Quantum Machine Intelligence (2024) 6:76
https://doi.org/10.1007/s42484-024-00210-y

RESEARCH ARTICLE

What we can do with one qubit in quantum machine learning: ten classical machine learning problems that can be solved with a single qubit

Manuel P. Cuéllar[1]

Received: 6 June 2024 / Accepted: 24 October 2024

Abstract
This work focuses on quantum machine learning and analyzes the power of one single qubit to solve classical machine learning problems. We explore possible strategies to address traditional supervised, unsupervised, and reinforcement learning tasks. In particular, we study binary and multinomial classification, regression problems, time series forecasting, clustering, and quantum reinforcement learning. Our results suggest that the same methodology could be used to address all three types of learning with different measurement strategies and, despite the strong limitation of reduced data dimensionality of the candidate problems, a single qubit can achieve similar or even improved performance with respect to state-of-the-art classic machine learning methods in many cases. As simulating the evolution of one qubit state is computationally efficient, our study enables the possibility to use the qubit model as a candidate solution to implement simple decision-making machine learning models in hardware with extremely low memory resources such as embedded systems or edge devices. The outcomes provided could be also of interest in academia and for the construction of demonstrators using low-scale contemporary quantum hardware.

Keywords Quantum machine learning · Quantum computing · Quantum supervised learning · Quantum unsupervised learning · Quantum reinforcement learning

1 Introduction

Research in quantum machine learning (QML) (Ganguly 2021) has increased substantially in the last decade thanks to advances in the construction of quantum computer hardware. The goal of QML is to adapt classical machine learning models and to create pure or hybrid QML ones that help solving supervised, unsupervised, and reinforcement learning tasks with an improvement in either efficiency or performance. The literature offers a wide variety of QML models in the three types of learning, ranging from classification (Schuld et al. 2014; Maheshwari et al. 2022) quantum neural networks, linear regression (Wang 2017), or support vector machines (Gentinetta et al. 2024) to more complex approaches including quantum generative adversarial networks (Zoufal et al. 2019) or convolutional networks (Rajesh and Naik 2021) in supervised learning, clustering with quantum K-means and K-medians (Aïmeur et al. 2007; Yarkoni et al. 2021) or autoencoders (Wu et al. 2024) in unsupervised learning, and parameterized quantum circuits (Chen et al. 2019) and policy optimization algorithms (Cherrat et al. 2023) in quantum reinforcement learning (Andres et al. 2022). There are also approaches in the field of knowledge representation, as for instance quantum decision trees (Heese et al. 2022) or quantum rule-based systems (Moret-Bonillo 2018; Cuellar et al. 2024).

The development progress of quantum computers is rising at an astonishing speed, and so it is the advances in QML. Current hardware and simulators enable the possibility to create more complex QML models with better efficiency than classical ones according to the aforementioned references, and the scalability is becoming a more pressing matter in the field as it is highlighted in Gentinetta et al. (2024). In this context, our work is motivated by the following question: Does a single qubit have a practical use in quantum machine learning? If so, the proposal would have

✉ Manuel P. Cuéllar
manupc@decsai.ugr.es

[1] Department of Computer Science and Artificial Intelligence, University of Granada, ETSIIT. C/. Pdta. Daniel Saucedo Aranda s.n., Granada, Granada 18014, Spain

Published online: 12 November 2024

obvious limitations regarding to the low dimensionality of the possible candidate problems to be addressed, as well as its scalability. Previous approaches have addressed classical low-dimensional machine learning problems with success, such as the *Iris flower* dataset (Piatrenka and Rusek 2022) in supervised learning, Variational quantum circuits in reinforcement learning benchmark environments (Skolik et al. 2022; Andres et al. 2023), or quantum clustering in synthetic benchmark data (Poggiali et al. 2024). However, all these approaches use more than one qubit to solve their problems.

There are not much works in the literature that have explored previously the power and limitations of using a single qubit in QML. One of them is the work (Yu et al. 2022), which studies the limitations of a quantum neural network with a single qubit for regression. The work (Tapia et al. 2023) proposes a didactic approach of a quantum neural network encoded into a single qubit to solve a real-world fraud detection dataset as a binary classification problem. On the other hand, the reference (Karimi et al. 2023) explores quantum coherence in a binary classification task with one qubit. Another work that addresses one qubit QML classification tasks is Pérez-Salinas et al. (2020), which explores how data re-uploading can help in the classification of three two-dimensional circles in an image containing up to four classes. The work was extended in Pérez-Salinas et al. (2021), where a proof that a qubit can hold universal approximation capabilities is discussed. From our point of view, this is a remarkable finding since a single qubit can fit a bounded continuous function as accurately as one can expect. However, this achievement could have limitations in practice, since the improvement in accuracy comes at the cost of increasing the model size in terms of parameters to be optimized and therefore the circuit depth. If this number is too high, then the optimization algorithms could get trapped into local optima solutions as it happens with the single-layer neural network universal approximators. The work also proposes a template circuit scheme as a baseline to solve 1-D and 2-D continuous functions experimentally as a proof of concept.

Other recent advances in the study of the capabilities of one single qubit to solve computer science and ML problems are Easom-McCaldin et al. (2024); Goswami et al. (2024): the work Easom-McCaldin et al. (2024) explored the encoding of images into a single qubit and solved three state-of-the-art classification machine learning problems: the digit and Fashion MNIST and the ORL face dataset. Performance results are below the classical machine learning standards, although the approach uses a number of resources much more lower than traditional methods. On the other hand, the article (Goswami et al. 2024) explores how a graph describing an instance of the traveling salesman problem can be solved with one qubit. Experimental results show the success of the approach to solve instances of symmetric and asymmetric graphs ranging from 5 to 8 nodes.

In this piece of research, we provide an empirical study about the benefits and limitations of using a single qubit to solve classical machine learning problems beyond classification and regression as they have previously addressed in the literature. Of course, they are relevant tasks within machine learning and we include datasets of these types, but we also extend our experiments to further problems such as time series forecasting, clustering, or reinforcement learning, covering the three main types of machine learning categories regarding supervised, unsupervised, and reinforcement learning. Besides the experimental study, we also provide methodological guidelines for the design of quantum encoding and measurement, which have been key aspects in our work. Our results suggest that a single qubit can be used to solve different types of problems with a performance similar to classical machine learning methods. Thus, the proposal stands as a possible way to save computational resources in quantum hardware and simulation, but also as a mechanism to substitute classical models in hardware with low resources such as microcontrollers, embedded systems, or edge devices. The remaining of the manuscript is structured as follows: Sect. 2 describe the tools from quantum computing that are relevant for our work, as well as the proposed qubit computation model and methodology. After that, Sect. 3 describes ten classical machine learning problems covering all three types of learning and discusses the results. Finally, Sect. 4 concludes.

2 Methods

2.1 Quantum computing tools

The fundamental tool used in this manuscript from quantum computing is the qubit. Usually, the mathematical model of a qubit is described as an element of a complex vector subspace in $\mathbb{C}^2$ as $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ subject to the constraint $|\alpha_0|^2 + |\alpha_1|^2 = 1$. As α_0, α_1 are complex numbers, other usual representations of a qubit express these coefficients in magnitude and phase as $|\psi\rangle = r_0 e^{i\theta_0}|0\rangle + r_1 e^{i\theta_1}|1\rangle$ or more commonly as $|\psi\rangle = e^{i\theta_0}\left(r_0|0\rangle + r_1 e^{i\theta}|1\rangle\right)$ where $\theta = \theta_1 - \theta_0$ is the relative phase and $e^{i\theta_0}$ a global phase. The global phase term is usually omitted if a quantum state is described over a single qubit because it does not affect measurement, so that the magnitudes and phase of the qubit can be also expressed in trigonometric form as $|\psi\rangle = \cos(\phi/2)|0\rangle + sin(\phi/2)e^{i\theta}|1\rangle$. In the remaining of the manuscript, we refer mostly to the magnitudes and relative phase representation, although the trigonometric form is also of interest as a link to the geometric representation of the qubit in the Bloch sphere.

The Bloch sphere is another resource from the quantum computing area that is widely referenced in this manuscript,

since all one qubit gates can be explained geometrically in terms of rotations of a qubit in the Bloch sphere up to a global phase effect. A single qubit can be mapped to a point in the surface of the Bloch sphere, and its coordinates are easily calculated from the trigonometric representation as shown in Eq. 1. Figure 1 shows a qubit $|\psi\rangle$ in the Bloch sphere, where the angle ϕ controls the value of the magnitudes and θ the value of the phase.

$$\begin{aligned} x &= \sin(\phi) \cdot \cos(\theta) \\ y &= \sin(\phi) \cdot \sin(\theta) \\ z &= \cos(\phi) \end{aligned} \tag{1}$$

In our approach, we evolve the qubit quantum state using the gate model, and we make an extensive use of parameterized quantum gates to do so. More specifically, our model structure includes $R_x(\phi)$, $R_y(\phi)$, and $R_z(\lambda)$ rotation gates in the three axes of the Bloch sphere. The unitary matrices of these gates are included in Eqs. 2–4 for article self-completeness.

$$R_x(\phi) = \begin{bmatrix} \cos(\frac{\phi}{2}) & -i \cdot \sin(\frac{\phi}{2}) \\ -i \cdot \sin(\frac{\phi}{2}) & \cos(\frac{\phi}{2}) \end{bmatrix} \tag{2}$$

$$R_y(\phi) = \begin{bmatrix} \cos(\frac{\phi}{2}) & -\sin(\frac{\phi}{2}) \\ \sin(\frac{\phi}{2}) & \cos(\frac{\phi}{2}) \end{bmatrix} \tag{3}$$

$$R_z(\lambda) = \begin{bmatrix} e^{-i\frac{\lambda}{2}} & 0 \\ 0 & e^{i\frac{\lambda}{2}} \end{bmatrix} \tag{4}$$

Finally, we advance that measurement plays a critical role in our approach since a suitable measurement strategy in one or more axes of the Bloch sphere can be used to exploit the information encoded in the quantum state of a qubit. For this reason, we use observables to calculate the outcomes of the approach in each problem studied and, more specifically, the $\sigma_x, \sigma_y, \sigma_z$ observables whose matrices are described in Eqs. 5–7.

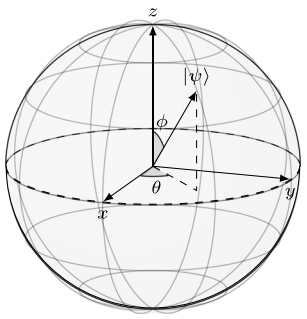


Fig. 1 An arbitrary qubit $|\psi\rangle$ and its geometric representation in the Bloch sphere

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{5}$$

$$\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{6}$$

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{7}$$

2.2 Template of the qubit QML model structure

The solutions proposed in the experimental subsections use the same model structure to solve different machine learning tasks, except for the sequence processing and forecasting problems. The proposal contains a single qubit and makes an extensive use of parameterized quantum circuits to evolve a quantum state containing the representation of the input data to a target quantum state with the answer to the problem. The model can be subdivided into three stages:

- A parameterized state preparation layer where a quantum state $|\psi\rangle$ is constructed from the initial state $|0\rangle$. The goal of this layer is to move the initial quantum state of the qubit from $|0\rangle$ to another state $|\psi\rangle$ that can facilitate the encoding of the input data for further processing. This layer allows the qubit to rotate in the three axes of the Bloch sphere using three parameters to be optimized $\theta_0 = (\theta_0^1, \theta_0^2, \theta_0^3)$. We name the unitary matrix of this module as $U_s(\theta_0)$ in our experiments, and it is constructed as $U_s(\theta_0) = R_x(\theta_0^1)R_z(\theta_0^2)R_y(\theta_0^3)$. Although the use of this state preparation layer is not usual in the literature, we have found experimental evidences that it can help to improve classical data encoding into a quantum state.
- The data embedding layer. Since a qubit can encode information in magnitude or phase only, both values are considered in the general case. In our experiments, an input pattern x_i with two features $x_i = (x_i^1, x_i^2)$ encodes x_i^1 in magnitude using a Y-axis rotation of the Bloch sphere and x_i^2 in phase using a rotation in the Z axis. We name the unitary matrix of the data embedding module as $U_e(x_i)$, and it is designed as $U_e(x_i) = R_z(x_i^2)R_y(x_i^1)$. It is expected that the output of this layer is a quantum state $|x_i\rangle$ containing the information of the inputs given by the classical data x_i.
- The processing layer (ansatz). This layer implements a parameterized rotation by $\theta_1 = (\theta_1^1, \theta_1^2, \theta_1^3)$ in the three axes of the Bloch sphere to move the qubit containing the encoded data $|x_i\rangle$ to a point of the Bloch sphere whose quantum state $|\hat{y}_i(\theta)\rangle$ contains the required response for the input data x_i, where θ stands for the set of all parameters in the model structure $\theta = (\theta_0, \theta_1)$. The

design of the ansatz has the same structure as the state preparation layer and its unitary matrix is calculated as $U_a(\theta_1) = R_x(\theta_1^1)R_z(\theta_1^2)R_y(\theta_1^3)$.

Figure 2 shows the circuit model that implements the proposed structure to output the quantum state $|\hat{y}_i(\theta)\rangle$ calculated as $|\hat{y}_i(\theta)\rangle = U_a(\theta_1)U_e(x_i)U_s(\theta_0)|0\rangle$. However, solving sequence processing problems such as time series forecasting might require to input different patterns in sequence as $x_{t-1}, x_{t-2}, ..., x_{t-h}$ to the proposed model. We address this situation in our experiments with the repetition of the data embedding and ansatz layer structures in a sequence of length h, so that the output state $|\hat{y}_i(\theta)\rangle$ is calculated as $|\hat{y}_i(\theta)\rangle = \prod_{k=1}^{h}\left[U_a(\theta_k)U_e(x_{t-k})\right]U_s(\theta_0)|0\rangle$. The circuit diagram that implements this approach is depicted in Fig. 3. In the remaining of the manuscript, we call this general model structure as the qubit QML approach for short.

