Towards Learning Separation
Predicting Unknown Observables on Time Evolved States

Applied Quantum Algorithms Leiden - Final Project

Student:
Egor Semenţul

May 2025

Contents

1 Part 1. Theory Questions

1.1 Strongly correlated systems

Q1.

a. Let $a_j^\dagger$ and a_j denote the fermionic creation and annihilation operators for mode j. They satisfy the canonical anticommutation relations:

$$\begin{aligned}\{a_i, a_j\} &= a_i a_j + a_j a_i = 0\\ \{a_i^\dagger, a_j^\dagger\} &= a_i^\dagger a_j^\dagger + a_j^\dagger a_i^\dagger = 0\\ \{a_i, a_j^\dagger\} &= a_i a_j^\dagger + a_j^\dagger a_i = \delta_{ij}\end{aligned}$$

These relations encode the Pauli exclusion principle and ensure the antisymmetric nature of fermionic wavefunctions.

b. For each fermionic mode j, define two corresponding Majorana operators γ_{2j-1} and γ_{2j} as linear combinations of the fermionic creation and annihilation operators:

$$\begin{aligned}\gamma_{2j-1} &= a_j + a_j^\dagger\\ \gamma_{2j} &= -i(a_j - a_j^\dagger)\end{aligned}$$

These operators satisfy the following properties:

- Each γ_k is Hermitian: $\gamma_k^\dagger = \gamma_k$
- The Majorana operators are real and self-conjugate combinations of a_j and $a_j^\dagger$

c. The Majorana operators γ_k (where $k = 1, 2, \ldots, 2M$ for M fermionic modes) obey the following anticommutation relations:

$$\begin{aligned}&\{\gamma_k, \gamma_l\} = 2\delta_{kl}\\ \Rightarrow \quad &\gamma_k^2 = 1 \quad (\text{since } \{\gamma_k, \gamma_k\} = 2)\end{aligned}$$

Some properties that follow:

- The Majorana operators are Hermitian: $\gamma_k^\dagger = \gamma_k$
- For $k \neq l$, they anticommute: $\gamma_k \gamma_l = -\gamma_l \gamma_k$
- They generate a Clifford algebra.

Q2. The Jordan-Wigner transformation maps a fermionic annihilation operator a_j into a non-local qubit operator:

$$a_j = \left(\prod_{k=1}^{j-1} Z_k \right) \cdot \frac{X_j + iY_j}{2}$$

This operator acts non-trivially on:

- Qubits 1 through $j-1$ via the string of Z operators
- Qubit j via X_j and Y_j

Therefore, the operator is j-local.

To compute the *average locality* over all N modes, we take the average of j from 1 to N:

$$\langle \text{locality} \rangle = \frac{1}{N} \sum_{j=1}^{N} j = \frac{N+1}{2}$$

Q3. The fermionic Hubbard model describes interacting fermions on a lattice. For a 1D chain with L lattice sites and spin-$\frac{1}{2}$ fermions, the Hamiltonian is:

$$H = -t \sum_{\langle i,j \rangle, \sigma} \left(a_{i,\sigma}^\dagger a_{j,\sigma} + a_{j,\sigma}^\dagger a_{i,\sigma} \right) + U \sum_i n_{i,\uparrow} n_{i,\downarrow}$$

Where:

- t is the hopping amplitude (kinetic term)
- U is the on-site interaction strength
- $\sigma \in \{\uparrow, \downarrow\}$ is the spin index
- $a_{i,\sigma}^\dagger$, $a_{i,\sigma}$ are fermionic creation and annihilation operators
- $n_{i,\sigma} = a_{i,\sigma}^\dagger a_{i,\sigma}$ is the number operator
- The sum $\langle i, j \rangle$ runs over nearest-neighbor pairs on the 1D chain

Q4. The Jordan–Wigner transformation maps fermionic operators into qubit operators. For the fermionic annihilation and creation operators, we have:

$$a_j = \left(\prod_{k=1}^{j-1} Z_k \right) \cdot \frac{X_j + iY_j}{2},$$
$$a_j^\dagger = \left(\prod_{k=1}^{j-1} Z_k \right) \cdot \frac{X_j - iY_j}{2}.$$

Applying this transformation to the 1D Hubbard Hamiltonian results in a Hamiltonian expressed purely in terms of Pauli operators acting on qubits. Each hopping term $a^\dagger_{i,\sigma} a_{j,\sigma}$ becomes a product of X, Y, and a string of Z's. Each on-site interaction term $n_{i,\uparrow} n_{i,\downarrow}$ transforms into a product of Z operators.

Spectral equivalence: The Jordan–Wigner transformation is a unitary mapping between fermionic and qubit Hilbert spaces. Since unitary transformations preserve eigenvalues, the spectrum of the Hamiltonian remains unchanged.

Q5.

a. The Hubbard Hamiltonian includes:

- Hopping terms between nearest neighbors for each spin species: $\mathcal{O}(N)$ terms
- On-site interaction terms: one per site $\Rightarrow \mathcal{O}(N)$ terms

Result: The total number of terms scales linearly: $\mathcal{O}(N)$

b. The Heisenberg Hamiltonian has nearest-neighbor spin interactions of the form:

$$H = \sum_{\langle i,j \rangle} J_x X_i X_j + J_y Y_i Y_j + J_z Z_i Z_j$$

Each bond contributes 3 terms, and there are $N-1$ bonds in 1D.

