels ($\mathcal{H}_l$). Assuming the availability of a QRAM, the algorithm has a time complexity of $O(\log(ND))$ and a space complexity of $O(\log(ND))$ qubits.

2.3.3 Quantum k-nearest neighbors classifier

The quantum k-nearest neighbors classifier proposed by Afham et al. [17] and Ma et al. [18] is a quantum version of the k-nearest neighbors algorithm [23], which is one of the simplest algorithms for multiclass classification in machine learning. In particular, the quantum k-NN in question is based on the notion of fidelity of quantum states. Indeed, the algorithm selects the k nearest neighbors based on the fidelity of the states encoding the training and test feature vectors, with the fidelity F being defined as

$$F(|\psi\rangle, |\phi\rangle) = |\langle\psi|\phi\rangle|^2. \tag{17}$$

More in detail, the algorithm exploits the amplitude encoding for the data features, and the initial state is defined as

$$|\Psi\rangle = |0\rangle |x\rangle |\psi_x\rangle \in \mathcal{H}_a \otimes \mathcal{H}_d \otimes \mathcal{H}_d \otimes \mathcal{H}_n, \tag{18}$$

where

$$|\psi_x\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |x_i\rangle |i\rangle \in \mathcal{H}_d \otimes \mathcal{H}_n,$$

N is the number of training samples, and x_i is the feature vector of the i-th sample. Regarding the quantum circuit, it consists of a SWAP test and two measurements. Specifically, the first measurement is performed on the SWAP test ancillary qubit ($\mathcal{H}_a$), while the second one is performed on the index register $|i\rangle$. By iterating this procedure, it is possible to estimate, for each index i, the quantity $\mathbb{Q}(i)$, which is defined as

$$\mathbb{Q}(i) = \mathbb{P}(i|0) - \mathbb{P}(i|1) = \frac{2(F_i - \langle F\rangle)}{N(1 - \langle F\rangle^2)}, \tag{19}$$

where F_i is the fidelity of the quantum states $|x\rangle$ and $|x_i\rangle$, and $\langle F\rangle$ is the average value of F_i over i. Eventually, the k nearest neighbors are retrieved by classically sorting the training data according to $\mathbb{Q}(i)$, and the predicted label is obtained through a majority voting.

In practice, the circuit requires 1 ancillary qubit for the SWAP test ($\mathcal{H}_a$), $\lceil \log_2 N \rceil$ qubits for the index register ($\mathcal{H}_n$), and $2 \lceil \log_2 D \rceil$ qubits for the feature registers ($\mathcal{H}_d$, with D being the number of features). Assuming the presence of a QRAM, the circuit time complexity is $O(\log(ND))$ and the space complexity is $O(\log(ND))$ qubits.

3 Ensembles of Quantum Classifiers

In this work, a hybrid scheme characterised by classical ensembles and quantum internal classifiers is employed. In practice, the quantum instance-based classifiers described in Section 2.3 are used as internal models of the ensemble methods presented in Section 2.2. A high-level view of the interaction between classical and quantum components is shown in Figure 1. First of all, the input data is classically processed to produce the input for the quantum models. Then, after the quantum encoding of the classical information, the quantum circuits are run multiple times, with final measurements, in order to obtain sufficiently precise estimates of the output quantities. Eventually, the output of the quantum models is classically post-processed to either carry on the training procedure or provide the final output.

The hybrid scheme allows employing quantum classification algorithms in homogeneous and heterogeneous ensemble techniques taken from the literature. In this way, it is possible to analyse the advantages in accuracy and robustness of using quantum classifiers in ensemble schemes while being compatible with the hardware limitations of current architectures, which do not allow efficient quantum implementations of ensemble techniques yet.

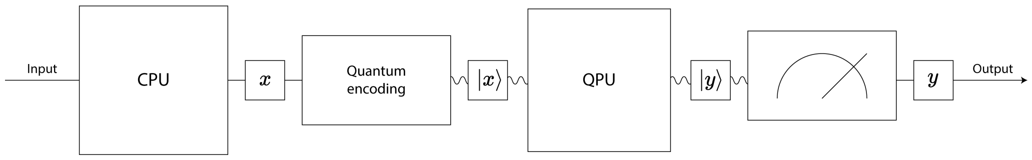


Fig. 1. High-level view of the interaction between classical and quantum components in the hybrid scheme employed.

3.1 Implementation

The hybrid scheme has been developed in Python language using Qiskit [24], the open-source SDK provided by IBM for building and running quantum circuits either on quantum hardware [25] or in simulation (the code is available at `https://github.com/emiliantolo/ensembles-quantum-classifiers`). In this work, the high-performance Aer simulator has been used for the execution of the algorithms. It is also worth mentioning that, although the scheme is theoretically valid for multiclass classification tasks, the code provided here supports only binary classification. Additional details about the implementation of the models are provided in the following.

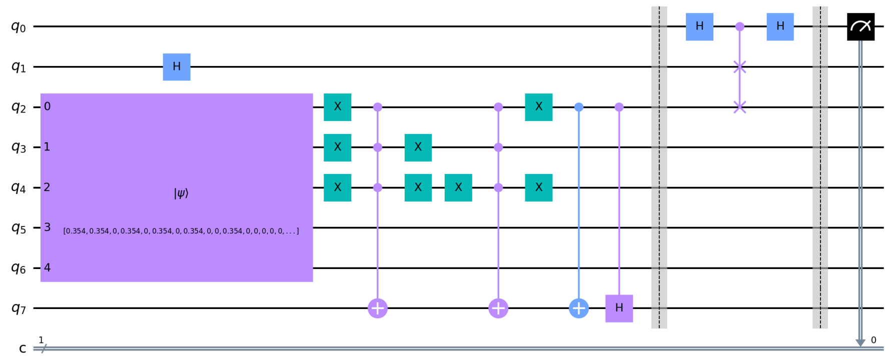
3.1.1 Ensemble techniques

Ensemble methods have been developed form scratch, ensuring a standard implementation. Regarding bootstrap (Section 2.2.1), the ensemble is built by sampling with replacement N subsets of S elements each from the training set; these subsets are then used as the training sets of the N internal quantum classifiers. Concerning boosting (Section 2.2.2), N training iterations are performed. In detail, at each step, a classifier is trained on a subset of S elements sampled from the training set according to the distribution determined by the previous iterations; then, the classifier is used to predict the training set labels, allowing the computation of the classifier weight in the ensemble and the definition of the new distribution. It is worth mentioning that an $\epsilon = 1e - 10$ value has been added in the computation of errors, in order to avoid divisions by zero. Eventually, for stacking, a k-fold cross validation procedure is run on the training set for each selected internal model, obtaining a prediction for each training instance; these predictions are then used as the training set for the meta-classifier, while the internal models are trained on the full training set. In particular, the meta-model used in this work takes as input not only the predicted output classes, but also the prediction confidences of the internal models. Concerning the prediction step, it is performed according to Eqs. (2), (3), and a variation of (4), respectively.

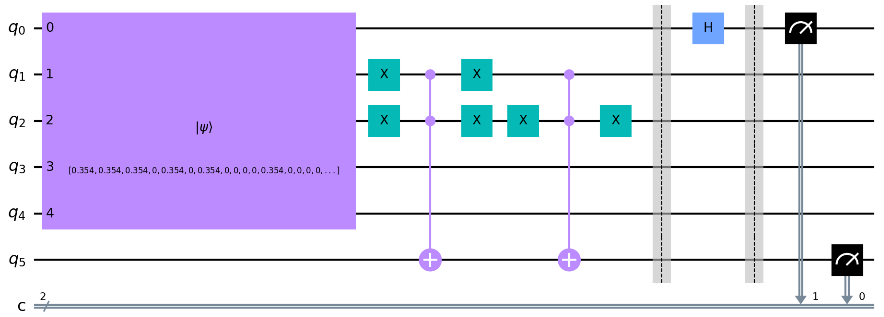
3.1.2 Quantum classifiers

First of all, the implementation of the quantum cosine classifier described in Section 2.3.1 has been taken from the work by Zardini et al. [26]. An example circuit for a toy dataset is shown in Figure 2a, and additional details about the implementation can be found in the original article.

