Applied Quantum Algorithms

Towards Learning Separation

Tim Neumann

MSc Quantum Information Science and Technology

Delft University of Technology / Leiden University

May 2025

PART 1

Strongly correlated systems

1. a. *Write down the commutation relations for Fermionic creation and annihilation operators.*

The fermionic annihilation and creation operators $a_i^\dagger$ and a_i on some mode i obey the following anticommutation relations:

$$\{a_i, a_j^\dagger\} = \delta_{ij}\mathbf{1}, \ \ \{a_i, a_j\} = \{a_i^\dagger, a_j^\dagger\} = 0. \tag{1}$$

Hence, the commutation relations are given by

$$[a_i, a_j^\dagger] = \delta_{ij}\mathbf{1} - 2a_j^\dagger a_i, \ \ [a_i, a_j] = 2a_i a_j, \ \ [a_i^\dagger, a_j^\dagger] = 2a_i^\dagger a_j^\dagger. \tag{2}$$

1. b. *Write down a transformation of fermionic creation and annihilation operators into Majorana operators.*

In order to transform the fermionic operators into Majorana operators, we apply the transformation

$$c_{i,0} = a_i + a_i^\dagger, \ \ c_{i,1} = \mathbf{i}(a_i - a_i^\dagger), \tag{3}$$

where $\mathbf{i}$ is the imaginary unit.

1. c. *Write down the commutation relations of the resulting Majorana operators.*

The Majorana operator anti-commutation relation are given by

$$\{c_{i,\alpha}, c_{j,\beta}\} = 2\delta_{i,j}\delta_{\alpha,\beta}\mathbf{1}. \tag{4}$$

2. *Derive the average locality of a single fermionic creation or annihilation operator after being transformed to a qubit operator by the Jordan-Wigner transform.*

A single fermionic creation or annihilation operator does not only act locally, but on a set of modes that anti-commute with one another. In particular, after mapping a or $a^\dagger$ to its corresponding qubit operator

$$a = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \frac{1}{2}(X + iY) \text{ or } a^\dagger = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} = \frac{1}{2}(X - iY) \tag{5}$$

through the Jordan-Wigner transformation, we need to account for this anti-commutation with other modes (that also map to qubits in the JW transform) by additionally applying Z gates on the affected qubits in the chain, as described for the case of annihilation operators in Equation 6:

$$\begin{aligned} a_p \left|f_1, \ldots, f_p, \ldots, f_N\right\rangle &= \delta_{f_p,1}(-1)^{\sum_{i=0}^{p-1}} \left|f_1, \ldots, f_p \oplus 1, \ldots, f_N\right\rangle \\ \equiv a_p \mapsto_{JW} Z_1 \otimes \ldots Z_{p-1} &\otimes \frac{X_p + iY_p}{2} \otimes 1_{p+1} \otimes \ldots 1_N. \end{aligned} \tag{6}$$

Hence, for a n-mode-to-n-qubit mapping, we find an average locality of

$$\frac{1}{n}\sum_{i=1}^{n} i = \frac{1}{n}\binom{n+1}{2} = \frac{n+1}{2}, \tag{7}$$

which implies that the average locality of a single creation or annihilation operator scales linearly with the system size. Note however that we usually deal with hopping terms of the form $a_{i+1}^\dagger a_i$.

Such terms act local after the Jordan-Wigner transform, since the corresponding Z gates cancel out, and only the creation and annihilation operators on modes/qubits $i+1$ and i persist.

3. *Write down the Hamiltonian for the Fermionic Hubbard model on a 1-dimensional chain.*

The Fermionic Hubbard model on a 1-dimensional chain is given by

$$H_{\text{FH}} = -\mu \sum_{j=1}^{n} \sum_{s=\uparrow,\downarrow} a_{j,s}^\dagger a_{j,s} + U \sum_{j=1}^{n} a_{j,\uparrow}^\dagger a_{j,\uparrow} a_{j,\downarrow}^\dagger a_{j,\downarrow} - t \sum_{j=1}^{n} \sum_{s=\uparrow,\downarrow} a_{j,s}^\dagger a_{j+1,s} + a_{j+1,s}^\dagger a_{j,s}, \tag{8}$$

where μ is the chemical potential, U is the on-site energy, and t the hopping energy [1]. Usually, we consider periodic boundary conditions, such that the indices in the above expression should be read in modulo n.

4. *Transform the Fermionic Hubbard Hamiltonian onto a qubit basis via the Jordan-Wigner transformation. Do these two Hamiltonians have the same eigenspectrum (please explain in words rather than calculating)?*

Note that we have indexed the creation and annihilation operators with both the mode and the spin state in the previous problem (i.e., we allowed two electrons with different spin at the same site, effectively doubling the number of modes). Thus, when mapping the Hamiltonian onto a qubit basis, we require $2n$ qubits, where we define the convention of mapping the annihilation operator $a_{j,\uparrow}$ to qubit $2j-1$, and the annihilation operator $a_{j,\downarrow}$ to qubit $2j$ (and likewise for creation operators). Then we can rewrite (throughout implicitly assuming the indices to be read in modulo $2n$) the first term in H_{FH} as

$$\begin{aligned}
&-\mu \sum_{j=1}^{n} \Big(\frac{\mathbf{1}-Z}{2}\Big)_{2j-1} + \frac{1}{4} Z_{2j-1}(X+iY)_{2j-1}(X-iY)_{2j} + \frac{1}{4} Z_{2j-1}(X-iY)_{2j-1}(X+iY)_{2j} + \Big(\frac{\mathbf{1}-Z}{2}\Big)_{2j} = \\
&-\mu \sum_{j=1}^{n} \Big(\frac{\mathbf{1}-Z}{2}\Big)_{2j-1} + \frac{1}{2} Z_{2j-1}(X_{2j-1}X_{2j} + Y_{2j-1}Y_{2j}) + \Big(\frac{\mathbf{1}-Z}{2}\Big)_{2j},
\end{aligned} \tag{9}$$

the second term as

$$U \sum_{j=1}^{n} \Big(\frac{\mathbf{1}-Z}{2}\Big)_{2j-1} \Big(\frac{\mathbf{1}-Z}{2}\Big)_{2j}, \tag{10}$$

and the last term as

$$\begin{aligned}
-t \sum_{j=1}^{n} &\frac{1}{4} Z_{2j-1}Z_{2j}(X+iY)_{2j-1}(X-iY)_{2j+1} + \frac{1}{4} Z_{2j-1}Z_{2j}Z_{2j+1}(X+iY)_{2j-1}(X-iY)_{2j+2} + \\
&\frac{1}{4} Z_{2j}(X+iY)_{2j}(X-iY)_{2j+1} + \frac{1}{4} Z_{2j}Z_{2j+1}(X+iY)_{2j}(X-iY)_{2j+2} + \\
&\frac{1}{4} Z_{2j-1}Z_{2j}(X-iY)_{2j-1}(X+iY)_{2j+1} + \frac{1}{4} Z_{2j-1}Z_{2j}Z_{2j+1}(X-iY)_{2j-1}(X+iY)_{2j+2} + \\
&\frac{1}{4} Z_{2j}(X-iY)_{2j}(X+iY)_{2j+1} + \frac{1}{4} Z_{2j}Z_{2j+1}(X-iY)_{2j}(X+iY)_{2j+2} = \\
-t \sum_{j=1}^{n} &\frac{1}{2} Z_{2j-1}Z_{2j}(X_{2j-1}X_{2j+1} + Y_{2j-1}Y_{2j+1}) + \frac{1}{2} Z_{2j-1}Z_{2j}Z_{2j+1}(X_{2j-1}X_{2j+2} + Y_{2j-1}Y_{2j+2}) + \\
&\frac{1}{2} Z_{2j}(X_{2j}X_{2j+1} + Y_{2j}Y_{2j+1}) + \frac{1}{2} Z_{2j}Z_{2j+1}(X_{2j}X_{2j+2} + Y_{2j}Y_{2j+2}).
\end{aligned} \tag{11}$$

Note that we assume the identity to act on all qubits that are not explicitly included in each of the terms. The Jordan-Wigner transform defines an isomorphism from the Fock space to the

Hilbert (qubit) space. Now we know that a change of basis fixes the spectrum of an operator, i.e., the spectrum is invariant under a change of basis. It follows that the operators maintain an identical spectrum (but in their respective spaces with their respective eigenspaces). Hence, we can assert without calculation that the two Hamiltonians (i.e., the Fermionic Hubbard Hamiltonian on Fock space and on qubit space) have the same eigenspectrum.