Result: The number of terms scales linearly: $\mathcal{O}(N)$

c. In second quantization, the Electronic Hamiltonian includes:

- $\mathcal{O}(N^2)$ one-body terms $a^\dagger_p a_q$
- $\mathcal{O}(N^4)$ two-body terms $a^\dagger_p a^\dagger_q a_r a_s$

After transformation (e.g., Jordan–Wigner or Bravyi–Kitaev), each becomes a sum over Pauli strings.

Result: The number of terms scales quartically: $\mathcal{O}(N^4)$

1.2 Quantum machine learning

Q1. Explicit model: Quantum variational classifier

This approach uses a parameterized quantum circuit to perform classification. The input $\vec{x}$ is encoded into a quantum state $\rho(\vec{x})$, which is then acted upon by a trainable observable $O(\vec{\theta})$. The model output is:

$$f_{\vec{\theta}}(\vec{x}) = \mathrm{Tr}[O(\vec{\theta})\rho(\vec{x})]$$

The parameters $\vec{\theta}$ are optimized to minimize a loss function over labeled data.

Implicit model: Quantum kernel estimator

In this approach, each input $\vec{x}$ is encoded into a quantum state $|\psi(\vec{x})\rangle$, and classification is performed by a classical SVM using the quantum kernel:

$$K(\vec{x},\vec{x}') = |\langle\psi(\vec{x})|\psi(\vec{x}')\rangle|^2$$

The kernel matrix K is computed via quantum overlaps between states. There are no trainable parameters in the quantum circuit; only the kernel is computed quantumly.

Q2.

a. A *feature map* $\Phi\colon \mathbb{R}^n \to \mathcal{H}$ embeds input vectors into a high-dimensional Hilbert space $\mathcal{H}$, where a linear classifier can be applied.

A *kernel function* $K(\vec{x},\vec{x}')$ computes the inner product in feature space:

$$K(\vec{x},\vec{x}') = \langle\Phi(\vec{x}),\Phi(\vec{x}')\rangle$$

In quantum settings, $\Phi(\vec{x}) = |\psi(\vec{x})\rangle$, and the quantum kernel is:

$$K(\vec{x},\vec{x}') = |\langle\psi(\vec{x})|\psi(\vec{x}')\rangle|^2$$

b. Kernels and feature maps are dual concepts. The kernel function is defined via the feature map as the inner product in feature space. Thus, the kernel determines similarity between inputs after mapping, enabling classification without explicitly constructing $\Phi(\vec{x})$.

c. A kernel $K(\vec{x},\vec{x}')$ corresponds to a feature map Φ if it is:

- Symmetric: $K(\vec{x},\vec{x}') = K(\vec{x}',\vec{x})$
- Positive semi-definite: for any finite set $\{\vec{x}_i\}$, the kernel matrix $K_{ij} = K(\vec{x}_i,\vec{x}_j)$ satisfies $K \geq 0$

These conditions are summarized in *Mercer's theorem*, which guarantees the existence of a Hilbert space and feature map such that $K(\vec{x},\vec{x}') = \langle\Phi(\vec{x}),\Phi(\vec{x}')\rangle$.

Q3. Yes, such learning problems exist. Consider the function class

$$\mathcal{F}_{H,O} = \{f_\alpha(x) = \mathrm{Tr}[\rho_H(x)O(\alpha)]\}\,,$$

where $x \in \{0,1\}^n$, $\rho_H(x) = e^{-iHt}|x\rangle\langle x|e^{iHt}$ is a time-evolved quantum state under a fixed local Hamiltonian H, and $O(\alpha) = \sum_j \alpha_j P_j$ is a sparse observable with known Pauli terms P_j.

A quantum learner can efficiently estimate the features $\phi_j(x) = \mathrm{Tr}[\rho_H(x)P_j]$ via local measurements and recover an accurate hypothesis using sparse linear regression, achieving polynomial sample and runtime complexity.

Now suppose a classical learner could efficiently PAC-learn $f_\alpha(x)$. Then, there would exist a polynomial-size classical circuit approximating f_α. If H is constructed to simulate a universal BQP circuit (e.g., via Feynman's circuit-to-Hamiltonian mapping), then evaluating $f_\alpha(x)$ is BQP-complete. Thus, an efficient classical learner would imply BQP $\subseteq$ P/poly, contradicting standard complexity assumptions.

Therefore, this constitutes a formal learning separation: the function class $\mathcal{F}_{H,O}$ is efficiently PAC-learnable by quantum algorithms but provably not by any classical algorithm unless BQP $\subseteq$ P/poly.

Q4. Let C be a quantum circuit acting on n qubits, with circuit depth $d = \mathcal{O}(1)$ independent of n. The task is to estimate the marginal probability that the first qubit is in state $|0\rangle$ after applying C, using a classical algorithm whose runtime is independent of n.

Key idea: In a depth-d quantum circuit composed of local gates, only a finite number of gates can influence the first qubit's final state. Specifically, only the gates within a light cone of depth d around qubit 1 affect its marginal distribution.

Construction: Let $L \subseteq [n]$ denote the subset of qubits within the causal light cone of qubit 1. Since the circuit depth is constant and each gate is local, the size of L is bounded by a constant $m = \mathcal{O}(1)$ — independent of n.

Now, restrict the full circuit C to only the m-qubit subsystem corresponding to L. This subsystem includes all gates that affect the first qubit's output. The reduced density matrix ρ_L on this subsystem captures all the information needed to compute the marginal probability on qubit 1.

Algorithm:

1. Extract the effective constant-size subcircuit acting on the m-qubit light cone.
2. Classically simulate this constant-size subcircuit exactly.
3. Compute the marginal probability $p_0 = \mathrm{Tr}[|0\rangle\langle 0|_1 \, \rho_L]$.

