Outline

This project consists of two parts. Part 1 contains 8 mandatory (and one extra credit) questions - covering theory of both quantum machine learning and strongly correlated systems **(you have to answer the questions to both topics)** - and it comprises 40% of the final mark for the project. For part 2 you choose **one** out of 8 topics **(i.e., unlike the theory part you can choose between either a project in quantum machine learning or strongly correlated systems)** and it comprises 60% of the final mark for the project.

Each project in part 2 has a list of "Tasks and Objectives" together with the required content of the report. Correctly carrying out the listed tasks and objectives together with an appropriate report will result in a passing grade that can go up to an 8,5. In order to obtain an even higher grade, you have to carry out the extra objective/task listed at the end of each of the projects.

At the end of this project **you will have to turn in a report**. The report should contain your solutions to the mandatory questions in part 1 and a write-up on the topic of your choice in part 2. This write up should be between 5-10 pages long and be concise and to the point. Also, please do not forget to submit your code for part 2, as this will be taken into account when grading.

The deadline for both parts of the project is 19th of May 2025.

Part 1

Strongly correlated systems

1. (a) Write down the commutation relations for Fermionic creation and annihilation operators.
   (b) Write down a transformation of fermionic creation and annihilation operators into Majorana operators.
   (c) Write down the commutation relations of the resulting Majorana operators.
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.
3. Write down the Hamiltonian for the Fermionic Hubbard model on a 1-dimensional chain.
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)?

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

   (a) the Hubbard model

   (b) the Heisenberg model

   (c) the Electronic structure problem

Quantum machine learning

1. Describe the implicit ("the quantum kernel estimator") and explicit ("the quantum variational classifier") of quantum variational SVMs. (Hint: `https://arxiv.org/abs/1804.11326`). Feel free to describe either the generic circuits, or the particular ones used in the provided reference.

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

   (b) What is the relationship between the two?

   (c) When does a kernel have a corresponding feature map?

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

4. (Hard question, extra credit) 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.

Part 2

Strongly correlated systems

Topic 2.A. Randomized measurements and classical shadows.
In this project you will implement classical shadows using randomized Pauli measurements on a quantum state.

Literature: The relevant parts are section I and II of the paper (the rest is about interesting applications, but is not required for this mini-project). For more information and intuition about classical shadows, you could watch the first hour or so from the youtube video included, a tutorial given by Richard Küng at QIP.

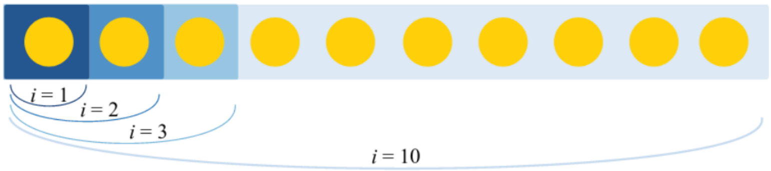
- `https://arxiv.org/abs/2203.11374`
- `https://www.youtube.com/watch?v=FXdJoJ0qcZY`

Objectives and tasks:

- Learn about the randomized measurement toolbox. Answer the following questions in your report:
  - What are the motivations for having a classical description of your quantum state?
  - Describe the data acquisition phase of the randomized measurement protocol. How do you construct a classical shadow? What do you actually store on your classical computer?
  - Describe the postprocessing protocol. How do you determine the expectation value of an observable?
  - What are the pros and cons of this measurement scheme compared to other schemes that we saw in the course? Explain how this depends on the observables you may want to measure: how many measurements do you need?
- Provide an implementation of the "vignette" application of section II.D (estimating purities and Rényi entropy of the time evolved Néel state with the XY-Hamiltonian with classical shadows).
  - Write a function that generates the circuit preparing the Néel state $|\psi\rangle$.
  - Using a first-order Trotter approximation, write a function that implements time-evolution $|\psi(t)\rangle = e^{-iH_{XY}t}|\psi\rangle$[1]. You will have to choose how many trotter steps you use.

[1] Note that you can write $H_{XY} = \sum_{i<j} J_{ij}\left(\sigma_i^+\sigma_j^- + \text{h.c.}\right) = \sum_{i<j} J_{ij}\frac{X_iX_j+Y_iY_j}{2}$ (prove this in your report!).

