Kernel-based quantum regressor models learning non-Markovianity

Diego Tancara,[1] Hossein T. Dinani,[2] Ariel Norambuena [ORCID],[3] Felipe F. Fanchini [ORCID],[4] and Raúl Coto [ORCID][5,6,*]

[1]*Centro de Óptica e Información Cuántica, Universidad Mayor, Vicerrectoría de Investigación, Santiago, Chile*
[2]*Escuela Data Science, Facultad de Ciencias, Ingenería y Tecnología, Universidad Mayor, Santiago, Chile*
[3]*Universidad Mayor, Vicerrectoría de Investigación, Santiago, Chile*
[4]*Faculdade de Ciências, UNESP - Universidade Estadual Paulista, Bauru, SP, 17033-360, Brazil*
[5]*Department of Physics, Florida International University, Miami, Florida 33199, USA*
[6]*Universidad Bernardo O Higgins, Santiago de Chile, Chile*

Quantum machine learning is a growing research field that aims to perform machine learning tasks assisted by a quantum computer. Kernel-based quantum machine learning models are paradigmatic examples where the kernel involves quantum states, and the Gram matrix is calculated from the overlap between these states. With the kernel at hand, a regular machine learning model is used for the learning process. In this paper we investigate the quantum support vector machine and quantum kernel ridge models to predict the degree of non-Markovianity of a quantum system. We perform digital quantum simulation of amplitude damping and phase damping channels to create our quantum data set. We elaborate on different kernel functions to map the data and kernel circuits to compute the overlap between quantum states. We show that our models deliver accurate predictions that are comparable with the fully classical models.

I. INTRODUCTION

During the last decades we have witnessed the rapidly growing fields of artificial intelligence (AI) and quantum computing (QC). The basis for AI and QC were developed in the past century. However, now this knowledge is widely available for research, business, and health, among others. AI aims to provide machines with human-like intelligence. From the very beginning, AI has been conceived in different ways, leading to the development of different branches, known as machine learning (ML) [1–3], deep learning [4], and reinforcement learning [5]. ML is based on statistical learning, where the machine learns from data that has already been labeled (supervised learning) or from unlabeled data (unsupervised learning). In recent years, supervised learning has undoubtedly impacted physics [2,3,6]. In particular, it is known for unraveling patterns from data sets that yield quantum phase transitions [7,8].

Quantum computing is also at the forefront of current technologies. Research groups have delivered highly functional and fault-tolerant quantum algorithms encompassing a wide variety of systems, including superconducting qubits [9,10], trapped ions [11], cold atoms [12], photonics [13,14], and color centers in diamond [15]. In the last years, quantum computers have pushed further the boundaries of physics, chemistry, biology, and computing itself, with groundbreaking achievements in the simulation of novel materials [16] and molecules [9,13,17,18] and in designing algorithms towards quantum supremacy [10,19] and quantum machine learning [20–40].

Among the main obstacles to be overcome in the development of quantum technologies is the interaction of the quantum system with the environment. This interaction disturbs the quantum state and, in general, can be divided into two types of processes: Markovian and non-Markovian [41]. Non-Markovian processes are those in which memory effects are taken into account, and their importance can be noted in several processes and protocols such as state teleportation [42], quantum metrology [43], and even in current quantum computers [44]. In this paper we use quantum machine learning to determine the degree of non-Markovianity of a two-time quantum process [45], i.e., a quantum channel. We focus on kernel-based machine learning models to learn from quantum states. Our results show that the quantum computer can create the data set, but also treat and learn from it, providing feedback on the very process in which it is involved.

The paper is organized as follows. In Sec. II, we introduce two quantum machine learning models based on kernels, namely, quantum support vector machine and quantum kernel ridge models. The goal of these models is to estimate the degree of non-Markovianity from a data set made of quantum states. Furthermore, we elaborate on the performance of the models based on three different kernel functions and four different kernel circuits to measure the overlap between two quantum states. All these possible combinations yield different Gram matrices. In Sec. III, we introduce the digital quantum simulation approach that we followed to describe the evolution of the system in amplitude damping and phase damping channels. In Sec. IV, we show our main results regarding the prediction of the degree of non-Markovianity. In Sec. V, we deliver the final remarks of this work.

II. KERNEL-BASED MACHINE LEARNING MODELS

Quantum machine learning aims to perform machine learning tasks assisted by a quantum computer. In

---

*raul.coto@protonmail.com

recent years, different implementations have been addressed, including variational quantum circuits [46–48], quantum nearest-neighbor methods [21], and quantum kernel methods [20,23,35]. The latter naturally appears in models that support a kernel function to represent the data into a feature space. Two well-understood examples are the support vector machine (SVM) and the kernel ridge regressor (KRR) models. Their extension to the quantum domain via a precomputed kernel is straightforward. Then, one could think of the kernel as a function (that in our case results from a quantum circuit) that we pass to the ML model to improve the learning process. Next, we describe the SVM and KRR models and their connection with the kernel.

A. Support vector machine

One of the most broadly used models in ML is support vector machines (SVM) [49]. This model can be used for classification [50,51] and regression [49,52,53] tasks. The former gives rise to an intuitive representation that relies on a hyperplane that splits the data set into different classes. Therefore, predicting the label of unknown data only depends on where the data samples fall regarding the hyperplane. In general, other models also use a hyperplane. However, the SVM sets the maximum margin, i.e., maximizing the distance between the hyperplane and some of the boundary training data, which are the data samples close to the edge of the class. These particular samples are known as support vectors (SVs). Since SVs are a subset of the training data set, this model is suitable for situations where the number of training data samples is small compared to the feature vector's dimension. Once the model has fitted the training data set, it can be used as a decision function that predicts new samples, without holding the training data set (eager learning algorithm) in memory. In this work we will focus on a regression task, which predicts a real number rather than a class. In what follows, we briefly describe the mathematical formulation of the optimization problem. More details can be found in Ref. [54].

SVM delivers the tools for finding a function $f(\vec{x})$ that fits the training data set $\{\vec{x}_i, y_i\}$, where $\vec{x}_i \in \mathbb{R}^d$ are the feature vectors with dimension d, and $y_i \in \mathbb{R}$ are the corresponding labels. Note that i runs over the number of training samples ($i = 1, 2, \ldots, l$). We begin with the linear function $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$, with $\vec{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ being fitting parameters. We shall discuss the case of nonlinear separable data later on. For ϵ-SVM [49], deviations of $f(\vec{x})$ from the labeled data (y_i) must be smaller than ϵ, i.e., $|f(\vec{x}) - y_i| \leqslant \epsilon$. Moreover, we must address the model complexity as given by the l_2-norm $\|\vec{w}\|^2$, and the tolerance for deviations ξ_i, ξ_i^* (slack variables) larger than ϵ, that are weighted by $C > 0$. Therefore, the optimization problem can be stated as [1,49,53]

$$
\begin{aligned}
\text{minimize} \quad & \tfrac{1}{2}\|\vec{w}\|^2 + C\sum_i (\xi_i + \xi_i^*) \\
\text{subjected to} \quad & \begin{cases} y_i - \vec{w} \cdot \vec{x}_i - b \leqslant \epsilon + \xi_i \\ \vec{w} \cdot \vec{x}_i + b - y_i \leqslant \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geqslant 0 \end{cases}
\end{aligned} \quad (1)
$$