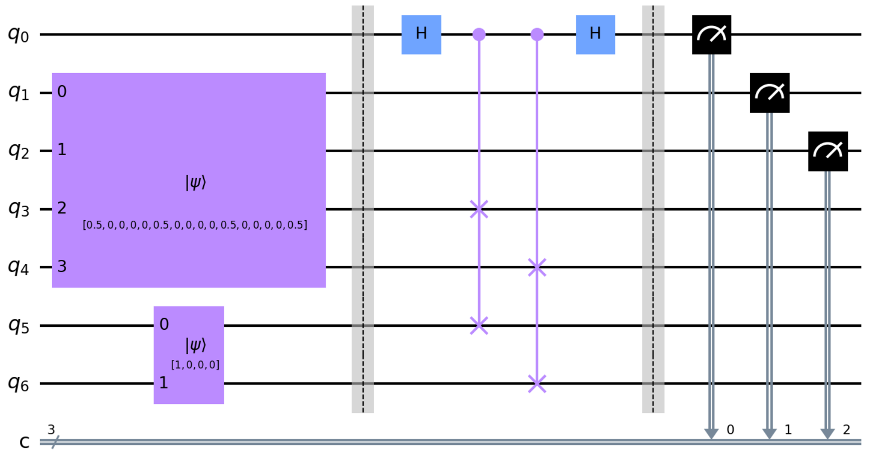
Instead, the quantum distance classifier illustrated in Section 2.3.2 has been implemented from scratch. An example circuit for the same toy dataset is provided in Figure 2b. In detail, the circuit is initialized by computing and directly setting the amplitudes of all qubits except



(a)



(b)



(c)

Fig. 2. Quantum circuit example for the quantum cosine classifier (a), the quantum distance classifier (b), and the quantum k-NN classifier; the dataset and the test instance considered are $X = \{([1,0,0,0],-1),([0,1,0,0],-1),([0,0,1,0],1),([0,0,0,1],1)\}$ and $x = [1,0,0,0]$, respectively.

the one (q_6) used to encode the training labels. Indeed, the labels are subsequently encoded in the circuit by applying NOT (X) and multi-controlled NOT gates. Then, a Hadamard gate (H) is applied to the ancillary qubit (q_0), and the state of the ancillary and the label qubits is measured. It is worth noting that, at the time of running the experiments, the conditional measurement required by the algorithm was not supported by Qiskit. Hence, all the iterations in which the outcome of the first measurement is 1 must be discarded when computing the probability estimate. In practice, if the data is standardized, the probability of obtaining 0 is around 0.5; otherwise, it is larger than 0.5.

Concerning the quantum k-NN described in Section 2.3.3, it has also been implemented from scratch. An example circuit for the same toy dataset is displayed in Figure 2c. In practice, the circuit is initialized by directly setting the amplitudes of two states: the state encoding the training set (q_1-q_4), and the state encoding the test instance (q_5-q_6). After that, a SWAP test is applied to the features registers of the two states (q_3-q_4 and q_5-q_6), with the ancillary qubit (q_0) as control qubit. Eventually, the state of the ancillary qubit and the state of the index register (q_1-q_2) are measured.

4 Empirical Evaluation

This section deals with the methods taken into account, the experimental setup used, the datasets considered, and the results obtained.

4.1 Methods and experimental setup

The ensemble techniques and the quantum classifiers considered in this work are summarised in Tables 1 and 1b, where *quantum_3NN* represents the quantum k-NN model with $k = 3$. Instead, Table 1c lists all the data normalization techniques taken into account. In particular, *none* corresponds to no normalization, *std* stands for standardization, and *minmax* is the so-called min-max normalization. More in detail, the standardization of the j-th feature of the i-th training instance in the $\{x^1, ..., x^n\}$ training set is defined as

$$\mathrm{std}(x^i_j) = \frac{x^i_j - \mu(x^1_j, ..., x^n_j)}{\sigma(x^1_j, ..., x^n_j)}, \tag{20}$$

where $\mu(x^1_j, ..., x^n_j)$ and $\sigma(x^1_j, ..., x^n_j)$ are respectively mean and standard deviation of the j-th feature over the training set. The corresponding min-max normalization is given by

$$\mathrm{minmax}(x^i_j) = \frac{x^i_j - \min_{i=1,...,n} x^i_j}{\max_{i=1,...,n} x^i_j - \min_{i=1,...,n} x^i_j}, \tag{21}$$

where $\min_{i=1,...,n} x^i_j$ and $\max_{i=1,...,n} x^i_j$ are the minimum and maximum feature values of the j-th feature in the training set. Hence, after the standardization, the features have zero mean and standard deviation equal to one, while, after the min-max normalization, they belong to the $[0, 1]$ interval (the test features are clipped to 0 or 1, if they exceed the interval edges).

All ensemble methods have been evaluated with all data normalization techniques. In addition, bootstrap and boosting have been evaluated with all quantum classifiers, also varying the number of internal classifiers (N) and the number of training samples per classifier (S). Instead, for stacking, the configuration reported in Table 2 has been used.

Table 1. Ensemble techniques (a), quantum classifiers (b), and normalization techniques (c) considered.

(a)

Ensemble techniques
bootstrap
boosting
stacking

(b)

Quantum classifiers
quantum_cosine
quantum_distance
quantum_3NN

(c)

Normalization techniques
none
std
minmax

Table 2. Stacking configuration tested.

Classifier	**Normalization**
Internal	
quantum_cosine	std
quantum_distance	std
quantum_1NN	minmax
quantum_3NN	minmax
Meta	
quantum_5NN	none

Regarding the quantum models, as stated in Section 3.1, the Aer simulator provided by Qiskit has been used for the execution of the algorithms. In particular, the number of measurements, also known as shots, has been set to 8192, which corresponds to the maximum allowed number of shots on real quantum IBM devices.

Eventually, it is worth highlighting that three simulation methods have been considered here: *statevector*, *local simulation* and *noisy simulation*. In the first case, the results are obtained by processing the final state vector of the circuit; hence, the probability estimates are exact. In the second method, the behaviour of the real machine is emulated by sampling state counts from the final probability distribution of the circuit, where no noise model has been taken into account; this method represents a best-case scenario. Eventually, the third method adds a noise model to the simulation, imitating the *ibm_brisbane* system with a 127-qubit *Eagle r3* QPU. The noisy simulation enables experiments that mimic the execution of real NISQ devices by utilizing the same properties, such as qubit coupling map, basis gates, and incorporating quantum and readout errors. Specifically, in the experiments, we have employed the gate fidelity values obtained from the latest calibration of the IBM quantum processor (on date 2023-11-15 16:20:29). However, due to the computational expensiveness of the noisy simulation, only a limited number of experiments have been performed utilizing *noisy simulation* in Section 4.3.4.

All results (if not specified differently) have been collected using a Monte Carlo (leave one group out) cross-validation technique [27], with 10 independent runs and a "80% training" - "20% validation" dataset split. In particular, for each dataset split, the values of the normalization techniques parameters have been computed on the training set samples.

4.2 Datasets

In the experiments, 11 datasets taken from the work by Zardini et al. [26] have been used. These datasets, whose properties are reported in Table 3, can be downloaded from the GitHub repository associated to the just mentioned article [28]. In particular, the original versions of these datasets come from the UCI Machine learning Repository [29], and most of them have been preprocessed to make them suitable for a binary classification task. It is also

Table 3. Datasets considered.

Name	# samples	# features	Class balance
iris_setosa_versicolor	100	4	balanced (50/50)
iris_setosa_virginica	100	4	balanced (50/50)
iris_versicolor_virginica	100	4	balanced (50/50)
vertebral_column_2C	310	6	unbalanced (100/210)
seeds_1_2	140	7	balanced (70/70)
ecoli_cp_im	220	7	unbalanced (77/143)
glasses_1_2	80	9	almost balanced (42/38)
breast_tissue_adi_fadmasgla	71	9	unbalanced (49/22)
breast_cancer	80	9	almost balanced (44/36)
accent_recognition_uk_us	80	12	unbalanced (63/17)
leaf_11_9	30	14	almost balanced (14/16)

worth mentioning that, for the algorithms tested in this work, the considered datasets lead to quantum circuits with sizes of at most 15 qubits, which can be simulated in a reasonable time.

4.3 Results

The results obtained are presented and discussed in the following sections.

4.3.1 Bootstrap and boosting hyperparameters

Bootstrap and boosting require to set two hyperparameters, namely, the number of internal classifiers N and the number of training samples for each classifier S. These parameters have a heavy impact on the performances of the ensembles. Hence, a grid search has been used in order to find the best configuration. In particular, the values taken into account are $N = [5, 10, 30, 50]$ and $S = [6, 8, 10, 20]$. The related plots are provided in Appendix 1, Figures A.1 and A.2.