- Write a function that, using a state-vector simulation, estimates exactly the second Rényi entropies $S(\rho_A) = -\log_2 \mathrm{Tr}\left(\rho_A^2\right)$ of subsystems A of $|\psi(t)\rangle$, as in Fig. 3b in the paper. The x-axis in the figure indicates the subsystems $A = [1 \rightarrow i]$:



- Write a function that implements random **Pauli** measurements, given a circuit.
- Use this function to compute a classical shadow of $|\psi(t)\rangle$ for different $t \in [0, 1]$.
- Using Eq. (3) or Eq. (4) in the paper, write a function that computes the second Rényi entropy given a classical shadow.
- Compute the entropies of the subsystems of $|\psi(t)\rangle$ with the above function. Compare these with the ones obtained by the state-vector simulation.
- Try different amounts of random measurement settings M and amount of repetitions per measurement K; how dependent is your result on these parameters?

The report should contain:

1. A brief description of the randomized measurements and classical shadows, and its pros and cons compared to regular measurement schemes.

2. An implementation (preferably in a jupyter notebook) of time evolution of the H_{XY} Hamiltonian on the Néel state, and a comparison of Rényi entropies for the exact case and with classical shadows using the randomized measurements protocol.

For 8+: An investigation into the dependence of your classical shadows on the amount of measurement settings M and repetitions per measurement setting K.
For 9+: Implement a noise model using density matrix simulators, and investigate noise for both the exact Rényi entropies and the ones obtained from classical shadows. Using noise, you should be able to reproduce closely the dotted line (noise simulation) of Fig. 3b.

Topic 2.B. Hamiltonian variational ansatze.
In this project you will implement and optimize the Hamiltonian variational ansatz of Wecker, Hastings and Troyer for the Hubbard model on a 2 × 2 lattice

Literature:

- `https://arxiv.org/abs/1507.08969`.

Objectives and tasks:

- Learn and understand the Hamiltonian variational ansatz.
  - What is the problem that is being solved?
  - What is the motivation for this choice of variational ansatz?
  - For which systems do the authors suggest this ansatz might be preferable?
- Understand the Hubbard model on a square lattice.
  - How many nearest neighbours does each site connect to on an infinite lattice?
  - How many nearest neighbours does each site connect to on a 2 × 2 lattice?
  - Write h_h, h_v, and h_U (defined below Eq. 2) in terms of creation and annihilation operators for a 2 × 2 lattice.
  - How many qubits are required to represent the Hubbard model on a single site?
  - What is the non-interacting Hamiltonian for this system?
- Provide an implementation of the Hamiltonian variational ansatz for the Hubbard model
  - Write a function to generate hermitian terms in h_h, h_v and h_U.
  - Write a function to generate the non-interacting and interacting Hamiltonians.
  - Write a function to generate circuits implementing $e^{i\theta h}$ terms given the above. Remember to leave θ a free parameter, and transform from fermions into qubits.
  - Write a function to generate the Hamiltonian variational ansatz using the above sub-functions.
  - Write a function to generate the non-interacting starting state using the above.

  - Implement the VQE using SciPy and the above, measure the energy at $t = 1$ and $U = 2$, and compare performance to the result found in ArXiv:1507.08969 as a function of the step size S. Be careful with optimization!

The report should contain:

1. A description of the Hamiltonian and system being studied.
2. A description and explanation of the algorithm and quantum-classical hybrid scheme.
3. An implementation of the ansatz and evaluation of performance (preferably in a jupyter notebook)
4. Optimize the VQE without noise. Then test its robustness with respect to circuit-level noise and/or sampling noise.

For 8+: Analyze the robustness of the VQE with respect to circuit or sampling noise during the optimization step.
For 10: Make a thorough, realistic, analysis of the effects of various sources of imperfection in the performance of the algorithm.

Topic 2.C. The Meta-Variational Quantum eigensolver.
In this project you will implement the Meta-Variational Quantum Eigensolver of Cervera-Lierta *et al.* to solve the ground state problem of a family of XXZ spin chain Hamiltonians.

Literature:

- `https://doi.org/10.1103/PRXQuantum.2.020329`.