The output $\hat{y}_i(\theta)$ provided in both model structures of Figs. 2 and 3 is obtained using measurement operators over the final quantum state $|\hat{y}_i(\theta)\rangle$ and a quantum-classical output mapping criterion. In our approach, measurement plays a key role in the whole process and is calculated using projective measurement and observables. To be more precise, we consider the observables $\sigma_x, \sigma_y, \sigma_z$ and calculate the expectation values $E(\sigma_k) = \langle\hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle$ according to the needs of a specific problem solution. The calculation of the final output of the model for an input pattern x_i takes into account the values of these expectations and a problem-dependent interpretation to map $\{E(\sigma_x), E(\sigma_y), E(\sigma_z)\}$ to a valid model response $\hat{y}_i(\theta)$.

2.3 Methodological details

As it was mentioned before, a single qubit can encode information in magnitude or phase. This is a very strong limitation to encode classical data with multidimensional features, and for this reason, a preliminary exploratory data analysis (EDA) and data preprocessing are key to success in our proposal. Moreover, the nature of problems addressed in Sect. 3 is varied and ranges from classification, regression, and time series forecasting in supervised learning, to unsupervised clustering and reinforcement learning. Having an adaptable yet effective methodology is crucial in our experiments. In this section, we decompose the steps required to solve a machine learning problem with one qubit in the necessary steps that must be taken into account.

We start with the hypothesis that the data of the problem at hand can be reduced to up to two dimensions used as input. If this condition is met after a suitable EDA is performed, then the proposed methodology follows the next steps:

1. **Data preprocessing**: This stage targets at preparing the classical data before they are fed to the quantum embedding module described in Sect. 2.2. In our work, we have considered different types of data preprocessing:

   - Feature transformation: This type of preprocessing is performed when a data transformation could be beneficial to encode the input data into a quantum state. It includes linear, quadratic, or logarithmic transformations of the data of a single feature in the problem addressed. As an example, the problem studied in Sect. 3.5 transforms a time series data with log-scale to reduce the negative effect of variance increase over time.
   - Feature normalization: The goal of this type of preprocessing is to change the domain of each input feature to reduce the negative effects of different ranges of scale. Moreover, it is a necessary step prior to quantum embedding to scale the data to the range $[0, \pi]$ for the $R_y(\theta)$ and $R_z(\lambda)$ encoding gates of the module.
   - Feature selection: This type of preprocessing involves selecting one or more features to be used as model inputs, while the remaining initial features of the problem are discarded. An example of this type of preprocessing is performed in Sect. 3.3 where the *petal width* and *petal length* features in the *Iris flower* dataset are selected as predictor variables. Thus, the remaining features *sepal width* and *sepal length* are discarded and not used to calculate the output of the proposed model.
   - Feature extraction: This step involves the combination of two or more initial features to create further facets that combine the information of the selected features into a single value. An example of this type of preprocessing is performed in Sect. 3.2 to solve the *Bank Note Authentication* dataset where the features

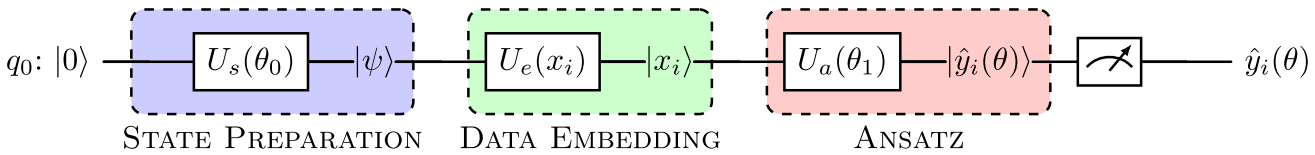


Fig. 2 General model structure: a state preparation layer builds an initial state $|\psi\rangle$ from $|0\rangle$ before data encoding. After that, the data embedding step encodes classical data x_i into a state $|x_i\rangle$. Finally, the ansatz provides the output state $|\hat{y}(\theta)\rangle$

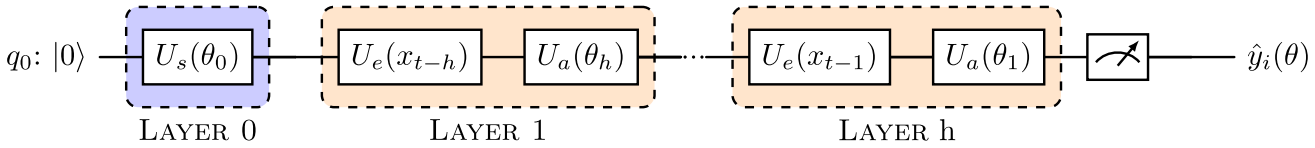


Fig. 3 Model structure for sequence processing. Layer 0 remains as the state preparation layer. The scheme of the data embedding and ansatz modules are replicated a number of times equals to the length of the sequence h

variance and *skewness* are combined to create a new feature that aggregates the information of both.

2. **Design of the measurement strategy and quantum-classical output mapping**: Whilst the previous step is aimed at preparing the input data, the design of the measurement strategy targets at selecting a suitable measurement method and the development of an interpretation that maps the model outcomes to a suitable response that solves the problem addressed. In our work, we use the expectation of an observable as model output, so that the design of the measurement strategy must take into account if the model output is continuous or discrete. In the first case, either the target output values or model predictions must be scaled according to the observable it is used for measurement. In the second case, it must be decided the number of observables that are needed to gather information from the output quantum state $|\hat{y}_i(\theta)\rangle$ and how these values must be interpreted to provide a correct output. As an illustrative example, Sect. 3.2 solves a binary classification problem and uses the observable σ_z for measurement. If the expectation of this observable $E(\sigma_z) \geq 0$, then the model output is mapped to $\hat{y}_i(\theta) = 1$ and $\hat{y}_i(\theta) = 0$ otherwise.
3. **Selection of a metric for model evaluation**: This selection is problem-dependent, and it will have a relevant impact in the evaluation of model performance. Example metrics we use in this work are the prediction accuracy for classification (8), the mean squared error for regression and time series forecasting (9), the Silhouette score for clustering (10), or the average return in reinforcement learning (11). In Eq. 8, the terms TP, TN, FP, and FN stand for the number of true positives and true negatives (success in the model prediction), and false positives and false negatives (failure in the model prediction) respectively, so that the metric represents the success rate of the model prediction and must be maximized. On the other hand, the terms N, y_i in Eq. 9 represent the number of data patterns in the dataset and the desired model output, respectively. The MSE must be minimized. The values $b(i)$, $a(i)$ in Eq. 10 encode the minimum distance of a cluster member i to a member of other clusters, and the average distance of the member i to the remaining elements of the same cluster, respectively. Thus, the Silhouette score evaluates a set of clusters considering both the intra-cluster and inter-cluster distances. The range of the S-score is $[-1, 1]$, and the clusters identified by an algorithm must maximize this metric. Finally, the values s_t^i, a_t^i, r in Eq. 11 stand for the environment state at time t in the i-th experiment in a reinforcement learning problem, the action selected at time t in the i-th experiment, and the reward obtained in the transition from state s_t^i to state s_{t+1}^i using action a_t^i. The metric in Eq. 11 should be maximized.

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i(\theta))^2 \tag{9}$$

$$\text{S-score} = \frac{1}{N}\sum_{i=1}^{N}\frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{10}$$

$$\overline{R} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}r(s_t^i, a_t^i, s_{t+1}^i) \tag{11}$$

4. **Selection of the parameter optimization algorithm**: This step is usually dependent of the choice of evaluation metric in the previous step. If the metric is derivable, then both gradient-based or gradient-free methods can be used to optimize the qubit QML model. Otherwise, the only choice is within the different families of gradient-free algorithms. In this work, we have selected evolution strategies (ES) as the optimization algorithm. To be more precise, the $ES(\mu + \lambda)$ (Hansen et al. 2015) algorithm was implemented due to its good performance to find suitable solutions in all the problems addressed, but also because some of the metrics we use for model evaluation are not derivable.
5. **Preliminary experimentation**: The goal of this stage is to find suitable hyperparameters of the training algorithm and the qubit QML model. In particular, the preliminary experimentation of this work followed a trial-and-error procedure to find the correct qubit QML structure to activate or not the state preparation layer, the number of layers in the time series forecasting problems, the parameters μ, λ, and the learning rate of the $ES(\mu + \lambda)$ algorithm.

6. **Training and test**: This final step generates the complete experimentation with the parameters found in the previous stage. In our work, it provided the results we describe in Sect. 3.

In our experience, after addressing all the problems described in the experimentation, we may conclude that steps 1 and 2 have special relevance to train a single qubit QML model to solve a problem. These two stages may affect the outcomes considerably, and specially the design of the measurement strategy and quantum-classical output mapping. The remaining steps 3–6 are also relevant although they might be considered as in the traditional pipeline of machine learning.

2.4 The effect of the state preparation layer

In this section, we discuss the benefits of the state preparation layer introduced in Sect. 2.2 and Fig. 2, since we are aware that this is an unusual setting in a traditional QML pipeline. As it was mentioned, the goal of the state preparation layer is to move the quantum state of the system to an initial value $|\psi\rangle$ different from $|0\rangle$ before classical-quantum data embedding. We recall that the value of $|\psi\rangle$ is problem-dependent and it is previously unknown, so that a parameterized unitary matrix $U_s(\theta_0) = R_x(\theta_0^1)R_z(\theta_0^2)R_y(\theta_0^3)$ is used to find $|\psi\rangle$ where the parameters $(\theta_0^1, \theta_0^2, \theta_0^3)$ need to be optimized. As we build $|\psi\rangle$ using $U_s(\theta_0)$, then the effect of the state preparation layer might be considered an initial change of basis in the quantum data space.

As it is widely known, many problems in engineering, including data science and machine learning, require a preprocessing step implementing linear or non-linear transformations to make data linearly separable. This is the case, for instance, of linear classifiers where data linear separability is a requirement to provide a correct output. And it is also a requirement in the data quantum space of our experiments due to the design of the measurement strategies we develop to solve most of the problems.

Let us illustrate the effect of the state preparation layer with an example considering a binary classification problem. In particular, we use the classical *XOR problem* with two input variables $(x^1, x^2) \in \mathbb{R}^2$ and one output $y \in \{0, 1\}$. The problem data map the points $(x_i^1, x_i^2) \mapsto y_i$ as $(0, 0) \mapsto 0$, $(0, 1) \mapsto 1$, $(1, 0) \mapsto 1$, and $(1, 1) \mapsto 0$. These data points are clearly not linearly separable as we cannot find a straight line that divides the data space with a correct classification criterion (see Fig. 4).

Something similar would happen if we use the raw data embedding layer described in Sect. 2.2 to encode the data of the *XOR* problem when we attempt to solve it using a quantum classifier. After a data normalization preprocessing to map $(x_i^1, x_i^2) \mapsto (x_i^1\pi, x_i^2\pi)$, the data embedding unitary matrix would be calculated as $U_e(x_i) = R_z(x_i^2)R_y(x_i^1)$ according to the guidelines described in Sect. 2.2. The effect is that the points $(0, 0)$ and $(0, 1)$ are encoded as $|0\rangle$ (subject to a global phase), and the points $(1, 0)$ and $(1, 1)$ are mapped to $|1\rangle$ (Fig. 5). Independently of the ansatz, the initial points $(0, 0)/(0, 1)$ and $(1, 0)/(1, 1)$ would be indistinguishable in their quantum state representation and no quantum classifier would be able to provide a good classification performance since data encoding is not suitable. Of course, there are ways to address this inconvenience such as changing the feature normalization function, or making a classical data preprocessing before quantum embedding. These possibilities involve an increase in the workload of the human analyst to set a proper experimental setting as the size of the dataset increases.

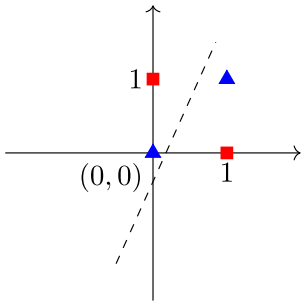


Fig. 4 Data of the XOR problem in $\mathbb{R}^2$. Triangles stand for class 0 and squares for class 1

If we move again to a classical machine learning approach that solves the *XOR* problem, a simple quadratic transformation such as the mapping $(x_i^1, x_i^2) \mapsto (x_i^1, (x_i^1 - x_i^2)^2)$ transforms the data space so that the resulting space is linearly separable, and a simple linear classifier successes in its task (Fig. 6). Another possible solution, which avoids non-linear transformations, could be to find an application that injects the data points into a higher dimensional space where they could be linearly separable. This is a well-known strategy in the classical machine learning community and the fundamental hypothesis of well-established models such as support vector machines. Also, we believe it is a key procedure that helped our experiments to success. In the case of the *XOR* problem, it should inject the 2-D *XOR* data into 3-D points in the surface of the Bloch sphere. However, an initial change of basis is required so that the data embedding layer is applied properly.

As an example, let us consider a state preparation layer that learns the parameters $(\theta_0^1, \theta_0^2, \theta_0^3)$ in some way that moves the initial state $|0\rangle$ of the quantum circuit to $|+\rangle$. Considering the same data embedding mechanism used in Fig. 5, now the problem data points are encoded as $(0, 0) \mapsto |+\rangle$, $(0, 1) \mapsto |-\rangle$, $(1, 0) \mapsto |-\rangle$, and $(1, 1) \mapsto |+\rangle$ subject to a global phase difference (see Fig. 7). The quantum states representing the problem data are now linearly separable in the

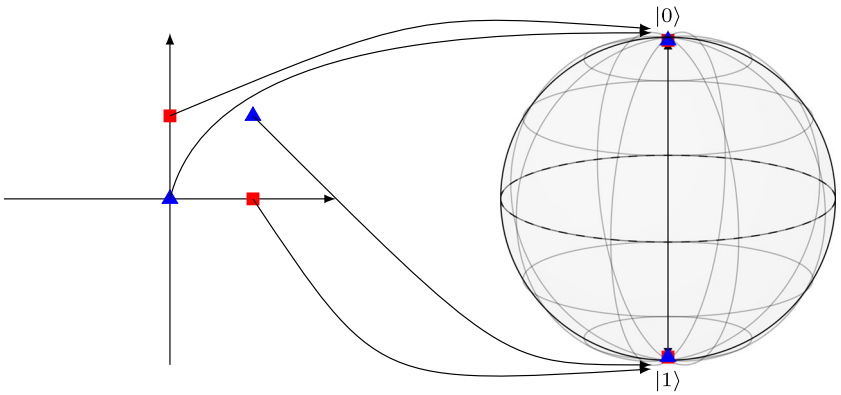


Fig. 5 Raw embedding of XOR problem data into quantum states using the proposed data embedding module

quantum state space, and a simple measurement in the X axis using the σ_x observable could lead to a correct classification without the need of an ansatz module. Thus, the state preparation layer has helped encoding the data into quantum states properly by means of applying an initial change of basis, and this effect helps to improve the encoding performance substantially as we discuss in the experiments.