As expected, the performance improve and the variance decreases by increasing the number of internal classifiers. Instead, for a fixed number of internal models N, the performance do not improve by increasing the number of training samples for each classifier. It is also possible to notice that, for even numbers of training samples, the accuracy is almost constant, whereas it drops for odds values. Indeed, odd numbers of training samples imply that the training sets cannot be balanced, since it is a binary classification task. This affects especially the cosine and the distance classifiers, because their prediction is an average value computed over all the training samples; instead, the k-NN classifier is less affected by this issue. Moreover, the optimal number of training samples has turned out to be dataset-dependent for the cosine and distance classifiers, while the accuracy of the k-NN classifier has always improved by increasing S (as expected). The considerations provided for bootstrap about the number of internal classifier and the number of training samples for each classifier hold also for boosting. Actually, the reduction in variance turns out to be more evident in this case.

In the end, the configuration $N = 30$, $S = 8$ has been chosen, since $N = 30$ already allows achieving good performance and $S = 8$ represents a good tradeoff between accuracy and runtime (the size of the index register is three qubits).

4.3.2 Performance comparison

The results achieved by all combinations of ensemble, base classifier, and normalization technique are shown in Figure 3. In detail, each box contains 110 points (one for each run on

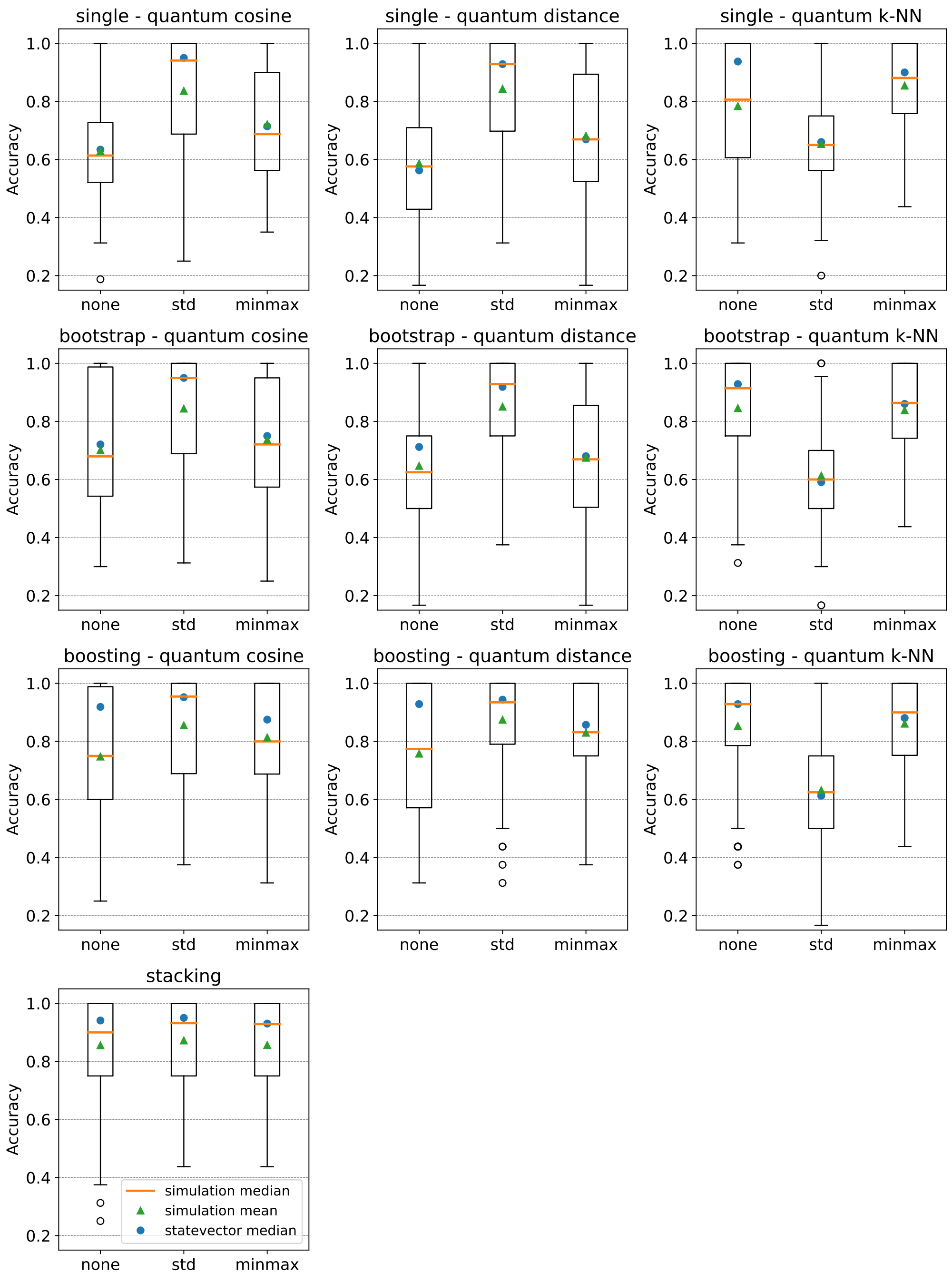


Fig. 3. Accuracy achieved by *local simulation* with 8192 shots, for all combinations of ensemble, base classifier, and data normalization technique. Each box contains 110 points (one for each run on each dataset).

each dataset), with each data point being the accuracy obtained by *local simulation* with 8192 shots. In addition, the orange line represents the median, the green triangle corresponds to the mean, while the blue circle represents the median of a *statevector* simulation executed on the same data. Focusing on the quantum classifiers executed without ensembles (first row), the quantum cosine and the quantum distance classifiers achieve the best performance when the input data is standardized. Instead, the quantum k-NN performs better with a min-max normalization. Indeed, in this quantum k-NN, the samples are sorted according to the squared cosine similarity with respect to the test instance; therefore, the features should belong to the same semi-axis to achieve good results.

Concerning bootstrap (second row), the introduction of the ensemble technique leads to a performance improvement for the quantum cosine classifier, while the behaviour with respect to the different normalization techniques remains unchanged. Similar considerations hold for the quantum distance classifier, with the best performance being achieved with the standardization of input data. Instead, the bootstrap ensemble with the quantum k-NN performs the best when the input data is not normalized; in this case, there is also a performance improvement with respect to using the single classifier, whereas the performance tend to worsen when a data normalization technique is used. Regarding boosting (third row), the results achieved with the quantum cosine classifier turn out to be better than those obtained by bootstrap and by the single classifier. Indeed, the accuracy is visibly better for no and min-max normalizations, and the variance is lower overall. The best results are still achieved with the standardization of input data. These considerations hold also for the quantum distance classifier, while, for the quantum k-NN classifier, boosting and bootstrap turn out to be almost equivalent (there is a little performance improvement). Eventually, the stacking ensemble (fourth row) shows very consistent performance regardless of the data normalization technique employed. Indeed, each internal classifier applies its own normalization technique.

Let us consider now only the best-performing normalization technique in *local simulation* for each "ensemble"-"classifier" pair (see Figure A.3 in Appendix 1 for an easier visual comparison). Focusing on the classifiers without ensembles, the cosine and the distance classifiers with standardization of input data perform similarly (the former being slightly better), and they both outperform the quantum k-NN with min-max normalization (the *statevector* median with no data normalization is also reported for the k-NN, since it is better than the min-max one). Concerning bootstrap, the classifier achieving the best results (also with respect to the single classifiers) turns out to be the quantum cosine classifier with data standardization. Indeed, both the quantum distance classifier with data standardization and the quantum k-NN without data normalization show a lower variance, but also a lower median accuracy. Regarding boosting, the configuration with cosine distance classifier and data standardization has the best median accuracy among all methods tested, while the version with distance classifier and data standardization has the best mean accuracy. Eventually, stacking shows good median and mean performance.