Conclusion: Since the circuit has constant depth and consists of local gates, only a constant number of qubits $m = \mathcal{O}(1)$ can influence the first qubit. Therefore, the marginal probability $p_0 = \mathrm{Tr}[|0\rangle\langle 0|_1 \, \rho_L]$ can be computed by classically simulating an m-qubit subcircuit. This simulation requires time exponential in m, but since m is independent of n, the total runtime is also independent of n, as required.

2 Part 2. Project 2.F. Case A

2.1 Introduction

Understanding the behavior of quantum many-body systems is a central challenge in quantum physics, with broad implications in condensed matter, quantum chemistry, and quantum information. A key task in this setting is to predict the expectation value of an observable O on a quantum state $\rho_H(x) = e^{-iHt}|x\rangle\langle x|e^{iHt}$, where H is a known local Hamiltonian and $x \in \{0,1\}^n$ is a computational basis state. This gives rise to the supervised learning problem: given training data $\{(x_\ell, y_\ell)\}$ where $y_\ell = \mathrm{Tr}[\rho_H(x_\ell)O]$, learn to predict y for unseen inputs x.

Recent works have shown that this task exhibits a *learning separation* [1, 2, 3]—that is, there exist physically natural choices of H and O such that a quantum learner can solve the learning problem efficiently (i.e., with polynomial resources), while any classical learner would require super-polynomial time or sample complexity unless BQP $\subseteq$ BPP. This phenomenon arises because evaluating the required features $\phi_j(x) = \mathrm{Tr}[\rho_H(x)P_j]$ (expectation values of local Pauli terms) is easy on a quantum device via measurement, but classically intractable due to the exponential cost of simulating e^{-iHt}.

In this project, we study a simplified version of the learning separation problem, motivated by the framework of Ref. [1], but adapted to time-evolved states. We construct a dataset $\{(x, y)\}$ where $y = \mathrm{Tr}[\rho_H(x)O(\alpha)]$ for a sparse Pauli observable $O(\alpha) = \sum_j \alpha_j P_j$. We simulate both exact features $\phi_{\mathrm{ex}}(x)$ (computed via classical matrix evolution) and noisy features $\phi_q(x)$ (extracted via Trotterized quantum circuits with finite shots). We compare these quantum learners with classical baselines trained directly on raw input bits.

Our experiments span several Hamiltonian families, including the all-to-all Heisenberg model, the 1D chain Heisenberg model, the transverse-field Ising model (TFIM), the XY model, and a global parity Hamiltonian. For each case, we train LASSO and neural network regressors on the available features and evaluate their generalization performance.

Our results confirm a learning separation: although classical models such as LASSO and neural networks can fit the task given quantum-accessible features, a purely classical learner — restricted to raw inputs or polynomial-time computable features — fails to generalize. These results support the theoretical prediction that quantum learners possess a provable advantage in learning certain structured physical prediction tasks.

2.2 Answers to Theory Sub-Questions

Question 1: *Understand and explain what we mean by separation in learning. Why is it different that a separation in computing? Can you give an example of a function which is hard to compute but not hard to learn?*

A **separation in learning** refers to a scenario where a function class is efficiently learnable in one computational model (e.g., quantum) but not in another (e.g., classical), under standard complexity-theoretic assumptions. This concept is formalized within the *Probably Approximately Correct* (PAC) learning framework [4], which studies the feasibility of learning functions from examples.

In the PAC model, a learner aims to find a hypothesis h that approximates an unknown target function f with high accuracy and confidence. Specifically, for a distribution $\mathcal{D}$ over inputs and parameters $\varepsilon, \delta > 0$, the goal is to ensure:

$$\Pr_{x\sim\mathcal{D}}[|h(x) - f(x)| > \varepsilon] \leq \delta.$$

A function class is said to be *efficiently PAC-learnable* if such a hypothesis can be found using a number of samples and computational resources polynomial in $1/\varepsilon$, $1/\delta$, and the input size n.

This notion contrasts with a **separation in computing**, which concerns the ability to evaluate a function exactly on arbitrary inputs. For instance, factoring is believed to be outside P but in BQP, meaning the problem is hard to compute classically but easy to compute quantumly using Shor's algorithm [5]. This represents a computational separation.

The key distinction lies in the objectives: computing focuses on exact evaluation, while learning emphasizes approximate prediction over a distribution of inputs. Consequently, a function may be hard to compute exactly but still be learnable in the PAC sense.

An illustrative example is **Simon's problem** [6], where the task is to find a hidden XOR mask s such that $f(x) = f(x \oplus s)$. Classically, solving this requires an exponential number of queries ($\Omega(2^{n/2})$), whereas a quantum algorithm can find s efficiently ($O(n)$) using the Fourier transform over $\mathbb{Z}_2^n$. This demonstrates a learning separation.

Recent works, such as [3], have further explored learning separations in physically motivated settings. They present scenarios where quantum learners can efficiently learn properties of quantum systems, while classical learners face exponential barriers, highlighting the practical relevance of learning separations.

Question 2: *What are the assumptions on the class of Hamiltonians in the references 2-3? Explain, from an high level point of view, the algorithm in reference 3 and the main idea of why it is guaranteed to work.*

References [2, 1] investigate the learnability of physical observables in quantum many-body systems. In both cases, the central task is to predict the value of an observable O on a quantum state $\rho_H(x)$, where $\rho_H(x)$ is the ground state of a local Hamiltonian $H(x)$ that depends on a classical bitstring input $x \in \{0,1\}^n$.

Assumptions on the Hamiltonians:

- Each $H(x)$ is a k**-local Hamiltonian** on n qubits — that is, it can be written as a sum of terms, each acting nontrivially on at most $k = O(1)$ qubits.

