Quantum ensemble of trained classifiers

Ismael C. S. Araujo
Departamento de Computação
Universidade Federal Rural de Pernambuco
Recife, Pernambuco, Brazil
ismael.cesar@ufrpe.br

Adenilton J. da Silva
Centro de informática
Universidade Federal de Pernambuco
Recife, Pernambuco, Brazil
ajsilva@cin.ufpe.br

***Abstract*—Through superposition, a quantum computer is capable of representing an exponentially large set of states, according to the number of qubits available. Quantum machine learning is a subfield of quantum computing that explores the potential of quantum computing to enhance machine learning algorithms. An approach of quantum machine learning named quantum ensembles of quantum classifiers consists of using superposition to build an exponentially large ensemble of classifiers to be trained with an optimization-free learning algorithm. In this work, we investigate how the quantum ensemble works with the addition of an optimization method. Experiments using benchmark datasets show the improvements obtained with the addition of the optimization step.**

***Index Terms*—quantum computing, quantum machine learning, quantum ensemble of classifiers.**

I. Introduction

Artificial intelligence (AI) is a field of computer science that studies the creation of programs designed to act as intelligent agents, created to evaluate and automatically make decisions based on the inputs [1]. Machine learning (ML) is a subfield of AI that studies the creation of algorithms and programs that are capable of not only make decisions based on the inputs but also learn with it to improve performance [2]–[4]. In this work, we name the programs and algorithms that involve some ML method as ML models, or only models.

Quantum computing is a field of computer science that studies the codification and processing of information in quantum systems [5]. The smallest unit that represents information in a quantum computer is a quantum bit or *qubit*. Unlike a classical bit, which can only assume one state at a time, either 0 or 1 exclusively, a *qubit* has the property of being in both states 0 and 1 at the same time. This superposition of information means that a quantum computer with multiple *qubits* is capable of representing an exponentially large number of states, according to the number of qubits available.

ML models usually have to process feature vectors with large dimensions, which can impact the cost of processing time and memory consumption. The power of state representation makes quantum computing a candidate for the improvement of ML models. Quantum Machine Learning (QML) is the subfield of quantum computing where researchers have been studying how to explore the potential of quantum computing to enhance ML [6], [7].

In [8], a QML approach named quantum ensemble of quantum classifiers is proposed. Classifier outputs weigh the degree of influence of each classifier in the final answer. In [8], the accuracy of each classifier weighs its significance, and bad performing classifiers would have a small impact on the final output.

In a quantum version of this kind of ensemble, it would be possible to use superposition to create an exponentially large set of classifiers, with the accuracy codified in the probability amplitude of the state. We could use this quantum ensemble as an optimization free model, with the supposition that an exponentially large ensemble of classifiers weighed by its accuracy would return good classifications [8].

In this work, we explore the idea of using quantum computers to efficiently create an ensemble of classifiers and perform experiments based on simulations using benchmark datasets. We propose a quantum ensemble strategy based on [8] with a training phase. We show the differences between quantum ensembles with optimized and unoptimized classifiers. The ML model used in the simulations was ANNs, so as for the therms optimized and unoptimized were taken as trained and untrained ANNs.

The rest of this work is organized as follows: Section II contains some introductory explanations concerning quantum computing. Section III introduces a more detailed description of quantum ensembles mentioned in the introduction, as well as replications of numerical analysis made in [8]. Section IV presents the main contribution, describes the simulations, presents metrics used to evaluate the model, and shows the results concerning the presented methodology. Finally, Section V contains some remarks about the results presented, as well as possible future works.

II. Quantum computing

The base unit of information used in a quantum computer is called a quantum bit or *qubit*. Mathematically a *qubit* can be represented in the form of a column vector, however, for simplicity and convenience, many works in the literature rather use the Dirac's notation [5], [9]. The symbol $|\cdot\rangle$ is called "ket" and the equivalence between Dirac's notation and column vector notation is made in the following equation:

$$\begin{aligned} \begin{bmatrix} 1 & 0 \end{bmatrix}^T &= |0\rangle \\ \begin{bmatrix} 0 & 1 \end{bmatrix}^T &= |1\rangle \end{aligned} \tag{1}$$

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

A *qubit* has the property to be in a state of superposition, that is, to assume more than one state at the same time. The superposition can be mathematically represented by a linear combination of the basis states. The coefficients in the linear combination would represent the amplitude probability of the system state to assume a specific state. More specifically, Let the state $|\psi\rangle$ in Eq. (2) be an arbitrary state of a single *qubit*.

The coefficients α and β are complex numbers representing the amplitude probability of $|\psi\rangle$.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2}$$

A quantum state can also be composed of multiple *qubits*.