The effect of introducing bootstrap and boosting with respect to using single quantum classifiers is shown more in detail in Figure 4. Concerning bootstrap (plot on the left), as stated previously, the performance tend to improve for both the quantum cosine and the quantum distance classifier, whereas they tend to worsen for the quantum k-NN. In terms of normalization technique, independently from the classifier used, a significant improvement can

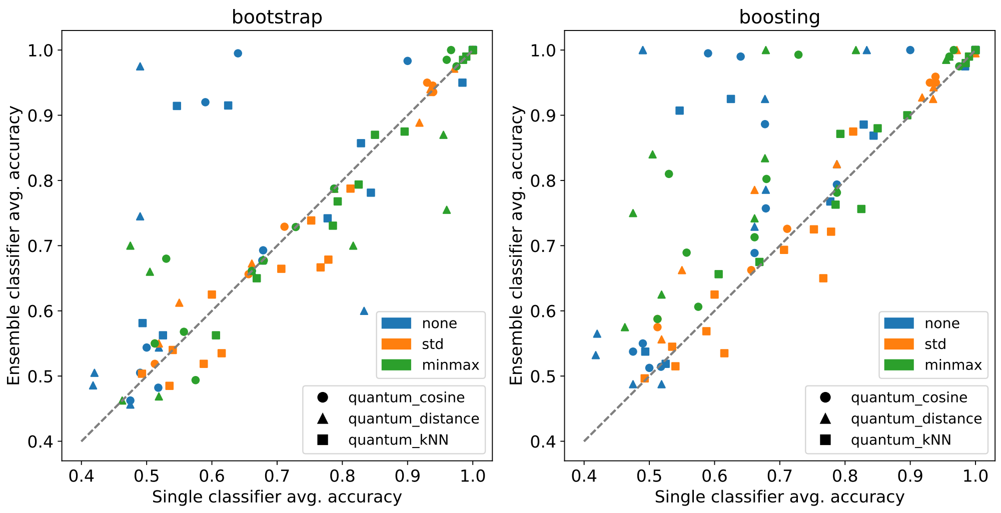


Fig. 4. Comparison between ensemble techniques and single classifiers in terms of average accuracy over 10 Monte Carlo runs. The left plot refers to bootstrap, the right plot to boosting. In both cases, all normalization techniques are taken into account, each data point corresponds to a different dataset, and *local simulation* with 8192 shots has been used.

be observed for no data normalization. Instead, there is a little advantage for standardization, and an overall neutral effect for min-max normalization. Regarding boosting (plot on the right), the performance of both the quantum cosine and the quantum distance classifiers improve significantly with the introduction of the ensemble technique, whereas there is not a clear benefit for the quantum k-NN. As in the previous case, standardization is the only normalization technique not taking advantage of the ensemble usage.

It is worth noting that both the quantum cosine and quantum distance classifiers are significantly affected by the class balance in the training data. Indeed, their predictions are given by weighted sums over the training set. The effect of the class balance on the performance of these classifiers is analysed in Appendix B.

4.3.3 Measurement sampling

As expected, the performance obtained with *local simulation* differ from the ideal ones, represented by *statevector*. Indeed, the repeated sampling from the final probability distribution of the qubits states inevitably introduces some uncertainty, which may lead to a wrong label prediction (for quantum cosine and quantum distance classifiers) or a wrong nearest neighbors ranking (for the quantum k-NN). Figure 5 shows the impact of sampling on the performance of the various methods. Focusing on the single classifiers (upper left), only in a few cases (located in the low-accuracy region) *local simulation* turns out to be better than *statevector*; in addition, the main outliers are all related to the quantum k-NN with no data normalization. Looking at bootstrap (upper right), almost all points are located near the main diagonal, with outliers mainly related to quantum cosine and quantum distance classifiers without data

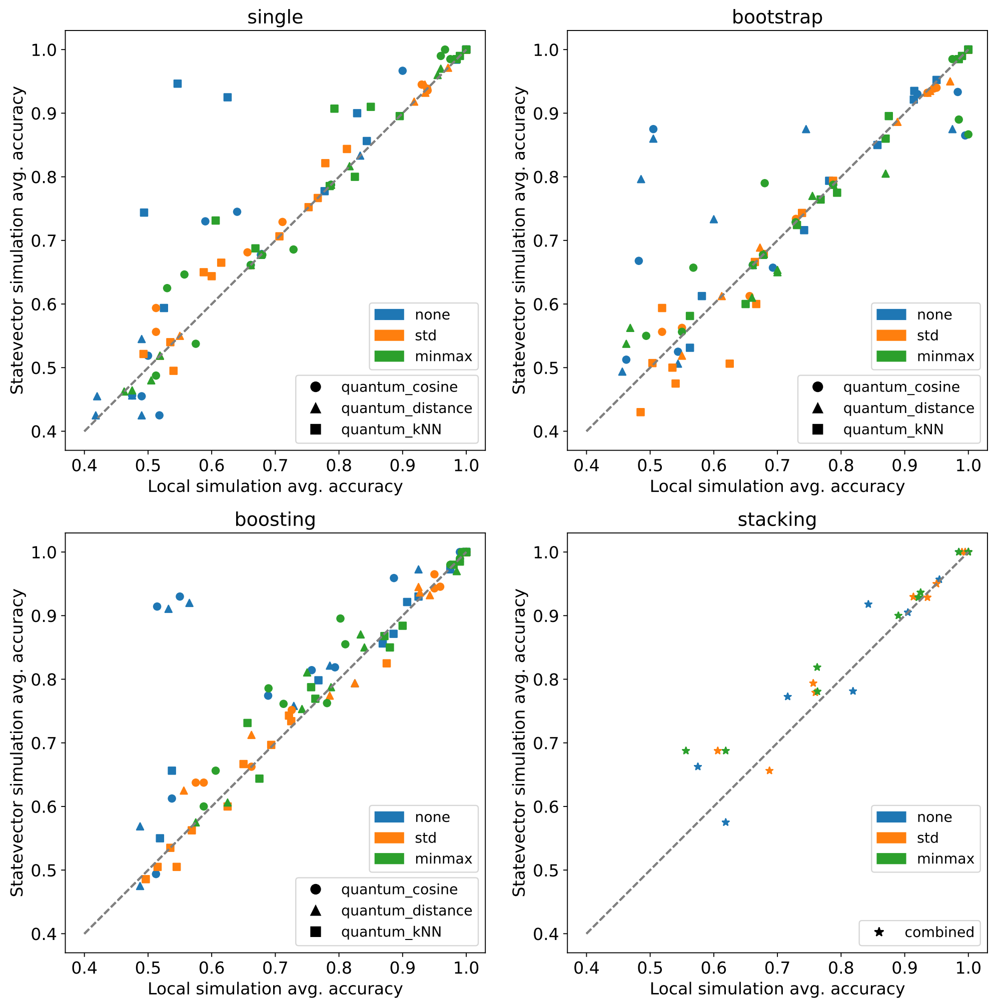


Fig. 5. Comparison between *statevector* and *local simulation* with 8192 shots in terms of average accuracy over 10 Monte Carlo runs. All combinations of internal classifier and normalization technique are shown for each ensemble technique, and each data point corresponds to a different dataset.

normalization. This suggests that the probability values are close to the decision threshold when the data is not normalized. Concerning boosting (bottom left), the performance drop when using *local simulation* turns out to be evident. This might be related to the ensemble building process; indeed, the uncertainty in the predictions of the internal models might lead to the computation of suboptimal boosting parameters. In addition, the outliers have the same properties of the bootstrap's ones. Eventually, for stacking (bottom right), it is possible to notice a performance drop in the low-middle accuracy region. To better understand the impact of sampling on each configuration, the distribution of the accuracy difference between *local simulation* and *statevector* for each combination of ensemble technique, quantum classifier, and data normalization is shown in Figure A.4 in Appendix 1.

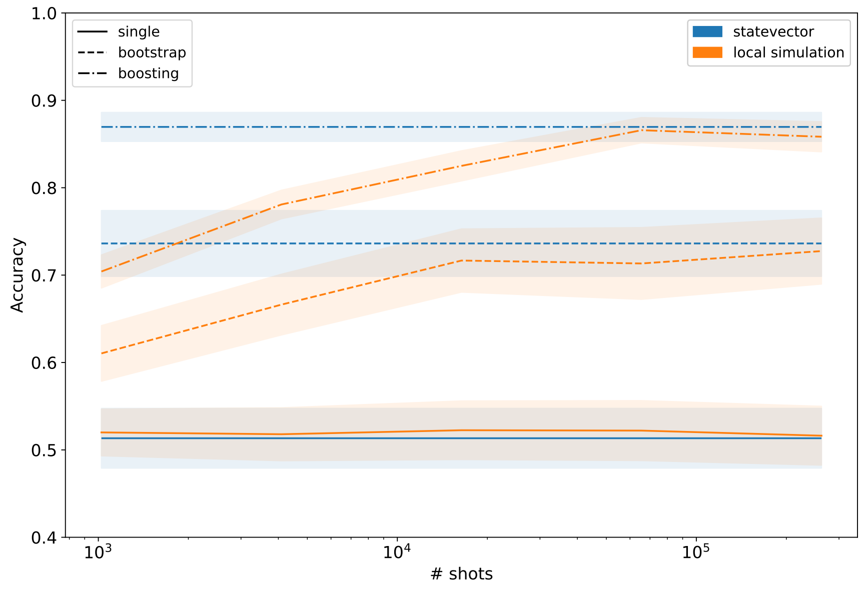


Fig. 6. Average accuracy over 120 runs on the *iris_versicolor_virginica* dataset as a function of the number of shots (ranging from 1024 to 262144). The plot refers to the quantum distance classifier with min-max data normalization, considering all ensemble techniques; the shaded regions correspond to 95% confidence intervals, which have been computed according to a Binomial distribution.