5. *What is the scaling of the number of terms in the Hamiltonian with the system size for...*

   a. *... the Hubbard model*

The Hubbard Hamiltonian 8 verifies that there are $2+1+4=7$ terms per site/qubit in the Hamiltonian, such that the number of terms in the Hamiltonian scales as $7n$ for n sites/qubits, i.e., linearly with the size of the system.

   b. *... the Heisenberg model*

The Heisenberg Hamiltonian (with uniform coupling strength) is given by

$$H_{\mathrm{FH}} = \sum_{i,j=1}^{n} J(X_iX_j + Y_iY_j + Z_iZ_j), \tag{12}$$

which scales as n^2 for n sites, i.e., quadratically with system size (if we are considering a chain, the scaling is linear again).

   c. *... the Electronic structure problem*

The electronic structure problem is given by

$$H_{\mathrm{e}} = \sum_{i,j=1}^{n} J_{ij} a_i^\dagger a_j + \frac{1}{2} \sum_{i,j,k,l=1}^{n} V_{ijkl} a_i^\dagger a_j^\dagger a_k^\dagger a_l^\dagger, \tag{13}$$

which scales as $n^2 + n^4$ for n sites, i.e., quartically with system size.

Quantum Machine Learning

1. *Describe the implicit ("the quantum kernel estimator") and explicit ("the quantum variational classifier") of quantum variational SVMs.*

Support Vector Machines (SVM) are commonly used for classification tasks of the form

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|^2 \\ &\text{s.t. } y_i(w^t x_i - b) \geq 1, \end{aligned} \tag{14}$$

where $\{(x_i, y_i)\}_i$ is the training data. In general, it is not possible to find a hyperplane that separates the data in the original space, which necessitates to introduce the idea of a feature map Φ, which maps the data x to some (usually higher-dimensional) feature space. The task of optimizing Equation 14 is then performed in the feature space.

In close analogy to this, for the explicit quantum variational SVM, we use a variational circuit that "generates a separating hyperplane in quantum feature space" by employing variational means [2]. This method is closely related to parameterized quantum circuits. Here, we define a unitary feature map $\mathcal{U}_{\Phi(x)}$ that acts on the vacuum state, where $\Phi(x)$ denotes the target state of the data-dependent mapping [2]. This allows us to map classical data to quantum state space. Then, we initialize some unitary evolution $W(\theta)$ with a number of parameters (in [2], the authors use a $2n(l+1)$-dimensional parameter vector, and an additional bias parameter b). Lastly, the label classification is obtained through (repreated) binary measurement (in [2], through measurement in the computational basis). The variational part of the algorithm consists of iteratively evolving the vacuum state as $W(\theta)\mathcal{U}_{\Phi(x)}|0\rangle^{\otimes n}$ and measuring, followed by an update of the parameter vector θ (and potentially a bias parameters). The training rule is provided by a loss function dependent on the correct labels of the classical data, which we assume access to.

In terms of density matrix formalism, we observe that the feature map $\mathcal{U}_\Phi$ induces some data-dependent density matrix $\rho(x)$, whereas the observable O (which we measure) and the variational time evolution W define the time evolution $O(\theta) = W(\theta)OW^\dagger(\theta)$. Then the entire circuit can be recast as a function

$$x \mapsto \mathrm{Tr}[\rho_x O(\theta)], \tag{15}$$

from which we can define a loss function for optimization, e.g.,

$$\mathcal{L}(\theta, x) = \max_\theta \min_x (-1)^{\mathrm{label}(x)} \mathrm{Tr}[\rho_x O(\theta)]. \tag{16}$$

In loss function 16, the minimum identifies the data point with the smallest margin in feature space. This is similar to identifying x_i for which the classification is as weak as possible, such that $y_i \cdot w^t \Phi(x_i)$ is minimal in the generalized optimization Problem 14. Then the maximum in Expression 16 identifies the parameter that maximizes the margin for this particular choice of smallest-margin data point(s); this is analogous to minimizing over w in Expression 17 (since minimizing $\|w\|$ implies maximizing the margin).

Overall, we see that the explicit quantum variational SVM recasts the method of classical SVMs to a quantum variational problem, in which the classification quality is encoded in the loss function of a variational circuit, which is updated iteratively.

On the contrary, in the implicit quantum variational SVM, we utilize the quantum computer only "to estimate the kernel function of the quantum feature space" [2] in order to accelerate a classical SVM method. This is a typical example of outsourcing a compute-intensive task in some classical algorithm to an accelerated quantum subroutine. In particular, the classical SVM formulation admits a dual problem which involves kernel terms $K(x_i, x_j)$.

This follows from recasting it as a Lagrangian optimization problem

$$\max_{\lambda\geq 0}\min_{w,b}\mathcal{L}(w,b,\lambda)=\max_{\lambda\geq 0}\min_{w,b}\frac{1}{2}\left\|w\right\|^2+\sum_{i=1}^{n}\lambda_i(1-y_i(w^tx_i-b)), \tag{17}$$

which we seek to minimize with respect to the primal variables w and b, and then maximize with respect to the dual variable λ. Performing the gradient calculations derived from the minimality condition $\frac{\partial L}{\partial w}=\frac{\partial L}{\partial b}=0$ and substituting the corresponding terms in Equation 17 yields the dual problem

$$\begin{aligned}&\max_{\lambda\geq 0}\sum_{i=0}^{n}\lambda_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_jy_iy_jx_i^tx_j\\ s.t.&\sum_{i=0}^{n}\lambda_iy_i=0,\lambda_i\geq 0.\end{aligned} \tag{18}$$

In a more general case, the dual formulation 18 now displays the aforementioned kernel

$$K(x_i,x_j)=\Phi(x_i)^t\Phi(x_j). \tag{19}$$

Instead of computing the kernel matrix classically (which is typically hard), we map the corresponding classical data to quantum state space via the map feature map $\mathcal{U}_\Phi$, and perform the kernel computation in this feature space. Quantumly, the "kernel entries are the fidelities between different feature vectors" [2], which can be (in the best case efficiently) computed using a quantum circuit. In particular, the kernel computation is achieved via measuring a parameterized quantum circuit, constructed such that the measurement outcome is the fidelity of two density matrices,

$$K(x,x')=\mathrm{Tr}[\rho_x\rho_{x'}], \tag{20}$$

as depicted in Figure 2c of [2]. Since the implementation resembles a fidelity measurement, the hope of this method is a more efficient way to determine the kernel function than classically possible. However, quantum kernels face problems of exponential concentration[3], so this is an on-going field of research.

Overall, we find two different approaches to the same problem. In the explicit formulation, a variational quantum circuit is used to minimize the cost function provided by the SVM formulation. This approach does not explore the entire subspace of states but gives a direct analogy to a classical SVM. In the implicit formulation, the classical SVM routine is accelerated through a kernel estimation quantum subroutine, which does explore the entire space, but necessitates choosing a correct PQC to model the kernel.

2. a. *What are feature maps and kernels in SVMs?*

Feature maps are maps that accept some data as input and map it to some feature space. As such, feature maps are functions from the space of the data to a target space that optimally has beneficial properties. The target space and function rule can vary greatly; for example, we can devise a feature map from classical data to classical data, where commonly the data is mapped into some higher-dimensional space such that an affine hyperplane can separate the mapped data. Analogously, we can also define the unitary maps $\mathcal{U}_\Phi : x\mapsto|\Phi(x)\rangle$, or $\mathcal{U'}_\Phi : x\mapsto|\Phi(x)\rangle\langle\Phi(x)|=\rho_{\Phi(x)}$, which maps classical data to quantum data, such as a quantum state or density matrix, respectively. Now kernels are the squared inner product of such mapped states. Borrowing from the above notation, such a kernel is of the form

$$K(x,y)=|\langle\Phi(x)|\Phi(y)\rangle|^2 \tag{21}$$

in general, where we interpret $\Phi(x)$ as a vector in feature space (this could both be a classical feature space or quantum Hilbert space). Note that quantumly, we can make use of the invariance of traces under cyclic permutation to reexpress Equation 21 as

$$K(x, y) = \mathrm{Tr}[\rho_{\Phi(x)}\rho_{\Phi(y)}]. \tag{22}$$

b. *What is the relationship between the two?*

As discussed above, the kernel is usually defined as the squared norm of the inner product of two data points in feature space.

c. *When does a kernel have a corresponding feature map?*

A kernel always has a corresponding feature map by Definition 21, since it is the squared norm of the inner product of such. For problems like the SVM, we are interested in identifying the kernel function associated to the feature map in order to simplify computations. In its consequence, this is the kernel trick: for some computational tasks, we only need the kernel, not the explicit feature mapping. In particular, Mercer's theorem asserts that every positive semi-definite function of two variables corresponds to an inner product in some feature space. That means, if we define *kernel* as some general function of two variables, we still find the correspondence between kernels and feature maps under the suitable conditions of Mercer's theorem. The benefit of this correspondence is that some feature maps might be hard to evaluate, but computing their kernel in the original space can be easy. In practice, we often disregard the exact form of the feature map entirely, and only consider 'expressive' kernels that correspond to *some* higher-dimensional map and allow for fast computation.