We use tensor products to represent multiple qubits. For instance, the two-*qubit* state $|0\rangle \otimes |1\rangle = |01\rangle$. Similarly to a single *qubit* state, the *multiqubit* state can be in superposition. Let $x \in \{0,1\}^n$ be a binary string, where n is the number of *qubits* in the string, and $|\psi'\rangle$ a *multiqubit* in superposition. Eq. (3) describes the *multiqubit* $|\psi\rangle$.

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x|x\rangle \tag{3}$$

To extract useful information from a superposed state, we need to perform a measurement operation. However, it is impossible to measure each component of a superposed state without affecting the entire system. That is because when we apply a measurement operation is applied to the system, the superposition of the system collapses, and the system assumes a specific state. The information contained in other components of the superposition is lost or changed. For example, if the qubit $|\psi\rangle$ in Eq. (2) is measured, $|\psi\rangle$ will collapse to the state $|0\rangle$ with $|\alpha|^2$ probability and $|1\rangle$ with $|\beta|^2$ probability, where $|\alpha|^2 + |\beta|^2 = 1$.

Another case to be analyzed is when a *multiqubit* system is partially measured. Suppose the two last qubits of the state in Eq.(3) are being measured. And let $\bar{x} \in \{\{0,1\}^{n-2} + 01\}$ be the regular expression that represents the binary strings of size n that end with 01. After the measurement of the two last qubits is performed, the system would return $|01\rangle$ with a probability $p = \sum_{\bar{x}} |\alpha_{\bar{x}}|^2$. And the state $|\psi'\rangle$ would collapse to the state $|\psi''\rangle$, as shown in Eq.(4)

$$|\psi''\rangle = \sum_{\bar{x}\in\{\{0,1\}^{n-2}+01\}} \frac{\alpha_{\bar{x}}|\bar{x}\rangle}{\sqrt{\sum_{\bar{x}} |\alpha_{\bar{x}}|^2}} \tag{4}$$

Quantum operators, or quantum gates, are used to operate and manipulate the states and amplitude probabilities of quantum systems. Let $N = 2^n$ be the number of states that can be represented with n *qubits*. Given a fixed basis, an $\mathbb{C}^{N\times N}$ unitary matrix represents a quantum gate over n qubits. Quantum gates can also be combined using tensor products to perform multiqubit operations. For instance, $I \otimes X \otimes I$, where I is the single-*qubit* identity gate and X the *not* gate.

Since quantum gates work as linear operators, these operators can modify different components of a superposition linearly, this property is known as quantum parallelism [5], [9], [10]. In Equation (5) an application of the gates $I \otimes X \otimes I$ on a superposed three-*qubit* state is exemplified.

$$(I \otimes X \otimes I)\,(\gamma|010\rangle + \phi|110\rangle) \rightarrow \gamma|000\rangle + \phi|100\rangle \tag{5}$$

A quantum computer can theoretically simulate any binary function of a classical computer [5]. Let A be a quantum operator that implements a classical function $f(x)$, such that $A|x\rangle|0\rangle = |x\rangle|f(x)\rangle$. Eq. (6) shows an example of the action of the A operator in a superposed state.

A is a linear operator and can be applied to each component of the superposition thanks to quantum parallelism. With superposition and quantum parallelism, it is possible to apply a function in an exponentially large number of states at the same time.

$$\sum_{x\in\{0,1\}^n} \alpha_x A|x\rangle|0\rangle = \sum_{x\in\{0,1\}^n} \alpha_x|x\rangle|f(x)\rangle \tag{6}$$

III. Quantum Ensembles

A classifier can be described as a function that maps features into the classes: $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ is the space of features and $\mathcal{Y}$ is the space of classes. A parameterized classifier makes the mapping of features according to the parameters that were set, such as the mapping $f : \mathcal{X} \times \Theta \rightarrow \mathcal{Y}$ where Θ is the space of the parameters. However in this work a parameter $\theta \in \Theta$ shall be referred to as the classifier itself. Given that in order to improve a classification model's performance, one can optimize its parameters θ, which is the case with artificial neural networks (ANNs) [4], [11].

Thus, an ensemble of classifiers is a classification method in which the output of different classifiers is combined to obtain a final answer. For example, given a binary classification problem where the set of classes is $Y = \{-1, 1\}$. Let the set of classifiers be $E = \{\theta_0, \theta_1, \ldots, \theta_{n-1}\}$. Given an unseen data sample $\tilde{x}$, the output of each classifier is combined in a sum in order to obtain the ensemble's final answer as it is shown in Eq. (7).

$$\tilde{y} = sign\left(\sum_{\theta\in E} w_\theta f(\tilde{x}, \theta)\right) \tag{7}$$

Where $sign$ is the sign function and $\tilde{y}$ is the answer (class) returned by the ensemble. Where the output of each classifier is pondered by a factor w_θ, which represents its desired degree of influence in the sum. A way to determine the value of the factor w_θ is by using the classifier's accuracy, which can be obtained by measuring the performance of each classifier in the dataset before using the ensemble to label new data [8]. This way, bad performing classifiers would have little influence in the sum.