- The Hamiltonians are drawn from **structured families** of local models, and in hardness arguments are often instantiated as random 2D spin systems, such as random Heisenberg models or spin glasses.
- For every input x, the Hamiltonian $H(x)$ has a unique ground state $\rho_H(x)$ that is efficiently preparable on a quantum computer but hard to simulate classically.

These assumptions ensure that the map $x \mapsto \rho_H(x)$ defines a class of quantum states that are physically plausible, and conjectured to be classically hard to sample from or simulate [3]. The hardness arises from known complexity-theoretic obstructions to simulating ground states of local Hamiltonians in higher dimensions.

High-level description of the algorithm (Ref 3):

This work studies the task of learning an observable $O(\alpha) = \sum_{j=1}^{m} \alpha_j P_j$, where the P_j are known local Pauli observables and $\alpha \in \mathbb{R}^m$ is an unknown, sparse coefficient vector. The goal is to learn a hypothesis $\hat{O}$ such that

$$x \mapsto y = \mathrm{Tr}[\rho_H(x)\, O(\alpha)]$$

can be approximated well on unseen inputs.

The algorithm proceeds as follows:

1. Fix a set of m local Pauli strings $\{P_j\}$.
2. For each training input x_ℓ, prepare the corresponding ground state $\rho_H(x_\ell)$ on a quantum device.
3. Estimate each feature $\phi_j(x_\ell) = \mathrm{Tr}[\rho_H(x_\ell) P_j]$ by performing measurements on $\rho_H(x_\ell)$.
4. Obtain the label $y_\ell = \mathrm{Tr}[\rho_H(x_\ell) O(\alpha)] = \sum_j \alpha_j \phi_j(x_\ell)$.
5. Use a classical algorithm (e.g., LASSO) to recover an estimate $\hat{\alpha}$ from the feature-label pairs $(\phi(x_\ell), y_\ell)$.

Why it works:

- The observable $O(\alpha)$ is sparse in a known basis, and the features $\phi_j(x)$ can be measured efficiently using a quantum device.
- The ground states $\rho_H(x)$ obey locality and exhibit low correlation length, so the number of relevant observables needed to express O is small.
- Under reasonable assumptions, such as incoherence and bounded noise, sparse recovery techniques like LASSO provably succeed in recovering α with polynomial sample complexity.
- The learning task is PAC-learnable quantumly, but any classical learner without access to $\rho_H(x)$ would need to simulate ground states for exponentially large systems — believed to be computationally intractable.

Question 3: *What are the main differences between the algorithms in references 2 and 3? (Resources, sample complexity, time complexity)*

Both papers propose different quantum learning strategies for predicting observables on ground states of local Hamiltonians.

Ref 2: Introduces a learning framework based on the *classical shadow tomography* technique. This method uses randomized Pauli measurements to estimate the expectation values of many observables in parallel, without assuming sparsity. It is general-purpose and designed to apply broadly to local observables in many-body systems. In Appendix F, Theorem 4, the authors provide a formal sample complexity bound: to learn an observable $O = \sum_i O_i$ such that the squared prediction error is at most ε, it suffices to use

$$N = B^2 m^{\mathcal{O}(C/\varepsilon)},$$

samples, where m is the input dimension, B bounds the observable norm, and C bounds the gradient norm of $\mathrm{Tr}(O\rho(x))$ with respect to x. The classical training and prediction time are also upper bounded by $\mathcal{O}((n+L)N)$, where L is the number of local terms in O. These bounds are derived under assumptions of locality, bounded operator norm, and smooth parameter dependence. Lower bounds for certain classes of observables are also given in Appendix G via information-theoretic arguments.

Ref 3: Focuses on learning a *sparse linear observable* $O(\alpha) = \sum_j \alpha_j P_j$, where the $\{P_j\}$ are known local Pauli strings and α is sparse. The algorithm assumes access to ground states $\rho_H(x)$ and uses direct Pauli measurements to estimate features $\phi_j(x) = \mathrm{Tr}[\rho_H(x)P_j]$. These features are then used in a supervised regression task, where α is recovered via LASSO. Ref 3 provides a theoretical guarantee: to learn an ε-accurate hypothesis with failure probability δ, the sample complexity scales as

$$N = \log(n/\delta) \cdot 2^{\mathrm{polylog}(1/\varepsilon)},$$

which is logarithmic in n and δ^{-1}, but quasi-exponential in $1/\varepsilon$. The total training and prediction time is $\mathcal{O}(nN)$, assuming standard LASSO solvers. These guarantees hold under the assumption that the observable is geometrically local and bounded in norm.

Key differences:

- **Observable assumptions:** Ref 3 assumes sparsity in a known Pauli basis; Ref 2 makes no sparsity assumption.
- **Measurement model:** Ref 2 uses randomized Pauli measurements (classical shadows); Ref 3 uses fixed local Pauli measurements.
- **Learning objective:** Ref 2 targets general observable estimation; Ref 3 solves a supervised regression task for a sparse observable.

- **Sample complexity:** Both works provide formal upper bounds. Ref 2 has sample complexity $N = B^2 m^{\mathcal{O}(C/\varepsilon)}$; Ref 3 has $N = \log(n/\delta) \cdot 2^{\mathrm{polylog}(1/\varepsilon)}$.

- **Time complexity:** Both papers analyze runtime. Ref 2 gives bounds on post-processing time as $\mathcal{O}((n+L)B^2 m^{\mathcal{O}(C/\varepsilon)})$; Ref 3 gives total complexity $\mathcal{O}(n \log(n/\delta) \cdot 2^{\mathrm{polylog}(1/\varepsilon)})$.