3. *Do there exist learning problems which quantum learners can learn but classical learners can not? If yes, provide an example.*

In general, the classes of problems that can be solved classically and quantumly coincide precisely, since a quantum circuit can evaluate any boolean function, and every quantum circuit is a (potentially very large) linear transformation. Assuming, however, that whether or not a learner *can* learn something refers to the question of whether it can learn something *efficiently* relative to the other type of learner, this question cannot be answered comprehensively at the moment.

Nonetheless, with common assumptions such as $BQP \subsetneq P/poly$, it has been proven that the learning of expectation values of polynomially-sized k-local Pauli operators or ground state energies under BQP-complete Hamiltonians display a learning separation [4]. which is discussed extensively in the report (second part of this paper).

4. *Given an n-qubit circuit of depth d (not dependent on n). Show that you can estimate the probability of the first qubit being in the state $|0\rangle$ in time independent of n using only a classical computer.*

Assume we have some quantum circuit of depth d, where d does not depend on the number of qubits n. Then, in particular, the number of interactions that the first qubit is involved in, either via single-qubit gates on this qubits, direct coupling to other qubits such as CNOTs, or secondary influences such as qubit-operations on entangled qubits, is upper-bounded by a term depending on d (and not n) in the worst case. This follows since heuristically, classical hardness of simulation arises when entanglement is present in a system. Now for a depth d circuit, the first qubit can only be entangled with a subsystem of the size 2^d; at each of the d layers, the number of qubits (implicitly or explicitly) interacting with the first qubit can at most double. Thus, independent of n, we only need to consider a subsystem of size $\leq 2^d$, which allows us to trace out irrelevant parts. Since d is fixed, we can ascribe some classical resource capacity for this, which does not scale with n, and can thus be considered efficient, even constant (depending on the difficulty of evaluating the interaction), in system size.

PART 2 – Project 2.F. (Case A)

Abstract

In this report, we aim to demonstrate a learning separation between classical and quantum learners by performing experiments on four different types of Hamiltonians. Therefore, we consider the problem of predicting Pauli string expectation values of a time-evolved quantum state, for which we train both a quantum model (LASSO on quantum data) and a classical neural network. We find that across all Hamiltonians, the quantum model outperforms the classical model due to the particular formulation of the learning problem, which reflects the fact that quantum state evolution is hard to simulate classically under appropriate system assumptions. We conclude that using real quantum hardware is necessary (although not sufficient) for a true demonstration of learning separation.

1 Introduction

Determining impactful problems that exhibit a learning separation between classical and quantum learners is pivotal in manifesting the utility of quantum computing in real-world applications. As demonstrated in [4, 5, 6], the task of predicting the expectation value of a Pauli string observable with unknown weights on a time-evolved state under some BQP-complete Hamiltonian falls into this category. In this report, we formalize the problem statement, present our experiments and discuss the computational results in the comparison of a quantum and classical learner, while elaborating on the concepts of learning separation and underlying assumptions throughout.

2 Theory & Background

Consider the task of predicting the expectation value of a Pauli string observable

$$O(\boldsymbol{\alpha}) = \sum_{i}^{m} \alpha_i P_i \tag{23}$$

on a time evolved state $p_H(\mathbf{x})$ under some Hamiltonian H. Formally, this learning task can be described as inferring the best-suited function $f^{\alpha}(\mathbf{x})$ out of a concept class

$$\mathcal{F} = \{f^{\boldsymbol{\alpha}}(\mathbf{x}) | \boldsymbol{\alpha} \in [-1,1]^m\}, \tag{24}$$

which contains parameterized functions that map some input bit string (some initial quantum state) $\mathbf{x}$ to its corresponding expectation value

$$Tr[\rho_H(\mathbf{x}) O(\boldsymbol{\alpha})]. \tag{25}$$

The learning process is defined by training a model on given data here; the data is classical, even though it might be generated by a quantum computer in the case of a quantum learner. Importantly, we consider all P_i to be k-local (in our experiment, we choose $k = 3$), and fix $\boldsymbol{\alpha} \in [-1,1]^m$ as coefficient vector of the m Pauli strings, where m scales polynomially as

$$3^k \binom{n}{k} \in \mathcal{O}(n^k).[1] \tag{26}$$

Eventually, we aim to provide evidence for a *learning separation*, which roughly means an "exponential advantage in learning"[4] of a quantum learner compared to a classical learner. Formally, such a separation exists if a quantum learner is capable of efficiently learning a concept class $\mathcal{F}$ *probably approximately correct* (PAC)[4, Definition 1], whereas a classical learner cannot. We will shortly see where such a separation arises from in the task of learning expectation values.

It should be noted that efficient learning differs from efficient computing. The former seeks to approximate an input-output relation typically through access to data (in the sense of an identification problem), whereas the latter requires knowledge of the underlying mechanism,

[1] It is also possible to keep the initial state fixed and vary $\boldsymbol{\alpha}$, which however defies the purpose of creating a learning separation, as argued in [4, p. 7].

and the capability to execute the computation. Hence on a very high level, learning describes the capability of generalizing given data to approximate the output, whereas computing refers to determining an output from an input. We then speak of learning a separation if a quantum learner can learn some data (quantum or classical) efficiently, whereas a classical learner cannot. Thus, the distinction between computing and learning often rests on a data gap; learning assumes access to data, whereas computing does not [7].

Further note that a function which is hard to compute classically might still be efficiently learnable; given that the assumption BPP $\subsetneq$ BQP holds, we may choose any concept $C \in$ BQP, $C \notin$ BPP, which implies that C is hard to compute classically. However, given access to data, a classical learner might still be able to learn C efficiently[7]. Another instance of this phenomenon is an algorithm that efficiently identifies a quantum circuit that performs a certain task, while not being able to execute (i.e., computing the circuit) it [8]. This identification vs. computation idea is a key concept in the theory of CC/CQ separation. Note that, on the other hand, we can also consider a problem that is hard to learn but easy to compute quantumly, an example of which is given by shallow circuits as elaborate in [9].

In the context of our problem, we strive to time-evolve our initial states under a Hamiltonian H such that we obtain a separation in *learning*. In particular, we prepare time-evolved states $\rho_H(\mathbf{x}_l)$ and compute their Pauli string expectation values

$$\phi_j(\mathbf{x}_l) = \langle P_j \rangle_l = Tr[\rho_H(\mathbf{x})_l P_j] \tag{27}$$

up to some precision ϵ_1[2], which are in turn provided to the model

$$h(\mathbf{x}) = \boldsymbol{\omega} \cdot \phi(\mathbf{x}). \tag{28}$$

We then train the model via LASSO regularization

$$\min_{\boldsymbol{\omega}\in\mathbb{R}^m, \|\boldsymbol{\omega}\|_1 \leq B} \frac{1}{N} \sum_{l=1}^{N} |\boldsymbol{\omega} \cdot \phi(\mathbf{x}_l) - y_l|^2. \tag{29}$$

Observe that this model (trained with linear regression + L1-regularization) itself is classical[3], but has access to quantum data (the Pauli expectation values). Training the model on a set of training, we want to find some $\boldsymbol{\omega}^*$ that approximates $\boldsymbol{\alpha}$ to sufficient accuracy in order to obtain a ϵ_2-bounded training error. Here, we require the model to predict expectation values on new data points, rather than learning $\boldsymbol{\alpha}$ itself (the quantum learner achieves both). In contrast, we define a neural network architecture and train it on the pairs of input bit strings and expectation values

$$\{(\mathbf{x}_l, Tr[\rho_H(\mathbf{x}_l)O(\boldsymbol{\alpha})])\}_l.$$

From this description of learners, it becomes apparent why we expect a learning separation on the given task; the quantum model has access to additional information about the expectation values, since it knows about the underlying generation mechanism, whereas the classical model does not. This seems like an unfair comparison at first sight, yet it should be noted that time-evolution under some Hamiltonian is efficient on a quantum computer, and since we assume a polynomially-sized Pauli string observable, computation of $\phi(x)$ is quantumly efficient[4]. This relates to the discussion of learnability vs. computability: Here, the computing efficiency of time evolution by means of a quantum computer gives rise to efficient learning.