A. Quantum ensemble described in quantum states

In order to implement a quantum version of an ensemble such as in Eq. (7), the system would have to be divided into five quantum registers. Taking into consideration the data

would be encoded in the qubits, the first register would be dedicated do store the data $|x\rangle$. The data can be processed in the system sample by sample, or it can be stored in superposition using a storing procedure from some kind probabilistic, or associative quantum memory [12]–[14]. For simplicity, a sample by sample approach will be considered.

The second register would be used for storing the classifiers $|\theta\rangle$. The third and fourth register would be used for storing the answer of each classifier $|\hat{y}_\theta\rangle$ and the correct answer $|y\rangle$ respectively. And the fifth register would be a one *qubit* register used as ancilla.

For the initial state of the ensemble, all the classifiers would have to be stored in superposition. And the ancillary register would be initialized in superposition as well. Let x be one data sample from a dataset $\mathcal{D}$ with its label being y. Both the data sample and its label would be stored in their respective registers and the answer register $|\hat{y}_\theta\rangle$ would be initialized in the state $|0\rangle$. Therefore, the initial state for the ensemble would be as in Eq. (8).

$$|x\rangle \frac{1}{\sqrt{|E|}} \sum_{\theta \in E} |\theta\rangle|0\rangle|y\rangle \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \tag{8}$$

Implementing a quantum equivalent of a classifier such as a neural network requires a sequence of unitary operators of different types that combined form the quantum operator that implements it, such as in [15]. To simplify our examples, let $\mathcal{F}$ be a unitary operator that implements the function $f(x, \theta)$, which is the function that returns the label given by the classifier, or the classifier's answer to input x. Applying $\mathcal{F}$ to the system, the result would be as in equation (9).

$$\begin{array}{c} \mathcal{F}|x\rangle \frac{1}{\sqrt{|E|}} \sum_{\theta \in E} |\theta\rangle|0\rangle|y\rangle \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right) \\ \downarrow \\ |x\rangle \frac{1}{\sqrt{|E|}} \sum_{\theta \in E} |\theta\rangle|\hat{y}_\theta\rangle|y\rangle \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right) \end{array} \tag{9}$$

The application of $\mathcal{F}$ would have to be repeated for every $x \in \mathcal{D}$. Where, everytime $|\hat{y}_\theta\rangle = |y\rangle$ the ancilary qubit would be rotated towards $|0\rangle$. Thus, the degree of influence of each classifier would be decoded on the phase of the quantum state, as in (10).

$$\frac{1}{\sqrt{|E|}} \sum_{\theta \in E} |\theta\rangle \left(\sqrt{a_\theta}|0\rangle + \sqrt{1 - a_\theta}|1\rangle\right) \tag{10}$$

Afterward, the ancillary *qubit* has to be measured. If after the measurement the *qubit* assumes the state $|1\rangle$ the whole process of building the ensemble has to be made again. However, if after the measurement the *qubit* assumes the state $|0\rangle$ the ensemble is ready to classify data samples from the validation set or any other unseen data. The resulting state of the ensemble would be as in (11).

$$\sum_{\theta \in E} \frac{\sqrt{a_\theta}}{\sqrt{\mathcal{X}|E|}} |\theta\rangle \tag{11}$$

Where $\mathcal{X}$ is a normalization factor resulting from the measurement of the ancillary *qubit*. Let $\tilde{x}$ be an unseen datasample. In order to obtain the ensemble's answer, one would only need to add the data sample into the state, apply the $\mathcal{F}$ operator and measure the answer qubit in (12).

$$\mathcal{F}|\tilde{x}\rangle \sum_{\theta \in E} \frac{\sqrt{a_\theta}}{\sqrt{\mathcal{X}|E|}} |\theta\rangle|0\rangle \rightarrow |\tilde{x}\rangle \sum_{\theta \in E} \frac{\sqrt{a_\theta}}{\sqrt{\mathcal{X}|E|}} |\theta\rangle|\hat{y}_\theta\rangle \tag{12}$$

However, by measuring the answer qubit the superposition of the ensemble would collapse, so the whole process of building the quantum ensemble would have to be repeated.