3 Experiments

This section describes possible solutions to a wide range of machine learning problems that can be reduced to two input features. First, Section 3.1 highlights the most relevant parts of the experimental settings used to solve each problem. After that, Sections 3.2–3.11 address binary and multinomial classification, regression, time series forecasting, clustering, and reinforcement learning problems so that the full range of types of machine learning are covered. Finally, Section 3.12 discusses the results.

3.1 General experimental settings

This section describes the experimental settings used that are not problem-dependent according to the methodological guidelines introduced in Sect. 2.3. Table 1 prints the problem enumeration (column 1), the μ, λ parameters of the evolution strategy training algorithm and the algorithm's learning rate α (column 2), and the maximum number of iterations allowed for training (column 3). An additional stopping criterion was set for each problem if it is solved with maximum accuracy according to the metric used for learning (column 4). We remark that problems 2 and 10 contain two versions in increasing order of complexity. Their corresponding rows separate with symbol – their experimental settings as the simpler (left hand) and the complete (right hand).

The qubit QML proposal was implemented in TensorFlow Quantum v0.7.1. Two different approaches including and not including the state preparation layer were considered (named as *QML (SP)* and *QML (no SP)* in the experiments, respectively). These models were compared in performance with state-of-the-art classical machine learning methods that vary depending of the problem nature:

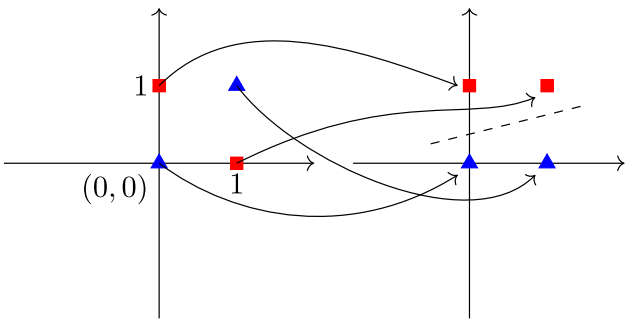


Fig. 6 Quadratic transformation of the XOR problem data for classical machine learning processing using linear classifiers

- In the case of classification tasks, we use a multilayer perceptron feedforward neural network (MLP) for classification and logistic regression as baseline methods. Both models were trained with common settings to minimize the cross-entropy loss function using the Adam optimizer for MLP with a default learning rate $\alpha = 0.001$ and LBFGS for the logistic regression procedure up to convergence. The activation function of hidden neurons was set to ReLU in MLP, as the most widely used contemporary activation function. In problem 1, the internal structure was set to 2 layers with 50 neurons each, 100 for Setosa flower determination and 200 for the full Iris flower in problem 2.
- We selected a MLP for regression as baseline in problems 3, 4, and 5. In these cases, the network topology contains 2 layers with 50 neurons each (problem 3), 100 neurons (problem 4), and 10 neurons (problem 5). In the case of problems 4 and 5, related to time series forecasting, other state-of-the-art methods such as recurrent neural

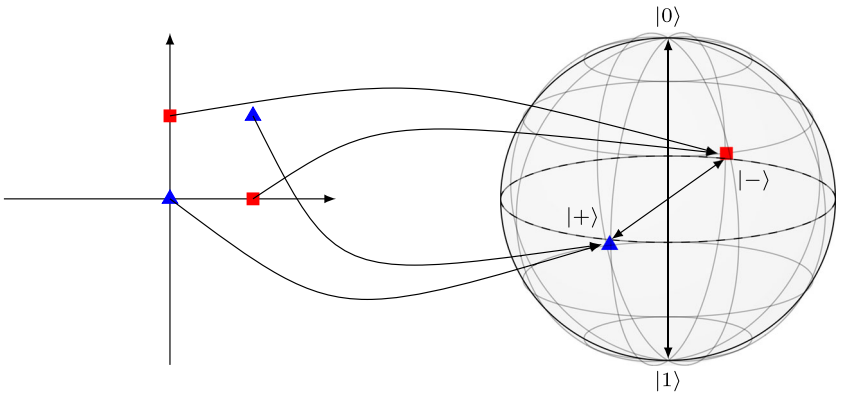


Fig. 7 Quantum embedding of XOR problem data after a state preparation layer that moves $|0\rangle$ to $|+\rangle$

networks were not included to make comparisons as fair as possible since quantum circuits are not able to contain loops that simulate recurrences.

- Problems 6 and 7 are clustering tasks. In these cases, the selected baseline algorithm is the classic K-means considering the sum of the squared differences between patterns as distance metric (the square of the Euclidean distance).
- The last problems 8, 9, and 10 come from the field of reinforcement learning. The selected baseline method is the double deep Q-network (DDQN) algorithm (Hasselt et al. 2016) with MLP as policy encoder containing 2 layers with 100 neurons in all experiments.

The experimentation with the baseline algorithms used for comparison was carried out with the Python library Scikit-learn v1.5.0 except for the DDQN which was implemented by scratch. All datasets were divided in two train (80%) and test (20%) sets as a standard procedure. Also, the same preprocessed data were fed to all qubit QML approach and corresponding baseline proposals of each problem for learning, so that all methods share the same initial conditions for training and test. The source code used in our experiments is available for free at the repository https://github.com/manupc/OneQubit hosted in GitHub.

Table 1 Experimental settings of the qubit QML proposal for each problem

Problem	$\mu/\lambda/\alpha$ (ES)	Iterations	Metric
1	5/30/0.1	100	Accuracy
2	1/5/0.1–5/30/0.2	50	Accuracy
3	5/20/0.1	30	MSE
4	10/50/0.1	100	MSE
5	10/50/0.3	100	MSE
6	5/10/0.5	20	S-score
7	10/50/0.3	30	S-score
8	2/10/0.1	50	$\overline{R}$
9	3/15/0.1	50	$\overline{R}$
10	10/30/0.1–10/50/0.3	500–200	$\overline{R}$

3.2 Problem 1: the bank note authentication dataset

The Bank Note Authentication (BNA) dataset describes a binary classification problem that contains data extracted from 1372 images of genuine and forged bank notes. A sample (x_i, y_i) in the dataset contains four continuous input features $x_i = (x_i^1, x_i^2, x_i^3, x_i^4)$ that correspond to the variance (x_i^1), skewness (x_i^2) and curtosis (x_i^3) of the wavelet transformed image, and the image entropy (x_i^4). The output y_i is a discrete 0/1 binary value to distinguish if the sample banknote x_i is legitimate or not.

The preprocessing step involved scaling the features to the range $[0, \pi]$ as a first step, so that they could be used as inputs in the data encoding layer of the proposed model structure. After that, a preliminary exploratory data analysis concluded that a useful feature extraction is to combine the variance and skewness into a single aggregated feature as $x_i^5 = \sqrt{x_i^1 + x_i^2}$. Then, it is possible to divide the decision space using a feature selection of values (x_i^5, x_i^3) as it is shown in Fig. 8.

A qubit model without state preparation with three parameters was used to solve this problem with the structure $U_s(\theta_0) = I$, $U_e(x_i) = R_z(x_i^3)R_y(x_i^5)$ and $U_a(\theta_1) = R_x(\theta_1^1)R_z(\theta_1^2)R_y(\theta_1^3)$. Also, the same qubit model including the state preparation layer was included in the experiments with $U_s(\theta_0) = R_x(\theta_0^1)R_z(\theta_0^2)R_y(\theta_0^3)$. Measurement was performed using the σ_z observable, and the model output $\hat{y}_i(\theta_1)$ was calculated as a problem-dependent mapping described in Eq. 12.

$$\hat{y}_i(\theta_1) = \begin{cases} 1 & , \quad \langle \hat{y}_i(\theta_1)|\sigma_z|\hat{y}_i(\theta_1)\rangle \geq 0 \\ 0 & , \quad \langle \hat{y}_i(\theta_1)|\sigma_z|\hat{y}_i(\theta_1)\rangle < 0 \end{cases} \tag{12}$$

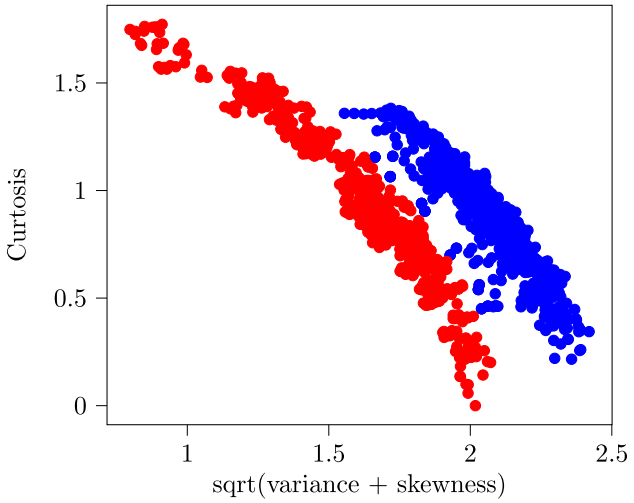


Fig. 8 Decision space of the BNA dataset after feature extraction and selection

The results were compared with a multi-layer perceptron for classification (MLP) and a logistic regression (LR) model as two of the most well-known classifiers in classical machine learning in 30 different experiments. Table 2 describes the results obtained for the qubit model (columns 2 and 3), the MLP (column 4), and the LR (column 5). Rows 2 and 3 print the average accuracy in the training and test sets together with the standard deviation in the 30 experiments performed. The subscripts in row 3 rank the algorithms from the best average results (value 1) to the worst (value 3) according to a Wilcoxon statistical test with 95% of confidence level. Row 4 describes the accuracy in train and test sets of the best solution found. Finally, row 5 remark the number of trainable parameters that were used by the three models.

According to Table 2, the method that provides the average best accuracy in the prediction of legitimate and forge banknotes is the qubit QML model, and specially when no state preparation layer is included in the structure. The statistical tests concluded that there are significant differences with respect to the second best methods (*QML (SP)* and *MLP*). As it could be expected, the worst model is LR since the decision space shown in Fig. 8 is not linearly separable. We remark that both QML and MLP predictors are able to provide solutions that solve the problem with 100% accuracy in both train and test sets, although the number of parameters required by the qubit model is substantially lower. Figure 9 provides a deeper insight about the behaviour of each model, and plots the decision boundaries of QML, MLP and LR in Fig. 9a, b, and c, respectively. Red triangles are asociated with the expected output $y_i = 0$ and blue circles to $y_i = 1$.

If we analyze the QML results from a geometric point of view considering the prediction model structure, we may realize that the ansatz operations $R_x(\theta_1^1)$, $R_z(\theta_1^2)$, $R_y(\theta_1^3)$ perform a rotation of the input state $|x_i\rangle$ around the three axes in the Bloch sphere to generate the output state $|\hat{y}_i(\theta_1)\rangle$. These operations influx a change of basis to the input state $|x_i\rangle$ for each pattern, so that the encoded output $|\hat{y}_i(\theta_1)\rangle$ makes the problem to be linearly separable in the quantum state space of the problem. Measurement provides non-linearity to map the quantum state $|\hat{y}_i(\theta_1)\rangle$ to the final output $\hat{y}_i(\theta_1)$. This non-linear mapping is clearly observed in the decision boundaries of Fig. 9a. A keen eye could also observe a non-linear decision boundary for MLP in Fig. 9b, which makes it possible to obtain 100% of accuracy both in train and test.

3.3 Problem 2: the Iris flower dataset

The *Iris flower* dataset is probably one of the most used classification problems in machine learning. It contains patterns (x_i, y_i) with four input features $x_i = (x_i^1, x_i^2, x_i^3, x_i^4)$ with the sepal length (x_i^1), sepal width (x_i^2), petal length (x_i^3), and petal width (x_i^4) of 150 samples of three types of Iris flower: *Setosa*, *Versicolor*, *Virginica*. The goal is to predict which type of flower y_i is a given specimen x_i considering the four input features. In a preliminary EDA, we may verify that the samples of type *Setosa* can be visually distinguished with 100% accuracy from the others using a feature selection of the *petal length* and *petal width*, as it can be verified in Fig. 10. Thus, the problem of the *Iris flower* is usually studied both as a binary or one-class classification task to determine if a specimen is *Setosa* or not, or as a multi-class classification with three possible classes.