One can solve this problem introducing the Lagrange multipliers $\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geqslant 0$, with the Lagrangian defined as

[49,52,53]

$$
\begin{aligned}
L = {} & \frac{1}{2}\|\vec{w}\|^2 + C\sum_i (\xi_i + \xi_i^*) - \sum_i (\eta_i \xi_i + \eta_i^* \xi_i^*) \\
& - \sum_i \alpha_i(\epsilon + \xi_i - y_i + \vec{w} \cdot \vec{x}_i + b) \\
& - \sum_i \alpha_i^*(\epsilon + \xi_i^* + y_i - \vec{w} \cdot \vec{x}_i - b).
\end{aligned} \quad (2)
$$

From the vanishing partial derivatives $\partial_b L$, $\partial_w L$, $\partial_\xi L$, and $\partial_{\xi^*} L$ the optimization problem can be recast as

$$
\begin{aligned}
\text{maximize} \quad & \begin{cases} -\frac{1}{2}\sum_{i,j}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j \rangle \\ -\epsilon \sum_i (\alpha_i + \alpha_i^*) + \sum_i y_i(\alpha_i - \alpha_i^*) \end{cases} \\
\text{subjected to} \quad & \begin{cases} \sum_i (\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, C] \end{cases}.
\end{aligned} \quad (3)
$$

For convenience, we have written the dot product as an inner product, $\langle x_i, x_j \rangle = \vec{x}_i \cdot \vec{x}_j$. From $\partial_w L = 0$ we find $\vec{w} = \sum_i (\alpha_i - \alpha_i^*)\vec{x}_i$, that leads to the decision function

$$
f(\vec{x}) = \sum_i (\alpha_i - \alpha_i^*)\langle x_i, x \rangle + b, \quad (4)
$$

that depends on the inner product between the unlabeled data ($\vec{x}$) and the training data ($\vec{x}_i$). We can recover b from the Karush-Kuhn-Tucker (KKT) condition, which states that at the solution point of the Lagrangian, the product between the Lagrange multipliers and the conditions vanishes. We remark that this calculation is computed internally in the scikit-learn library [1]. We would like to stress that the decision function in Eq. (4) has a sparse representation in terms of α_i, α_i^*. Only a small subset of the training data set (support vectors) contributes to the decision function. In Appendix A, we show the arguments for the sparsity and the calculation of b.

We have introduced so far a linear decision function that can handle linearly separated data. For nonlinearly separated data, it is possible to define a clever kernel function $k(x_i, x)$ that generalizes $\langle x_i, x \rangle$ by taking the samples to a higher dimensional space, where they are linearly separable. We elaborate further on this idea later on.

B. Kernel ridge regressor

KRR is another important nonlinear machine learning model. It has been successfully used to predict the evolution of quantum systems [55]. It combines ridge regression with the kernel trick [1,56]. The former, provides a linear solution based on least squares with l_2 regularization that penalizes large coefficients. Like in SVM, the l_2 norm prevents model complexity, while the kernel allows the model to learn a nonlinear function in the original space. This model offers a straightforward optimization problem stated by [1]

$$
\text{minimize} \quad \sum_{i=1}^{N} \|\vec{w} \cdot \vec{x}_i - y_i\|^2 + \alpha\|\vec{w}\|. \quad (5)
$$

The above problem can be written in an equivalent way as [56]

$$
\begin{aligned}
&\text{minimize} && \sum_{i=1}^{N} (y_i - \vec{w} \cdot \vec{x}_i - b)^2, \\
&\text{subjected to} && \|\vec{w}\|^2 \leqslant \alpha_d,
\end{aligned}
\tag{6}
$$

where there is a one-to-one correspondence between the hyperparameters α and α_d. Introducing the Lagrange multipliers as in the previous subsection, the decision function can be found as

$$
f(\vec{x}) = \sum_i \beta_i k(x_i, x) + b,
\tag{7}
$$

with β_i and b being fitting parameters.

It is worth noting that SVM and KRR are similar in terms of the l_2 regularization and that both use the kernel trick, but the loss function is different. While SVM relies on a linear ϵ-insensitive loss, KRR uses squared error loss. The former implies that all the training points that result in errors that fall inside the ϵ tube do not contribute in the solution, which originates sparseness. In contrast, KRR considers all the training points. This yields differences in the performance of these models.

Machine learning algorithms have greatly profited from kernel functions [6,28,35]. Therefore, we now introduce a generalization of the decision function to learn from nonlinear data. The kernel can be understood as a measure of similarities between two vectors, and it supports representations ranging from polynomial to exponential functions [1]. Along this paper we consider three different functions for the kernel $k(x_i, x_j)$, namely, linear $\langle x_i, x_j \rangle + c$, polynomial $(\langle x_i, x_j \rangle + c)^d$, and exponential $\exp(-\sigma \sqrt{1 - \langle x_i, x_j \rangle})$.

We have so far addressed the classical part (optimization problem) of this hybrid quantum machine learning approach. In the next subsection we will focus on implementing the kernel through a quantum circuit.

C. Quantum kernels

We have noted that the kernel provides efficient separability in nonlinear regions. The main idea behind the kernel is that it allows to map the data to a higher-dimensional space, termed as a "featured space" [54]. In general lines, let us consider a feature map $\phi : x \in \chi \rightarrow \phi(x) \in H$ that encodes information from a certain domain χ (commonly $\chi \in R^n$) to a feature space H. The advantages of using the map rely on the "kernel trick" [6], which allows us to set the decision function without the explicit calculation of $\phi(x)$. This idea has encouraged researchers to bridge classical and quantum machine learning [25,26,35]. Let us consider a Hilbert space H that contains the states of a quantum system. Now, instead of encoding the information of χ in a feature space given by functions $\phi(x)$, with $x \in \chi$, the information is encoded in quantum states $|\phi(x)\rangle \in H$ [35,57,58], which is known as quantum embedding. Quantum embedding is a crucial step in the process and, in some cases, may lead to a disadvantage against classical models—for instance, because of the noise in the embedding ansatz. To overcome this, we resorted to performing digital quantum simulation of the quantum

dynamics rather than classical simulation [54], which allows us to handle quantum states to build up the kernel. We use the kernel to calculate a symmetric and semipositive definite matrix (Gram matrix) over all possible combinations in the training data set and we use it to train our model.

A natural choice to obtain the kernel from the training samples ρ_i is the pairwise trace distance between the quantum states ($\text{Tr}[\rho_i \rho_j]$), which is commonly carried by the swap test [59,60]. In what follows we describe the circuit implementation. First, we encode the information into two different qubits. Each of these qubits undergoes a NM (non-Markovian) evolution, induced by independent ancilla qubits. Then, the overlap between states ρ_i and ρ_j yields the matrix element $k(\theta_i, \theta_j) = \text{Tr}[\rho_i \rho_j]$, where θ_i is the parameter that control the NM evolution. We note that for the case of pure states, $\rho_i = |\psi_i\rangle\langle\psi_i|$ and $\rho_j = |\psi_j\rangle\langle\psi_j|$, the kernel simply reduces to $|\langle\psi_i|\psi_j\rangle|^2$.