These considerations motivate the choice of a Hamiltonian that is not efficiently simulatable classically. Under the (reasonable) assumption that the complexity class BQP is not contained in P/poly[5], we should choose Hamiltonians that

[2]This error is due to sampling. For small system sizes, we can conduct the experiments analytically using the density matrix formalism, but this defeats the purpose of elaborating on learning separation, since then the model becomes classically constrained.

[3]The data determines the initial state, and we apply a given quantum time evolution, from which we extract quantum expectation values as training data. This is nothing else than a data-dependent feature map, which is used by a classical LASSO model.

[4]In our experiment, we emulated this with Hamiltonian simulation through a shallow Trotter circuit.

[5]The former denotes the complexity class of bounded-error quantum polynomial time algorithms, the latter one the class of decision problems that are "solvable by a polynomial time deterministic algorithm equipped with an 'advice' bitstring" [4].

are BQP-complete for our time-evolution. As stated in [10], under appropriate formalization, many of the commonly used Hamiltonians such as the "antiferromagnetic Heisenberg and XY model [or] the Fermi-Hubbard model" are indeed BQP-complete.

3 Experiment and Results

For the implementation of the aforementioned experiment of predicting expectation values of time-evolved states, we use the *PennyLane* library with the *lightning.qubit* device for quantum state simulation (including expectation value measurement of such states), the *sklearn* library to train the LASSO model, as well as the *torch* library for training the neural network.

Firstly, we generate data for the Heisenberg Hamiltonian on a 2×5 lattice and train the LASSO model for different values of the hyperparameter B'. Note that in this formulation, B' refers to the penalty coefficient in the Lagrangian formulation of the LASSO problem, not to the upper bound of the L1 norm. Secondly, we analyse this data to find a reasonable regularization parameter B' with which we then generate data for the 10-qubit system under four types of Hamiltonians; the Heisenberg, antiferromagentic XY, 2D Ising and a simple Z Hamiltonian (which can be thought of as induced by a static magnetic field along the Z direction). For the exact formulation of Hamiltonians used, we refer to Appendix 5.1. Lastly, we train and compare the two models across the four types of Hamiltonians.

In the data generation phase, for each experiment we sample 4500 time-evolved states ρ_x, using 400 shots per Pauli measurement[6]. For the observable, we generate the $3^3\binom{10}{3} = 3240$-dimensional vector of coefficients $\boldsymbol{\alpha}$ for the vector of Pauli string observables, according to the uniform distribution over $[-1, 1]$. The motivation behind these hyperparameters is that both the LASSO as well as the neural network training are performed on 80% of the available data, such that for our experiments we have access to 3600 labeled training samples. The resulting system is then overdetermined, i.e., underparameterized, which allows us to leverage the full capability of the LASSO method. As we will discuss later, Hamiltonian simulation is efficient on a quantum computer, and since we assume the observable to be polynomially-sized in the number of qubits, data generation is in general efficient quantumly[7]. The time-evolution of the Hamiltonian is simulated by Trotterization, using three first-order Trotter steps. We reference to Appendix 5.2 for more information on Trotterization. We can also opt to generate data analytically instead of via sampling, as outlined in Appendix 5.3[8]. This can give insight into the learning mechanism, but is ultimately not desirable in production since it abstracts away the quantumness of the measurement process. The neural network consists of three hidden layers of node sizes 256, 128 and 64, respectively, and is trained for 5000 epochs using the ADAM optimizer. A treatment of hyperparameter optimization can be found in Appendix 5.7. To ensure commensurability, we perform 5-fold cross-validation to test the quantum model, and leave out a test set of 20% of the data to test the neural network.

3.1 Choosing the regularization hyperparameter B'

Ideally, for a given lattice size, we would sweep over all values $B \leq \|\alpha\|$[9] to find the parameter that minimizes the test error, in order to explore the tradeoff between encouraging too much sparsity (underfitting) and too little sparsity (potential overfitting). However, since the number of training samples is comparable to the dimension of the vector to be learned for the given system sizes, and since the loss on the test data is generally low by the construction of the linear

[6]Recall that we choose the number of shots as N in order to achieve sampling error $\epsilon_1 \approx 1/\sqrt{N}$, so this is chosen in order to achieve a sampling error $\epsilon_1 \approx 0.05$.

[7]This already foreshadows the choice of this particular problem to demonstrate learning separation.

[8]Instead of measuring the Pauli expectation values of quantum states in PennyLane, this approach entails computing the time-evolved density matrix ρ_x and determining the expectation values of the Pauli strings according to Expression 25.

[9]Note that this refers to B, not B'.

problem, we want to encourage a relatively low level of sparsity in the solution vector $\boldsymbol{\omega}^*$.

Comparing different hyperparameters of the LASSO optimization, we find that regular linear regression ($B' = 0$)[10] is capable of perfectly learning the coefficient vector $\boldsymbol{\alpha}$, and thus exhibits optimal predictive power, as expected for the overdetermined system in consideration. For further insight, Appendix 5.5 features the corresponding plots of the (unconstrained) linear regression experiment.

As Table 1 shows, increasing the penalty on the L1 norm (regularization) of the coefficient vector estimate $\boldsymbol{\omega}^*$ decreases the predictive power of the LASSO model[11]. In particular, we find that choices of $B' \in [10^{-7}, 10^{-5}]$ perform very well on both training and test data. Recalling our goal to approximate $\boldsymbol{\alpha}$ such that the training error is less than a critical threshold ϵ_1, the choice of $B' = 10^{-5}$ exhibits a desirably low training error of $6.86 \cdot 10^{-4}$, for example.

To visualize the predictions of the Pauli expectation values made with the choice of $B' = 10^{-7}$, Figure 1 displays how LASSO matches the underlying ground truth well for a 2×5 lattice. It should be noted that for larger B', the coefficient vector $\boldsymbol{\omega}^*$ is forced to be uniformly zero, as can be seen in Appendix 5.4.

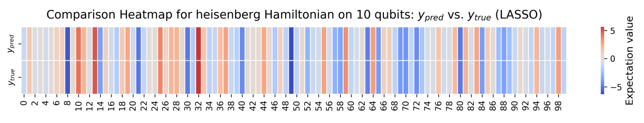

Figure 1: Comparison of observable expectation value of first 100 test set data points. Top Row: Prediction by LASSO model with $B = 10^{-7}$, Bottom Row: Ground truth as obtained from sampling.

We are only interested in a certain ϵ_2-suboptimality (perfect accuracy is not required) and may encourage some sparsity to potentially reduce measurement overhead in production[12]. While maintaining low training and test losses, we thus opt for the choice of $B' = 10^{-6}$ for our comprehensive experiments on different Hamiltonians, whose results are described in Section 3.2[13].

B'	0	10^{-7}	10^{-5}	10^{-3}	10^{-1}
Avg. loss	0	$4.73 \cdot 10^{-4}$	$4.46 \cdot 10^{-2}$	1.77	7.07

Table 1: Comparison of MSE loss of quantum model (LASSO regression) on test data for different values of hyperparameter B' (regularizaton penalty). Training and test data generated by time-evolution under the Heisenberg Hamiltonian on 2×5 lattice.

3.2 Predicting expectation values on 2×5 lattice

In the following, we summarize our results in numbers and figures, gathering evidence for a learning separation between the quantum and classical learner. Firstly, as Table 2 verifies, the MSE loss of LASSO predictions with $B' = 10^{-6}$ is generally multiple orders of magnitude lower than this of the neural network[14]. Secondly, the LASSO regularization accounts perfectly for the variance in the test data as reflected in the R^2 metric, which is unanimously 1.00 across all Hamiltonians.