Different quantum machine learning classifiers have also used this data set as a proof of concept as in Piatrenka and

Table 2 Accuracy results in the BNA dataset. Row 1 contains the name of the models. Rows 2 and 3 print the average percentage of accuracy of the models and the standard deviation in all executions. Row 4 describes the percentage of models accuracy in both train and test sets, respectively. Row 5 includes the number of parameters to be optimized for the models

	QML (SP)	QML (no SP)	MLP	Logistic regression
Avg. training Acc	99.94 ± 0.15	100.00 ± 0.00	99.55 ± 0.42	98.81 ± 0.00
Avg. test Acc	$99.93 \pm 0.15_{(2)}$	$99.95 \pm 0.12_{(1)}$	$99.83 \pm 0.29_{(2)}$	$99.27 \pm 0.00_{(3)}$
Best	100.00/100.00	100.00/100.00	100.00/100.00	98.82/99.27
Parameters	6	3	2751	3

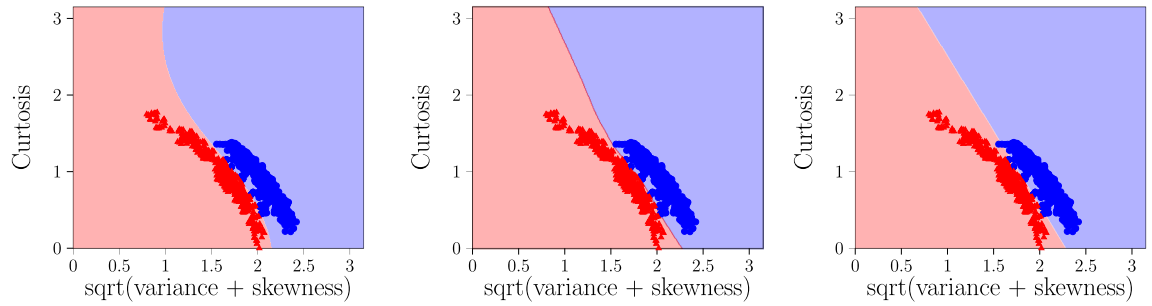


Fig. 9 Decision boundaries of each trained model in the BNA dataset

Rusek (2022), although using a larger number of qubits, typically two. The very same structure of the qubit model used in Sect. 3.2 can be used to build a binary classifier for the *Setosa* flower using a single qubit. In this case, the preprocessing step only involves scaling the petal length x_i^3 and width x_i^4 to the range $[0, \pi]$. The unitary matrix for data encoding is given by $U_e(x_i) = R_z(x_i^4)R_y(x_i^3)$ so that the petal length is encoded into the magnitudes of the qubit and the petal width in the phase. After that, a parameterized ansatz $U_a(\theta_1)$ with three parameters is used as computation layer. Measurement is performed using the σ_z observable as in the BNA problem, and we also use the mapping of Eq. 12 to distinguish if a flower specimen is *Setosa* ($\hat{y}_i(\theta_1) = 0$) or not ($\hat{y}_i(\theta_1) = 1$). The summary of results are shown in Table 3 with the same format as Table 2 for 30 different experiments and comparison with a MLP classifier and a LR model.

Determining if a flower specimen is *Setosa* or not is clearly linearly separable, so that the four models are equivalent in terms of performance and achieve 100% of accuracy in train and test sets. Moreover, LR and qubit *QML (no SP)* are also equivalent in the number of parameters to be optimized, and both have the minimum possible set of parameters. Figure 11 gives support to these results and plots the decision boundaries of the three QML, MLP, and LR predictors. It is worth noting the non-linear decision bounds introduced by measurement in the QML model, which could provide a behaviour near overtraining if more samples would be available in the dataset.

Besides binary classification, we can also perform multiclass prediction tasks with one qubit if we extend measurement to a larger number of observables rather than σ_z. As the *Iris flower* dataset contains three classes, one possible strategy could match each axis of the Bloch sphere to a flower type, then calculate the expectation of observables $\sigma_x, \sigma_y, \sigma_z$ and design a suitable mapping strategy to provide the QML model output $\hat{y}_i(\theta_1)$. The designed solution in this manuscript is described in Eq. 13 where $\hat{y}_i(\theta) = 0$ is for *Setosa* classification, $\hat{y}_i(\theta) = 1$ if the specimen is *Versicolor* and $\hat{y}_i(\theta) = 2$ if the flower sample is *Virginica*.

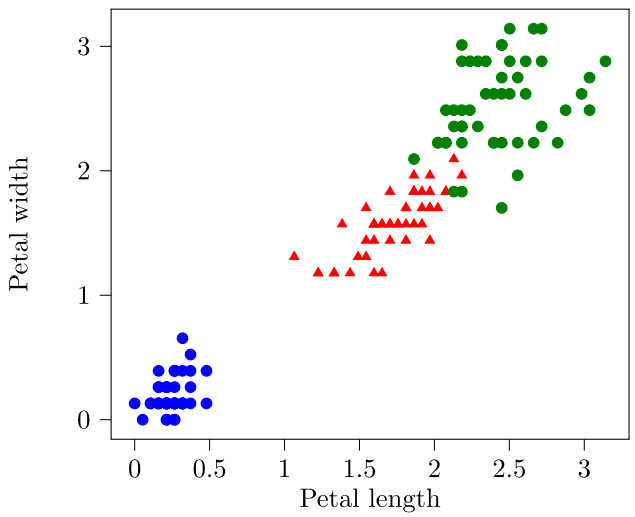


Fig. 10 Decision space of the Iris dataset after feature selection. Blue circles, *Setosa*; red triangles, *Versicolor*; green circles, *Virginica*

$$\hat{y}_i(\theta) = \begin{cases} 0 & , \quad z = argmax_{k\in\{x,y,z\}} \langle \hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle \\ 1 & , \quad x = argmax_{k\in\{x,y,z\}} \langle \hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle \\ 2 & , \quad y = argmax_{k\in\{x,y,z\}} \langle \hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle \end{cases} \tag{13}$$

In our experiments, we needed no further preprocessing than the one used for the binary *Setosa* flower determination, although setting a state preparation stage $U_s(\theta_0)$ with three parameters was required to achieve a maximum performance of the qubit QML model. Table 4 describes the results of the experimentation in 30 different runs against the MLP and LR classical machine learning models with the same format as Tables 2 and 3. In this case, the Wilcoxon statistical test concluded that there are no significant differences between the classical MLP and LR with the qubit *QML (SP)* proposal in terms of average accuracy among runs, although the number of parameters used by *QML (SP)* is lower than in the other two techniques. However, the qubit QML methods were the only models that were able to provide a solution with 100% of accuracy in the test set, and *QML (SP)* achieved a state-

Table 3 Accuracy results in the Iris Setosa classification problem with the same format as Table 2

	QML (SP)	QML (no SP)	MLP	Logistic regression
Avg. training Acc	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
Avg. test Acc	$100.00 \pm 0.00_{(1)}$	$100.00 \pm 0.00_{(1)}$	$100.00 \pm 0.00_{(1)}$	$100.00 \pm 0.00_{(1)}$
Best	100.00/100.00	100.00/100.00	100.00/100.00	100.00/100.00
Parameters	6	3	10, 504	3

of-the-art global accuracy of 98% in the whole dataset using just two input features. The classical MLP can also achieve this 98% of accuracy but using the information provided by more input features. If the input information is reduced to features (x_i^3, x_i^4) as in out experiments, the global accuracy drops to 5 missclassified samples while the *QML (SP)* stands at 3 sample misclassifications.

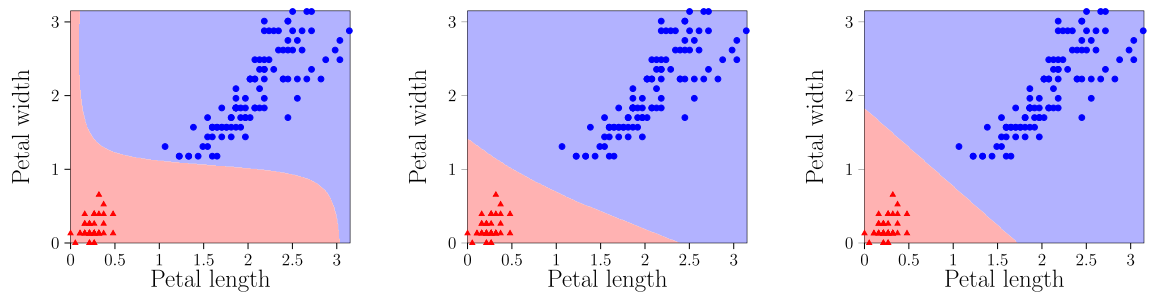
On the other hand, if we compare the results of the full *QML (SP)* model using the state preparation layer and *QML (no SP)* in columns 2 and 3, we may verify a significant increase in performance in the former with respect to the latter. As it was discussed in Sect. 2.4, the state preparation layer moves the initial state of the quantum circuit $|0\rangle$ to another state $|\psi\rangle$ using a change of basis, and this step might influence the quality of the data encoding in the data embedding module. If no state preparation is present, under the same experimental settings the *QML (no SP)* proposal drops performance in average to a 71.58% of training classification accuracy, which is significantly worse than all the remaining models. The statistical test concluded that there are significant differences between this model and the other three, which leads to the conclusion that it is a not so desirable solution with respect to the full *QML (SP)* in this problem. In spite of this decrease of performance, the *QML (no SP)* model is still able to achieve near-optimal results with a 96.67% of classification success in training and 100% in test, according to the best solution found in 30 executions. It also holds the advantage of using 3 trainable parameters only.

As we did for the BNA problem and the *Iris Setosa* determination, Fig. 12 plots the decision boundaries of each model to fully classify all three types of *Iris flowers* with the qubit QML (SP), MLP, and LR models. In the case of the qubit QML proposal, Fig. 12a shows clearly the non-linear behaviour in the decision bounds of the model, introduced by the measurement operator and the quantum-classical decision mapping of Eq. 13, which leads to the missclassification of three *Versicolor* flower samples as *Virginica*. In the case of the MLP, three *Versicolor* samples were classified as *Virginica* and two *Virginica* as *Versicolor*. Finally, LR missed in the classification of 4 samples of *Virginica* as *Versicolor* and 2 specimens of *Versicolor* as *Virginica* using linear discrimination.

3.4 Problem 3: the combined cycle power plant dataset

The combined cycle power plant dataset (CCP) is a regression problem over real (Fig. 13) data collected from a combined cycle power plant from 2006 to 2011. It contains 9568 patterns (x_i, y_i) with four features $x_i = (x_i^1, x_i^2, x_i^3, x_i^4)$ containing the average ambient temperature (x_i^1), average ambient pressure (x_i^2), relative humidity (x_i^3), and exhaust vacuum (x_i^4). These four variables are used to predict y_i, the hourly electrical energy output provided by the plant.

The preprocessing of this dataset starts with a change of scale of all features to the range $[0, \pi]$ and y_i to the range of the σ_z observable in $[-1, 1]$. After that, a preliminary EDA concluded that a feature extraction step could be benefitial to create two additional features x_i^5, x_i^6 as linear combinations of the original ones as $x_i^5 = 0.5(x_i^1 + x_i^2)$ and $x_i^6 = 0.5(x_i^3 + x_i^4)$. These two features x_i^5, x_i^6 were used as inputs to the regression models used in our experimen-



(a) qubit QML (no SP) (b) Multilayer Perceptron (c) Logistic Regression

Fig. 11 Decision boundaries of each model in the Iris Setosa classification problem. Blue circles, *Setosa*; red triangles, *Setosa*; blue circles, *Versicolor* or *Virginica*

Table 4 Accuracy results in the Iris flower classification problem with the same format as Table 2

	QML (SP)	QML (no SP)	MLP	Logistic regression
Avg. training Acc	93.00 ± 10.39	71.58 ± 11.00	96.22 ± 0.42	95.00 ± 0.00
Avg. test Acc	$93.22 \pm 10.49_{(1)}$	$71.67 \pm 12.13_{(2)}$	$96.67 \pm 0.00_{(1)}$	$96.67 \pm 0.00_{(1)}$
Best	97.50/100.00	96.67/100.0	96.67/96.67	95.00/96.67
Parameters	6	3	41, 403	9

tation. We used the standard qubit QML model with 3 (no state preparation layer) and 6 parameters (full *QML (SP)*) to address the problem as $|\hat{y}(\theta)\rangle = U_a(\theta_1)U_e(x_i)U_s(\theta_0)|0\rangle$, where $U_e(x_i) = R_z(x_i^6)R_y(x_i^5)$, and trained the model 30 times with different random initial solutions to perform statistical analysis. The classical machine learning baseline used for comparison is a multi-layer perceptron for regression, as this is one of the most widely known classical models to solve this type of problems. The target goal is to minimize the mean square error measure (MSE) in both cases.

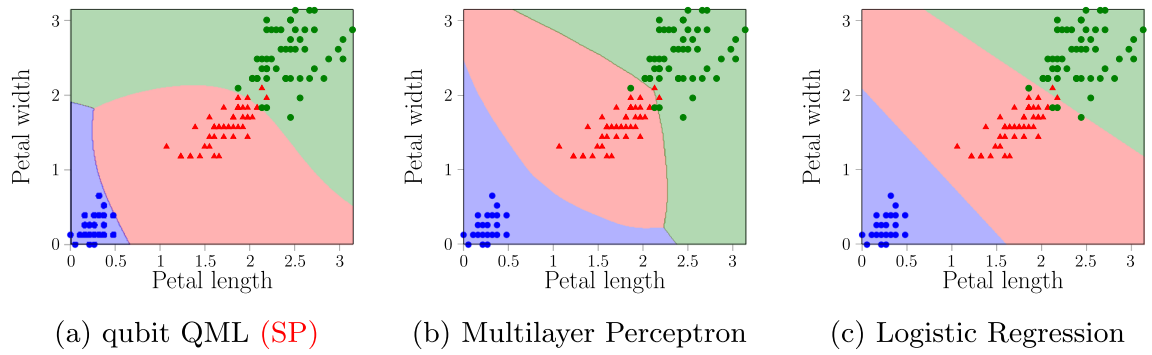
Table 5 displays the summary of results for the qubit QML models and MLP, both in training and test sets. According to a Wilcoxon statistical test with 95% of confidence level, there are significant differences in the performance of QML and MLP, and the latter provides better accuracy with respect to QML. On the other hand, if we focus on the comparison of *QML (SP)* and *QML (no SP)*, in this problem, it is also shown a big difference of one order of magnitude in the MSE metric. The statistical Wilcoxon test concluded that there are significant differences between both prediction error data distributions in test. Thus, this problem is another example of the benefits that the state preparation layer could provide before data embedding.

We plot the prediction results of the best QML and MLP solutions in the last 100 test data in Fig. 14, together with the true target values. In Fig. 14, we may verify that MLP provides a higher precision in the predictions. Moreover, we can observe that the qubit *QML (SP)* was also able to learn the underlying dependecy of the outputs y_i with respect the inputs x_i, although with a lower accuracy. If we compare both models in terms of complexity, the qubit *QML (SP)* method still needs 6 parameters to be optimized while the MLP requires 151 in this problem.