Additionally, the relationship between number of measurements (shots) and performance has been analysed for a single case, namely, the quantum distance classifier with min-max data normalization. For this purpose, all ensemble techniques and a single dataset have been taken into account. The results are shown in Figure 6. In detail, the single classifier achieves comparable results to *statevector* already with 1024 shots, while, for bootstrap and boosting, the number of measurements required to reach the ideal performance is in the order of 10^5. Moreover, bootstrap performs worse than boosting, but it reaches the 95% confidence interval with 10^4 shots. Instead, boosting achieves comparable performance to bootstrap in the *statevector* execution even with a low number of shots, but it requires ten times repetitions to reach the 95% confidence interval. In general, a large number of shots is needed to achieve near-to-exact performance (the number of shots grows quadratically with respect to the size of the confidence interval).

4.3.4 Noisy experiments

The *noisy simulation* method permits a more accurate representation of the behaviour of the current quantum devices, but it has a high computational cost, limiting the experiments that can be performed in a reasonable amount of time. As a consequence, we have chosen to evaluate only the quantum distance classifier with bootstrap and boosting ensembles. Indeed, simulating the quantum cosine classifier requires way more CPU-hours compared to the

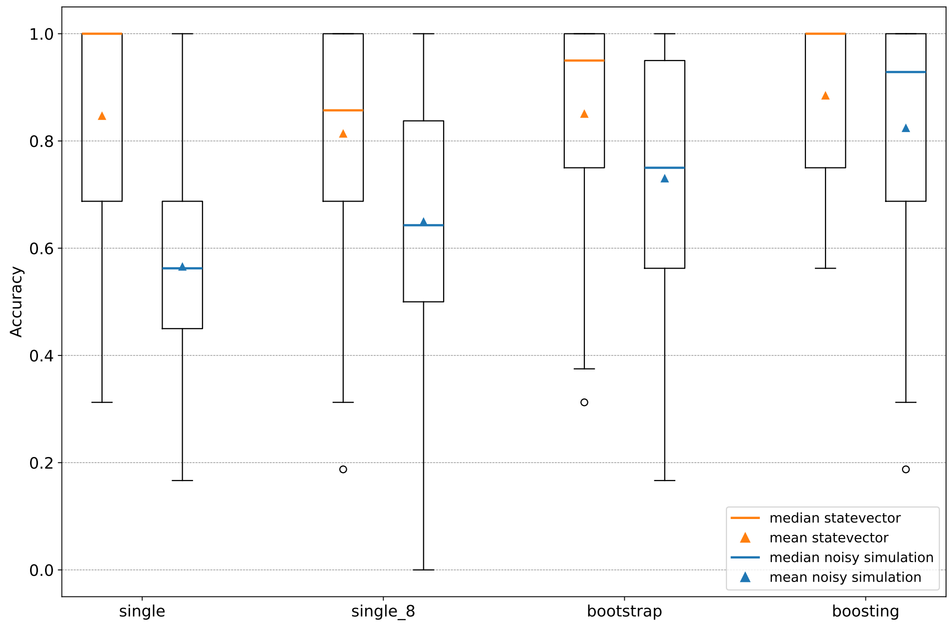


Fig. 7. Accuracy achieved by different configurations of the quantum distance classifier with *std* data normalization on the 8 considered datasets for *statevector* and *noisy simulation* modalities. In detail, the configurations considered are: single classifier with the full training set (single), single classifier with 8 bootstrapped data samples (single_8), bootstrap and boosting ensembles. The number of shots used for *noisy simulation* is 8192, and the number of points per box is 80 (one for each run on each dataset).

distance classifier. The data normalization technique used is *std*, as the previous experiments have shown that it yields the best results for the distance classifier. Lastly, regarding the datasets, we have selected those containing 100 data samples or less (thus, 8 out of 11).

The results obtained, which are displayed in Figure 7, reveal the ensembles' superiority over the single classifier in a noisy environment. Actually, the comparison includes also a quantum distance classifier with 8 training samples (sampled with replacement from the training set), which corresponds to an internal classifier of the ensemble methods. It turns out that, in a noisy environment, a classifier with only 8 data samples performs better on average than a classifier trained on the entire dataset, despite the expectations based on the *statevector* results. The reason is that larger circuits are more affected by noise. Thus, small classifiers, like those exploited by the ensemble methods, prove to be advantageous in a noisy environment. Eventually, it is worth observing that both bootstrap and boosting have obtained a higher average accuracy compared to the single small classifier (both in the ideal case and in the noisy simulation), confirming again the effectiveness of the ensembles.

Instead, Figure 8 shows the comparison between the ensemble methods and the single (large) quantum distance classifier for each pair of dataset and Monte Carlo run. This view allows evaluating the accuracy enhancement provided by the ensemble techniques on each dataset in the case of a noisy simulation. In particular, the majority of data points are located above the bisector, demonstrating the effectiveness of bootstrap and boosting even

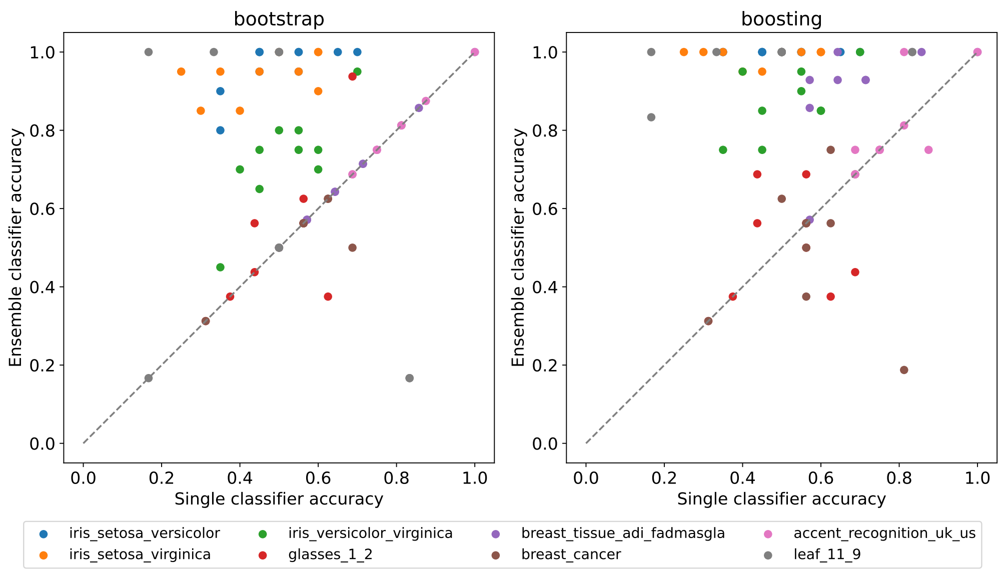


Fig. 8. Comparison between ensemble methods and quantum distance classifier with *std* data normalization in terms of accuracy over 10 Monte Carlo runs. The left plot refers to bootstrap, the right plot to boosting. Each data point corresponds to a different pair of dataset and Monte Carlo run, and *noisy simulation* with 8192 shots has been used.

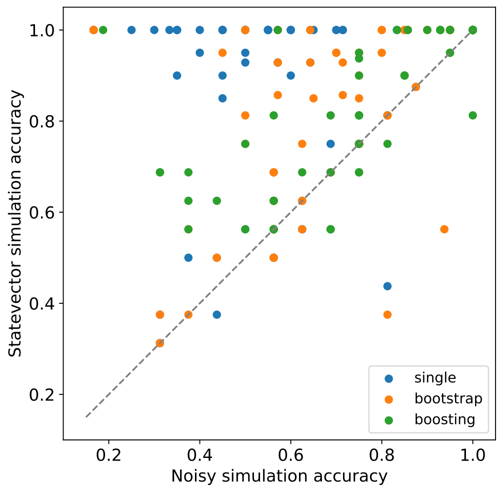


Fig. 9. Comparison between *statevector* and *noisy simulation* with 8192 shots in terms of accuracy for the quantum distance classifier with *std* data normalization. Each data point corresponds to a different combination of ensemble technique, Monte Carlo run, and dataset.

in a noisy setup. A few points, associated mainly with the *glasses_1_2* and *breast_cancer* datasets, fall below the bisector. However, they represent a clear minority.

Lastly, Figure 9 provides an alternative view (with respect to Figure 7) for the comparison between *statevector* and *noisy simulation*. In particular, the points related to the bootstrap and boosting ensembles are closer to the bisector than those related to the single classifier. This confirms that the single classifier is more affected by noise than the ensemble methods, which demonstrate greater robustness (probably due to the usage of small internal classifiers).