We describe next different implementations for the overlapping.

1. Swap test

The swap test is a high-level sequence of quantum operations that involves two system qubits, an ancilla qubit, two-qubit gates (CNOT), one-qubit gates, and a final measurement on the ancilla [59], see Fig. 1. By measuring the probability of finding the ancilla in state $|0\rangle$ (P_0), one obtains the state overlapping by computing $\text{Tr}[\rho_i \rho_j] = 2P_0 - 1$.

2. Inversion test

Our second kernel considers the quantum state of a closed system (unitary evolution) that encompasses the system qubit and the environment ancilla qubit [61]. It begins with two quantum states driven by unitary evolution $U(\theta)$, such that $|\Psi_\theta\rangle = U(\theta)|00\rangle$, with $|00\rangle = |0\rangle_s \otimes |0\rangle_a$. The kernel is defined as the probability to reach the initial state after two subsequent evolutions, assuming that the inverse evolution $U^\dagger(\theta)$ can be implemented. The matrix elements read

$$
\begin{aligned}
k(\theta_i, \theta_j) &= |\langle\Psi_{\theta_i}|\Psi_{\theta_j}\rangle|^2 = |\langle 00|U^\dagger(\theta_i)U(\theta_j)|00\rangle|^2 \\
&= |\langle 00|\Theta\rangle|^2,
\end{aligned}
\tag{8}
$$

where $|\Theta\rangle = U^\dagger(\theta_i)U(\theta_j)|00\rangle$. In contrast to the swap test kernel, this one requires two measurements, which allows us to decrease the number of quantum registers (Fig. 1). We remark that this kernel is not experimentally feasible for the particular goal of detecting non-Markovianity. In general, one has no access to perform measurements upon the environment. In addition, it requires reverse unitary interactions of the system-environment dynamics. Nevertheless, we consider it because it may be applied to other machine learning tasks [61] and it delivers the best accuracy we found in this paper.

3. Ancilla-based algorithm

The Ancilla-based algorithm (ABA) is a variation of the swap test that conveniently reduces the number of gates. It was first discovered in the context of quantum optics [62], and rediscovered later with the assistance of a neural network and introduced for quantum circuits [60]. The circuit is depicted in Fig. 1, and it is composed by two-qubit CNOT
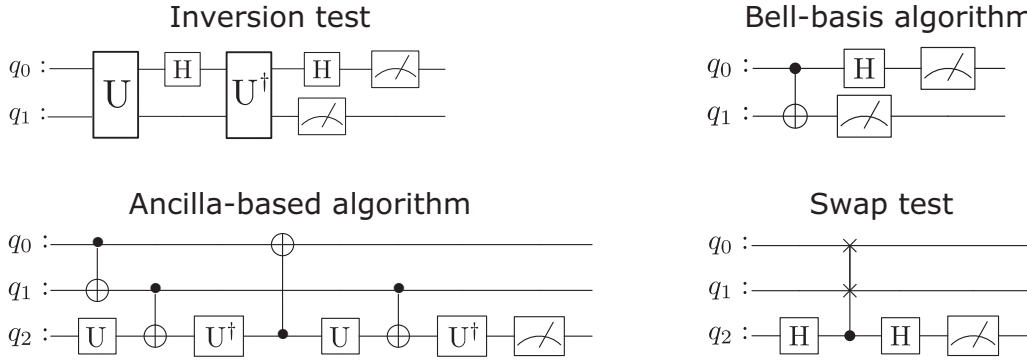
FIG. 1. Quantum circuits compute the overlap between two quantum states in the kernel function to calculate the Gram matrix. For the inversion test U represents either the amplitude damping or phase damping channel depicted in Fig. 2. For the ancilla-based algorithm (ABA) $U = T^\dagger H$ [60].

gates and single-qubit $U = T^\dagger H$ gates. T introduces a $\pi/4$ rotation around the z axis, and H is the Hadamard gate. Similar to the swap test, the final outcome is computed as $2P_0 - 1$.

4. Bell-basis algorithm

The Bell-basis algorithm (BBA) considers less resources than the previous one (ABA), but demands Bell-basis measurements on all the system qubits [60]. The circuit is depicted in Fig. 1, and it only requires a CNOT gate and a Hadamard gate. However, the postprocessing is more involved and demands computing all outcomes as $P_{00} + P_{01} + P_{10} - P_{11}$.

In this paper we do not intend to explicitly compare the accuracy of all these approaches for estimating the overlapping (for a comparison between swap test, ABA, and BBA, see Ref. [60]). We will compare them in terms of the accuracy of the decision function.

In the next section we describe the quantum circuits that account for the interaction between the system qubit with the environment ancilla qubit that ultimately yields non-Markovianity.

III. DIGITAL QUANTUM SIMULATION OF NON-MARKOVIAN CHANNELS

The main purpose of this paper is to determine the degree of non-Markovianity of a quantum process using a quantum machine learning algorithm. We begin with simulating two non-Markovian channels, amplitude damping and phase damping, whose degree of non-Markovianity can be controlled. For this purpose we simulate the processes using usual circuit routines, taking auxiliary qubits to represent the environment. In this section, we show how the degree of non-Markovianity is calculated and present how the non-Markovian amplitude damping and phase damping processes can be simulated using a quantum circuit.

A. Calculating the degree of non-Markovianity

There are different ways to measure the degree of non-Markovianity. The most popular measures are based on the trace distance dynamics [63], the dynamics of entanglement [64,65], and mutual information [66], among others [45]. In this paper we consider the measure based on entanglement

dynamics of a bipartite quantum state that encompasses the system that interacts with the environment and an ancilla qubit that is isolated from it [65]. It is worthwhile to notice that this ancilla only serves the purpose of quantifying non-Markovianity and it is not implemented in the quantum circuits, in contrast to the ancilla used to simulate the effect of the environment for the amplitude damping and phase damping processes.

A monotonic decrease in the entanglement of the bipartite system implies that the dynamics is Markovian. An increase in the entanglement during the evolution is a result of memory effects and thus non-Markovian. The measure can be calculated as

$$\mathcal{N} = \max \int_{dE(t)/dt > 0} \frac{dE(t)}{dt} dt, \tag{9}$$

where the maximization is done over all possible initial states and E is the measure of entanglement. It has been found that the maximization is achieved for Bell states [67]. Therefore, we consider a bipartite system in a Bell state and use the concurrence as the measure of entanglement [68].

B. Amplitude damping

For the amplitude damping (AD) channel, we consider a qubit interacting with a bath of harmonic oscillators, given by the Hamiltonian ($\hbar = 1$) [69,70]

$$H = \omega_0 \sigma_+ \sigma_- + \sum_k \omega_k a_k^\dagger a_k + \sum_k (g_k^* \sigma_+ a_k + g_k \sigma_- a_k^\dagger). \tag{10}$$

Here, $\sigma_+ = \sigma_-^\dagger = |1\rangle\langle 0|$ with $|1\rangle$ ($|0\rangle$) corresponding to the excited (ground) state of the qubit with transition frequency ω_0, $a_k(a_k^\dagger)$ is the annihilation (creation) operator of the kth mode of the bath with frequency ω_k, and g_k is the coupling between the qubit and the kth mode. We assume that the bath has a Lorentzian spectral density

$$J(\omega) = \frac{1}{2\pi} \frac{\gamma_0 \lambda^2}{(\omega_0 - \omega)^2 + \lambda^2}, \tag{11}$$

where $\lambda \approx 1/\tau_r$, with τ_r being the environment correlation time, $\gamma_0 \approx 1/\tau_s$, where τ_s is the typical time scale of the system.