2.3 Methodology and Experimental Setup

2.3.1 Overview of the Learning Task

The goal of this experiment is to learn a predictive model for the expectation value of an observable $O(\alpha)$ on quantum states of the form

$$\rho_H(x) = e^{-iHt}|x\rangle\langle x|e^{iHt},$$

where $x \in \{0,1\}^n$ is a computational basis input string, H is a known k-local Hamiltonian acting on n qubits, and $t > 0$ is a fixed evolution time. The observable $O(\alpha)$ is constructed as a sparse linear combination of local Pauli operators:

$$O(\alpha) = \sum_{j=1}^{k} \alpha_j P_j,$$

where each P_j is a 3-local Pauli string sampled from a pool of useful (to be explained later) candidates, and α_j are randomly drawn real coefficients.

We generate a dataset $\{(x_\ell, y_\ell)\}_{\ell=1}^N$ where each input x_ℓ is a uniformly random bitstring of length n, and the corresponding label is the exact observable expectation:

$$y_\ell = \mathrm{Tr}[\rho_H(x_\ell)\, O(\alpha)].$$

The machine learning task is to learn a function $h : \{0,1\}^n \to \mathbb{R}$ that approximates $x \mapsto y = \mathrm{Tr}[\rho_H(x)O(\alpha)]$ given access to the training data. Since α and H are fixed for each dataset, and the observable depends only on evolved states derived from classical bitstrings x, the problem becomes a supervised regression task over quantum-generated labels.

In this work, we study three types of input features:

- **Raw bitstrings** x (baseline classical model),

- **Exact features** $\phi_{\mathrm{ex}}(x) = (\mathrm{Tr}[\rho_H(x)P_j])_{j=1}^k$, computed via classical matrix simulation,

- **Noisy quantum features** $\phi_q(x)$ estimated from Trotterized circuits and finite-shot measurements on each P_j.

2.3.2 Hamiltonians and Evolution

We consider five distinct local Hamiltonians H acting on n qubits, each representing a physically motivated quantum many-body system. The models differ in their spatial structure, connectivity, and interaction types.

The implemented Hamiltonians are:

- **All-to-All Heisenberg**: A fully connected Heisenberg model with random symmetric couplings J_{ij}:
$$H = \sum_{i<j} J_{ij}(X_i X_j + Y_i Y_j + Z_i Z_j).$$
- **Chain Heisenberg**: A 1D Heisenberg chain with uniform nearest-neighbor interactions and optional periodic boundary conditions:
$$H = \sum_{i} J_{i,i+1}(X_i X_{i+1} + Y_i Y_{i+1} + Z_i Z_{i+1}).$$
- **Transverse-Field Ising Model (TFIM)**: A random ZZ-coupled model with uniform transverse field strength h:
$$H = \sum_{i<j} J_{ij} Z_i Z_j + h \sum_{i} X_i.$$
- **XY Model**: A variant of the Heisenberg model including only $X_i X_j + Y_i Y_j$ couplings:
$$H = \sum_{i<j} J_{ij}(X_i X_j + Y_i Y_j).$$
- **Parity Hamiltonian**: A non-interacting model given by the product of X on all qubits:
$$H = X_1 \otimes X_2 \otimes \cdots \otimes X_n.$$

In all models with couplings J_{ij}, the coefficients are drawn from a uniform distribution on $[0, 2)$ and symmetrized as $J_{ij} = J_{ji}$. The models vary in degree of connectivity and operator structure, enabling the study of learning performance across distinct Hamiltonian classes.

Once H is constructed, the corresponding unitary evolution operator $U = e^{-iHt}$ is computed via exact matrix exponentiation. The evolved quantum state is then
$$|\psi(x)\rangle = U|x\rangle, \quad \text{and} \quad \rho_H(x) = |\psi(x)\rangle\langle\psi(x)|.$$

This procedure is repeated for every input x to generate the labels and exact features.

2.3.3 Observable Construction

In each experiment, the target observable $O(\alpha)$ is constructed as a sparse linear combination of k randomly chosen local Pauli operators:

$$O(\alpha) = \sum_{j=1}^{k} \alpha_j P_j,$$

where each P_j is a 3-local Pauli string acting nontrivially on three randomly selected qubits, and $\alpha_j \in [0, 2)$ is a coefficient drawn uniformly at random. All P_j are Hermitian and drawn from the tensor product set $\{X, Y, Z\}^{\otimes 3}$.

To ensure the Pauli terms are informative and contribute non-trivially to the learning task, we apply a variance-based filtering method. Specifically, a candidate Pauli string P is retained only if its expectation values across the dataset exhibit variance above a threshold:

$$\mathrm{Var}\left(\mathrm{Tr}[\rho_H(x_\ell)P]\right) \geq \theta,$$

where θ is a small fixed cutoff and the variance is estimated over the input samples $\{x_\ell\}$.

This filtering step ensures that only Pauli terms with non-constant response over the dataset are included in $O(\alpha)$, avoiding degenerate observables that would be difficult to learn. The accepted Pauli strings and their coefficients define a unique, fixed observable for the entire dataset, used to compute the exact labels

$$y_\ell = \mathrm{Tr}[\rho_H(x_\ell)\, O(\alpha)].$$

The same Pauli terms $\{P_j\}$ are also used as features during training, both in the exact simulation case and in the Trotterized circuit case. This guarantees alignment between the observable and the input feature representation.