Objectives and tasks:

- Learn and understand Meta-VQE
  - What is the goal of Meta-VQE? What are its inputs and outputs?
  - How does Meta-VQE differ from a standard VQE? What is the connection with machine learning?
  - What are the potential advantages and uses of Meta-VQE?
  - How do the encoding and processing steps work? What is the loss function?
  - What is Opt-Meta-VQE?
- Provide an implementation of Meta-VQE on the one-dimensional XXZ model, and compare it to VQE and Opt-Meta-VQE:
  - Write a function to generate the Hamiltonian of the XXZ chain $H(\lambda, \Delta)$ for a given system size n and value of parameters λ and Δ).
  - Write a generator that yields the gates for the processing step given a vector of parameters $\vec{\theta}$.
  - Write a generator that yields the gates for the encoding step given Δ, a vector of parameters $\vec{\varphi}$, and the function f (you can reuse the processing step generator!).
  - Combine these to write a function that returns the expectation value of the Hamiltonian on the ansatz state.
  - Construct a training set and the relative loss function. Optimize the parametrized model using an optimizer of your choice. [*note*: as the optimization might be slow for these many parameters, you can use a smaller system size n than in the paper. $n = 8$ or 10 should be a good compromise]
  - Construct a standard VQE with the same parametrized quantum circuit structure.

  - Optimize the VQE with the relevant initial parameters, to obtain both VQE and Opt-Meta-VQE results.
  - Reproduce the plot in Fig.2 of the linked paper (you can skip the GA-VQE and Opt-GA-VQE).

The report should contain:

1. A brief description of the Meta-VQE, underlying its specific characteristics and its potential advantages.
2. An implementation (preferably in a Jupyter notebook) of the Meta-VQE on the 1-dimensional XXZ model.
3. A discussion of the results, including any personal observations.

For 8+: Study the convergence of Meta-VQE, Opt-Meta-VQE and standard VQE on the same training set. Compare the total number of quantum circuit evaluations needed in the optimization process (ignoring sampling noise) and the relative accuracies.
For 9+: Go beyond the results in the paper by treating a research question of your choice. **For example**, study the role of the encoding function $f(\Delta, \phi)$ and of the circuit ansatz on accuracy and efficiency of the protocol; **or** compare Meta-VQE and VQE in a realistic setting with sampling noise in the optimization; **or** implement another idea of your choice (you can always discuss it with the TAs).

Quantum machine learning

Topic 2.D. (*Expressivity of the generating set*). Recently research has pointed to the encoding of data into a parameterised quantum circuit as being a key part of successful quantum machine learning, in this project you will attempt to characterise its importance.

In this project, you will assess the importance of the type, quantity and variety of encoding layers on the expressivity of parameterised quantum circuits, and study how different choices make parametrized quantum circuits (a.k.a. quantum nerual networks) different from classical neural networks.

You will test this by using a parameterised quantum circuit with fixed input encoding [2] to generate a synthetic data set, then experiment with how expressive a parameterised quantum circuit must be to learn this data set. Repeating these experiments for multiple different architectures of the generating parameterised quantum circuits will allow for comparisons and orderings.

You may also experiment with how neural networks perform on the same data set, experimenting with different architectures to characterise how modifications to the generating architecture impact the need for more complicated neural networks.

Three possible choices of inquiry:

- Generators v. generators: understand the role of the spectrum of generators of rotations in the functions the circuits generate. Select two different classes of generators, and corresponding PQC-based models A and B. For a fixed number of qubits (2-3) Generate artificial data with A, learn with B and vice versa. study the quality of fit, and generalization performance, and how many parameters (depth) are needed for good fits (training error) (but what is the generalization/test error)? Discuss.

- Depth v. width: understand the importance of data-reuploading. Select two qubit numbers, say 1 and 3, and generate corresponding PQC-based models A and B with those qubit numbers, defined for the same data dimension (use data reuploading). Generate artificial data with A, learn with B and vice versa. study the quality of fit, and generalization performance, and how much depth is needed (in the shallower circuit) for good fits (training error) (but what is the generalization/test error)? Discuss.