3.5 Problem 4: the air passengers time series

The air passenger dataset is a classic time series dataset widely used in academia and in testing of time series forecasting models. It contains the number of international airline passengers in thousands from 1949 to 1960 with monthly granularity (144 data points). A typical initial preprocessing step in this time series aims to reduce the increase of time series variance over time by means of a logarithmic transformation of the whole dataset. After that, a linear trend is also fitted and removed to center the time series data in zero. This preprocessed time series $x(t)$ was used in our experiments to forecast the last 20% of the time series data with a forecasting horizon of 1 as $x(t+1) = f(x(t), x(t-1), ..., x(t-h), \theta)$ where f is the prediction model hypothesis, θ the model parameters, and h the historical data time horizon allowed to predict the next time series value. Figure 15 plots the original time series data and distinguishes between the training and test sets.

After the usual logarithmic transformation and trend preprocessing of this time series, we removed the time component and created a set of patterns (x_i, y_i) where $x_i = (x(i-1), x(i-2), ..., x(i-h))$ and $y_i = x(i)$. The selected baseline classical method to compare the performance of the qubit QML structure is a MLP for regression as it is one of the most widely known classical machine learning feedforward models used in time series forecasting. An initial experimental setup tested different values of h so that both MLP and



(a) qubit QML (SP) (b) Multilayer Perceptron (c) Logistic Regression

Fig. 12 Decision boundaries of each model in the full Iris flower classification problem. Blue circles, *Setosa*; red triangles, *Versicolor*; green circles, *Virginica*

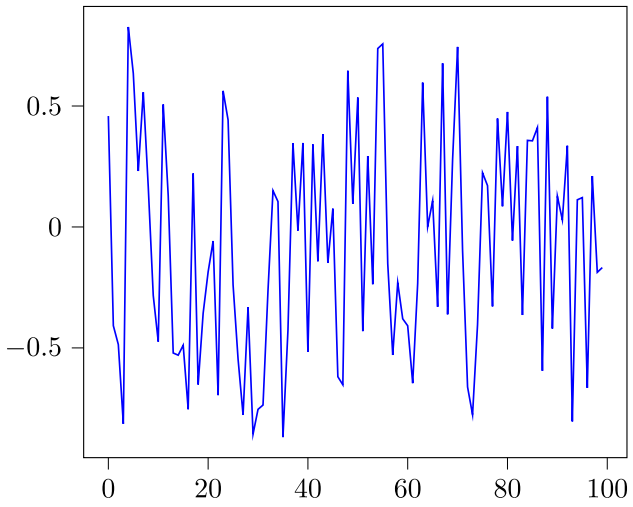


Fig. 13 Last 100 test data of the combined cycle power plant dataset after classical preprocessing

qubit QML methods could fit the time series data accurately. We found that a value $h = 4$ was the minimum historical time horizon able to provide a precise time series forecasting, and trained 30 different qubit QML and MLP models to be able to analyze results statistically. In both models, the target is to minimize the mean squared error between the expected output y_i and the model output $\hat{y}_i(\theta)$.

The structure of the qubit QML approach differs from the ones shown in the previous problems, and it contains a number of h processing layers according to the diagram of Fig. 3. The full structure of the model used in the experimentation is $|\hat{y}_i(\theta)\rangle = \prod_{k=h}^{1} [U_a(\theta_k)U_e(x(i-k))]\, U_s(\theta_0)|0\rangle$ for the *QML (SP)* model, where $U_e(x(i-k)) = R_y(x(i-k))$, and $U_s(\theta_0) = I$ for the *QML (no SP)* approach. Measurement was carried out using the σ_z observable, and the quantum-classical output mapping is calculated as $\hat{y}_i(\theta) = \langle \hat{y}_i(\theta)|\sigma_z|\hat{y}_i(\theta)\rangle$. Thus, as a preliminary preprocessing step before training, the values of the input patterns x_i were scaled to the range $[0, \pi]$ so that they could be used as $R_y(\phi)$ angle rotations and their corresponding outputs y_i to the range of the observable $[-1, 1]$.

Table 6 summarizes the results obtained in training and test sets for both approaches. It can be verified that MLP is able to provide a better prediction accuracy than the QML approach in both training and test sets, and a Wilcoxon statistical test with 95% of confidence level helped to verify this result. Moreover, the best solution found by MLP performed better than the best solution provided by the qubit QML. However, the number of parameters of qubit *QML (SP)* is 15 (3 for each processing layer and state preparation), while the MLP contains more than 10,000 parameters. If we compare the results of MLP and *QML (no SP)*, we may observe a similar behaviour to the one found in Sect. 3.4 as the *QML (no SP)* model was unable to learn the data properly and provided an error metric of one order of magnitude worse. In this problem, we also verify that the use of the state preparation layer is an useful strategy to ease data encoding and later model training.

Figure 16 plots the predictions of models *QML (SP)* and MLP in the preprocessed test set with the logarithmic transformation and removed trend. Here, we also included further results using a historical time horizon of h=2 and h=3 in Fig. 16a and b to show the deficiency of both MLP and qubit QML to approximate the true target value y_i. In these cases, it seems that both *QML (SP)* and MLP attempt to approximate the output y_i by means of the replication of the past historical data. On the other hand, Fig. 16c contains the predictions using h=4. It complements the data of Table 6 and shows graphically how the qubit QML was able to learn the dependencies of the outputs y_i with respect to the inputs x_i, although the prediction accuracy was worse than using the classical MLP.

3.6 Problem 5: the OikoLab weather dataset

The OikoLab weather dataset is composed of eight time series including hourly data of ambient temperature, dew-point temperature, wind speed, mean sea level pressure, relative humidity, surface solar radiation, surface thermal radiation, and total cloud cover. It was collected from January of 2010 to December of 2021 nearby Monash University in Victoria, Australia. In our manuscript, we focus on the average ambient temperature time series with monthly granularity, therefore reducing the time series length from 100058 multidimensional data points to 138 values of one dimension. We used the first 80% of time series data for training and the remaining 20% for test.

Table 5 Prediction results in the CCP regression problem: Row 1 prints the model names. Rows 2 and 3 contain the average training and test mean square error (MSE) obtained by the models in 30 experiments, and their corresponding standard deviation. Row 4 describes the MSE in train and test of the best solution found, and row 5 shows the number of free parameters of the models

	QML (SP)	QML (no SP)	MLP
Avg. training MSE	0.048 ± 0.0001	0.108 ± 0.121	0.019 ± 0.0011
Avg. test MSE	$0.051 \pm 0.0003_{(2)}$	$0.112 \pm 0.125_{(3)}$	$0.018 \pm 0.0011_{(1)}$
Best	0.048/0.050	0.053/0.056	0.017/0.017
Parameters	6	3	151

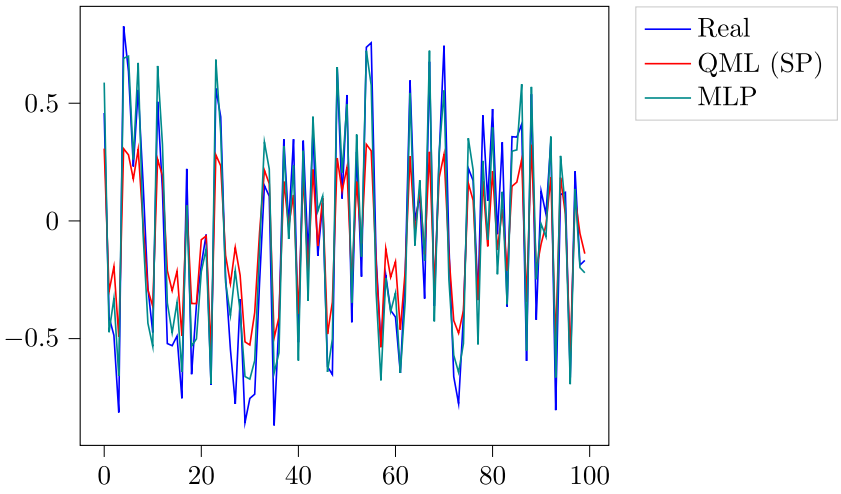


Fig. 14 Estimation of hourly electrical energy output by QML (SP) and multilayer perceptron in the combined cycle power plant test dataset

The methodology used to solve this problem is the same as the one used to solve the air passengers time series forecasting dataset, so that the qubit QML model structure follows the diagram in Fig. 3, and it calculates the output quantum state as $|\hat{y}_i(\theta)\rangle = \prod_{k=h}^{1} [U_a(\theta_k)U_e(x(i-k))]\, U_s(\theta_0)|0\rangle$, where $U_e(x(i-k)) = R_y(x(i-k))$ for the model *QML (SP)*, and $U_s(\theta_0) = I$ for *QML (no SP)*. We also used the same measurement and quantum-classical output mapping strategy as it was introduced in Sect. 3.5: the σ_z observable was used, and the final model output was calculated as $\hat{y}_i(\theta) = \langle \hat{y}_i(\theta)|\sigma_z|\hat{y}_i(\theta)\rangle$. Once the initial data were aggregated with monthly granularity, we selected $x(t)$ as the average monthly ambient temperature time series and created a set of patterns (x_i, y_i) as $x_i = (x(i-1), x(i-2), ...x(i-h))$, and $y_i = x(i)$. The input values of patterns x_i were scaled to the range $[0, \pi]$ and the output values y_i to the range $[-1, 1]$. Figure 17 describes the time series values in both training and test sets once the output data were scaled to $[-1, 1]$.

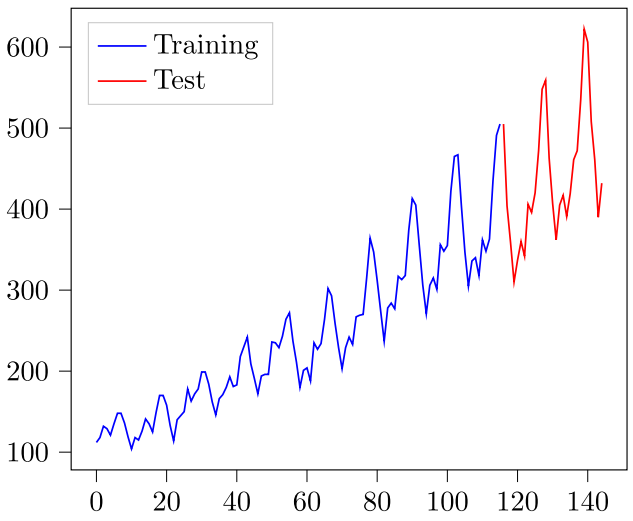


Fig. 15 Air passengers time series data. The X axis is associated to the time evolution (measured in months), and the Y axis contains the number of passengers (in thousands)

Table 7 describes the results of training the qubit QML models in 30 separate experiments and also a baseline MLP algorithm for regression. In this case, we see that the MSE accuracy in both training and test sets is similar in the three models, and a Wilcoxon test with 95% of confidence level concluded that there are no significant differences in their performance. Also, the best solution found by qubit QML approaches and MLP provides very similar performance in this problem. The major difference can be found in the number of parameters of each algorithm: The qubit QML approaches need 12 and 9 parameters (*QML (SP)* and *QML (no SP)*, respectively) for the three processing layers and the initial state preparation, while the MLP requires 161 parameters. Figure 18 plots the predictions provided by the best solution found of the *QML (SP)* and MLP models with different historical time horizons h=2 and h=3. In Fig. 18a, it can be verified that a time horizon of h=2 does not provide enough information to either the qubit QML or the MLP to provide suitable results. In fact, we observe a similar behaviour to the one we displayed previously in Fig. 16a and b to solve the air passengers dataset, and the models attempt to provide a prediction using the most recent historical data that are fed as inputs. On the other hand, Fig. 18b shows the predictions of the best qubit QML and MLP models with h=3 as a complementary information to Table 7. Here, we verify visually than the predictions provided by both approaches are very similar, according to the similarities found in Table 7.

Table 6 Accuracy results in the air passengers time series forecasting problem with h=4, with the same format as Table 5

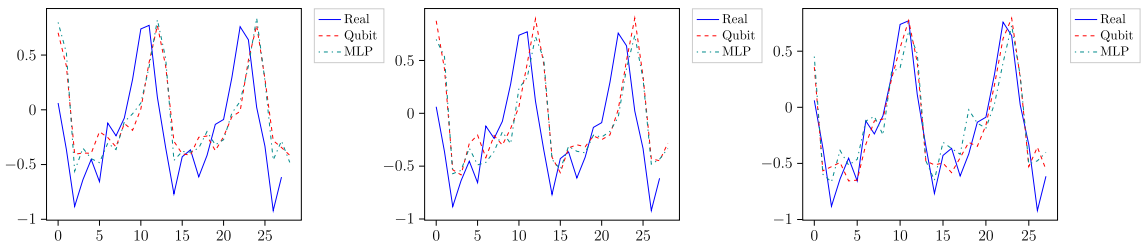
	QML (SP)	QML (no SP)	MLP
Avg. training MSE	0.069 ± 0.011	0.106 ± 0.006	0.031 ± 0.003
Avg. test MSE	$0.083 \pm 0.024_{(2)}$	$0.141 \pm 0.032_{(3)}$	$0.064 \pm 0.005_{(1)}$
Best	0.066/0.043	0.106/0.080	0.028/0.054
Parameters	15	12	10, 701

3.7 Problem 6: Old Faithful geyser eruptions clustering

Old Faithful is a geyser located in the Yellowstone National Park in USA. The Old Faithful Geyser dataset comprises 272 samples $x_i = (x_i^1, x_i^2)$ containing the waiting time between eruptions (x_i^1) and the duration of each eruption (x_i^2), respectively, and it is a classical dataset used for clustering in unsupervised learning. A preliminary EDA in Fig. 19 shows that both the waiting time and the duration are suitable candidates to distinguish between two possible categories of eruptions (number of clusters).