B. Replicating results from numerical analysis

In [8] the authors analyze the use of an exponentially large quantum ensemble using quantum classifiers. Presenting the idea of using such a quantum ensemble inspired in Bayesian learning, proposing a quantum classification method to be used as optimization free (or untrained) type of learning. With the premiss that a reasonably large ensemble with weak but accurate classifiers would still return good classifications [16], [17].

A numerical analysis is presented further in [8] showing how such kind of ensemble would behave. The analysis was presented using simplified toy examples in a classical computer. The classes used in the examples were $Y = \{-1, 1\}$.

Where two binary classification problems were used, with the first case being an uni-dimensional case, where the function f is to perform mappings of one-dimensional data points from both classes randomly generated with a normal distribution. Where the hyperparameters relative to each class will be referred to using a plus or minus signals, such as σ_+ and μ_+ for the class 1 and σ_- and μ_- for class -1.

$$f(x) = \frac{1}{\sigma_\pm \sqrt{2\pi}} e^{\left(-\frac{x-\mu_\pm}{\sqrt{2}\sigma_\pm}\right)^2} \tag{13}$$

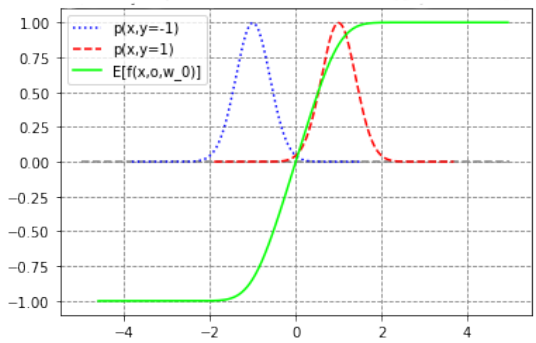


Fig. 1: Decision region defined by expectatios value of the data points of the unidimentional case.

Fig. 1 shows some results of the replications concerning the calculation of the decision region for the one-dimensional case. The data points to be classified were randomly generated

with the parameter $\sigma_{\pm} = 0.4$, and with $\mu_{-} = -1$ and $\mu_{+} = 1$. It can be seen that the decision boundary region can be found on the intersection of the distributions $p(\tilde{x}, y = -1)$ and $p(\tilde{x}, y = 1)$ from data points with class -1 and 1 respectively. However, if the variance of one of the classes' distribution is changed, the decision boundary is shifted towards one of the classes' distribution as shown in Fig. 2. Where the variance was changed from $\sigma_{+} = 0.4$ to $\sigma_{+} = 0.9$. Showing how this optimization free kind of classification is dependent on how well distributed are the data points for each class.

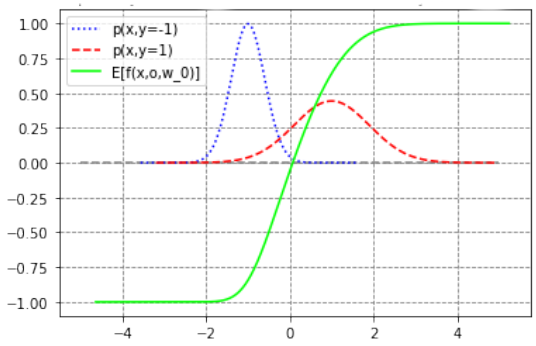
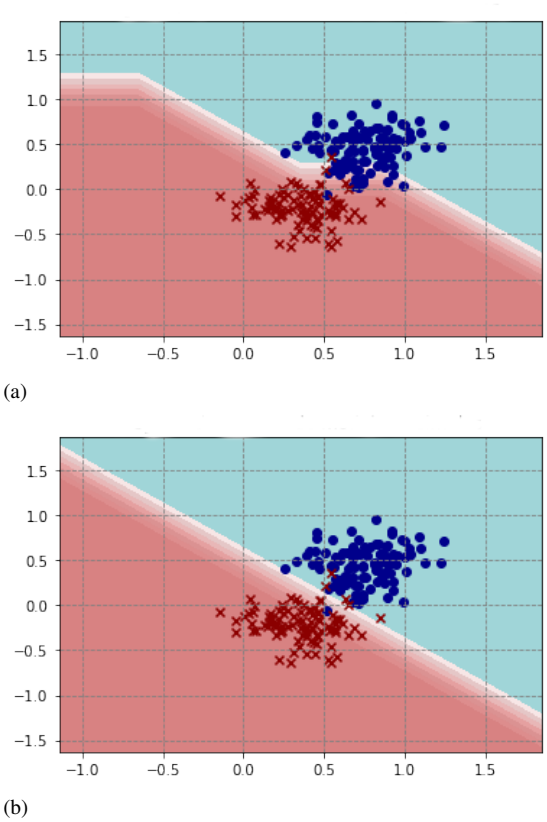


Fig. 2: Shifting decision boundary by changing the distribtuions' variance of data points with class 1 .