- QNNs v. NNs: qualitative comparison of NN functions vs QNN functions. Consider two models A and B, where A is PQC-based, and B is a neural network, with the same dimension of inputs and outputs. Generate artificial data

[2] an input encoding is the scheme used to input data into a parameterised quantum circuit, it is normally composed of rotation gates that rotate the state by some amount dependent on the size of the input angle x_i

with A, learn with B and vice versa. study the quality of fit, and generalization performance, and how parameter numbers and depth influence quality of fits (training error) (but what is the generalization/test error)? Discuss. Can NNs learn all QNNs can generate as efficiently and vice versa? What would you expect.

Literature: We have provided a number of sources to assist you. Papers 1 and 2 address how a quantum computer can be used for machine learning by using parameterised quantum circuits. Sources 3 and 4 are theoretical works that describe how different encoding layers may affect generalisation, these may provide inspiration for what types of encoding layers you may wish to study. Sources 5 and 6 are again practical implementations of quantum machine learning.

1. `https://arxiv.org/pdf/1804.11326.pdf`
2. `https://arxiv.org/pdf/1804.00633.pdf`
3. `https://journals.aps.org/pra/abstract/10.1103/PhysRevA.103.032430`
4. `https://arxiv.org/pdf/2106.03880.pdf`
5. `https://www.tensorflow.org/quantum/tutorials/mnist`
6. `https://pennylane.ai/qml/demos/tutorial_variational_classifier.html`

Objectives and tasks:

- Generate artificial data from a quantum circuit by fixing the relevant hyperparameters and parameters.

  Remark. *I.e., analogous to the artificial data in* `arxiv.org/pdf/1804.11326.pdf`*.*

- Experiment with the learning performance of different parameterized quantum circuits (or neural networks) on learning this data set, varying their architecture.

- Repeat this experimentation for different artificial datasets that originate from different hyperparameters.

- Compare the performance of several different encoding layers. How do changes to encoding affect training? How do changes in the test set affect this? Do the two effects cancel each other out?

The report should contain:

1. An explanation of how the artificial data is generated.

2. An explanation of the code that generates the artificial data and trains a parameterized quantum circuit model/neural net to classify this data.
3. Comparison of different encoding schemes/architectures on the training task.

For an 8+: Experiment with using the encoding as a *regularization technique*.

Topic 2.E. (*Learning graph invariants with PQCs*). Variational quantum algorithms are candidates for advantage on near-term quantum hardware. The choice of ansatz to solve a specific problem plays an important role in trainibility and performance of the algorithm. So we are constantly looking for more informed ansatzes and the ansatz is used to define the learning model. One way of embedding bias, that is, prior knowledge that will enable your model to learn faster, in your model is through exploiting the symmetries of the problem. In this mini project, you are going to use a particular ansatz, an equivariant ansatz, in a data re-uploading scheme to learn a permutation invariant property of graphs, namely connectivity. A graph is said to be connected if there is a path between every pair of vertices. Then you are going to discuss the expressivity of the quantum model and compare its performance to a classical neural network.

Literature:

1. `https://arxiv.org/pdf/2210.07980.pdf`
2. `https://arxiv.org/pdf/2112.05261.pdf`

Theory questions:

Given an arbitrary matrix $M \in U(N)$, which in our case will be defined in next section and is in fact an equivariant encoding of a graph, and a sequence of matrices $\{W_i\} \in U(N)$, we define the k-th re-uploading circuit to be $U_k = \prod_{i=1}^{k}(W_i M)$ and the function family realized by this circuit to be $h_k(M) = \langle 0|U_k|0\rangle$ [3] .

1. By inserting resolutions of identity[4] matrix show that this function family is k-th degree multivariate polynomials of the entries of the matrix M of the form:

$$h_k(M) = \sum_{i_1=0}^{n-1} \cdots \sum_{i_k=0}^{n-1} m_{i_1} \times \ldots \times m_{i_k} \times w_{i_1}^{(1)} \ldots w_{i_k}^{(k)}$$

   where i_l are multi-indices selecting an entry of a matrix, $n = N^2$ and $w_{i_l}^{(j)}$ are the i_l entry of the matrix W_j.

2. Show that there are $n * k$ many parameters to set the coefficients $w_{i_l}^{(j)}$ while having $\binom{n+k-1}{k}$ many equations. what does this tell us about the expressivity of this model? Note that we would consider this model to be maximally expressive if one can get all the coefficients.