Regarding our criterion on training loss, the LASSO method converges to a loss on the order of 10^{-5}, validating that the training objective is met. The training loss of the neural network, on the other side, remains large even after 5000 epochs, as visualized in Figure 2.

[10]The smaller B, the lower the penalty on the norm of the solution vector; in the limit of $B' \mapsto 0$, we perform linear regression.

[11]All predictions were obtained on comparable yet different samples; however, the logical causation between higher B' and higher MSE loss is clearly reflected in the data. We do not give uncertainty estimates here, since all experiments were only performed once due to high computational cost, and since small deviations are insignificant compared to the magnitudes of losses we are comparing.

[12]In general, we want to use LASSO because it prevents overfitting and enables generalization, even though in this specific case, the efficient data generation always allows us to at least match the number of unknowns to be estimated.

[13]We could take more metrics into account, but to motivate the choice of hyperparameter, we deem the MSE loss and the given problem context to be sufficient.

[14]Again, we do not give uncertainty estimates here, but small deviations are insignificant compared to the magnitudes of losses we are comparing to.

Hamiltonian	Heisenberg	XY	Ising	Z
MSE loss LASSO	$1.11 \cdot 10^{-3}$	$1.06 \cdot 10^{-3}$	$1.27 \cdot 10^{-3}$	$1.10 \cdot 10^{-3}$
$\frac{\text{MAE loss LASSO}}{\text{range expectation value}}$	$1.7 \cdot 10^{-3}$	$1.57 \cdot 10^{-3}$	$7.23 \cdot 10^{-4}$	$7.56 \cdot 10^{-4}$
MSE loss NN	3.51	3.42	6.34	6.71
R^2 LASSO	1.00	1.00	1.00	1.00
R^2 NN	0.41	0.54	0.86	0.83

Table 2: Comparison of MSE loss of quantum model (LASSO regression) and neural network for Heisenberg, XY, Ising and Z Hamiltonian on 2×5 lattice, according to various metric.

Here, we have used a standard neural network architecture as described in Subsection 3, but a more detailed treatment of hyperparameter optimization can be found in Appendix 5.7. While we cannot make any claims about the ϵ_2-suboptimality (since the model learns $\boldsymbol{\omega}^*$ only implicitly), this strongly suggests that the classical learner does not meet the learning requirement[15].

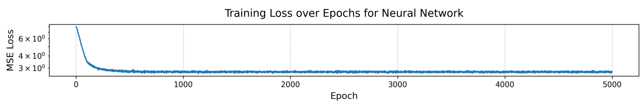

Figure 2: Neural network loss (logarithmic y-axis) over epochs for XY time evolution.

Even though we did not fully optimize the performance of the neural network, the significant difference in losses hints at a structural difference between the quantum and classical learner (the classical learner is not capable of learning the data efficiently), which is further elaborated on in Section 3.3. To visualize the predictions made by each learner, we compile the plots in Figures 3 and 4, showing the exemplary predictions of the LASSO model and neural network on the expectation values of the time-evolved states under the XY Hamiltonian.

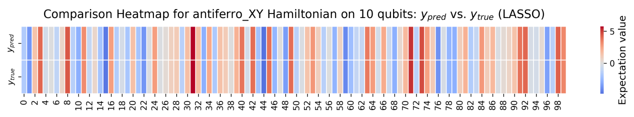

Figure 3: Comparison of observable expectation value of first 100 test set data points under XY time evolution. Top Row: Prediction by LASSO model with $B' = 10^{-6}$, Bottom Row: Ground truth as obtained from sampling.

The other three classes of Hamiltonians display very similar results, which can be accessed in detail at the author's GitHub page: https://github.com/timneumann1/learning_separation_time_evolution.

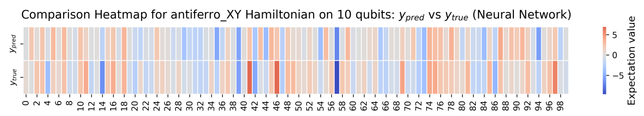

Figure 4: Comparison of observable expectation value of first 100 test set data points under XY time evolution. Top Row: Prediction by neural network, Bottom Row: Ground truth as obtained from sampling.

We highlight the beneficial property of the quantum learner to not only make accurate predictions, but to do this in an interpretable way by learning the underlying structure of the concept class. For reference, the exemplary learning of the coefficients under the 2D Ising Hamiltonian is depicted in Figure 5.

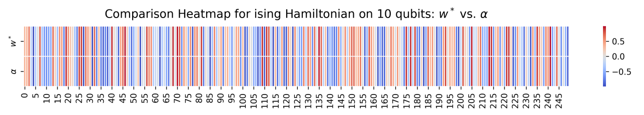

Figure 5: Comparison of first 250 observable coefficients under Ising time evolution. Top Row: Prediction by LASSO model with $B' = 10^{-6}$. Bottom Row: Ground truth as obtained by fixing $\boldsymbol{\alpha}$.

3.3 Comparing classical and quantum learner

The above results suggest that there does exist a learning separation between classical and quantum learners on the given task. We next

[15]For completeness, it should be noted that the classical learner predicts the expectation values perfectly when provided the string of Pauli data instead of bit strings, which of course violates our assumption of full classicality.

interpret the results of Section 3.2 by providing some theoretical considerations of why the learning separation should hold, while also exploring potential weaknesses of the method.

To begin with, we make concrete why we expect a learning separation for certain Hamiltonians and sufficiently large systems. The main reason for this is the qualitative difference in learning context: The quantum learner can access the Pauli string expectation values of time-evolved states, and infers the coefficients by solving a linear system (up to regularization)[16]. Meanwhile, the classical learner is only provided with a mapping of bitstrings to expectation values, i.e., scalars. The critical component here is the classical hardness to perform time-evolution of a quantum state under a BQP-complete Hamiltonian[4]. As outlined there, such Hamiltonians do not only arise from (artificial) cryptographic problems, but also from physically relevant systems. The Heisenberg and antiferromagnetic XY Hamiltonian admit BQP-complete formulations[10], and the 2D Ising Hamiltonian also features non-trivial correlations. The role of the simple Z magnetization Hamiltonian, which is simulatable classically, will be determined below. From a data science perspective, the key distinction between the learners is thus availability of data, arising from the quantum learner having access to a quantum way of efficiently performing the time evolution. This agrees well with the observation that learning advantages occur when "concepts are hard but data are easily generatable"[7].

On a theoretical level, Theorem 1 in [4] asserts the existence of such a learning separation under reasonable assumptions for some Hamiltonian (cf. [4, Table 1] as well). The advice bitstring from Definition 3 in [4] are the Pauli string measurements (of which there are polynomially many), which the classical learner does *not* have access to. Similar to the endeavor of Hamiltonian learning in [10, p. 5.3], we have the concept class 24, and we have training data generated by a "genuine quantum process"[10]. Importantly, and different to some other settings, the classical learner is not asked to identify the correct concept of a parameterized concept class, but learn a complete blackbox mapping $\Phi : \{0,1\}^{10} \mapsto \mathbb{R}$, which additionally is highly non-linear in the input due to projective measurement. Meanwhile, the quantum learner has access to the parameterized concept class and simply needs to identify the correct one (the coefficient vector $\boldsymbol{\alpha}$).

The comparison between learners that identify a concept class with greatly varying amounts of prior knowledge might sound somewhat contrived. This idea of an 'unfair' comparison often underlies the task of evaluating classical vs. quantum performances, since the regimes of computation are different. Nonetheless, the overall argument is valid, and our results suggest that we were able to directly observe a learning separation in our experiments on overdetermined systems. Still, there are some fundamental concerns that might be raised about the setup of our experiments, which we address next.

Firstly, apart from the fact that the LASSO regression took advantage of the linearity in the data available to the quantum learner, which is not available to the classical learner, we might ask about the quality of the classical learner itself. In order to truly demonstrate a learning separation empirically, one would have to test against *any* possible (non-oracular) classical model[17], which is of course infeasible. Nonetheless, in order to ensure a minimal level of commensurability[18], a fair effort should be devoted to optimize the classical learner, in our case the neural network.