The second classification case was also a binary classification problem but, with bi-dimensional data points. Where the data points used in the replication of this case were randomly generated using scikit-learn's [18] $blob\_function$. With parameters used to generate the data points for each class being $\mu_{-} = [-1, 1]$ and $\mu_{+} = [1, -1]$. With a standard deviation of 0.5 for data points in both classes. As in [8], 8000 perceptrons were created to classify the data points, with weights and biases in the interval $[-1, 1]$. Fig. 3 shows the results of the replication of the bi-dimensional case for both trained and untrained perceptrons. And as can be seen, the decision boundary finds in the center of both sets in the case using trained perceptrons.

We went a bit further with the replications and executed a bi-dimensional case, where the data points from both classes are distributed differently. Fig. 4 shows the new decision region dividing the two newly generated sets. A difference in the definition of the decision boundary can be remarked between the cases using trained and untrained perceptrons. Where the difference becomes less subtle when taking to account the outliers, or the points located further away from the center of each of their respective classes.

The replications presented in this work were made with the purpose to show how dependant an optimization free kind of ensemble would be to the data points distribution for each class. However, those are but replications of toy examples of numerical analysis made in a classical manner, where the differences between trained and untrained models can be subtle.



(a)

(b)

Fig. 3: Computing decision region for bi-dimensional data points. 3a Decision boundary computed using 8000 untrained perceptrons. 3b Decision boundary computed using 8000 trained perceptrons.

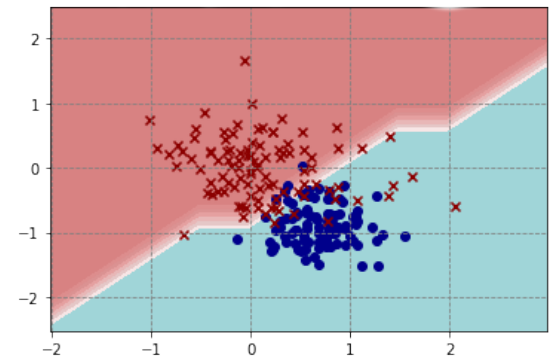
IV. Quantum ensemble of trained classifiers

Taking into consideration how a quantum ensemble of quantum classifiers work. We decided to evaluate what would be the results of such a quantum ensemble using benchmark datasets. The datasets used in this approach were the sets made available by the *scikit-learn*'s [18] library: iris, breast-cancer, and wine.

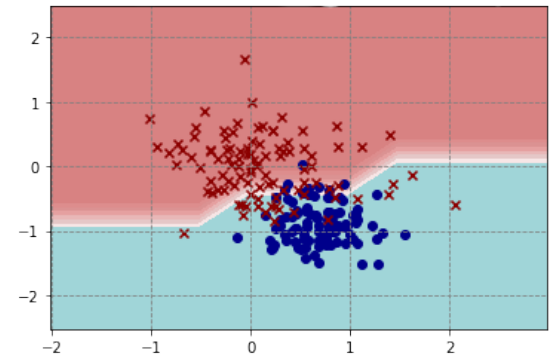
Unfortunately, quantum computers made available today are still very limited to perform this kind of experiment. Therefore, we decided to make a simulation-based on calculations of the probability amplitudes in the system. The type of classifier used in the simulations was the multilayer perceptron (MLP). All of them with 1 hidden layer with 10 neurons. The models were implemented using with the *python* programming language, using the *pytorch* library [19].

The cross-validation methodology used was hold-out. Being 70% of the data points used for training and 30% of the data points used for validation. Whose accuracy values shall be referred to as training and validation respectively.

Two experiments were executed for the simulations and two metrics were used to evaluate both experiments. The First metric was the Mean Probability Per Sample of the ensemble to return the correct label or $MPPS_{hit}$. A quantum circuit



(a)



(b)

Fig. 4: Computing decision region for bi-dimensional data points, with standard deviation of the distribution of data points with class -1 changed to 0.3. 3a Decision boundary computed using 8000 untrained perceptrons. 3b Decision boundary computed using 8000 trained perceptrons.

returns its answer in a probabilistic way. So, to compute the $MPPS_{hit}$ we first computed the probability of the system to return the correct label for each sample, a Probability Per Sample (or PPS_{hit}). Then, the arithmetic mean was calculated to obtain the $MPPS_{hit}$.