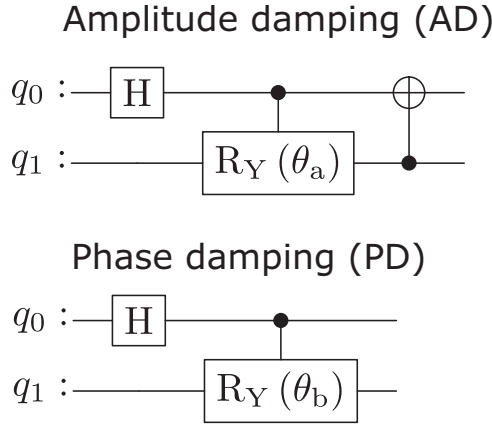## Amplitude damping (AD)

Phase damping (PD)

FIG. 2. Quantum circuits for simulating AD and PD channels.

The dynamics of the qubit that is coupled resonantly with the environment can be expressed as

$$\rho(t) = \sum_{i=0}^{1} M_i(t)\rho(0)M_i^{\dagger}(t), \tag{12}$$

where the Kraus operators are given by [71,72]

$$M_0(t) = |0\rangle\langle 0| + \sqrt{p(t)}|1\rangle\langle 1|, \tag{13}$$

$$M_1(t) = \sqrt{1 - p(t)}|0\rangle\langle 1|, \tag{14}$$

in which

$$p(t) = e^{-\lambda t}\left[\frac{\lambda}{d}\sinh(dt/2) + \cosh(dt/2)\right]^2, \tag{15}$$

with $d = \sqrt{\lambda^2 - 2\gamma_0\lambda}$. The dynamics is known to be non-Markovian in the strong coupling regime $\lambda < 2\gamma_0$ ($\tau_s < 2\tau_r$) [73]. We have scaled our AD simulations with $\gamma_0 = 1$, and considered λ in the range [0.1, 3].

The AD process can be simulated for a general scenario with a quantum circuit via an ancilla qubit [71,72]. After tracing out the ancilla qubit we obtain the desired mixed state. Figure 2 shows the quantum circuit. The Hadamard gate prepares the qubit in the superposition state $(|0\rangle + |1\rangle)/\sqrt{2}$, while the controlled rotation and CNOT gates simulate the interaction of the qubit with the environment. In this circuit, the angle θ_a is given by [71,72]

$$\theta_a = 2\arccos(\sqrt{p(t)}), \tag{16}$$

where $p(t)$ is given in Eq. (15).

C. Phase damping

For the phase damping (PD) channel, following Ref. [74], we consider a qubit undergoing decoherence induced by a colored noise given by the time-dependent Hamiltonian ($\hbar = 1$)

$$H(t) = \Gamma(t)\sigma_z. \tag{17}$$

Here, $\Gamma(t)$ is a random variable which obeys the statistics of a random telegraph signal defined as $\Gamma(t) = \alpha(-1)^{n(t)}$, where α is the coupling between the qubit and the external

influences, $n(t)$ is a random variable with Poisson distribution with mean $t/(2\tau)$, and σ_z is the Pauli z operator. In this case, the dynamics of the qubit is given by the following Kraus operators [74]:

$$M_0(t) = \sqrt{\frac{1 + \Lambda(t)}{2}}\mathbb{I}, \tag{18}$$

$$M_1(t) = \sqrt{\frac{1 - \Lambda(t)}{2}}\sigma_z, \tag{19}$$

where

$$\Lambda(t) = e^{-t/(2\tau)}\left[\cos\left(\frac{\mu t}{2\tau}\right) + \frac{1}{\mu}\sin\left(\frac{\mu t}{2\tau}\right)\right], \tag{20}$$

with $\mu = \sqrt{(4\alpha\tau)^2 - 1}$, and $\mathbb{I}$ being the identity matrix.

For $\alpha\tau > 1/4$ the dynamics is non-Markovian, while for $\alpha\tau < 1/4$ it is Markovian. We have scaled our PD simulations with $\alpha = 1$, and considered τ in the range [0.1, 0.75]. The PD channel can be simulated using a quantum circuit, shown in Fig. 2 [71]. In this circuit, the Hadamard gate prepares the qubit into the superposition state and the controlled rotation simulates the interaction with the environment. The angle θ_p is given by

$$\theta_p = 2\arccos[\Lambda(t)], \tag{21}$$

where $\Lambda(t)$ is given in Eq. (20).

IV. RESULTS

We perform our simulations with the statevector_simulator and qasm_simulator, integrated in the Aer package from IBM Qiskit [75]. For comparison, we also run simulations using the Pennylane library [76], obtaining similar outcomes. The statevector_simulator is an ideal simulator that considers the evolution of the wave function. In contrast, the qasm_simulator mimics the open dynamics of the IBM quantum computer. This means that it considers losses and shot noise. However, it allows us to set all qubits equal and fully connected, without relying on a specific quantum hardware.

It is well known that the quantum state of a qubit can be represented as a point in a sphere of radius one (Bloch's sphere). A generic state can be represented in the Bloch's sphere in terms of the expectation values as

$$\rho = \frac{1}{2}\left(\mathbb{I} + \sum_{i=x,y,z}\langle\sigma_i\rangle\sigma_i\right), \tag{22}$$

where $\mathbb{I}$ is the 2×2 identity matrix.