To conclude, it is important to highlight that, in the experiments performed here, most of the quantum circuit logic deals with data encoding. Indeed, ideally, the preparation of the initial state should scale logarithmically with respect to the number of training data instances. However, without a QRAM, it scales linearly. As a consequence, the impact of noise in the noisy experiments turns out to be higher. Instead, the gate complexity for executing the quantum algorithms remains constant for the quantum distance and cosine classifiers, and logarithmic in the number of features for the quantum k-NN.

5 Conclusions

In this work, a hybrid ensemble scheme characterised by classical ensemble techniques and quantum instance-based classification algorithms has been considered and empirically evaluated, in simulation with and without noise, on a binary classification task. In particular, the ensemble techniques taken into account are bootstrap, boosting, and stacking, while the considered quantum classifiers are a quantum cosine classifier, a quantum distance classifier, and a quantum k-nearest neighbors classifier. In addition, three data normalization techniques, namely, no normalization, standardization, and min-max normalization, have been taken into account. The results have shown that the introduction of the ensemble techniques leads to a performance improvement with respect to using single quantum classifiers. In detail, the ensemble models have demonstrated a slight advantage in accuracy compared to single classifiers when considering an ideal execution with a suitable data normalization technique. At the same time, they have shown the ability to mitigate both unsuitable data normalizations and noise, making quantum classifiers more stable. Indeed, a strong dependency of the classifiers' performance on the data normalization technique used has been observed, as well as the benefit of having class-balanced datasets for classifiers like the quantum distance one.

More in detail, in a noiseless environment, the single quantum cosine and quantum distance classifiers have achieved the best performance with data standardization, while the quantum k-NN has performed better with the other two techniques. Both bootstrap and boosting have improved the performance of the single classifiers, especially when the base performance was poor, with the quantum k-NN with data standardization being an anomaly in this sense (the performance did not improve). Actually, boosting has proven to be the best ensemble technique in terms of absolute performance but also the most sensitive to measurements uncertainty (due to the iterative structure). It is also worth mentioning that, for both bootstrap and boosting, the number of shots required to reach near-to-exact results has turned out to be quite high. Eventually, stacking has shown good accuracy results and good stability with respect to different data normalization techniques (the internal classifiers perform a subsequent normalization). Concerning the noisy experiments, both bootstrap and boosting have outperformed the quantum distance classifier, showing their capability of mitigating the impact of noise on performance by leveraging smaller internal models that are less affected by noise.

Given the promising results obtained, demonstrating how ensembles improve classification accuracy and mitigate data normalization and quantum circuit noise issues, thereby enhancing performance while constraining circuit size, an interesting possibility for future work could be the development of quantum ensemble techniques. For instance, quantum ensembles could be implemented by means of variational quantum circuits (VQCs) based on Hardware Efficient Ansatzes (HEAs) [30] with parametrized weights induced by the ensembles parameters.

Acknowledgements

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. In addition, E.Z. was supported by by Q@TN, the joint lab between University of Trento, FBK-Fondazione Bruno Kessler, INFN-National Institute for Nuclear Physics and CNR-National Research Council. E.T. was supported by the MUR National Recovery and Resilience Plan (PNRR) M4C1I4.1, funded by the European Union under NextGenerationEU. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or The European Research Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. We would also like to thank the reviewer for the effort in reviewing the manuscript, as well as for giving insightful observations and suggestions that contributed to improve the quality of the work.

References

1. J. Biamonte and P. Wittek and N. Pancotti and P. Rebentrost and N. Wiebe and S. Lloyd (2017), *Quantum machine learning*, Nature, Springer Science and Business Media LLC, Vol. 549, Num. 7671, pp. 195-202, 10.1038/nature23474.
2. J. Preskill (2018), *Quantum Computing in the NISQ era and beyond*, Quantum, Verein zur Forderung des Open Access Publizierens in den Quantenwissenschaften, Vol. 2, pp. 79, 10.22331/q-2018-08-06-79.
3. R. Polikar (2006), *Ensemble based systems in decision making*, IEEE Circuits and Systems Magazine, Vol. 6, Num. 3, pp. 21-45, 10.1109/MCAS.2006.1688199.
4. M. Schuld and F. Petruccione (2018), *Quantum ensembles of quantum classifiers*, Scientific Reports, Vol. 8, Num. 2772, 10.1038/s41598-018-20403-3.
5. A. Abbas and M. Schuld and F. Petruccione (2020), *On quantum ensembles of quantum classifiers*, Quantum Machine Intelligence, Vol. 2, Num. 6, 10.1007/s42484-020-00018-6.
6. I. C. S. Araujo and A. J. da Silva (2020), *Quantum ensemble of trained classifiers*, arXiv, quant-ph, 2007.09293.
7. A. Macaluso and L. Clissa and S. Lodi and C. Sartori (2022), *Quantum Ensemble for Classification*, arXiv, cs.LG, 2007.01028.
8. D. Windridge and R. Nagarajan (2017), *Quantum Bootstrap Aggregation*, Quantum Interaction, Springer International Publishing, pp. 115-121, 978-3-319-52289-0.
9. R. Qin and Z. Liang and J. Cheng and P. Kogge and Y. Shi (2022), *Improving Quantum Classifier Performance in NISQ Computers by Voting Strategy from Ensemble Learning*, arXiv, quant-ph, 2210.01656.
10. X. Zhang and M. Wang (2022), *An efficient combination strategy for hybird quantum ensemble classifier*, arXiv, quant-ph, 2210.06785.
11. M. Incudini and M. Grossi and A. Ceschin and A. Mandarino and M. Panella and S. Vallecorsa and D. Windridge (2023), *Resource saving via ensemble techniques for quantum neural networks*, Quantum Machine Intelligence, Vol. 5, Num. 39, https://doi.org/10.1007/s42484-023-00126-z.
12. L. Breiman (1996), *Bagging predictors*, Machine Learning, pp. 123–140, https://doi.org/10.1007/BF00058655.

13. R. Schapire (2013), *Explaining adaboost*, Empirical inference, Springer, pp. 37-52, 978-3-642-41136-6.
14. D. H. Wolpert (1992), *Stacked generalization*, Neural Networks, Vol. 5, Num. 2, pp. 241-259, ISSN 0893-6080, https://doi.org/10.1016/S0893-6080(05)80023-1.
15. D. Pastorello and E. Blanzieri (2021), *A Quantum Binary Classifier based on Cosine Similarity*, 2021 IEEE International Conference on Quantum Computing and Engineering (QCE), 10.1109/qce52317.2021.00086.
16. M. Schuld and M. Fingerhuth and F. Petruccione (2017), *Implementing a distance-based classifier with a quantum interference circuit*,Eventually EPL (Europhysics Letters), IOP Publishing, Vol. 119, Num. 6, pp. 60002, 10.1209/0295-5075/119/60002.
17. A. Afham and A. Basheer and S. K. Goyal (2020), *Quantum k-nearest neighbor machine learning algorithm*, arXiv, arXiv:2003.09187v1.
18. Y. Ma and H. Song and J. Zhang (2021), *Quantum Algorithm for K-Nearest Neighbors Classification Based on the Categorical Tensor Network States*, International Journal of Theoretical Physics, Vol. 60, pp. 1164-1174, 10.1007/s10773-021-04742-y.
19. V. Giovannetti and S. Lloyd and L. Maccone (2008), *Quantum Random Access Memory*, Physical Review Letters, American Physical Society (APS), Vol. 100, Num. 16, 10.1103/physrevlett.100.160501.
20. S. Lloyd and M. Mohseni and P. Rebentrost (2013), *Quantum algorithms for supervised and unsupervised machine learning*, arXiv, 10.48550/ARXIV.1307.0411.
21. M. A. Nielsen and I. L. Chuang (2010), *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, 10.1017/CBO9780511976667.
22. H. Buhrman and R. Cleve and J. Watrous and R. de Wolf (2001), *Quantum Fingerprinting*, Phys. Rev. Lett., American Physical Society, Vol. 87, Num. 16, pp. 167902-167905, 10.1103/PhysRevLett.87.167902.
23. T. Cover and P. Hart (1967), *Nearest neighbor pattern classification*, IEEE Transactions on Information Theory, Vol. 13, Num. 1, pp. 21-27, 10.1109/TIT.1967.1053964.
24. M. S. Anis and H. Abraham and AduOffei and R. Agarwal and G. Agliardi and others (2021), *Qiskit: An Open-source Framework for Quantum Computing*, 10.5281/zenodo.2573505.
25. IBM (2021), *IBM Quantum*, https://quantum-computing.ibm.com/.
26. E. Zardini and E. Blanzieri and D. Pastorello (2023), *Implementation and empirical evaluation of a quantum machine learning pipeline for local classification*, PLOS ONE, Vol. 18, Num. 11, pp. 1-28, https://doi.org/10.1371/journal.pone.0287869.
27. Q. Xu, Y. Liang (2001), *Monte Carlo cross validation*, Chemometrics and Intelligent Laboratory Systems, Vol. 56, Num. 1, pp. 1-11, ISSN 0169-7439, https://doi.org/10.1016/S0169-7439(00)00122-2.
28. E. Zardini (2022), *Implementation and empirical evaluation of a quantum machine learning pipeline for local classification*, GitHub, GitHub repository, https://github.com/ZarHenry96/quantum-ml-pipeline.
29. D. Dua and C. Graff (2017), *UCI Machine Learning Repository*, University of California, Irvine, School of Information and Computer Sciences, http://archive.ics.uci.edu/ml.
30. A. Kandala and A. Mezzacapo and K. Temme and M. Takita and M. Brink and J. M. Chow and J. M. Gambetta (2017), *Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets*, Nature, Springer Science and Business Media LLC, Vol. 549, Num. 7671, pp. 242-246, 10.1038/nature23879.