Therefore, we performed hyperparameter optimization on our multiple-layer network with decaying layer size. Fixing the common (leaky) ReLU activation function as well as an early stopping condition and a weight decay term, we varied batch size and learning rate as outlined in Appendix 5.7. We also ensured to provide

[16]Admittedly, the system is quite large; in our case, the equation $Ax = b$ has to be solved for a 4500×3240 matrix A. However, linear regression is a standard and well-supported task.

[17]Non-existence of evidence is not evidence of non-existence.

[18]We compare a simple but very well-suited quantum learner with a more or less arbitrary classical learner, so it is desirable to put effort into optimizing the learner that has no access to more structural information about the problem.

the same amount of test samples (5-fold cross-validation for LASSO and 20% test data split for the neural network). However, the training and test loss remained at a high level for all Hamiltonians in hyperparameter experiments, so we defaulted to the simple neural network outlined in 'prediction.py' for our experiments. In this context, further lines of work could explore changing the data input to the network (e.g., using one-hot encoding instead of bitstrings), even though we deem it unlikely that such a change will bridge the learning gap for a 10-qubit system. Additional work could be devoted to identifying other classical (potential non-ML) learners altogether, even though the choice of neural networks as universal function approximators seems sensible.

Secondly, and most importantly, there is a subtle point about the setup of the experiment itself. While in theory, there does exist a learning separation for the given problem[19], it is by construction impossible to faithfully reproduce this empirically without access to a quantum computer. This opens two potential ways to approach this project; either by giving both learners access to all data, which would imply that we do not explore any quantum benefit (since we perform all experiments on a classical device), or by creating two different learning scenarios, which is not faithful to the actual data generation mechanism yet more strongly resembles a real-world scenario in which a quantum machine can be used. In order to investigate the power of quantum simulation, we chose the second option. It is clear, however, that we observed the separation only because we enforced a data availability on the learners that is derived from theoretical arguments, not numerical insight.

This insight finally sheds light on why the simple Z Hamiltonian displayed a learning separation as well, even though it is efficiently simulatable classically. It could be argued that we should have trained the classical model on the Pauli expectation values, since we expect classical simulation to be efficient for this type of Hamiltonian[20]. However, under this setting it is not even necessary to train another model, since then the LASSO model can be considered a classical model as well, and learning separation disappears.[21] In some sense, the inclusion of a non-BQP-complete Hamiltonian in this project is therefore redundant, which is our reason for omitting discussion of its results other than listing its performance in Table 2.

Thus, a proper demonstration of a learning separation relies on the existence of an appropriately large and true quantum simulator, potentially in a post-NISQ era.[22] Nonetheless, our experiments aim to replicate behavior in the limit of a system size of hard Hamiltonians, for which learning separation is formally proved, which partly justifies the choice of different learning contexts.

4 Conclusion and Outlook

Throughout this report, we have investigated the problem of learning expectation values of time-evolved quantum states. We provided numerical evidence that a quantum learner performs exceptionally well on this task due to access to data that is efficient to generate quantumly. On the other hand, a classical learner did not meet the learning requirements. These insights were reflected in the MSE loss and R^2 metrics depicted in 3.2. We also tried to preemptively address some concerns about the results, and pointed out that a faithful replication of learning separation can only be achieved using quantum hardware.

[19]As listed in [4, p. 11], the Heisenberg and XY Hamiltonian we considered typically give rise to BQP-complete problems, and additionally, the 2D-Ising model involves Z-correlations between qubits.

[20]The argument would be that fairness of comparison requires that everything that can be generated classically should be supplied to the classical learner.

[21]Indeed, in the context of this problem, choosing the first variant of the experiments as described above (equal access to data) would have made the project trivial, since then the optimally performing LASSO model would have acted as both a quantum and a classical model.

[22]Note that for an empiric learning separation, we should then allow the classical learner access to polynomial-time resources to approximate the final density matrices and gather more information, at least. In the current setting, the classical learner is entirely agnostic to the Hamiltonian.

Due to structural similarity, we may compare these deliberations to the similar problem of predicting ground state properties on an unknown observable, where rigorous guarantees require certain assumptions on the Hamiltonians. A treatment of this can be found in Appendix 5.6.

It will be interesting to see evidence of learning separation on large systems using actual quantum hardware, while acknowledging the hardness of defining general principles for commensurability between quantum and classical algorithms. Possible further lines of research entail considering more general (non-nearest-neighbor) correlations, and different Hamiltonians (such as the transverse-field Ising model). Another interesting question is in how far a generic feature map ϕ' (that does not contains as much information about the evolved states as the Pauli expectation values but is still expressive) could improve classical learning.

Code availability

The code for this project and the generated experimental data is available on the author's GitHub page at `https://github.com/timneumann1/learning_separation_time_evolution`.

References

[1] Jan-Michael Reiner et al. "Emulating the one-dimensional Fermi-Hubbard model by a double chain of qubits". In: *Physical Review A* 94.3 (Sept. 2016). ISSN: 2469-9934. DOI: 10.1103/physreva.94.032338. URL: http://dx.doi.org/10.1103/PhysRevA.94.032338.

[2] Vojtech Havlicek et al. "Supervised learning with quantum enhanced feature spaces". In: *Nature* 567.7747 (Mar. 2019). arXiv:1804.11326 [quant-ph], pp. 209–212. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-019-0980-2. URL: http://arxiv.org/abs/1804.11326 (visited on 04/08/2025).

[3] Supanut Thanasilp et al. *Exponential concentration in quantum kernel methods*. 2024. arXiv: 2208.11060 [quant-ph]. URL: https://arxiv.org/abs/2208.11060.

[4] Riccardo Molteni, Casper Gyurik, and Vedran Dunjko. *Exponential quantum advantages in learning quantum observables from classical data*. 2024. arXiv: 2405.02027 [quant-ph]. URL: https://arxiv.org/abs/2405.02027.

[5] Hsin-Yuan Huang et al. "Provably efficient machine learning for quantum many-body problems". In: *Science* 377.6613 (Sept. 2022). ISSN: 1095-9203. DOI: 10.1126/science.abk3333. URL: http://dx.doi.org/10.1126/science.abk3333.

[6] Laura Lewis et al. "Improved machine learning algorithm for predicting ground state properties". In: *Nature Communications* 15.1 (Jan. 2024). ISSN: 2041-1723. DOI: 10.1038/s41467-024-45014-7. URL: http://dx.doi.org/10.1038/s41467-024-45014-7.

[7] Riccardo Molteni. *Exponential quantum advantages in learning quantum observables from classical data*. https://www.youtube.com/watch?v=4OCN8v-TMfA&t. 2024.

[8] Vedran Dunjko. *Exponential separations between classical and quantum learners – IPAM at UCLA*. https://www.youtube.com/watch?v=IlEixl4mq0o&t=1275s. Institute for Pure & Applied Mathematics (IPAM), Accessed: 2025-05-09. 2021.

[9] Alexander Nietner et al. *On the average-case complexity of learning output distributions of quantum circuits*. 2023. arXiv: 2305.05765 [quant-ph]. URL: https://arxiv.org/abs/2305.05765.

[10] Casper Gyurik and Vedran Dunjko. *Exponential separations between classical and quantum learners*. 2024. arXiv: 2306.16028 [quant-ph]. URL: https://arxiv.org/abs/2306.16028.

5 Appendix

5.1 Hamiltonian formulation for numerical experiments

We consider four types of Hamiltonians. Firstly, the Heisenberg Hamiltonian

$$H_{Heisenberg}(J) = \sum_{i,j} J_{ij}\Big(X_i X_j + Y_i Y_j + Z_i Z_j\Big) \tag{30}$$

where the sum ranges over all nearest-neighbor interactions, and J_{ij} is sampled uniformly random from $[-1, 1]$; secondly, the antiferromagnetic XY Hamiltonian

$$H_{XY}(J) = \sum_{i,j} J_{ij}\Big(X_i X_j + Y_i Y_j\Big), \tag{31}$$

with the same conditions as for Hamiltonian 30; thirdly, the 2D Ising Hamiltonian with local Z magnetizations[23]

$$H_{Ising}(J, h) = \sum_{i,j} J_{ij}\Big(Z_i Z_j\Big) + \sum_i h_i Z_i; \tag{32}$$

and lastly, a simple Z-type Hamiltonian

$$HZ(J) = \sum_i J_{ij}\Big(Z_i\Big), \tag{33}$$

where the sum ranges over all sites.