For illustration we firstly focus on the amplitude damping channel. In Fig. 3 we show the expectation values calculated using the statevector_simulator and qasm_simulator. The former provides outcomes with no dispersion (top), as expected from the ideal simulation. On the other hand, qasm_simulator delivers more realistic results that include dispersion (bottom). This dispersion will be important for selecting the best algorithm that computes the overlap. In contrast, statevector_simulator brings no significant difference in the prediction. Therefore, it can be misleading when selecting a machine learning model, and thus hereafter we restrict our analysis to qasm_simulator.
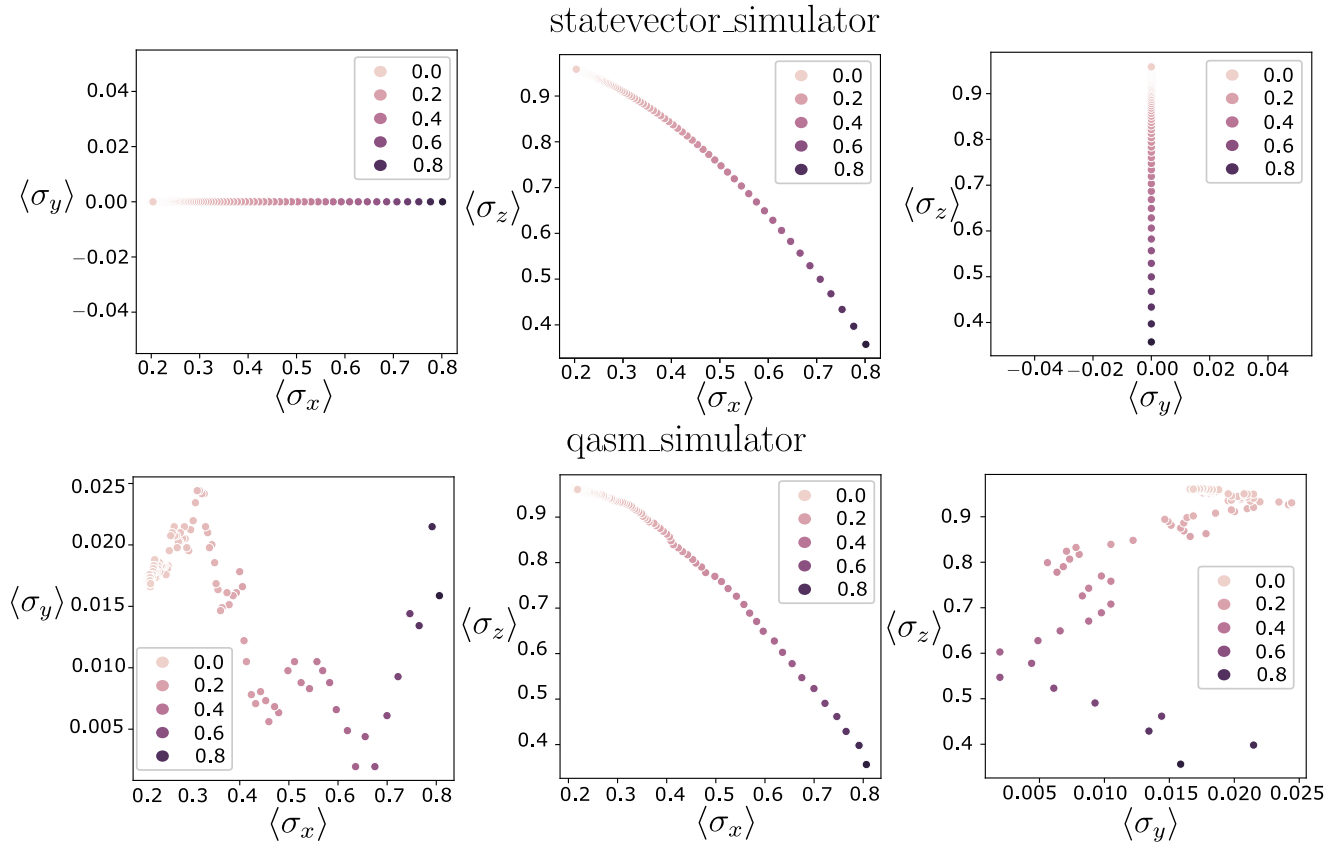
FIG. 3. Expectation values delivered by the noisy qasm_simulator exhibit small dispersion after 8000 shots, in contrast to the ideal statevector_simulator. We only observe correlations in the plane defined by $\langle \sigma_x \rangle$ and $\langle \sigma_z \rangle$.

In Fig. 4 we show the degree of NM for the amplitude damping channel as a function of the parameter θ (rotation angle that controls NM introduced in Sec. III B). For the calculations, we used qasm_simulator with the exponential kernel function that yields the best accuracy, as shown in Appendix B. For exploration of the algorithms we only focus on QSVM. We manually seek optimal hyperparameters and report the prediction on the training data set. A more robust analysis will be given later on. We can observe that the inversion test leads to a feature space that allows better prediction of the degree of NM. We note that the BBA algorithm can be improved for a different set of parameters, but it still underperforms.

We now compare the performance between quantum SVM (QSVM) and quantum KRR (QKRR). Hereafter, we focus on simulations on the qasm_simulator for the inversion test with exponential function. To prevent overfitting, we use two steps for cross-validation. First, we use the train_test_split function in scikit-learn [1] to randomly split the training set from the test set. Then, we use the GridSearchCV function to explore the best-fitting hyperparameters for each model, and we use a fivefold cross-validation. Thus, GridSearchCV provides the best estimator for the range of given parameters averaged over five different samplings of the training set. Finally, we used these estimators to predict the test set, which contains the data that the model has not seen. In Fig. 5 we show our predictions for amplitude damping and phase damping. One can observe that both models succeeded in predicting the degree

of non-Markovianity, besides small differences in the score (mean squared error). However, there are important aspects that might be taken into account before selecting one over the other. First, we remark that QSVM requires less training data to deliver good fittings. This is known, and it results from the sparseness in the training samples (only SVs contribute). Therefore, QSVM provides a major advantage given that the most time-consuming operation is the calculation of the Gram matrix. Thus, less training samples reduces the overall computation time. In contrast, we observe that as the number of data samples increases, QKRR improves.

For comparison, we estimate the degree of non-Markovianity using a classical kernel, i.e., the radial basis function (RBF). We follow the procedure reported in Ref. [54], where the training is carried out with the expectation values $\langle \sigma_x \rangle$, $\langle \sigma_y \rangle$, and $\langle \sigma_z \rangle$. Thus, instead of using quantum states to build up a kernel, we resort to using classical data, i.e., measurement outcomes. However, the process to obtain the states to be measured is the same we outlined in Sec. III—in Ref.[54] the authors used a master equation approach instead of digital quantum simulation.

In Table I we show the mean squared errors for each model for the AD and PD channels. We remark that the quantum versions, where the kernel is calculated from the overlap between quantum states, deliver accurate predictions that are comparable with the classical models, albeit we found that SVM with a RBF kernel provides the best accuracy, as evidenced in terms of the mean squared error and the coefficient
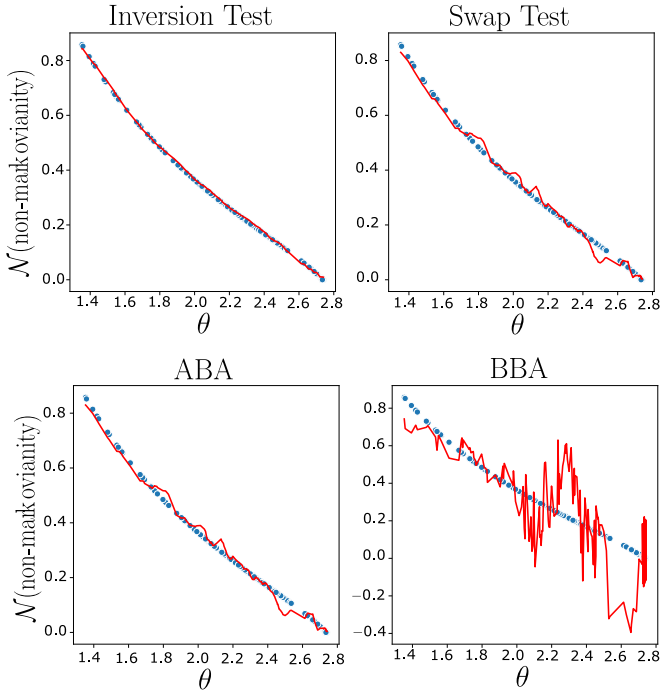
FIG. 4. QSVM prediction of non-Markovianity as a function of the rotation angle θ for different kernel circuits. The inversion test outperforms the others. We considered the amplitude damping channel, the exponential kernel function, and the hyperparameters $\{C = 0.5, \epsilon = 0.01\}$.

of determination R^2 (not shown here). This particular problem illustrates that extending the kernel to be quantum provides interesting insights and contributes to concatenate quantum blocks of operations. It does not necessarily outperform a fully classical training process but delivers useful outcomes.