A qubit model $|\hat{y}_i(\theta)\rangle = U_a(\theta_1)U_e(x_i)U_s(\theta_0)|0\rangle$ with three parameters for state preparation in $U_s(\theta_0)$ *QML (SP)* model, or no state preparation layer (*QML (no SP)*), and three parameters for data processing in $U_a(\theta_1)$ can be used to perform the clustering task. If we assume a number of clusters $K = 2$, then the observable σ_z can be used for measurement, and the model output $\hat{y}_i(\theta)$ can be calculated as in a binary classification problem using the quantum-classical output mapping of Eq. 12. If $\hat{y}_i(\theta) = 0$, then the input x_i is assigned to a cluster and to the other cluster in case of $\hat{y}_i(\theta) = 1$. The parameter learning of the qubit model is carried out through the maximization of the Silhouette score as target metric to be optimized. This way of processing helps us to avoid the calculation of distances between different data patterns x_i, x_j that would require further qubits to achieve a quantum clustering solution such as quantum K-means or K-medians (Yarkoni et al. 2021). We compared the results of our approach with the classical K-means algorithm as baseline. As in the previous problems, we performed 30 runs of each algorithm, and Table 8 summarizes the results.

According to Table 8, the S-score used as learning metric in QML helps the qubit QML structure to achieve the same best performance as the classical K-means both in training and test sets. Also, the exact same results were obtained with *QML (SP)* and *QML (no SP)*, meaning that the use of the state preparation layer is not required in this problem. A Wilcoxon test with 95% of confidence level confirms this statement and suggests that there are no significant differences in the resulting performance distributions. Also, we remark that all executions of K-means returned a solution with the same S-score both in training and test, and also the 30 executions of QML returned the same S-score in training with very few differences in the test set. However, the K-means algorithm requires 4 parameters only (two parameters for each centroid) while the QML method needs 6 and 3 parameters for *QML (SP)* and *QML (no SP)*, respectively. Figure 20 shows the decision boundaries of both QML and K-means algorithms. In the case of Fig. 20a, the non-linearity introduced by measurement can be observed visually with respect to the linear boundary of K-means in Fig. 20b. In fact, we may verify that the only difference between Fig. 20b and a is a single point located approximately at the center that was assigned



(a) Time horizon h=2 (b) Time horizon h=3 (c) Time horizon h=4

Fig. 16 Estimation of air passengers in the test set by QML (SP) and multilayer perceptron in the air passengers time series

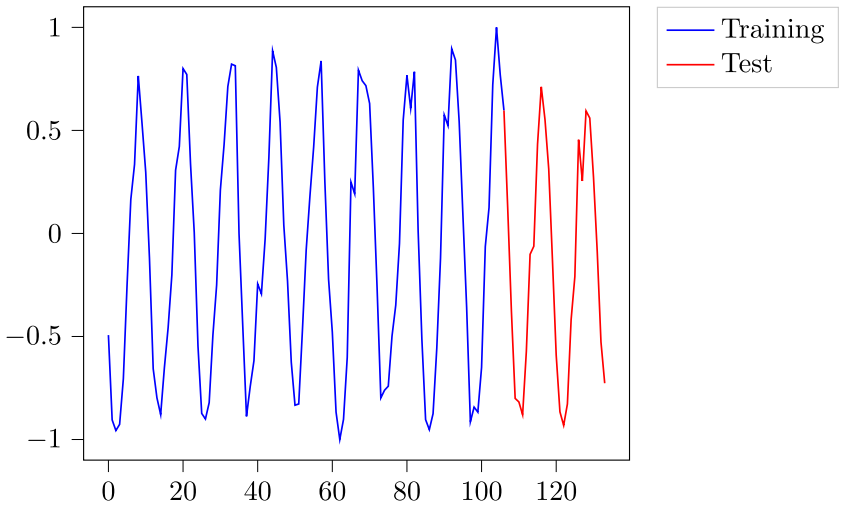


Fig. 17 OikoLab dataset. Test monthly average temperature time series data

to the cluster in the right in Fig. 20a and the cluster in the left in Fig. 20b.

3.8 Problem 7: Blobs clustering

The Blobs cluster is a synthetic dataset generated with the Python Scikit-Learn Machine Learning library, and it is useful to test different types of clustering methods. In contrast to the previous sections, where the datasets are related to problems with real data, here, we use a synthetic dataset to test the limits in the number of outputs that a single qubit can provide. To do so, in our experiments, we assume a synthetic Blobs dataset with 180 samples with two features $x_i = (x_i^1, x_i^2)$ and six target clusters (Fig. 21).

and σ_y to the probability of obtaining $|i\rangle$ or $|-i\rangle$ in the circular basis. There are six possible named quantum states in total, the same number as the count of possible clusters in our dataset. Our strategy for measurement and output mapping from $|\hat{y}_i(\theta)\rangle$ to $\hat{y}_i(\theta)$ considers the matching of each one of these quantum states to a cluster and to decide which is the correct output cluster as the most likely one considering all possibilities. As the three selected observables have eigenvalues $+1$ and -1 for their respective eigenvectors, we already know that $\langle\hat{y}_i(\theta)|\sigma_z|\hat{y}_i(\theta)\rangle < 0$ means that it is more likely to obtain $|1\rangle$ than $|0\rangle$ after measurement, and the same happens to the other observables with their eigenvectors respectively. If we enumerate the six possible clusters from 1 to 6, then this idea can be formalized in Eq. 14.

$$\hat{y}_i(\theta) = \begin{cases} 1 \;,\; z = argmax_{k\in\{x,y,z\}} \left\{|\langle\hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle|\right\} \wedge \langle\hat{y}_i(\theta)|\sigma_z|\hat{y}_i(\theta)\rangle \geq 0 \\ 2 \;,\; z = argmax_{k\in\{x,y,z\}} \left\{|\langle\hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle|\right\} \wedge \langle\hat{y}_i(\theta)|\sigma_z|\hat{y}_i(\theta)\rangle < 0 \\ 3 \;,\; x = argmax_{k\in\{x,y,z\}} \left\{|\langle\hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle|\right\} \wedge \langle\hat{y}_i(\theta)|\sigma_x|\hat{y}_i(\theta)\rangle \geq 0 \\ 4 \;,\; x = argmax_{k\in\{x,y,z\}} \left\{|\langle\hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle|\right\} \wedge \langle\hat{y}_i(\theta)|\sigma_x|\hat{y}_i(\theta)\rangle < 0 \\ 5 \;,\; y = argmax_{k\in\{x,y,z\}} \left\{|\langle\hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle|\right\} \wedge \langle\hat{y}_i(\theta)|\sigma_y|\hat{y}_i(\theta)\rangle \geq 0 \\ 6 \;,\; y = argmax_{k\in\{x,y,z\}} \left\{|\langle\hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle|\right\} \wedge \langle\hat{y}_i(\theta)|\sigma_y|\hat{y}_i(\theta)\rangle < 0 \end{cases} \tag{14}$$

The qubit *QML (SP)* and *QML (no SP)* models were used in this problem to provide the output $|\hat{y}_i(\theta)\rangle = U_a(\theta_1)U_e(x_i)U_s(\theta_0)|0\rangle$, where $U_s(\theta_0) = I$ in the case of *QML (no SP)*. The main difference that makes the solution to this problem unique is the measurement and quantum-classic output mapping procedures. We consider three observables $\sigma_z, \sigma_x, \sigma_y$ for measurement. We remark that σ_z is strongly related to the probability of obtaining $|0\rangle$ or $|1\rangle$ in the computational basis after measurement, while σ_x is to the probability of obtaining $|+\rangle$ or $|-\rangle$ in the Hadamard basis

As we did in Sect. 3.7, the classical K-means algorithm was used as a baseline method against the qubit QML proposal with a number of clusters $K = 6$. In the case of QML, the target goal was also to maximize the Silhouette score. Table 9 prints a summary of results in train and test sets. The qubit *QML (SP)* method was able to achieve the same best performance as the classical K-means in terms of the S-score metric both in training and test sets. While the K-means converged to the same solution in all experiments, the qubit *QML (SP)* procedure provided worse results in average with a low standard deviation in training. A Wilcoxon test

Table 7 Accuracy results in the OikoLab time series forecasting problem with h=3, with the same format as Table 5

	QML (SP)	QML (no SP)	MLP
Avg. training MSE	0.038 ± 0.002	0.044 ± 0.001	0.043 ± 0.010
Avg. test MSE	$0.033 \pm 0.006_{(1)}$	$0.042 \pm 0.002_{(1)}$	$0.035 \pm 0.010_{(1)}$
Best	0.034/0.025	0.036/0.024	0.036/0.024
Parameters	12	9	161

with 95% of confidence level concluded that there are significant differences in performance results among different executions between both methods. On the other hand, if we compare the results of *QML (SP)* and *QML (no SP)* in performance, we may verify that there are significant differences that make the latter method not suitable to solve this problem. As it happened previously in other problems, the State Preparation layer applied before data embedding plays an important role to improve classical data encoding into quantum data states thanks to a change of basis that moves the initial state $|0\rangle$ to a parameterized initial state $|\psi\rangle$.

Figure 22 provides complementary information and plots the decision boundaries among the six clusters of the best solution found in *QML (SP)* and K-means algorithms. As expected, K-means supports a wider region for each cluster due to the algorithm's design and the Euclidean distance it uses to assign each element to a cluster (Fig. 22b), although the qubit *QML (SP)* approach is able to generate non-linear bounds in Fig. 22a. Also, the number of trainable parameters in the qubit *QML (SP)* model is 6 while the required number of parameters in K-means is 12 (two for each cluster).

We remark that the measurement and classical output mapping strategy used in this problem can be theoretically generalized to provide a larger number of discrete outputs using a single qubit, since measurement in the three axes of the Bloch sphere can be used to approximate the amplitudes of a qubit accurately up to a global phase. As an example, if we have a number of clusters equal to eight, then the possible output quantum states $|\hat{y}_i(\theta)\rangle$ could be matched with a cluster as the corresponding octant of its location in the Bloch sphere. We tested this approach in preliminary experiments, although the results we obtained were not able to provide accurate solutions. Many factors can influence this failure, ranging from the need of more powerful search and parameter optimization algorithms to finer preprocessing requirements or the design of the input and output data representation in the quantum state space. As this experiment lies out of the scope of this paper, we left a deeper study to a future research work.

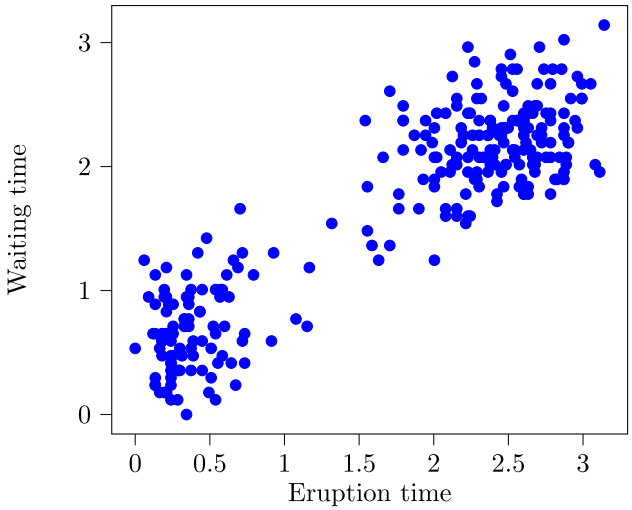
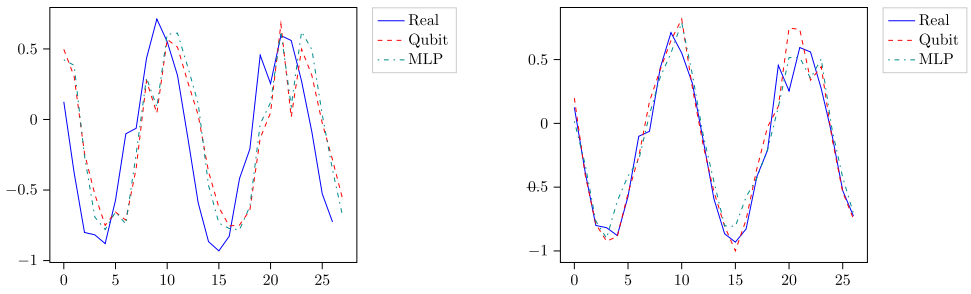


Fig. 19 Old Faithful dataset. Each point corresponds to a geyser eruption. The X axis plots the normalized eruption time, and the Y axis the normalized waiting time between two different eruptions



(a) Time horizon h=2 (b) Time horizon h=3

Fig. 18 Estimation of average monthly temperature in the OikoLab weather dataset with *QML (SP)* and multilayer perceptron models

Table 8 Clustering results in the Old Faithful dataset: row 1 describes the model names. Rows 2 and 3 print the average training and test S-score metric obtained by the models in 30 different experiments, and their respective standard deviation. Row 4 contains the training and test S-score of the best solution found by the models, and row 5 their number of trainable parameters

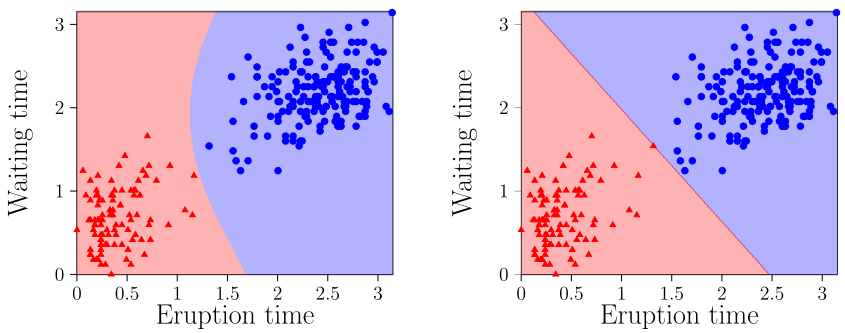
	QML (SP)	QML (no SP)	K-means
Avg. training S-score	0.92 ± 0.00	0.92 ± 0.00	0.92 ± 0.00
Avg. test S-score	$0.89 \pm 0.03_{(1)}$	$0.89 \pm 0.03_{(1)}$	$0.91 \pm 0.00_{(1)}$
Best	0.92/0.91	0.92/0.91	0.92/0.91
Parameters	6	3	4

3.9 Problem 8: the CartPole environment

The CartPole is a control problem and one of the best-known reinforcement learning environments in the Farama Foundation's Gymnasium simulator (formerly OpenAI's Gym). It is composed of a moving cart with a movable pole attached to its center (Fig. 23). The goal of the CartPole environment is to prevent that the pole falls for as many time instants as possible. The agent perception at a given time instant t, $x_t = (x_t^1, x_t^2, x_t^3, x_t^4)$, contains four features: a cart position (x_t^1) and velocity (x_t^2) in the computer screen and a pole angle (x_t^3) and angular velocity (x_t^4), respectively. On the other hand, the action set is discrete and contains two actions $\{a_1, a_2\}$: The first pushes the cart to the left with constant force, and the other to the right. The agent receives a reward +1 for each time instant that the pole is on top, and the environment ends either when the pole falls or the cart leaves the computer screen. This environment is considered solved in the existing literature if an agent is able to hold the pole on top of the cart for at least 500 time steps in 100 consecutive environment tests. According to our previous study (Cuellar et al. 2024), the CartPole can be addressed with quantum reinforcement learning and solved using the two pole features x_t^3, x_t^4 only.