Appendix A Additional plots

Additional plots related to the noiseless experiments are provided in this appendix. Specifically, Figures A.1 and A.2 show the classification accuracy obtained by bootstrap and boosting with different parameter configurations. Instead, Figure A.3 shows the results achieved by the ensemble methods when the optimal data normalization technique is considered for each classifier. Lastly, Figure A.4 shows the comparison between *local simulation* and *statevector* for all combinations of ensemble technique, quantum classifier, and data normalization.

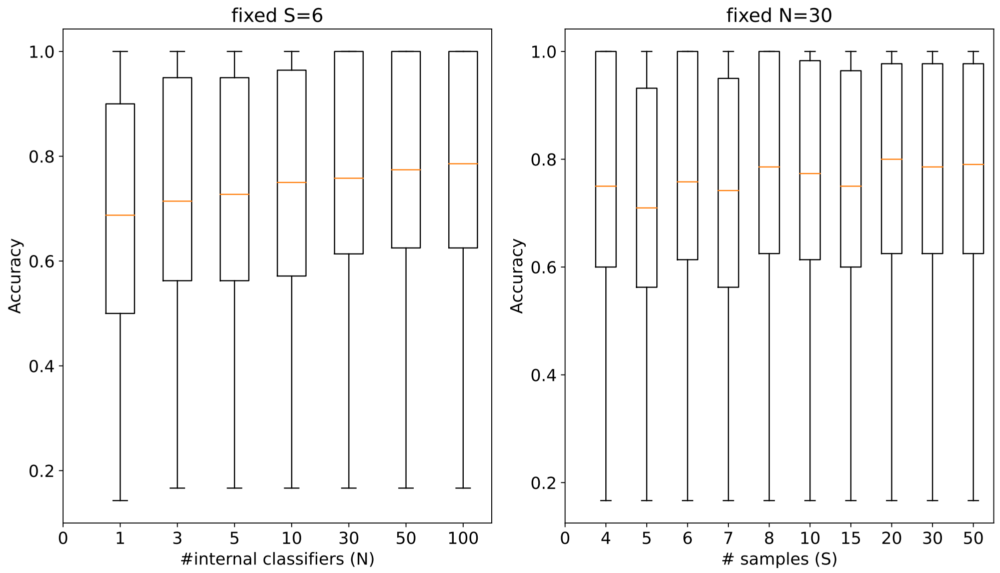


Fig. A.1. Accuracy comparison for the bootstrap technique, varying the number of internal classifiers N (left) and the number of training samples for each classifier S (right) while keeping fixed the other parameter ($S = 6$ in the left plot, $N = 30$ in the right plot). Each box contains 990 points, with each data point being the accuracy obtained in a run on a certain dataset by a combination of quantum classifier and normalization technique.

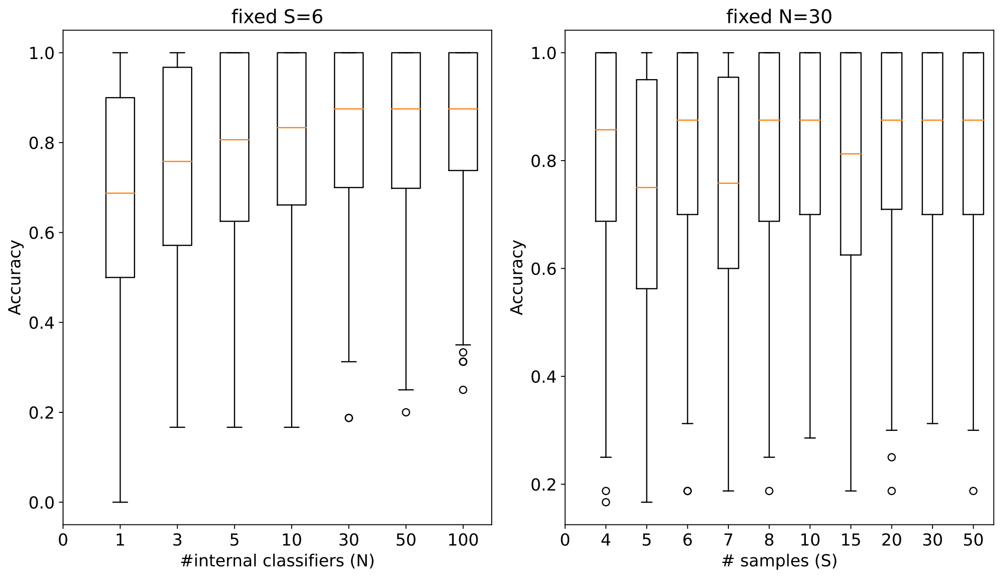


Fig. A.2. Accuracy comparison for the boosting technique, varying the number of internal classifiers N (left) and the number of training samples for each classifier S (right) while keeping fixed the other parameter ($S = 6$ in the left plot, $N = 30$ in the right plot). Each box contains 990 points, with each data point being the accuracy obtained in a run on a certain dataset by a combination of quantum classifier and normalization technique.

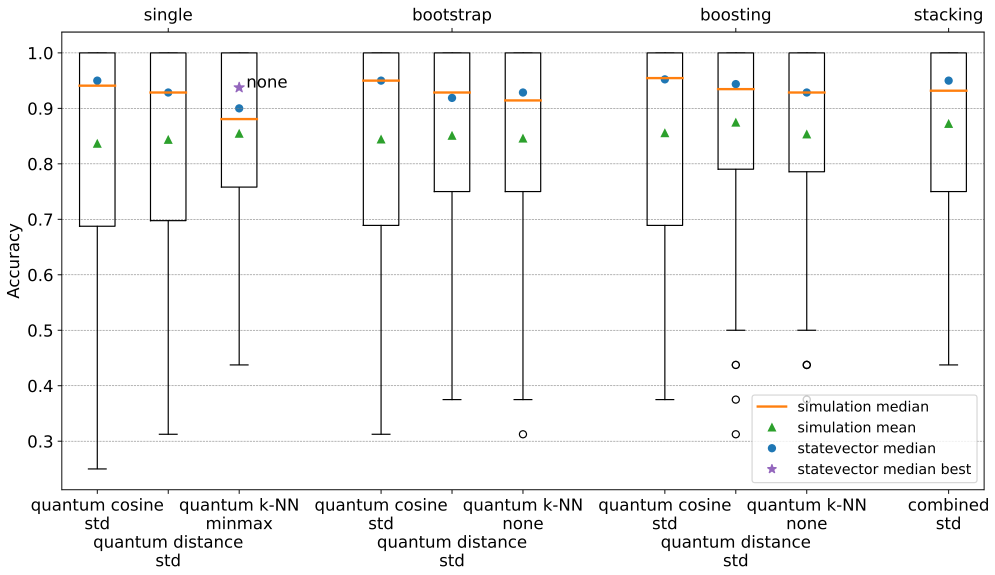


Fig. A.3. Accuracy achieved by *local simulation* with 8192 shots, for all combinations of ensemble and base classifier. Each box contains 110 points (one for each run on each dataset), and only the best-performing normalization technique (in *local simulation*) is shown for each models pair.