[3] To be precise, this $h_k(M)$ is not fully defined but its absolute value is. However, we still use the former format without absolute values as it simplifies the construction in the next part.

[4] $\mathbb{I} = \sum_{i=0}^{2^m-1} |i\rangle\langle i|$ represents the identity operator in an m-qubit Hilbert space

3. Explain what a permutation equivariant ansatz is and how it is connected to graph symmetries. You can use the references to answer this question.

Programming problems: You will be using the data re-uploading quantum model with specific data encoding and variational layers to learn the connectivity problem. Then compare its performance to a classical model to draw your own conclusions about the expressivity of the quantum model.

Data Generation:
Use a graph generation algorithm to create a set of random graphs, specify the graphs by their adjacency matrix. To each adjacency matrix, assign a label by whether the graph is connected or disconnected. If the graph is connected, label it as 0; if it is disconnected, label it as 1. This will give you the training set $T = \{(A_i, y_i)\}_{i=1}^{N}$ where A_i are the adjacency matrices and $y_i = \{0, 1\}$.

Learning with a quantum and a classical model:

1. Quantum model: The quantum circuit itself is constructed from L layers, each consisting of a data encoding circuit block M and a trainable circuit block $W_i(\theta_i)$ controlled by the parameters θ_i.

   a- Let $M = e^{iH_G}$ where $H_G = \sum_{(i,j)\in\epsilon} \sigma_z^i \sigma_z^j$ is the Ising Hamiltonian
   For every adjacency matrix A of the graph G in the training data use the Ising Hamiltonian to encode the graphs into quantum circuits .

   b- For variational parts of the circuit, take $W_j(\theta_j) = \bigotimes_{i=1}^{n} R_x(\theta_j)$ where θ_j is the trainable parameter in the layer j. Here we have one parameter to train per layer.

   c- We can get more parameters by modifying the variational ansatz to $W_j(\theta_j^{(1)}, \theta_j^{(2)}) = \bigotimes_{l=1}^{n} R_x(\theta_j^{(1)}) R_y(\theta_j^{(2)})$. This helps with the expressivity of the model.

   d- **Theory question:** Show that this circuit ansatz is permutation equivariant.

   e- Train the model for different numbers of re-uploading layers for both variational ansatz. compare the performance.

   f- You can also allow parametrization in the data encoding layers. allow $M_j = e^{i\gamma_j H_G}$ where $H_G = \sum_{(i,j)\in\epsilon} \sigma_z^i \sigma_z^j$ and γ_j is the trainable parameter in layer j. Compare the performance.

2. Classical model: Define a classical learning model where the hypothesis family are multivariate k degree polynomial functions:

$$h_k(M) = \sum_{i_1=0}^{n-1} \cdots \sum_{i_k=0}^{n-1} m_{i_1} \times \ldots \times m_{i_k} \times w_{i_1 \ldots i_k}$$

where the coefficients $w_{i_1 \ldots i_k}$ are completely free. Train the model and optimize the parameters $w_{i_1 \ldots i_k}$ to learn connectivity of the graphs. Discuss the performance of the model and compare it to the quantum model with k layers.

The report should contain:

1. Answer to the questions in the theory part.

2. Report of the implementation of the quantum model with all the two suggested variational layers. Comparison of them to each other. Comparison of their performance to the case where you allow parametrization in the data encoding layers. Answers to the theory question (proving that the ansatz is permutation equivariant)

3. Report of the implementation of the classical model and its comparison to the quantum model.

4. **For grade 8+** We suggested $W_j = \bigotimes_{l=1}^{n} R_x(\theta_j^{(1)}) R_y(\theta_j^{(2)})$ for variational layers. This design, allows two parameters per qubit per layer to be trained. We can have three parameters by having $W_j = \bigotimes_{l=1}^{n} R_x(\theta_j^{(1)}) R_y(\theta_j^{(2)}) R_z(\theta_j^{(3)})$. Adding more parameters help with expressivity of the model. Design another equivariant variational ansatz with more trainable parameters in each layer to get better performance. Show that your ansatz is equivariant. Train the model and compare the performance.