Another metric taken into account was the ensemble's overall accuracy, that is, the accuracy of the answers given by the set of classifiers. Let p_x to be the probability of the ensemble to answer a datapoint correctly. In order to measure the ensemble's overall accuracy, a threshold probability (p_t) was used, so that when $p_x \geq p_t$ a correct answer would be computed to the ensemble's overall accuracy.

The threshold probability value chosen for the experiments was 0.7. This value was chosen after a series of trials and errors, in order to define a threshold that was not too low valued, lest bad performing ensemble would have a great influence on the final results. And to define not a too much high valued threshold, lest it would be too prohibitive for reasonable performing ensembles to be computed in the final results.

During the execution of the experiments, the number of members in the ensemble was variated in the set $|E| \in \{100, 200, 300, 400, 500\}$.

Algorithm 1 Pseudocode of the simulations were performed

procedure QUANTUM ENSEMBLE($\mathcal{D}$,$|E|$)
  for Each data set $D \in \mathcal{D}$ **do**
    Create ensemble of classifiers $E = \{\theta_1, \ldots, \theta_{|E|}\}$
    Divide D into training (D_t) and validation (D_v) sets
    Initialize the $array$ $PPS_{hit} \leftarrow \varnothing$
    if Use the training step **then**
      for Each training epoch $\mathcal{T}$ **do**
        for Each classifier $\theta_i \in \{\theta_1, \ldots, \theta_{|E|}\}$ **do**
          Train θ_i
          Calculate the accuracy a_θ using D_t
        end for
      end for
    else
      for Each classifier $\theta_i \in \{\theta_1, \ldots, \theta_{|E|}\}$ **do**
        Calculate the accuracy a_θ using D_t
      end for
    end if
    Create the $array$ of probability amplitudes $\left[\sqrt{\frac{a_{\theta_1}}{\mathcal{X}|E|}}, \ldots, \sqrt{\frac{a_{\theta_{|E|}}}{\mathcal{X}|E|}}\right]$
    Process D_v with classifiers $\{\theta_1, \ldots, \theta_{|E|}\}$
    for Each datapoint $(x, y) \in D_v$ **do**
      Calculate $p_{hit} = \sum_{\theta \in E} \sqrt{\frac{a_{\theta_{|E|}}}{\mathcal{X}|E|}}$ for each classifier $\theta \in E$ that hits y
      if $p_{hit} >= p_t$ **then**
        Increment overall accuracy of the ensemble
      end if
      Save p_{hit} into the $array$ PPS_{hit}
    end for
    Compute the $MPPS_{hit}$ using the data from $array$ PPS_{hit}
  end for
end procedure

From the experiments executed, the first was to perform classification using untrained classifiers to verify the results of an optimization free learning over the benchmark datasets. And the second kind of experiment was to perform classification using trained models to verify if by adding a training step in the system it is possible to improve the ensemble's performance, or if it remains unchanged. Algorithm 1 contains pseudocode with a high-level explanation of how the experiments were executed.

A. Simulating a quantum ensemble with untrained classifiers

In the simulations, the overall accuracy of a quantum ensemble using untrained classifiers was 0.0 for all ensemble sizes defined, using a threshold of 0.7. Table I shows the results concerning the $MPPS_{hit}$ and its standard deviation ($StdD$) from quantum ensembles of different sizes using untrained classifiers. It can be observed that for all ensemble sizes the

$MPPS_{hit}$ remained below 0.5 which explains the overall accuracy of 0.0.

TABLE I: Results from a quantum ensemble with untrained classifiers on the Iris dataset.

$\|E\|$	$MPPS_{hit}$	$StdD$
100	0.37	0.053
200	0.37	0.084
300	0.38	0.042
400	0.39	0.059
500	0.39	0.043

Tables II and III show the results concerning $MPPS_{hit}$ and its $StdD$ on the Wine and Breast-Cancer dataset. From those results, slightly better performance of the quantum ensemble with untrained classifiers on the Breast-Cancer dataset can be observed relative to the Iris dataset, according to the values of $MPPS_{hit}$. And slightly worse results are obtained with untrained classifiers on the Wine dataset. The results from both datasets toghether with the Iris dataset's results show that the overall $MPPS_{hit}$ on all the datasets remained below the threshold value of 0.7, thus resulting in an overall accuracy of 0.0 for all ensemble sizes on all the datasets.

TABLE II: Results from a quantum ensemble with untrained classifiers on the Wine dataset.

$\|E\|$	$MPPS_{hit}$	$StdD$
100	0.37	0.086
200	0.37	0.04
300	0.38	0.03
400	0.36	0.028
500	0.35	0.041

TABLE III: Results from a quantum ensemble with untrained classifiers on the Breast-Cancer dataset.