2.3.4 Feature Extraction

For each input $x \in \{0, 1\}^n$ and a fixed observable $O(\alpha) = \sum_j \alpha_j P_j$, we extract two types of features: exact features computed classically, and noisy features obtained from simulated quantum measurements.

Exact Features. Given access to the full matrix representation of $U = e^{-iHt}$, we compute the evolved state $|\psi(x)\rangle = U|x\rangle$ explicitly. For each Pauli term P_j used in the observable, we compute the expectation value

$$\phi_j^{\mathrm{ex}}(x) = \langle \psi(x)|P_j|\psi(x)\rangle.$$

These values are collected into a feature vector $\phi_{\mathrm{ex}}(x) \in \mathbb{R}^k$. This procedure is repeated for each sample x in the dataset, yielding an exact feature matrix $\Phi_{\mathrm{ex}} \in \mathbb{R}^{N \times k}$.

Noisy Quantum Features. To simulate realistic quantum feature access, we construct a Trotterized quantum circuit that approximates $U = e^{-iHt}$ using first-order Trotter decomposition with a fixed number of steps s:

$$U \approx \left(\prod_{\ell} e^{-ih_\ell \Delta t} \right)^s, \quad \Delta t = \frac{t}{s},$$

where each h_ℓ is a local term in H. For each input x, we prepare $|x\rangle$, apply the Trotterized circuit, and measure each Pauli term P_j in its eigenbasis using R measurement shots. For each P_j, the feature is estimated as the average parity of outcomes:

$$\phi_j^q(x) = \frac{1}{R} \sum_{r=1}^{R} \text{parity}_{P_j}(x_r),$$

where $\text{parity}_{P_j}(x_r) \in \{-1, +1\}$ depends on the measurement outcome in the rotated basis of P_j.

This yields a noisy feature matrix $\Phi_q \in \mathbb{R}^{N \times k}$, which approximates Φ_{ex} up to Trotter and sampling error. To control fidelity, we perform a sweep over (s, R) pairs and select the minimal configuration that achieves average correlation above a threshold (Pearson correlation ≥ 0.95) with the exact features on a subset of the data.

2.3.5 Learning Models

To evaluate the predictive power of each feature representation, we train and test two standard regression models: LASSO for sparse linear recovery and a feedforward neural network for general nonlinear regression.

LASSO Regression. Given input features $\phi(x) \in \mathbb{R}^k$ and labels $y = \text{Tr}[\rho_H(x) O(\alpha)]$, we fit a sparse linear model via ℓ_1-regularized least squares:

$$\min_{w \in \mathbb{R}^k} \frac{1}{N} \sum_{\ell=1}^{N} (w \cdot \phi(x_\ell) - y_\ell)^2 + \lambda \|w\|_1.$$

We use the built-in `LassoCV` model from `scikit-learn`, which performs internal k-fold cross-validation to select the regularization parameter λ from a predefined grid. This model is applied separately to each of the three feature types: exact features Φ_{ex}, noisy features Φ_q, and raw bitstrings x.

Neural Network. We also train a multilayer perceptron regressor using the `MLPRegressor` class from `scikit-learn`. The architecture consists of a single hidden layer with ReLU activation and a fixed number of units, optimized via the Adam optimizer. The model is trained to minimize the squared error loss between predictions and labels. Early stopping and tolerance-based convergence criteria are used to ensure stable training.

Training and Evaluation. For each dataset, we perform an 80/20 train-test split. Models are trained exclusively on the training set, and evaluated on both train and test data using two metrics:

- **Mean squared error (MSE):** $\mathrm{MSE} = \frac{1}{N}\sum_{\ell}(h(x_{\ell}) - y_{\ell})^2$,
- **Coefficient of determination (R^2):** $R^2 = 1 - \frac{\mathrm{MSE}}{\mathrm{Var}(y)}$.

These metrics are computed for all models on all three feature representations.

2.3.6 Implementation Details

All experiments were implemented in Python using `NumPy`, `SciPy`, and `scikit-learn` for numerical linear algebra, optimization, and regression. Quantum circuits were simulated using `qiskit`'s `AerSimulator` backend. The full experimental pipeline was executed in a Jupyter notebook.

The reference setting used across all experiments is:

- Number of qubits: $n = 10$,
- Dataset size: $N = 100$ input bitstrings $x \in \{0,1\}^{10}$,
- Number of Pauli terms in the observable: $k = 3$,
- Total evolution time: $t = 1.0$,
- Variance threshold for Pauli filtering: $\theta = 10^{-3}$.

Each observable $O(\alpha)$ is generated independently per experiment using k 3-local Pauli strings selected from random qubit indices and Pauli labels $\{X, Y, Z\}$. The coefficients α_j are drawn uniformly from $[0, 2)$.

All models are trained on an 80% training split and evaluated on both train and test sets. LASSO is fit via cross-validated `LassoCV`, and neural networks are implemented with one hidden layer of size 50 using `MLPRegressor`.

2.4 Results

We report the results of the regression models applied to each Hamiltonian and feature type. For every setting, we evaluate test set performance using both MSE and R^2. Feature types include raw input bitstrings, exact Pauli expectations, and noisy expectations extracted via Trotterized quantum simulation. Figure 1 shows the test R^2 scores across all Hamiltonians and model/feature combinations, excluding raw-bit features. Figure 2 displays the predicted vs. true observable values across all Hamiltonians, using both exact and noisy features. Table 1 summarizes the selected Trotter parameters and test performance for all models and Hamiltonians.