V. CONCLUSIONS

In this paper we have thoroughly studied kernel-based quantum machine learning models to predict the degree of non-Markovianity using quantum data (quantum states). Each state is obtained through digital quantum simulation, where an ancilla qubit originates the non-Markovian behavior. We focus on two different decoherence channels, amplitude damping and phase damping. These quantum states are mapped to a Gram matrix by calculating its overlap. We investigate different kernel functions (linear, polynomial and exponential) and

TABLE I. The table shows the accuracy of the quantum and classical versions of the studied machine learning models. The hyperparameters for AD (PD) are as follows: QSVM: $C = 4 \times 10^{-1}(2 \times 10^{-1})$, $\epsilon = 10^{-2}$; QKRR: $\alpha = 10^{-1}(2 \times 10^{-1})$; SVM: $C = 10^2$, $\epsilon = 10^{-3}$; KRR: $\alpha = 10^{-4}(10^{-5})$.

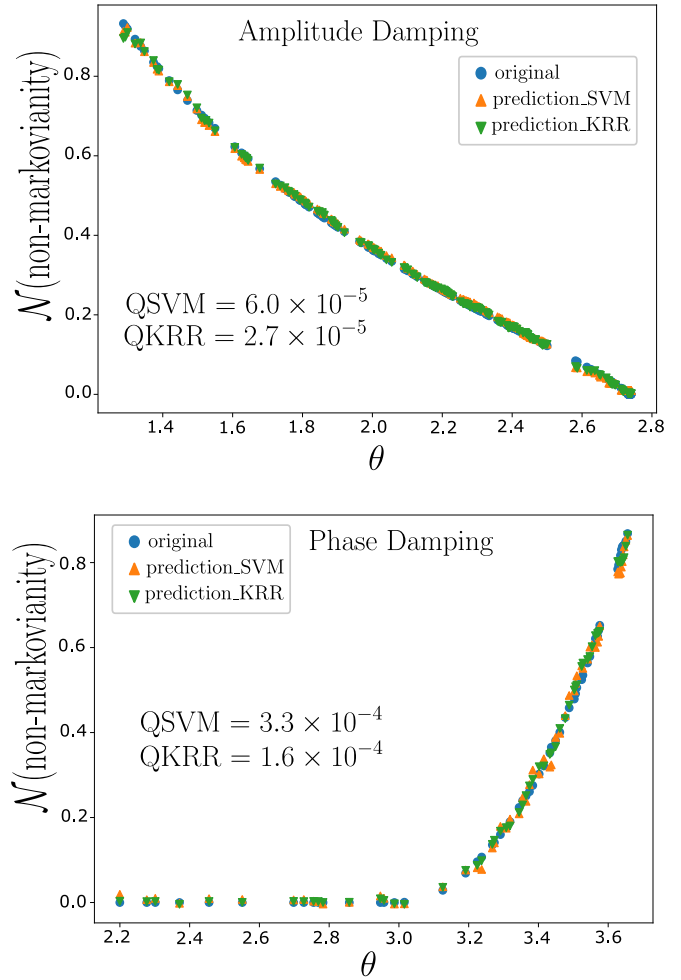|	QSVM	QKRR	SVM	KRR
AD	6.0×10^{-5}	2.7×10^{-5}	2.6×10^{-6}	1.4×10^{-5}
PD	3.3×10^{-4}	1.6×10^{-4}	5.9×10^{-5}	1.8×10^{-4}

FIG. 5. Both QSVM and QKRR deliver accurate predictions of the degree of non-Markovianity, based on the mean squared error score. For a small training data set QSVM performs better (not shown here). For a sufficiently large number of points QKRR provides a smaller mean squared error.

different kernel circuits to compute the overlap (inversion test, bell-basis algorithm, ancilla-based algorithm, and the swap test). We found that the inversion test with the exponential function delivers the best results. We draw our attention to two well-known kernel-based machine learning models, SVM and KRR. When the models are trained with precomputed quantum kernels we dubbed them QSVM and QKRR, respectively. By optimizing the learning process through cross-validation steps and grid search, we found a good accuracy in our models. We found QSVM to be slightly better than QKRR, not only in the prediction's accuracy but also in requiring less training samples.

Finally, we compare our results with their classical counterpart, i.e., when using classical data (expectation values) to train the models. While there are not significant differences, we observe that SVM with an RBF kernel delivers the best performance. This means that in this particular case it is better to measure upon the system and then process the measurement outcomes with machine learning techniques.

ACKNOWLEDGMENTS

APPENDIX A: LAGRANGIAN CALCULATIONS WITH SVM

We begin with the Lagrangian in Eq. (2),

$$L = \frac{1}{2}\|\vec{w}\|^2 + C\sum_i (\xi_i + \xi_i^*) - \sum_i (\eta_i \xi_i + \eta_i^* \xi_i^*)$$
$$- \sum_i \alpha_i(\epsilon + \xi_i - y_i + \vec{w}\cdot\vec{x}_i + b)$$
$$- \sum_i \alpha_i^*(\epsilon + \xi_i^* + y_i - \vec{w}\cdot\vec{x}_i - b). \quad (A1)$$

Taking the partial derivatives with respect to the primal variables (b, w, ξ_i, ξ_i^*) yields

$$\partial_b L = \sum_i (\alpha_i^* - \alpha_i) = 0, \quad (A2)$$

$$\partial_w L = w - \sum_i (\alpha_i - \alpha_i^*)x_i = 0, \quad (A3)$$

$$\partial_\xi L = C - \alpha_i - \eta_i = 0, \quad (A4)$$

$$\partial_{\xi^*} L = C - \alpha_i^* - \eta_i^* = 0. \quad (A5)$$

First, from the KKT condition we obtain $\eta_i\xi_i = 0$. Multiplying Eq. (A4) by ξ_i, we deduce the relation as
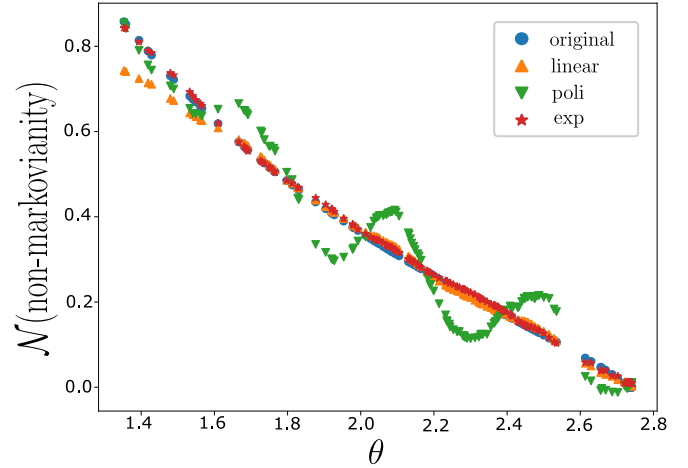
$$(C - \alpha_i)\xi_i = 0. \quad (A6)$$

FIG. 6. Exponential kernel function delivers the best prediction of non-Markovianity.