$\|E\|$	$MPPS_{hit}$	$StdD$
100	0.54	0.034
200	0.55	0.038
300	0.54	0.036
400	0.52	0.047
500	0.53	0.029

These results show that the optimization free approach is not a good option for a quantum ensemble when using real-life datasets. To improve the ensemble's performance we decided to add a training (optimization) step, right after the creation of the models, as it is shown in Algorithm 1.

B. Simulating a quantum ensemble with trained classifiers

In the simulations the models were trained with different training epochs, variating in the set $\{5, 10, 15\}$. In all training epochs for all values in $|E|$, the optimization method used in the training step as Stochastic Gradient Descent (SGD) [20]. With a learning rate of 1.2 and a momentum of 0.9 and a batch size of 10 datapoints. The values concerning the number of training epochs, batch size, learning rate, and momentum values were defined after a series of tests that involved trial and error. Tables IV, V and VI show the overall accuracy for every ensemble size in $|E|$ according to each of the tree training epochs number aforementioned. As in the untrained case, the results are according to the Iris, Wine and Breast-Cancer datasets respectively.

The best results for the Iris dataset in Table IV were the quantum ensemble with 200 and 300 ensemble members, both having an overall accuracy of 0.97. With the classifiers in the ensemble of size 200 trained with 10 training epochs. Where the $MPPS_{hit}$ in this case was 0.97, with $StdD$ of 0.051. And the classifiers trained in the ensemble of size 300 trained with 5 and 15 training epochs. Which for those cases the $MPPS_{hit}$ were both 0.96, with $StdD$ of 0.083 and 0.146 respective to each case of training epochs presented.

TABLE IV: Overall accuracies of a quantum ensemble with different number of ensemble members over the **Iris** dataset

$\|E\|$	No. of epochs	overall accuracy
100	5	0.93
	10	0.95
	15	0.95
200	5	0.95
	10	0.97
	15	0.93
300	5	0.97
	10	0.95
	15	0.97
400	5	0.88
	10	0.95
	15	0.95
500	5	0.93
	10	0.95
	15	0.93

The best results for the Wine dataset in Table V were from the quantum ensemble with 100, 300 and 400 ensemble members, all three ensemble sizes containing results with an overall accuracy of 1.0. With the ensemble size of 100 and 300 with models trained with 15 epochs, and the ensemble size of 400 with models trained with 10 epochs. The $MPPS_{hit}$ for the quantum ensemble with 100 and 300 models was 0.98, with a $StdD$ of 0.042 and 0.032 respectively.

And $MPPS_{hit}$ for the quantum ensemble with 400 models was 0.98, with a $StdD$ of 0.018.

TABLE V: Overall accuracies of a quantum ensemble with different number of ensemble members over the **Wine** dataset

$\|E\|$	No. of epochs	overall accuracy
100	5	0.96
	10	0.96
	15	1.00
200	5	0.98
	10	0.96
	15	0.96
300	5	0.96
	10	0.98
	15	1.00
400	5	0.96
	10	1.00
	15	0.96
500	5	0.98
	10	0.92
	15	0.98

The best results for the Breast Cancer dataset in Table VI were from quantum ensembles with 300 and 500 ensemble

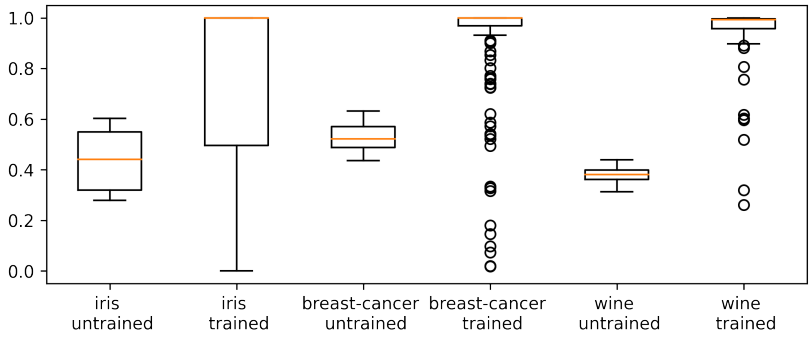


Fig. 5: PPS_{hit} of an ensemble with 300 classifiers untrained and trained with 15 $epochs$ on all the detasets mentioned.

members, trained with 5 and 15 training epochs respectively. Where the $MPPS_{hit}$ for the quantum ensemble with 300 members of 0.98, with $StdD$ of 0.077. And the $MPPS_{hit}$ for the quantum ensemble with 500 members of 0.98, with $StdD$ of 0.07.