The experiments with one qubit in this problem consider a full qubit QML structure as $|\hat{y}_t(\theta)\rangle = U_a(\theta_1)U_e(x_t)U_s(\theta_0)|0\rangle$ with six parameters to be optimized for *QML (SP)* and three for *QML (no SP)*. The σ_z observable for measurement. The quantum-classical output mapping is equivalent to the binary classification strategy in Eq. 12 where $\hat{y}_t(\theta) = 0$ selects action a_1 and $\hat{y}_t(\theta) = 1$ chooses action a_2. Data encoding is carried out as $U_e(x_t) = R_z(x_t^4)R_y(x_t^3)$ assuming x_t^3, x_t^4 are scaled to the range $[0, \pi]$. We compared the results of the qubit QML proposal with the standard Double Deep Q-Network (DDQN) algorithm (Hasselt et al. 2016).

Table 10 shows the results obtained by the qubit QML proposal and the classical DDQN algorithm in 30 different runs. Unlike in the previous problems, the subscripts in row 2 describe the number of times that each algorithm solved the problem in the 30 experiments performed and the standard deviation in the 30 experiments. We may see that the three algorithms were able to achieve the optimal solution in all executions and have equivalent performance, calculated as the average return of the model in 100 different environment tests among 30 executions. Despite this equivalence, we remark that the number of parameters of the qubit QML models is significantly lower than the number of parameters in the DDQN algorithm. According to the existing literature in quantum reinforcement learning, this is the first approach to solve the environment with one qubit (Skolik et al. 2022; Cuellar et al. 2024).



(a) qubit QML (b) K-Means

Fig. 20 Decision boundaries of each model in the Old Faithful clustering problem

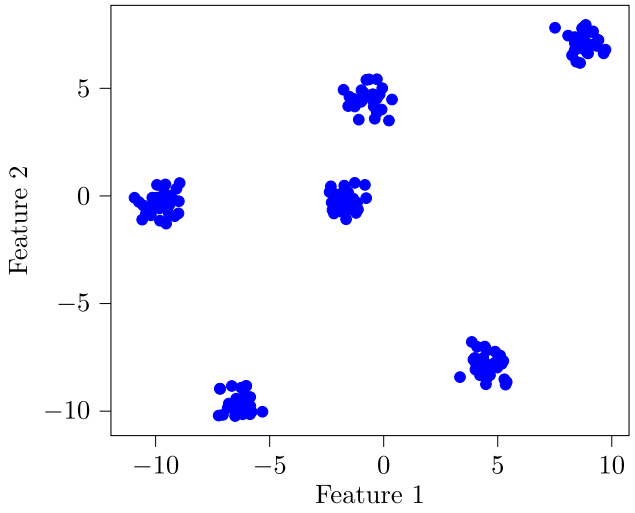


Fig. 21 Blobs dataset

3.10 Problem 9: the MountainCar environment

MountainCar is another reinforcement learning environment of the Farama Foundation's Gymnasium simulator and it is a control problem. It starts with a car stochastically located in a valley between two hills (Fig. 24). An agent is in charge of applying a constant force to the left or right to move the car, or to not apply a force at all (three discrete actions $\{a_1, a_2, a_3\}$). The goal is to reach the top of the right-hand hill using $x_t = (x_t^1, x_t^2)$ as observation at every time instant t. These perception values contain the car's location in the X axis (x_t^1) and its velocity (x_t^2) (two continuous values). A reward value of -1 is provided to the agent at each time step. An episode ends either if the agent succeeds or 200 time steps passed without reaching a goal state. The current literature considers that the environment is solved if the agent achieves an average return of -110 or higher in 100 consecutive environment tests.

The qubit QML structure used to solve this problem does not differ from the one used in the CartPole environment in Sect. 3.9 except for the encoding subcircuit that it is designed as $U_e(x_t) = R_z(x_t^2)R_y(x_t^1)$ in the current problem. Moreover, the measurement strategy and quantum-classical output mapping need to be adapted to the three possible actions that the qubit may provide as an outcome. In this case, we use the same strategy we applied in Sect. 3.3 to solve the multinomial *Iris flower* classification dataset using the measurement in the three axes with the observables $\sigma_z, \sigma_x, \sigma_y$ and the mapping described in Eq. 13. Thus, if $\hat{y}_t(\theta) = 0$, then the agent selects action a_1, it picks action a_2 if $\hat{y}_t(\theta) = 1$, and it chooses action a_3 if $\hat{y}_t(\theta) = 2$. The approach is compared with the DDQN algorithm as a baseline as it was performed in the previous section.

Table 11 describes the results obtained by each method in 30 different executions. The values displayed in row 2 correspond to the average mean return obtained in 100 consecutive episodes for 30 different executions and the standard deviation, and row 3 prints the mean return of the best solution found for each algorithm in 100 consecutive episodes. Subscripts in row 2 describe the number of executions of each algorithm that solved the problem. In this case, although both *QML (SP) and DDQN* algorithms were able to solve the problem, we can see a superiority in the classical DDQN since it solved the environment in every execution while the qubit *QML (SP)* method solved the problem 40% of times with the experimental settings used. If we compare the performance of *QML (SP)* and *QML (no SP)*, we may notice a substantial difference between both approaches. In this case, using the State Preparation layer was useful to change the way classical data are encoded into a quantum state. This fact also influenced the learning algorithm, which was able to avoid local optima to achieve near-optimal solutions in *QML (SP)*.

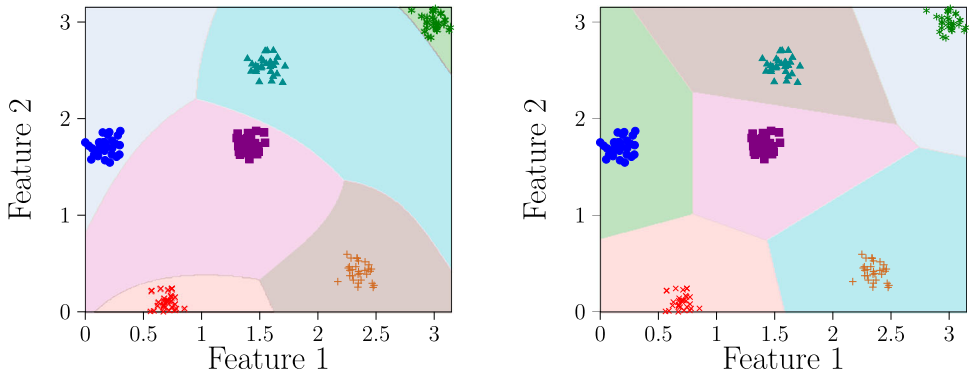
However, we may see a clear superiority of the qubit QML algorithm with respect to the number of parameters in row 4. As it happened with the CartPole environment, this approach is the first one to solve MountainCar with a single qubit (Cuellar et al. 2024).

3.11 Problem 10: the FrozenLake environment

The Frozen Lake is another simulated environment for reinforcement learning included in Farama's Gymnasium library. It assumes a 4×4 grid world where the agent starts at the top left corner and its goal is to reach the bottom-right corner without falling into holes located in the floor (Fig. 25). To do so, the agent can perceive x_t, an integer value from 0 to 15 describing its cell location at any given time instant t, and execute one of four actions $\{a_1, a_2, a_3, a_4\}$: to move up (a_1),

Table 9 Clustering results in the Blobs dataset, with the same format as Table 8

	QML (SP)	QML (no SP)	K-means
Avg. training S-score	0.93 ± 0.02	0.83 ± 0.01	0.98 ± 0.00
Avg. test S-score	$0.90 \pm 0.10_{(2)}$	$0.82 \pm 0.04_{(3)}$	$0.98 \pm 0.00_{(1)}$
Best	0.98/0.98	0.85/0.83	0.98/0.98
Parameters	6	3	12



(a) qubit QML (SP) (b) K-Means

Fig. 22 Decision boundaries of each model in the Blobs clustering problem

right (a_2), down (a_3), or left (a_4). The simulation ends either when the agent reaches the goal cell or it falls into a hole. At every time step t, the agent always receives a reward of 0 except when the goal state is found with reward +1. There are two versions of this environment: *slippery*, where the effect of an action is stochastic and simulates that the agent can slip when it moves, and *non-slippery*, where the target location is completely determined by the current location of the agent and the action selected. In this work, we solve both the non-slippery and slippery versions for completeness. The non-slippery FrozenLake is considered solved if the agent is able to obtain a reward of 1 in one environment simulation. As the environment observation and action spaces are discrete and the spaces are small, it is possible to calculate an optimal policy to solve the problem with dynamic programming in the stochastic (slippery) version using the value iteration algorithm. The average return of an agent following this optimal policy in this case converges to 0.74 as the number of episodes goes to infinity and, for that reason, we consider that the slippery version is solved if the agent receives an average return of 0.74 in 100 consecutive episodes.

This environment differs substantially from the others studied in Sects. 3.9 and 3.10 since the observation space is discrete and it contains a larger action space. The qubit QML structure used to solve it remains the same as $|\hat{y}_t(\theta)\rangle = U_a(\theta_1)U_e(x_t)U_s(\theta_0)$ with six parameters. However, a preprocessing is required to transform the discrete observation value into two features x_t^1, x_t^2 that are encoded in the quantum circuit as $U_e(x_t) = R_z(x_t^2)R_y(x_t^1)$. It is easy to decompose the agent's current cell observation x_t as a pair (row, column): the position's row can be calculated as $x_t^1 = \lfloor x_t/4 \rfloor$ and the column as $x_t^2 = x_t \% 4$, where % stands for the module operator. As a second preprocessing step, these two values x_t^1, x_t^2 are scaled to the range $[0, \pi]$.

On the other hand, measurement is carried out in the X and Y axes of the Bloch sphere using the observables σ_x, σ_y. As the number of possible agent actions is four, we follow a strategy similar to the one presented in Sect. 3.8 to design the quantum-classical output mapping. We remark that the agent has no semantic interpretation modules and it is not aware of the effect of each action in the environment. However, we can include this expert knowledge in a natural way in this problem as follows: We know that actions up/down and left/right have opposite effects so that if action *up* is matched with measurement of ket $|+\rangle$, then action *down* could be matched with the opposite ket $|-\rangle$. We apply the same design to action *right* matched with ket $|i\rangle$ and action *left* with ket $|-i\rangle$. This is not the usual setting of a pure classical reinforcement learning procedure where the agent has no prior knowledge about the environment or its actions, but we included this possibility

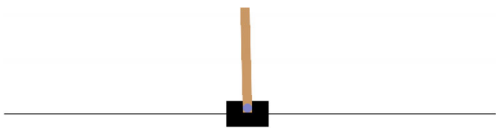


Fig. 23 Image of the CartPole simulator

Table 10 Results in the CartPole environment: row 1 plots the model names. Row 2 prints the average training return and the standard deviation in 30 experiments. Row 3 contains the average return in 100 test experiments of the best solution found, and row 4 the parameters required to train the models

	QML (SP)	QML (no SP)	DDQN
Avg. training return	$500 \pm 0_{(30)}$	$500 \pm 0_{(30)}$	$500 \pm 0_{(30)}$
Avg. return (best solution)	500.0	500.0	500.0
Parameters	6	3	10, 602

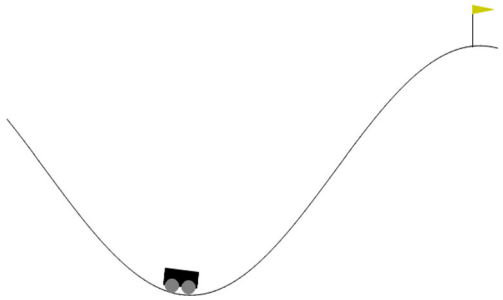


Fig. 24 Image of the MountainCar simulator

in this section for illustrative purposes. Then, the quantum-classical output mapping is formalized as it is described in Eq. 15.

$$\hat{y}_i(\theta) = \begin{cases} a_1 \;, & x = argmax_{k\in\{x,y\}} \left\{ |\langle \hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle| \right\} \wedge \langle \hat{y}_i(\theta)|\sigma_x|\hat{y}_i(\theta)\rangle \geq 0 \\ a_3 \;, & x = argmax_{k\in\{x,y\}} \left\{ |\langle \hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle| \right\} \wedge \langle \hat{y}_i(\theta)|\sigma_x|\hat{y}_i(\theta)\rangle < 0 \\ a_2 \;, & y = argmax_{k\in\{x,y\}} \left\{ |\langle \hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle| \right\} \wedge \langle \hat{y}_i(\theta)|\sigma_y|\hat{y}_i(\theta)\rangle \geq 0 \\ a_4 \;, & y = argmax_{k\in\{x,y\}} \left\{ |\langle \hat{y}_i(\theta)|\sigma_k|\hat{y}_i(\theta)\rangle| \right\} \wedge \langle \hat{y}_i(\theta)|\sigma_y|\hat{y}_i(\theta)\rangle < 0 \end{cases} \tag{15}$$

The aforementioned setting was used to train qubit *QML (SP)* and *QML (no SP)* structures, and a DDQN baseline method in both versions of the problem for 30 experiments. Table 12 describes the results obtained for the non-slippery FrozenLake, and Table 13 for the slippery version. Regarding the non-slippery FrozenLake, we can verify that the best solutions of both algorithms were able to solve the problem, although the qubit *QML (SP)* model was able to achieve the optimal behavior in 29 of 30 experiments and DDQN in 27 of them. A Wilcoxon test with 95% of confidence level suggested that there are no significant differences in the results of both methods and the result distributions could be considered as equivalent. On the other hand, if we observe the results of the slippery version in Table 13, we can see that both the qubit *QML (SP)* and DDQN had the same performance in average and were able to solve the problem. However, as it happened in all previous problems studied, the number of parameters in the qubit QML remains lower than the number of parameters of DDQN, which suggests that the model could be more suitable for its implementation in hardware with low resources.