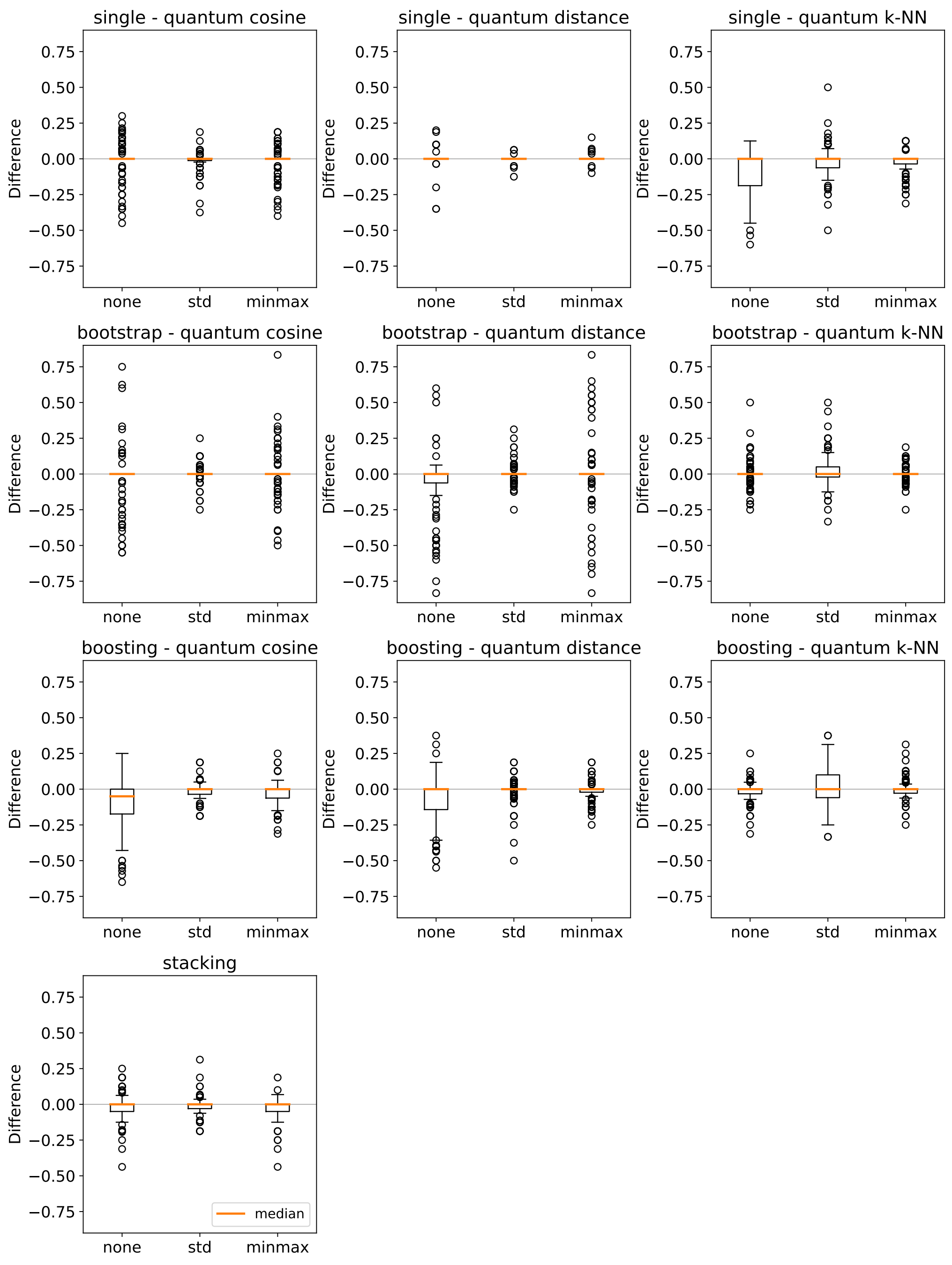


Fig. A.4. Comparison of *local simulation* with 8192 shots and *statevector* in terms of accuracy difference for all combinations of ensemble technique, quantum classifier, and data normalization technique. The number of points per box is 110, with each data point being related to a run on a certain dataset.

Appendix B Class imbalance

Both the quantum cosine and the quantum distance classifiers are strongly influenced by the training data class balance. Indeed, the label predictions are given according to a weighted sum over the training set (Eqs. (5) and (12)). Hence, if the classes are not balanced, the most frequent one is preferred. Moreover, the cosine similarity can take negative values. Instead, in the quantum distance classifier, the weight belongs to the interval $[0, 1]$; as a consequence, every data instance gives a non-negative contribution to the corresponding label. In conclusion, if the weights magnitude is small, the prediction is determined mainly by the ratio of classes in the training set.

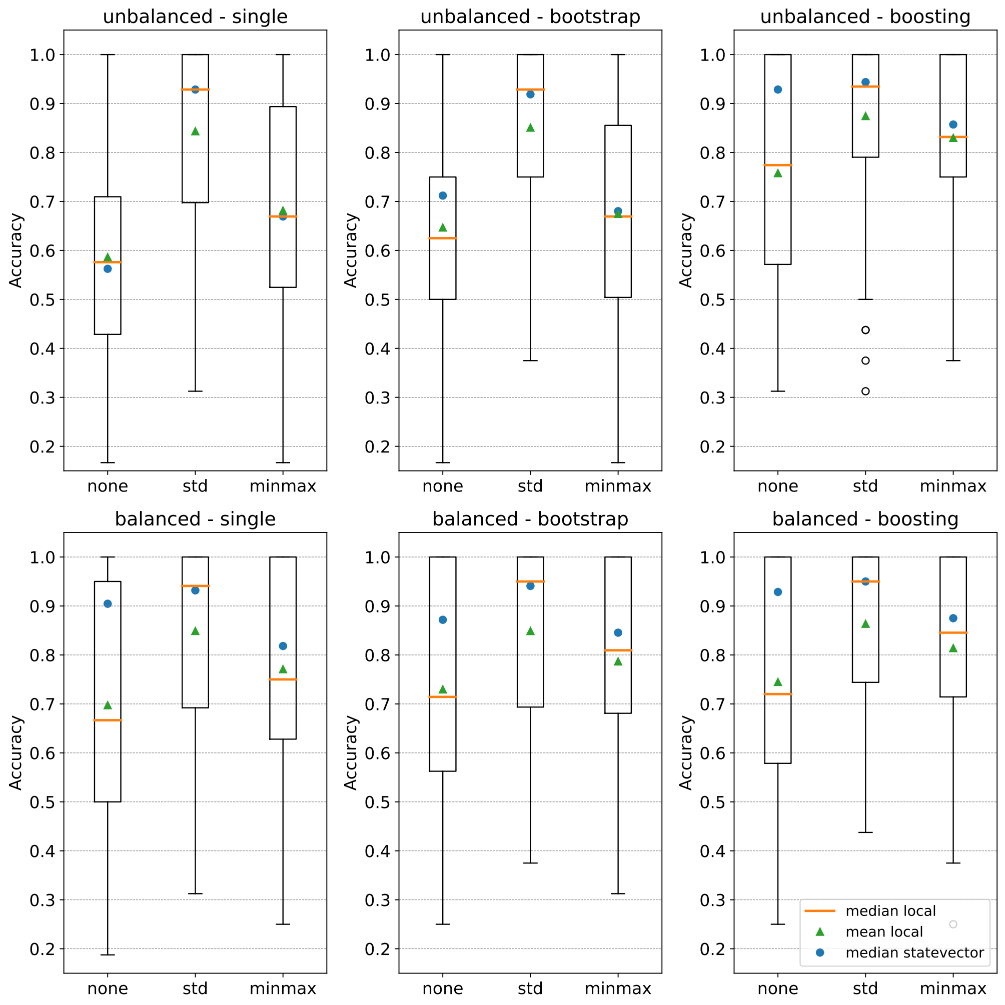


Fig. B.1. Accuracy achieved by the quantum distance classifier with non-balanced and balanced training sets, for all combinations of ensembles and data normalization techniques. The number of shots used for *local simulation* is 8192, and the number of points per box is 110 (one for each run on each dataset).

Figure B.1 shows the results achieved by the quantum distance classifier with non-balanced and balanced training sets, for all combinations of ensemble and data normalization techniques. In particular, for the single classifier, the most frequent class has been subsampled in order to match the number of instances of the other class; instead, for the ensembles, the balance has been obtained by sampling the same number of data instances from each class when building the training sets for the internal models. Focusing on the single classifiers, it is possible to notice that, while the results with data standardization are almost identical in the two cases, there is a significant performance improvement for no and min-max data normalizations when forcing the class balance. Nevertheless, a remarkable difference with respect to the ideal performance can be observed. Furthermore, standardization (*std*) turns out to be still the best normalization technique for the quantum distance classifier. Similar considerations hold for bootstrap, whose ideal performance with balanced data are also not always better than the ideal ones for the single classifier; however, the results achieved in simulation are still better. Eventually, boosting seems to be not as sensitive to class imbalance as the other classifiers, and its results are in line with the ones achieved with non-balanced class. In the end, it remains the best-performing ensemble technique.