Topic 2.F. (*Towards learning separation*). In this project you will use a quantum algorithm to solve a ML problem related to quantum many body physics. Specifically, the task is to learn to predict from data expectation values of unknown observables on time evolved quantum states under a known Hamiltonian. Alternatively, you can consider the task of learning to predict expectation values on ground states of a class of local Hamiltonians. As it will be discussed in the lectures, for a special class of Hamiltonians one could show that those tasks exhibit a provable learning separation with respect to classical algorithms. In this project, we will focus on a class of physically motivated Hamiltonians.

Literature:

1. `https://arxiv.org/pdf/2306.16028.pdf`
2. `https://www.science.org/doi/full/10.1126/science.abk3333`
3. `https://arxiv.org/pdf/2301.13169.pdf`
4. `https://arxiv.org/abs/1912.13146`
5. `https://arxiv.org/abs/2205.06278`

Questions:

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

Implementation: There are two possible variants of the learning problem we can consider: in **case A** the task is to predict observables on time evolved quantum states, in **case B** we will be dealing with ground states instead. Notice that **case A** will be easier to implement as the quantum algorithm will just have to perform a fixed time evolution on input states. The implementation of **case B** will be instead more involved, as the quantum algorithm will have to find the ground state for the Hamiltonians associated at each data point. However, **case B** allows for a direct comparison with relevant recent results in the literature (Ref. 2-3) and may lead to interesting unexplored scenarios.
Anyway, you can choose either **case A** or **case B**. Notice that the final grade of the

miniproject will not be affected by the choice of **case A** or **case B**.

Case A: Predicting unknown observables on time evolved states The ML problem you will consider is of the same kind of the ones in Ref 2-3, however we consider the simpler case where we predict observables on time evolved states. Let H be a Hamiltonian and consider the associated concept class

$$\mathcal{F}_{H,O} = \{f^{\alpha}(x) \;\mid \alpha \in [-1,1]^m\} \tag{1}$$

$$\text{with: } f^{\alpha}(x): \;\; x \in \{0,1\}^n \to f^{\alpha}(x) = \mathrm{Tr}[\rho_H(x)O(\alpha)] \;\; \& \;\; O(\alpha) = \sum_{i=1}^{m} \alpha_i P_i \tag{2}$$

Where x specifies the initial state $\rho_0(x) = |x\rangle\langle x|$ and $\rho_H(x)$ is the evolved state under the Hamiltonian H, i.e. $\rho_H(x) = U\rho_0(x)U^{\dagger}$ with $U = e^{iHt}$. The ML algorithm has to learn a model $h(x)$ which approximates the function f^{α^*} for an unknown α^* using $\mathcal{T}^{\alpha^*} = \{(x_\ell, y_\ell)\}_{\ell=1}^{N}$ as training data. In the ML problem you will consider, the input x will come from an underlying distribution. You can choose the uniform distribution over $x \in \{0,1\}^n$ for example.

1. We will consider the same class of Hamiltonian considered in Ref. 3 given by the 2D anti-ferromagnetic random Heisenberg model $H(J) = \sum_{i,j} J_{ij}(X_iX_j + Y_iY_j + Z_iZ_j)$, you can set the coupling J_{ij} randomly. The Hamiltonian will act on a system of $N = 10$ qubits.
2. As unknown observable you can choose an arbitrary combination of k local Pauli string $O(\alpha_i) = \sum_i \alpha_i P_i$. You can set $k = 3$ for example. This means that the observable $O(\alpha)$ will be the linear combination of all the 3-local Pauli matrices with an arbitrary (of your choice) but fixed α.
3. At this point you can generate data classically by explicitly constructing the initial state $\rho_0(x)$ and computing $\rho_H(x)$ by matrix multiplication.
4. Construct the quantum model and train it using the LASSO regression algorithm:
   (a) For every training point in $T^{\alpha} = \{(x_\ell, y_\ell)\}_{i=1}^{N}$ the quantum algorithm prepares multiple copies of the state $\rho_H(x_\ell)$ and computes the estimates of the expectation values $\langle P_j\rangle_\ell = \mathrm{Tr}[\rho_H(x_\ell)P_j] \;\; \forall j = 1, ..., m$ up to a certain precision ϵ_1. The number of copies needed will depend on the desired error ϵ_1 on the expectation values of each $\langle P_j\rangle_\ell$. As you saw in the lectures, you will have to take in consideration a sampling error which scales polynomially with the number of copies of the state.