5.2 Trotterization

The Hamiltonians H we encounter in this report contain non-commuting terms. In order to enable simulation of time-evolution $U = e^{-itH}$ on a gate-based quantum circuit, we use the Trotter-Suzuki decomposition

$$e^{A+B} = \lim_{n \mapsto \infty}\Big(e^{A/n} e^{B/n}\Big)^n. \tag{34}$$

Assuming that a Hamiltonian in question can be written as $H = H_1 + H_2$ for $[H_1, H_2] \neq 0$, we approximate the Hamiltonian-induced time evolution as

$$U_H = e^{-itH} = \Big(e^{H_1 t/n} e^{H_2 t/n}\Big)^n + O\Big(\frac{1}{n}\Big), \tag{35}$$

where in particular we choose $n = 3$ (three Trotter steps) and $t = 1$ (for simplicity)[24]. Equation 35 is the first-order Trotter approximation, which we consider sufficient in the scope of this project.

5.3 Comparison of learners for analytical time evolution

Whereas real quantum computers suffer from sampling error, the small system size allows us to get some insight into precise state evolution. Using the expectation value formalism on density matrices outlined in Equation 25, we conduct an experiment on a 2×3 lattice. The results are summarized in Table 3.

[23]This is not the transverse field Ising model, which is also of interest for further exeriments.

[24]Note that we also assumed $\hbar = 1$.

Hamiltonian	Heisenberg
Avg. loss LASSO	$1.34 \cdot 10^{-5}$
Avg. loss NN	$9.56 \cdot 10^{-3}$

Table 3: Comparison of MSE loss of quantum model (LASSO regression) and Neural Network on test set for analytical expectation value computation, for Heisenberg Hamiltonian on 2×3 lattice.

For the small 6-qubit system, the system is strongly overdetermined/underparameterized (the solution vector $\boldsymbol{\alpha}$ only has 540 entries), so we see that the LASSO predictions match the analytical values well, as can be verified in the excerpt of expectation values in Figures 6 and 7. It also becomes apparent that the classical learner can fulfill the task to high accuracy as well (learning separation makes no claim about small system sizes in which classical state simulation may still be efficient).

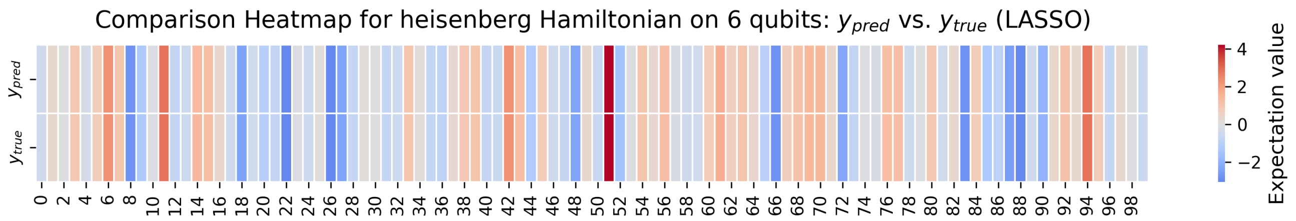

Figure 6: Comparison of observable expectation value of first 100 test set data points. Top Row: Prediction by LASSO model with $B' = 10^{-4}$. Bottom Row: Ground truth as obtained by fixing $\boldsymbol{\alpha}$.

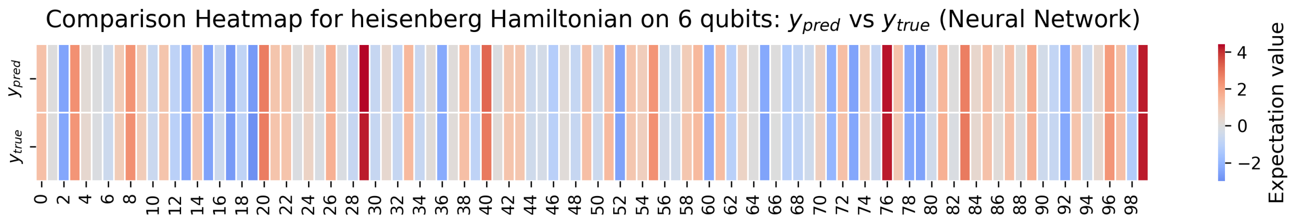

Figure 7: Comparison of first 100 test set Pauli string expectation values. Top Row: Prediction by Neural Network. Bottom Row: Ground truth as obtained by fixing $\boldsymbol{\alpha}$.

5.4 Comparison of different hyperparameters for regularization

A visual inspection helps to determine a good choice for the hyperparameter B': For B' too large, the coefficients and predictions are forced to be very small, and zero in the limit. This can be seen in Figure 8, which displays the solution vector $\boldsymbol{w}^*$, and Figure 9, which shows the the predictions thereof, for $B' = 10^{-1}$. A lattice size of 2×5 was used for these hyperparameter experiments.

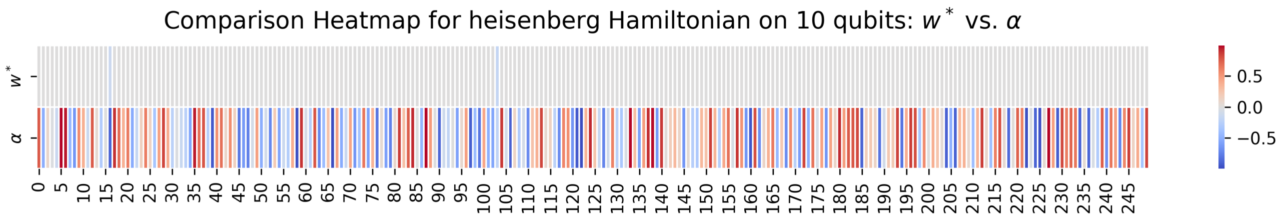

Figure 8: Comparison of first 250 observable coefficients. Top Row: Prediction by LASSO model with $B' = 10^{-1}$. Bottom Row: Ground truth as obtained by fixing $\boldsymbol{\alpha}$.

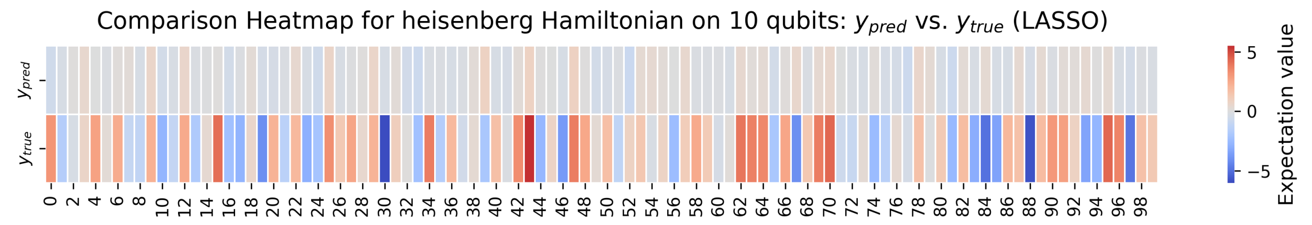

Figure 9: Comparison of observable expectation value of first 100 test set data points. Top Row: Prediction by LASSO model with $B' = 10^{-1}$, Bottom Row: Ground truth as obtained from sampling.

We also find that this choice of hyperparameter violates our condition on the training loss; the training set MSE error is 7.05 here, which is very large compared to the average expectation values that are being estimated.

5.5 Linear regression model for prediction of expectation values

Omitting the regularization term in the quantum regression model leads to a solution that is not sparse, as seen in Figure 10; instead, the coefficient vector is perfectly replicated, which is also a consequence of the choice of overdetermined system. The loss on both training and testing data is on the order of 10^{-28}, as the visualization of the corresponding predictions of expectation values of a 2×5 lattice under the Heisenberg Hamiltonian in Figure 11 exemplifies[25]. Similar experiments for underdetermined systems showed good yet not perfect alignment, which is sensible.