TABLE VI: Overall Accuracies of a Quantum Ensemble With Different Number of Ensemble Members Over the **Breast-Cancer** Dataset

$\|E\|$	No. of epochs	overall accuracy
100	5	0.95
	10	0.97
	15	0.95
200	5	0.95
	10	0.95
	15	0.95
300	5	0.99
	10	0.95
	15	0.95
400	5	0.97
	10	0.95
	15	0.96
500	5	0.97
	10	0.95
	15	0.98

By presenting these results we intend to show the advantage of using the training step in an ensemble quantum ensemble. Fig. 5 presents a summary of the PPS_{hit} for an ensemble of 300 classifiers with the training and the optimization free strategy (untrained). In the trained case the classifiers were trained with 15 epochs.

V. Conclusions

From the results presented we can infer that using a quantum ensemble of quantum classifiers as a form of optimization free learning is not a good approach to perform classification in a quantum computer. Mainly because by simply adding a training step it was possible to significantly improve the performance of the ensemble relative to the overall accuracy and the mean probability of obtaining the correct answer.

Because of the hardware limitations concerning the recent quantum computers, it was not possible to perform an experiment by using an exponentially large quantum ensemble, as it would be theoretically possible on an actual general-purpose quantum computer.

Despite the fact the results in this work were obtained from simulations by calculating the probability amplitudes of the system, they serve as experimental evidence of the great advantage of still using a training step, or optimization step, in a quantum ensemble of quantum classifiers, which is the main contribution of this work. Even if adding the optimization step means increasing the cost relative to gate application in building the quantum circuit.

The models used in the simulations were artificial neural networks. Other works investigating the codification and training of neural networks can be found in literature [15], [21], [22]. This work investigates the training of a quantum ensemble of quantum classifiers using neural networks with a fixed architecture. A future work might be the addition of architecture selection of neural networks [23].

Some quantum processors were made available, which can be accessed through the cloud. However, Noisy Intermediate Scale Quantum processors are still limited in the number of qubits. Another possible future work might be to make a reduced case scenario to be used in those currently small scale quantum processors.

Acknowledgement

This work was supported by CNPq, CAPES and FACEPE (Brazilian research agencies).

References

[1] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.
[2] K. Faceli, A. C. Lorena, J. Gama, A. C. P. d. L. Carvalho *et al.*, "Inteligência artificial: Uma abordagem de aprendizado de máquina," 2011.
[3] T. M. Mitchell, "Machine learning," 1997.
[4] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
[5] M. A. Nielsen and I. L. Chuang, *Quantum computation and Quantum information*. Cambridge University Press India, 2000.
[6] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, p. 195, 2017.
[7] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, 2015.
[8] M. Schuld and F. Petruccione, "Quantum ensembles of quantum classifiers," *Scientific reports*, vol. 8, no. 1, p. 2772, 2018.
[9] D. McMahon, *Quantum computing explained*. John Wiley & Sons, 2007.

[10] N. S. Yanofsky, M. A. Mannucci, and M. A. Mannucci, *Quantum computing for computer scientists*. Cambridge University Press Cambridge, 2008, vol. 20.

[11] S. Haykin, "Neural networks: principles and practice," *Bookman*, 2001.

[12] D. Ventura, "Artificial associative memory using quantum processes," in *Proceedings of the International Conference on Computational Intelligence and Neuroscience*, vol. 2, 1998, pp. 218–221.

[13] D. Ventura and T. Martinez, "A quantum associative memory based on grover's algorithm," in *Artificial Neural Nets and Genetic Algorithms*, 1999, pp. 22–27.

[14] C. A. Trugenberger, "Quantum pattern recognition," *Quantum Information Processing*, vol. 1, no. 6, pp. 471–493, 2002.

[15] A. Fawaz, P. Klein, S. Piat, S. Severini, and P. Mountney, "Training and meta-training binary neural networks with quantum computing," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1674–1681.

[16] M. Kearns and L. Valiant, "Learning boolean formulae or finite automata is as hard as factoring. harvard university," *Center for Research in Computing Technology, Aiken Computation Laboratory*, 1988.

[17] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[20] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[21] B. Ricks and D. Ventura, "Training a quantum neural network," in *Advances in neural information processing systems*, 2004, pp. 1019–1026.

[22] A. J. da Silva, T. B. Ludermir, and W. R. de Oliveira, "Quantum perceptron over a field and neural network architecture selection in a quantum computer," *Neural Networks*, vol. 76, pp. 55–64, 2016.

[23] P. G. Dos Santos, R. S. Sousa, I. C. Araujo, and A. J. da Silva, "Quantum enhanced cross-validation for near-optimal neural networks architecture selection," *International Journal of Quantum Information*, vol. 16, no. 08, p. 1840005, 2018.