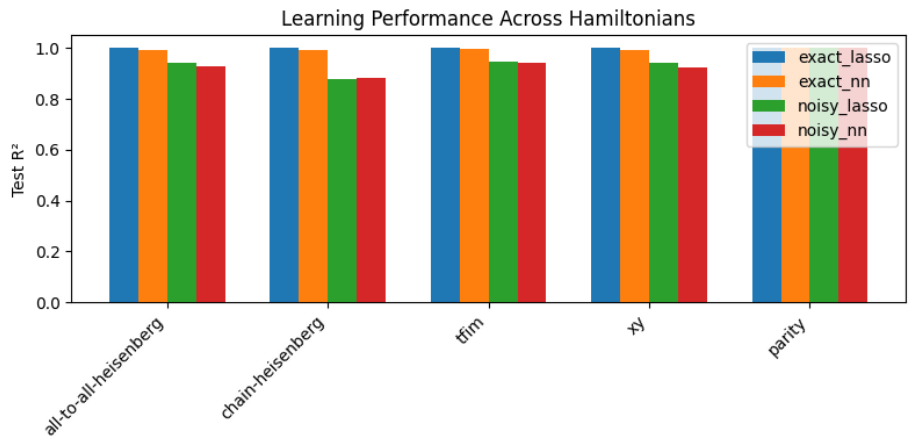

Figure 1: Test R^2 score across Hamiltonians. Each bar corresponds to a specific regressor and feature type.

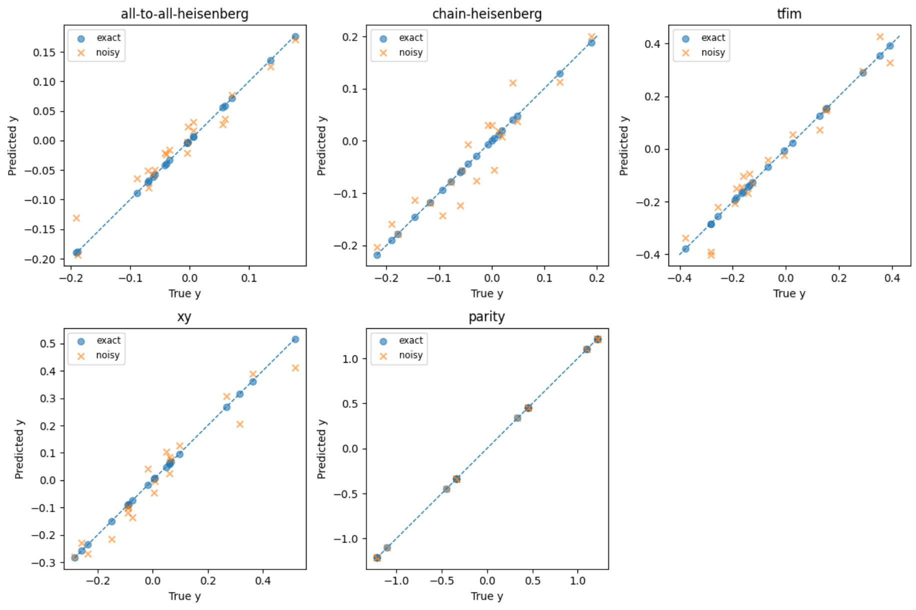

Figure 2: Scatter plots of predicted vs. true $y = \mathrm{Tr}[\rho_H(x)O]$ values. Blue circles use exact features; orange crosses use noisy Trotter features.

2.5 Comparison Across Hamiltonians and Discussion

We now compare the learning performance across different Hamiltonians and interpret the results in light of the task's physical structure and feature accessi-

Table 1: Detailed test performance across Hamiltonians. Each entry reports either Trotter parameters or model evaluation on the test set.

Metric / Model	**All-to-All**	**Chain**	**TFIM**	**XY**	**Parity**
Steps	254	128	16	128	2
Shots	10000	20000	10000	5000	10
MSE (LASSO)					
Raw bits	0.0080	0.0110	0.0489	0.0442	0.7719
Exact features	0.0000	0.0000	0.0000	0.0000	0.0000
Noisy features	0.0004	0.0012	0.0026	0.0025	0.0000
MSE (NN)					
Raw bits	0.0089	0.0579	0.0676	0.0604	0.6340
Exact features	0.0001	0.0001	0.0001	0.0004	0.0002
Noisy features	0.0006	0.0012	0.0027	0.0032	0.0001
R^2 **(LASSO)**					
Raw bits	-0.0253	-0.0823	-0.0354	-0.0595	-0.0158
Exact features	1.0000	1.0000	1.0000	1.0000	1.0000
Noisy features	0.9433	0.8800	0.9455	0.9411	1.0000
R^2 **(NN)**					
Raw bits	-0.1327	-4.7154	-0.4324	-0.4461	0.1657
Exact features	0.9905	0.9928	0.9987	0.9915	0.9997
Noisy features	0.9268	0.8807	0.9430	0.9229	0.9998

bility. Our goal is to understand how the complexity of the quantum dynamics and the feature extraction method affect model generalization.

Quantum vs Classical Features. Across all Hamiltonians, regression models trained on exact quantum features $\phi_{\mathrm{ex}}(x)$ consistently achieved near-perfect accuracy ($R^2 \approx 1.0$). This demonstrates that the map $x \mapsto \mathrm{Tr}[\rho_H(x)O]$ is learnable when given access to physically meaningful observables. In contrast, models trained on raw bitstrings—without access to evolved quantum states—performed poorly, with negative or near-zero R^2.

Effect of Trotterization and Noise. When features are extracted via noisy quantum circuits, performance remains high but degrades slightly depending on the Hamiltonian. For example, all-to-all Heisenberg required 254 Trotter steps and 10,000 shots to reach the target fidelity threshold, whereas TFIM and parity reached sufficient accuracy with far fewer resources. This suggests that the effective circuit depth required to extract useful features varies with the entanglement structure and locality of the Hamiltonian.