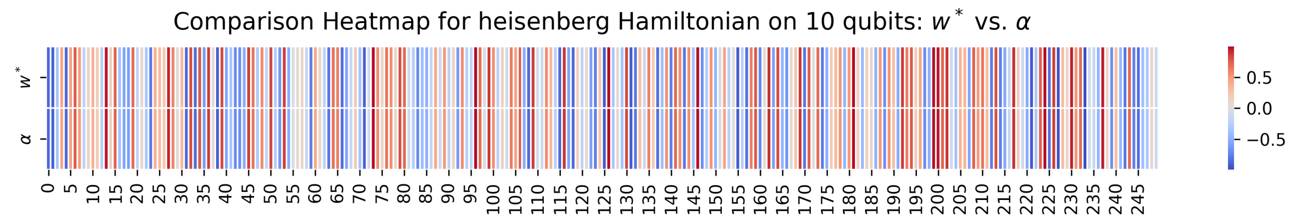

Figure 10: Comparison of first 250 observable coefficients. Top Row: Prediction by Linear Regression model without norm penalty. Bottom Row: Ground truth: fixed $\boldsymbol{\alpha}$.

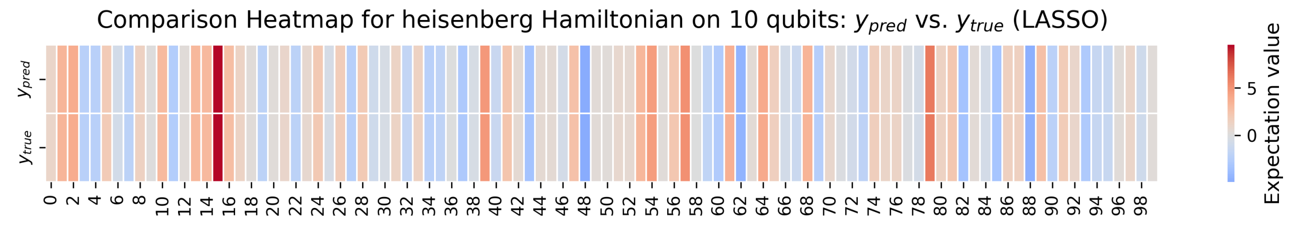

Figure 11: Comparison of observable expectation value of first 100 data points (of entire data set). Top Row: Prediction by linear regression model. Bottom Row: Ground truth as obtained by fixing $\boldsymbol{\alpha}$.

5.6 Learning separation for ground states

As featured in [4], the problem of estimating ground states of Hamiltonians parameterized by some coupling features a learning separation between classical and quantum learners. Now in a

[25]Instead of using the LASSO model, we used LinearRegression model provided by sklearn for this task.

similar direction of work, both [5] and [6] describe ML algorithms that seek to provide evidence for a separation between ML methods and classical (non-ML) methods. While this seems to be a different separation than the one described in this report, we should note that data generation for the given ML methods is quantumly hard also in these cases, which to some extent reconciles the distinction. Now both the algorithms as well as the assumptions on the Hamiltonians in [5] and [6] differ quite significantly, so this section is concerned with outlining the differences of the two approaches.

Regarding assumptions, [5] works with a "smooth family of Hamiltonians with a constant spectral gap", whereas in [6], we consider a parameterized "n-qubit gapped geometrically local Hamiltonian", consisting of few-body interactions. The latter assumes the geometry to be known (in contrast to the former), but not how it is parameterized. Hence, the latter set of assumption is somewhat stronger than the former, but it also leads to more efficient utilization of resources.

From an algorithmic viewpoint, [5] introduces a kernel-based neural network. While this seems to be a non-quantum approach, the provided data to train this kernel contains the ground states under the given Hamiltonian, and is thus similar in spirit to the encoding of expectation values we encountered in Expression 27.

Now [6] takes a slightly different approach, defining a LASSO model almost identical in formulation to the one considered in Expression 29.[26] Here as well, we have access to training data $\{(\mathbf{x_l}, y_l)\}_l$, where $y_l \approx Tr[\rho(\mathbf{x_l})O])$, which we want to learn using the model $h^*(\mathbf{x}) = \boldsymbol{\omega^*} \cdot \phi(\mathbf{x})$. Importantly, the feature mapping ϕ is defined as local Pauli operators encoded with an indicator function, appropriate for the learning task, but quite similar in nature to the idea of encoding Pauli string expectation values in [4]. This is reasonable since the feature map encodes the energies of local couplings, which are then linearly combined to yield the ground state energy. Notably, these indicator functions are merely used for providing a rigorous guarantee, but the actual implementation uses the commonly used Fourier feature map. The core idea here is thus to use a feature mapping to approximate a kernel function. The guarantee for this approach to work is rooted in the claim that for the type of Hamiltonians in consideration, the ground state energy can be approximated well using a sufficiently expressive feature map. Even though the guarantee does not hold for the Fourier feature map, it performs well in experiment. This could inspire the work in our paper to adapt a more generic feature map ϕ that does not contains as much information about the evolved states.

In terms of sample complexity, the algorithm proposed in [5] displays a lower bounds of $N = n^{\Omega(1/\epsilon)}$, i.e., $\mathcal{O}(n^c)$, whereas the improved ML algorithm in [6] uses a dataset of size $N = \mathcal{O}(\log(n))$, which is a significant improvement. In particular, the sample complexity in [5] "depends exponentially on $\frac{1}{\epsilon}$"[6], whereas [6] establishes a "quasi-polynomial dependence on $\frac{1}{\epsilon}$". Additionally, the new algorithm works over any distribution, whereas the previous only worked over $[-1, 1]^m$. Regarding computational time, both algorithms exhibit scaling with $\mathcal{O}(nN)$, which implies a "nearly linear computational time" for the improved algorithm in [6]. Hence, we find an overall improvement across the categories.

The main takeaway of this digression into a different type of problem that shows learning separation is that the correct choice of method and the details of assumptions made have a vast impact on theoretical guarantees.

5.7 Hyperparameter optimization for neural network

To ensure some commensurability, we perform a systematic testing of the hyperparameters of learning rate and batch size to see whether the loss is affected. In particular, we construct a

[26]However, the inputs here are not the bitstrings, but the couplings of the Hamiltonian.

neural network with four hidden layers, accepting a 10-bit bitstring and returning a scalar. Since we found that adding or removing layers (as well as varying or removing dropout for regularization) did not impact the performance significantly for a standard choice of parameters, we opted to keep this architectural choice fixed. This includes an early stopping condition, a weight decay term and a regularization via dropout. The detailed specifications and implementation can be accessed in the file 'neural_net_opt.py' on the author's GitHub page. We refrained from rescaling the input and target variables since we operate with bitstrings and relatively small expectation values. The results of our hyperparamter optimization for the Heisenberg Hamiltonian are summarized in Table 4.

learning rate \ batch size	32	128	512	3240
10^{-4}	3.50 (0.41)	3.50 (0.41)	4.18 (0.30)	4.50 (0.24)
10^{-3}	3.59 (0.40)	3.51 (0.41)	3.40 (0.43)	3.50 (0.41)
10^{-2}	3.40 (0.43)	3.58 (0.40)	3.64 (0.39)	3.54 (0.41)
10^{-1}	4.38 (0.27)	3.70 (0.38)	4.42 (0.26)	4.62 (0.26)

Table 4: Comparison of MSE loss and R^2 metric (R^2 loss in parentheses) for different choices of hyperparameters in the neural network for Heisenberg Hamiltonian on 2×5 lattice.

An exemplary loss function plot is displayed in Figure 12, which shows that the learning condition is not met.

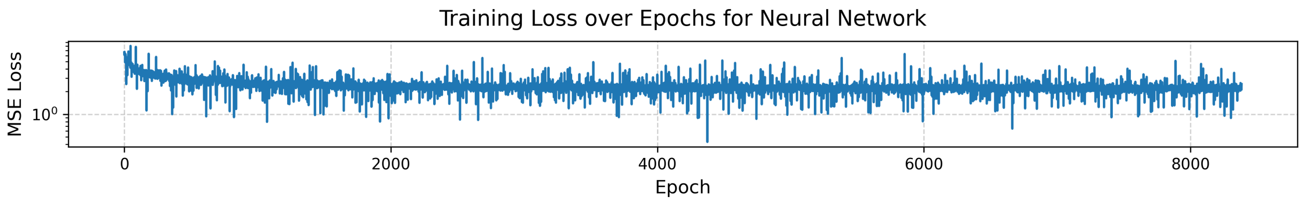

Figure 12: Loss vs. epoch for neural network model training of Heisenberg Hamiltonian with parameters $lr = 0.01$ and $batch\_size = 512$.

In general, the models are able to explain variance in the output better for the Z Hamiltonian; nonetheless, across all choices of hyperparameters, predictive power is quite poor compared to the quantum learner discussed in Section 3.2.