This means that only samples with $\alpha_i = C$ lie outside the ϵ tube ($\xi_i \neq 0$). We now consider the second constraint,

$$\alpha_i(\epsilon + \xi_i - y_i + \vec{w}\cdot\vec{x}_i + b) = 0. \quad (A7)$$

Note that all samples inside the ϵ tube ($|f(\vec{x}_i) - y_i| < \epsilon$) have a vanishing Lagrange multiplier α_i, which leads to the sparse representation of $f(\vec{x})$ in Eq. (4). A similar procedure can be followed for $\xi_i^*, \eta_i^*, \alpha_i^*$, which allows to approach the value for b [53].

APPENDIX B: KERNEL FUNCTIONS PERFORMANCE

We now compare three different functions for the kernel $k(x_i, x_j)$: linear $\langle x_i, x_j\rangle$, polynomial $(\langle x_i, x_j\rangle + 0.1)^3$, and exponential $\exp(-3\sqrt{1 - \langle x_i, x_j\rangle})$. Figure 6 shows that the exponential kernel function provides the best fitting. The polynomial function is only considered for completeness, since a more thorough exploration of the parameters may lead to a better fitting.

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. **12**, 2825 (2011).

[2] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. **91**, 045002 (2019).

[3] P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, A high-bias, low-variance introduction to machine learning for physicists, Phys. Rep. **810**, 1 (2019).

[4] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, Automatic differentiation in PyTorch, *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA* (NIPS, 2017).

[5] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. (MIT Press, Cambridge, 1998).

[6] V. Dunjko and H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: A review of recent progress, Rep. Prog. Phys. **81**, 074001 (2018).

[7] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nat. Phys. **13**, 431 (2017).

[8] A. Canabarro, F. F. Fanchini, A. L. Malvezzi, R. Pereira, and R. Chaves, Unveiling phase transitions with machine learning, Phys. Rev. B **100**, 045129 (2019).

[9] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature (London) **549**, 242 (2017).

[10] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell *et al.*, Quantum supremacy using a programmable

superconducting processor, Nature (London) **574**, 505 (2019).

[11] K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J. S. Chen, N. C. Pisenti, M. Chmielewski, C. Collins, K. M. Hudek, J. Mizrahi, J. D. Wong-Campos, S. Allen, J. Apisdorf, P. Solomon, M. Williams, A. M. Ducore, A. Blinov, S. M. Kreikemeier *et al.*, Benchmarking an 11-qubit quantum computer, Nat. Commun. **10**, 5464 (2019).

[12] T. M. Graham, Y. Song, J. Scott, C. Poole, L. Phuttitarn, K. Jooya, P. Eichler, X. Jiang, A. Marra, B. Grinkemeyer, M. Kwon, M. Ebert, J. Cherek, M. T. Lichtman, M. Gillette, J. Gilbert, D. Bowman, T. Ballance, C. Campbell, E. D. Dahl *et al.*, Multi-qubit entanglement and algorithms on a neutral-atom quantum computer, Nature (London) **604**, 457 (2022).

[13] A. Peruzzo, J. McClean, P. Shadbolt, Man-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. **5**, 4213 (2014).

[14] J. M. Arrazola, T. R. Bromley, J. Izaac, C. R. Myers, K. Brádler, and N. Killoran, Machine learning method for state preparation and gate synthesis on photonic quantum computers, Quantum Sci. Technol. **4**, 024004 (2019).

[15] M. H. Abobeih, Y. Wang, J. Randall, S. J. H. Loenen, C. E. Bradley, M. Markham, D. J. Twitchen, B. M. Terhal, and T. H. Taminiau, Fault-tolerant operation of a logical qubit in a diamond quantum processor, Nature (London) **606**, 884 (2022).

[16] R. Babbush, N. Wiebe, J. McClean, J. McClain, H. Neven, and G. K.-L. Chan, Low-Depth Quantum Simulation of Materials, Phys. Rev. X **8**, 011044 (2018).

[17] P. J. J. O'Malley, R. Babbush, I. D. Kivlichan, J. Romero, J. R. McClean, R. Barends, J. Kelly, P. Roushan, A. Tranter, N. Ding, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, E. Jeffrey, E. Lucero, A. Megrant, J. Y. Mutus *et al.*, Scalable Quantum Simulation of Molecular Energies, Phys. Rev. X **6**, 031007 (2016).

[18] K. M. Nakanishi, K. Mitarai, and K. Fujii, Subspace-search variational quantum eigensolver for excited states, Phys. Rev. Res. **1**, 033062 (2019).

[19] X. Peng, Z. Liao, N. Xu, G. Qin, X. Zhou, D. Suter, and J. Du, Quantum Adiabatic Algorithm for Factorization and Its Experimental Implementation, Phys. Rev. Lett. **101**, 220405 (2008).

[20] M. Rebentrost, P. Mohseni, and S. Lloyd, Quantum Support Vector Machine for Big Data Classification, Phys. Rev. Lett. **113**, 130503 (2014).

[21] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning, Quantum Inf. Comput. **15**, 0318 (2015).

[22] X.-D. Cai, D. Wu, Z.-E. Su, M.-C. Chen, X.-L. Wang, L. Li, N.-L. Liu, C.-Y. Lu, and J.-W. Pan, Entanglement-Based Machine Learning on a Quantum Computer, Phys. Rev. Lett. **114**, 110504 (2015).

[23] Z. Li, X. Liu, N. Xu, and J. Du, Experimental Realization of a Quantum Support Vector Machine, Phys. Rev. Lett. **114**, 140504 (2015).

[24] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature (London) **549**, 195 (2017).

[25] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature (London) **567**, 209 (2019).

[26] M. Schuld and N. Killoran, Quantum Machine Learning in Feature Hilbert Spaces, Phys. Rev. Lett. **122**, 040504 (2019).

[27] Z. He, L. Li, S. Zheng, X. Zou, and H. Situ, Quantum speedup for pool-based active learning, Quantum Inf. Process. **18**, 345 (2019).

[28] R. Mengoni and A. Di Pierro, Kernel methods in quantum machine learning, Quantum Mach. Intell. **1**, 65 (2019).

[29] K. Bartkiewicz, C. Gneiting, A. Černoch, K. Jiráková, K. Lemr, and F. Nori, Experimental kernel-based quantum machine learning in finite feature space, Sci. Rep. **10**, 12356 (2020).

[30] S. Johri, S. Debnath, A. Mocherla, A. Singk, A. Prakash, J. Kim, and I. Kerenidis, Nearest centroid classification on a trapped ion quantum computer, npj Quant. Info. **7**, 122 (2021).

[31] D. Willsch, M. Willsch, H. De Raedt, and K. Michielsen, Support vector machines on the d-wave quantum annealer, Comput. Phys. Commun. **248**, 107006 (2020).

[32] Y. Zhang and Q. Ni, Recent advances in quantum machine learning, Quantum Eng. **2**, e34 (2020).