If we compare the behavior of the state preparation layer in *QML (SP)* and its absence in *QML (no SP)*, we may see a common pattern in both slippery and non-slippery settings of the problem. In both cases, the *QML (no SP)* was unable to provide suitable results, which suggests that the state preparation layer was successful in the task of finding a preliminary change of basis that improves the data encoding. As it happened in several problems studied in this manuscript, this layer takes on special relevance to improve quantum data embedding in our problems and it could be considered as a good alternative to improve learning of quantum models.

To end up with this section, we would like to remark that this work is also the first one to solve FrozenLake with one single qubit in both the slippery and non-slippery cases (Cuellar et al. 2024).

3.12 Discussion

After the experimental study covering a full range of supervised, unsupervised, and reinforcement learning problems solved with one qubit, we can conclude that the qubit QML model proposed is an effective tool with practical applications in machine learning although limited to problems with low dimensionality that can be reduced to two features. In fact, the qubit model was able to achieve state-of-the-art results in 8 of 10 problems with a substantial improvement in model complexity with respect to classical methods. It is especially striking that the remaining two problems that were not solved with similar performance to classic machine learning belong to the family of continuous function approximation, while the majority of the other eight must take decisions from a discrete set. However, the analysis of results of these two datasets (problems 3 and 4) suggests that the

Table 11 Results in the MountainCar environment with the same format as Table 10

	QML (SP)	QML (no SP)	DDQN
Avg. training return	$-122.67 \pm 19.45_{(12)}$	$-199.16 \pm 4.17_{(0)}$	$-106.28 \pm 1.85_{(30)}$
Avg. return (best solution)	-107.66	-176.72	-102.77
Parameters	6	3	10, 703

Fig. 25 Image of the FrozenLake simulator

qubit model was able to learn the internal dependencies between model inputs and outputs, although with less precision than a classical MLP. As a matter of fact, we believe that our experimental study can be considered an important step forward in the pave to achieve resource optimization in quantum machine learning, but also as a relevant contribution to solve problems that were addressed previously with QML but using a larger number of qubits, such as the *Iris flower* dataset (Piatrenka and Rusek 2022) and all the reinforcement learning environments addressed (Cuellar et al. 2024; Skolik et al. 2022; Chen et al. 2019).

Our empirical study has provided us with further lessons learned beyond the experimental evidence: First, the design of a suitable measurement methodology is crucial not only to gather the system outcomes, but also to increase the number of possible model outputs. In our experience, this task must be carried out considering how the evolution of a quantum state with input data $|x_i\rangle$ provides the quantum state encoding the output $|\hat{y}_i(\theta)\rangle$. This task gets special relevance if we notice that both the inputs and outputs might share the same representation space in the Bloch sphere. A suitable geometric study about the problem of linear or non-linear separability could help in this design.

Another aspect of interest regards preprocessing. It is widely known that preprocessing plays a very important role in classical machine learning, and the same happens in QML. Using a single qubit as a computational model, preprocessing involves compacting all the relevant data of the problem into two features that must evolve to a linearly separable decision in the quantum state space depending of the measurement strategy. Thus, feature selection and feature extraction procedures are key tools to success.

We end this section with a discussion about possible practical applications of the approach in industry. It is widely known that the Internet of Things (IoT) plays an important role in our daily lives with applications ranging from home automation to wearable devices, home or industry appliances, etc. The current trend is that IoT devices increase both in use and performance services including the integration with artificial intelligence and machine learning models. However, the main limitation of these devices regards to their size and energy requirements, and therefore the power of the microcontrollers used in their manufacturing. There are other limiting aspects that can be considered, as country regulations that require industrial certifications of how an artificial intelligence model works and provides its outputs, for example. We believe that the proposed qubit QML approach studied in this manuscript could have a place in this context, as the simulation of the evolution of a qubit's quantum state is computationally efficient, specially considering the small size of the proposed model. Moreover, we believe that its behavior can be explained in terms of industry certifications specially if we compare it against other powerful models such as neural networks, with the added benefit of the possibility to provide non-linear behavior.

4 Conclusions

In this work, we have studied experimentally the benefits and limitations of using a single qubit to solve classical machine learning problems covering the full spectrum of supervised, unsupervised, and reinforcement learning. We have proposed a variational quantum circuit including three stages, state preparation, data encoding, and ansatz, and analyzed the results both in terms of the geometrical interpretation of the qubit response and performance. We have put in a manifest that a single qubit is a powerful tool to have into account when solving simple problems whose dimensionality can be reduced to two features, although there are strong limitations concerning scalability to complex problems containing tens

Table 12 Results in the non-slippery FrozenLake environment with the same format as Table 10

	QML (SP)	QML (no SP)	DDQN
Avg. training return	$0.97 \pm 0.18_{(29)}$	$0.00 \pm 0.00_{(0)}$	$0.90 \pm 0.30_{(27)}$
Avg. return (best solution)	1.0	0.0	1.0
Parameters	6	3	10, 804

Table 13 Results in the slippery FrozenLake environment with the same format as Table 10

	QML (SP)	QML (no SP)	DDQN
Avg. training return	$0.75 \pm 0.02_{(29)}$	$0.33 \pm 0.03_{(0)}$	$0.74 \pm 0.14_{(29)}$
Avg. return (best solution)	0.81	0.4	0.82
Parameters	6	3	10, 804

or hundreds of features. In spite of this disadvantage, the qubit model was able to achieve state-of-the-art performance using fewer parameters with respect to classical machine learning methods in the problems studied, and this fact suggests that it could be a good choice to be considered for implementation in hardware with low resources such as embedded systems, microcontrollers or edge devices that need to perform decision tasks with machine learning tools, since simulating the evolution of a single qubit quantum state is computationally efficient in our proposal.

Acknowledgements This work was performed in cooperation with the IEEE CS Task Force in Quantum Computational Intelligence (https://cmte.ieee.org/quantum-computational-intelligence/).

Author Contribution The author declares that he contributed to the design, implementation, analysis, or results of the research and writing of the article.

Funding This article was funded by the project QUANERGY (Ref. TED2021-129360B-I00), Ecological and Digital Transition R&D projects call 2022 by MCIN/AEI/10.13039/501100011033 and European Union NextGeneration EU/PRTR, and supported by Grant PID2021-128970OA-I00 funded by MCIN/AEI/10.13039/501100011033/FEDER.

Data Availability The datasets used in this manuscript can be accessed for free online:
- Problem 1: UCI Machine Learning Repository at https://archive.ics.uci.edu/dataset/267/banknote+authentication.
- Problem 2: UCI Machine Learning Repository at https://archive.ics.uci.edu/dataset/53/iris.
- Problem 3: UCI Machine Learning Repository at https://archive.ics.uci.edu/dataset/294/combined+cycle+power+plant.
- Problem 4: Kaggle online platform at https://www.kaggle.com/datasets/chirag19/air-passengers.
- Problem 5: Kaggle online platform at https://www.kaggle.com/datasets/nirmalsankalana/oikolab-weather-dataset.
- Problem 6: Kaggle online platform at https://www.kaggle.com/datasets/janithwanni/old-faithful.
- Problem 7: Python's scikit learn library at https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html.
- Problems 8, 9, 10: Farama's foundation Gymnasium library at https://gymnasium.farama.org/index.html.

Code Availability The source code used in this manuscript can be accessed for free at the GitHub repository https://github.com/manupc/OneQubit.

Declarations

Conflict of Interest The author declares no competing interests.

References

Aïmeur E, Brassard G, Gambs S (2007) Quantum clustering algorithms. In: Proceedings of the 24th international conference on machine learning. ICML '07, pp 1–8. Association for Computing Machinery, New York, USA. https://doi.org/10.1145/1273496.1273497

Andres E, Cuellar MP, Navarro G (2023) Efficient dimensionality reduction strategies for quantum reinforcement learning. IEEE Access 11:104534–104553. https://doi.org/10.1109/ACCESS.2023.3318173

Andres E, Cuellar MP, Navarro G (2022) On the use of quantum reinforcement learning in energy-efficiency scenarios. Energies 15(16). https://doi.org/10.3390/en15166034

Chen SY-C, Yang C-HH, Qi J, Chen P-Y, Ma X, Goan H-S (2019) Variational quantum circuits for deep reinforcement learning. IEEE Access 8:141007–141024

Cherrat EA, Kerenidis I, Prakash A (2023) Quantum reinforcement learning via policy iteration. Quantum Mach Intell 5(2):30. https://doi.org/10.1007/s42484-023-00116-1

Cuellar MP, Pegalajar MC, Cano C (2024) Automatic evolutionary design of quantum rule-based systems and applications to quantum reinforcement learning. Quantum Inf Process 23(5):179. https://doi.org/10.1007/s11128-024-04391-0

Easom-McCaldin P, Bouridane A, Belatreche A, Jiang R, Al-Maadeed S (2024) Efficient quantum image classification using single qubit encoding. IEEE Trans Neural Netw Learn Syst 35(2):1472–1486. https://doi.org/10.1109/TNNLS.2022.3179354

Ganguly S (2021) Quantum machine learning: an applied approach. Apress, New York

Gentinetta G, Thomsen A, Sutter D, Woerner S (2024) The complexity of quantum support vector machines. Quantum 8:1225. https://doi.org/10.22331/q-2024-01-11-1225

Goswami K, Veereshi GA, Schmelcher P, Mukherjee R (2024) Solving the travelling salesman problem using a single qubit. arXiv:2407.17207

Hansen N, Arnold DV, Auger A (2015) In: Kacprzyk J, Pedrycz W (eds.) Evolution strategies, pp 871–898. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-43505-2_44

Hasselt Hv, Guez A, Silver D (2016) Deep reinforcement learning with double Q-learning. In: Proceedings of the thirtieth AAAI conference on artificial intelligence. AAAI'16, pp 2094–2100. AAAI Press, Phoenix, Arizona

Heese R, Bickert P, Niederle AE (2022) Representation of binary classification trees with binary features by quantum circuits. Quantum 6:676. https://doi.org/10.22331/q-2022-03-30-676

Karimi M, Javadi-Abhari A, Simon C, Ghobadi R (2023) The power of one clean qubit in supervised machine learning. Scientific Reports 13(1):19975. https://doi.org/10.1038/s41598-023-46497-y

Maheshwari D, Sierra-Sosa D, Garcia-Zapirain B (2022) Variational quantum classifier for binary classification: real vs synthetic dataset. IEEE Access 10:3705–3715. https://doi.org/10.1109/ACCESS.2021.3139323

Moret-Bonillo V (2018) Emerging technologies in artificial intelligence: quantum rulebased systems. Progress Artif Intell 7. https://doi.org/10.1007/s13748-017-0140-6

Pérez-Salinas A, López-Núñez D, García-Sáez A, Forn-Díaz P, Latorre JI (2021) One qubit as a universal approximant. Phys Rev A 104:012405. https://doi.org/10.1103/PhysRevA.104.012405

Pérez-Salinas A, Cervera-Lierta A, Gil-Fuster E, Latorre JI (2020) Data reuploading for a universal quantum classifier. Quantum 4:226. https://doi.org/10.22331/q-2020-02-06-226

Piatrenka I, Rusek M (2022) Quantum variational multi-class classifier for the iris data set. In: Groen D, Mulatier C, Paszynski M, Krzhizhanovskaya VV, Dongarra JJ, Sloot PMA (eds) Computational Science – ICCS 2022. Springer, Cham, pp 247–260

Poggiali A, Berti A, Bernasconi A, Del Corso GM, Guidotti R (2024) Quantum clustering with k-means: a hybrid approach. Theoretical Comput Sci 992. https://doi.org/10.1016/j.tcs.2024.114466

Rajesh V, Naik UP (2021) Quantum convolutional neural networks (QCNN) using deep learning for computer vision applications. In: 2021 International conference on recent trends on electronics, information, communication & technology (RTEICT), pp 728–734. https://doi.org/10.1109/RTEICT52294.2021.9574030

Schuld M, Sinayskiy I, Petruccione F (2014) Quantum computing for pattern classification. In: Pham D-N, Park S-B (eds) PRICAI 2014: trends in artificial intelligence. Springer, Cham, pp 208–220

Skolik A, Jerbi S, Dunjko V (2022) Quantum agents in the Gym: a variational quantum algorithm for deep Q-learning. Quantum 6:720. https://doi.org/10.22331/q-2022-05-24-720

Tapia EP, Scarpa G, Pozas-Kerstjens A (2023) A didactic approach to quantum machine learning with a single qubit. Phys Scripta 98(5):054001. https://doi.org/10.1088/1402-4896/acc5b8

Wang G (2017) Quantum algorithm for linear regression. Phys Rev A 96:012335. https://doi.org/10.1103/PhysRevA.96.012335

Wu J, Fu H, Zhu M, Zhang H, Xie W, Li X-Y (2024) Quantum circuit autoencoder. Phys Rev A 109:032623. https://doi.org/10.1103/PhysRevA.109.032623

Yarkoni S, Kleshchonok A, Dzerin Y, Neukart F, Hilbert M (2021) Semi-supervised time series classification method for quantum computing. Quantum Mach Intell 3(12):1–11. https://doi.org/10.1007/s42484-021-00042-0

Yu Z, Yao H, Li M, Wang X (2022) Power and limitations of single-qubit native quantum neural networks. In: Oh AH, Agarwal A, Belgrave D, Cho K (eds.) Advances in neural information processing systems (2022). https://openreview.net/forum?id=XNjCGDr8N-W

Zoufal C, Lucchi A, Woerner S (2019) Quantum generative adversarial networks for learning and loading random distributions. npj Quantum Inf 5(1), 103. https://doi.org/10.1038/s41534-019-0223-2