Comparison Across Hamiltonians. While exact-feature models achieved similar performance across all systems, the quality of noisy features and the difficulty of learning varied. The chain-Heisenberg and TFIM models exhibited greater sensitivity to measurement noise, requiring more Trotter steps or shots to maintain performance. The parity Hamiltonian, in contrast, was exceptionally easy to learn: both noisy and exact models achieved perfect predictions even with minimal circuit depth. This aligns with the observable's global, highly symmetric structure and low expressivity.

Sparsity and Interpretability. LASSO regression consistently recovered sparse models with three active features matching the ground truth observable construction. Even in noisy conditions, the learned coefficients were close to the exact ones, confirming the reliability of sparse recovery techniques in quantum regression. This also demonstrates that the Pauli feature space is not only expressive but also interpretable.

2.6 Formal Argument on Learning Separation (9+)

Claim: There exists a task that can be efficiently learned by a quantum learner with access to evolved quantum states and local measurements, but cannot be learned by any efficient classical algorithm under standard complexity assumptions.

Problem Setup. Consider the function class

$$\mathcal{F}_{H,O} = \left\{ f_\alpha(x) = \mathrm{Tr}[\rho_H(x) O(\alpha)] \mid \alpha \in \mathbb{R}^k,\ \text{sparse} \right\},$$

where $x \in \{0,1\}^n$ is a classical input bitstring, $\rho_H(x) = e^{-iHt}|x\rangle\langle x|e^{iHt}$ is a time-evolved quantum state under a fixed local Hamiltonian H, and $O(\alpha) = \sum_{j=1}^{k} \alpha_j P_j$ is a sparse linear observable, with known Pauli terms $\{P_j\}$ and unknown sparse coefficients α.

Quantum Learnability. As shown in [7], a quantum learner can efficiently extract features $\phi_j(x) = \mathrm{Tr}[\rho_H(x)P_j]$ using local measurements, and apply sparse linear regression to recover an accurate hypothesis $\hat{\alpha}$. The overall sample complexity scales polynomially in n and $1/\varepsilon$, and the total runtime is efficient assuming access to a quantum circuit that prepares $\rho_H(x)$.

Classical Intractability. Suppose there existed a classical learner that PAC-learns $f_\alpha(x)$ with sample and time complexity polynomial in n and $1/\varepsilon$. Then, by standard results in computational learning theory [8], there exists a classical circuit of polynomial size that approximates $f_\alpha(x)$ for any input x. But if H is chosen to simulate a BQP circuit, then evaluating $f_\alpha(x)$ encodes a BQP-complete decision problem.

Thus, the existence of an efficient classical learner implies that the BQP-complete language associated with f_α is in P/poly, which contradicts the widely believed complexity assumption

$$\mathbf{BQP} \not\subseteq \mathbf{P/poly}.$$

Example Hamiltonian. An explicit example of such a Hamiltonian H is constructed in [7] using Feynman's circuit-to-Hamiltonian mapping. For any BQP circuit U, the corresponding H is a geometrically local Hamiltonian such that the time evolution e^{-iHt} effectively simulates U. An observable of the form $O = Z \otimes I^{\otimes(n-1)}$ can be used to extract the output of the computation. Learning the function $x \mapsto \mathrm{Tr}[\rho_H(x)O]$ is thus equivalent to learning the outcome of a quantum computation.

Conclusion. There exists a quantum learning task that is efficiently PAC-learnable by a quantum algorithm, but provably not learnable by any classical algorithm unless BQP $\subseteq$ P/poly. This formalizes the notion of a learning separation in the observable prediction task.

2.7 Conclusion

In this project, we investigated the problem of predicting the expectation value of an unknown observable on time-evolved quantum states. We implemented a full learning pipeline including Hamiltonian construction, observable generation, feature extraction, and supervised learning using LASSO and neural networks.

Across all Hamiltonians, we observed that models trained on quantum features—whether computed exactly or extracted via Trotterized circuits—achieve high generalization performance, while models trained on raw bitstrings fail to generalize. This constitutes an empirical learning separation: good features exist, but are inaccessible to purely classical learners. This separation is not just a practical limitation of raw-input models. The relevant quantum features depend on the evolved state $\rho_H(x) = e^{-iHt}|x\rangle\langle x|e^{iHt}$, and simulating this evolution is classically intractable for generic local Hamiltonians. Thus, under standard complexity assumptions, no efficient classical learner can compute the necessary features to learn the task—whereas a quantum learner can access them via direct measurement.

References

[1] L. Lewis, H.-Y. Huang, V. T. Tran, S. Lehner, R. Kueng, and J. Preskill, "Improved machine learning algorithm for predicting ground state properties," *Nature Communications*, vol. 15, no. 1, p. 895, 2024.

[2] H.-Y. Huang, R. Kueng, G. Torlai, V. V. Albert, and J. Preskill, "Provably efficient machine learning for quantum many-body problems," *Science*, vol. 377, Sept. 2022.

[3] C. Gyurik and V. Dunjko, "Exponential separations between classical and quantum learners," 2024.

[4] L. G. Valiant, "A theory of the learnable," *Commun. ACM*, vol. 27, p. 1134–1142, Nov. 1984.

[5] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, p. 1484–1509, Oct. 1997.

[6] D. Simon, "On the power of quantum computation," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 116–123, 1994.

[7] R. Molteni, C. Gyurik, and V. Dunjko, "Exponential quantum advantages in learning quantum observables from classical data," 2024.

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