[33] D. K. Park, C. Blank, and F. Petruccione, The theory of the quantum kernel-based binary classifier, Phys. Lett. A **384**, 126422 (2020).

[34] T. M. Khan and A. Robles-Kelly, Machine learning: Quantum vs classical, IEEE Access **8**, 219275 (2020).

[35] M. Schuld, Quantum machine learning models are kernel methods, arXiv:2101.11020.

[36] T. Goto, Q. H. Tran, and K. Nakajima, Universal Approximation Property of Quantum Machine Learning Models in Quantum-Enhanced Feature Spaces, Phys. Rev. Lett. **127**, 090506 (2021).

[37] X. Wang, Y. Du, Y. Luo, and D. Tao, Towards understanding the power of quantum kernels in the NISQ era, Quantum **5**, 531 (2021).

[38] C. Gyurik, D. van Vreumingen, and V. Dunjko, Structural risk minimization for quantum linear classifiers, Quantum **7**, 893 (2023).

[39] S. Saeedi, A. Panahi, and T. Arodz, Quantum semi-supervised kernel learning, Quantum Mach. Intell. **3**, 24 (2021).

[40] C. Ding, T.-Y. Bao, and H.-L. Huang, Quantum-inspired support vector machine, *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2021.

[41] H.-P. Breuer and F. Petruccione, *The Theory of Open Quantum Systems* (Oxford University Press, Oxford, 2007).

[42] E.-M. Laine, H.-P. Breuer, and J. Piilo, Nonlocal memory effects allow perfect teleportation with mixed states, Sci. Rep. **4**, 4620 (2014).

[43] A. W. Chin, S. F. Huelga, and M. B. Plenio, Quantum Metrology in Non-Markovian Environments, Phys. Rev. Lett. **109**, 233601 (2012).

[44] G. A. L. White, C. D. Hill, F. A. Pollock, L. C. L. Hollenberg, and K. Modi, Demonstration of non-Markovian process characterisation and control on a quantum processor, Nat. Commun. **11**, 6301 (2020).

[45] F. A. Pollock, C. Rodríguez-Rosario, T. Frauenheim, M. Paternostro, and K. Modi, Operational Markov Condition for Quantum Processes, Phys. Rev. Lett. **120**, 040405 (2018).

[46] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Parameterized quantum circuits as machine learning models, Quantum Sci. Technol. **4**, 043001 (2019).

[47] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, Variational quantum algorithms, Nat. Rev. Phys. **3**, 625 (2021).

[48] H. T. Dinani, D. Tancara, F. F. Fanchini, A. Norambuena, and R. Coto, Estimating the degree of non-Markovianity using variational quantum circuits, arXiv:2202.13964v3 (2022).

[49] V. Vapnik, *The Nature of Statistical Learning Theory* (Springer Verlag, New York, 1995).

[50] C. J. C. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Discov. **2**, 121 (1998).

[51] M. Opper and R. Urbanczik, Universal Learning Curves of Support Vector Machines, Phys. Rev. Lett. **86**, 4410 (2001).

[52] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, New support vector algorithms, Neural Comput. **12**, 1207 (2000).

[53] A. J. Smola and B. Schölkopf, A tutorial on support vector regression, Stat. Comput. **14**, 199 (2004).

[54] F. F. Fanchini, G. Karpat, D. Z. Rossatto, A. Norambuena, and R. Coto, Estimating the degree of non-Markovianity using machine learning, Phys. Rev. A **103**, 022425 (2021).

[55] L. E. H. Rodriguez, A. Ullah, K. J. R. Espinosa, P. O. Dral, and A. A. Kananenka, A comparative study of different machine learning methods for dissipative quantum dynamics, Mach. Learn.: Sci. Technol. **3**, 045016 (2022).

[56] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer, New York, 2009).

[57] R. LaRose and B. Coyle, Robust data encodings for quantum classifiers, Phys. Rev. A **102**, 032420 (2020).

[58] M. Weigold, J. Barzen, F. Leymann, and M. Salm, Encoding patterns for quantum algorithms, IET Quantum Commun. **2**, 141 (2021).

[59] J. A. Smolin and D. P. DiVincenzo, Five two-bit quantum gates are sufficient to implement the quantum Fredkin gate, Phys. Rev. A **53**, 2855 (1996).

[60] L. Cincio, Y. Subaşı, A. T. Sornborger, and P. J. Coles, Learning the quantum algorithm for state overlap, New J. Phys. **20**, 113022 (2018).

[61] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, Quantum embeddings for machine learning, arXiv:2001.03622v2 (2020).

[62] J. C. Garcia-Escartin and P. Chamorro-Posada, Swap test and Hong-Ou-Mandel effect are equivalent, Phys. Rev. A **87**, 052330 (2013).

[63] H.-P. Breuer, E.-M. Laine, and J. Piilo, Measure for the Degree of Non-Markovian Behavior of Quantum Processes in Open Systems, Phys. Rev. Lett. **103**, 210401 (2009).

[64] D. Chruściński, A. Kossakowski, and Á. Rivas, Measures of non-Markovianity: Divisibility versus backflow of information, Phys. Rev. A **83**, 052128 (2011).

[65] Á. Rivas, S. F. Huelga, and M. B. Plenio, Entanglement and Non-Markovianity of Quantum Evolutions, Phys. Rev. Lett. **105**, 050403 (2010).

[66] S. Luo, S. Fu, and H. Song, Quantifying non-Markovianity via correlations, Phys. Rev. A **86**, 044101 (2012).

[67] A. C. Neto, G. Karpat, and F. F. Fanchini, Inequivalence of correlation-based measures of non-Markovianity, Phys. Rev. A **94**, 032105 (2016).

[68] S. Hill and W. K. Wootters, Entanglement of a Pair of Quantum Bits, Phys. Rev. Lett. **78**, 5022 (1997).

[69] P. Haikka and S. Maniscalco, Non-Markovian dynamics of a damped driven two-state system, Phys. Rev. A **81**, 052103 (2010).

[70] S. J. Whalen and H. J. Carmichael, Time-local Heisenberg-Langevin equations and the driven qubit, Phys. Rev. A **93**, 063820 (2016).

[71] M. A. Nielsen and I. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2000).

[72] G. García-Pérez, M. A. C. Rossi, and S. Maniscalco, IBM Q experience as a versatile experimental testbed for simulating open quantum systems, npj Quantum Inf. **6**, 1 (2020).

[73] B. Bellomo, R. L. Franco, and G. Compagno, Non-Markovian Effects on the Dynamics of Entanglement, Phys. Rev. Lett. **99**, 160502 (2007).

[74] S. Daffer, K. Wódkiewicz, J. D. Cresser, and J. K. McIver, Depolarizing channel as a completely positive map with memory, Phys. Rev. A **70**, 010304(R) (2004).

[75] D. C. McKay, T. Alexander, L. Bello, M. J. Biercuk, L. Bishop, J. Chen, J. M. Chow, A. D. Córcoles, D. Egger, S. Filipp *et al.*, Qiskit backend specifications for OpenQASM and OpenPulse experiments, arXiv:1809.03452v1 (2018).

[76] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi *et al.*, Pennylane: Automatic differentiation of hybrid quantum-classical computations, arXiv:1811.04968v4 (2018).