Note, also, that m scales at most polynomially in n as $\{P_j\}_{j=1}^m$ are local observables.

(b) To prepare the states $\rho_H(x_\ell)$ you can use Trotterization methods that you studied in the course. (You will have to choose an adequate number of Trotter steps!)

(c) Define the model $h(x) = w \cdot \phi(x)$, where $\phi(x)$ is the vector of the Pauli string expectation values $\phi(x) = [\mathrm{Tr}[\rho_H(x)P_1], ..., \mathrm{Tr}[\rho_H(x)P_m]]$ computed at Step 1. Set the right hyperparameter $B \geq 0$ (motivate the choice!) and run the LASSO regression finds an optimal w^* from the following optimization process:

$$\min_{\substack{w\in\mathbb{R}^m \\ ||w||_1 \leq B}} \quad \frac{1}{N}\sum_{l=1}^{N} |w \cdot \phi(x_l) - y_l|^2 \tag{3}$$

with $\{(x_l, y_l = \mathrm{Tr}[\rho(x_l)O(\alpha)])\}_{l=1}^N$ being the training data.

Importantly, to meet the learning condition the optimization does not need to be solved exactly, i.e. $w^* = \alpha$. As reported in Ref. 3 it is sufficient to obtain a w^* whose training error is ϵ_2 larger than the optimal one.

5. Compare the results using a classical model (you can use your favourite Neural Network).

The report should contain:

1. Answer to the questions above
2. Report of the ML experiment and comparison of the results between the classical and quantum method.
3. **8+** Try using different Hamiltonians and compare the results, specifically you should look at how the performance of the quantum model vs the classical one varies with respect to the Hamiltonian considered.
4. **9+** Discussion with formal arguments on why there should exist a Hamiltonian which gives rise to a learning separation for this kind of task. Identify an example of such Hamiltonian providing proofs of your claims.

Case B (Hard): Predicting unknown observables on ground states The ML problem you will consider is of the same as the ones in Ref 2-3. Let

$\mathcal{H} = \{H(x) \mid x \in \{0,1\}^n\}$ be a family of local Hamiltonians and consider the associated concept class

$$\mathcal{F}_{\mathcal{H},O} = \{f^{\alpha}(x) \mid \alpha \in [-1,1]^m\} \tag{4}$$

$$\text{with: } f^{\alpha}(x): \quad x \in \{-1,1\}^n \to f^{\alpha}(x) = \mathrm{Tr}[\rho_H(x)O(\alpha)] \quad \& \quad O(\alpha) = \sum_{i=1}^{m} \alpha_i P_i \tag{5}$$

Where x specifies the hamiltonian $H(x)$ in the family $\mathcal{H}$ and $\rho_H(x)$ is the ground state of the Hamiltonian $H(x)$. The ML algorithm has to learn a model $h(x)$ which approximates the function f^{α^*} for an unknown α^* using $\mathcal{T}^{\alpha^*} = \{(x_\ell, y_\ell)\}_{\ell=1}^N$ as training data.

1. We will consider the same class of Hamiltonian considered in Ref. 2 given by the 2D anti-ferromagnetic random Heisenberg model $H(J) = \sum_{i,j} J_{ij}(X_iX_j + Y_iY_j + Z_iZ_j)$, for $N = 10$ qubits. Notice that in this case the couplings J_{ij} will exactly be your input vectors x for $x \in \{-1,1\}^n$. You may assume they come from a underlying distributions (e.g. uniform distribution).
2. As unknown observable you can choose an arbitrary combination of k local Pauli string $O(\alpha_i) = \sum_i \alpha_i P_i$. You can set $k = 3$ for example. This means that the observable $O(\alpha)$ will be the linear combination of all the 3-local Pauli matrices with an arbitrary (of your choice) but fixed α.
3. At this point you can generate data classically by diagonalizing $H(J)$ for different vectors J coming (as said before) from an underlying distribution. Notice that the theoretical guarantees for the classical ML algorithm of Ref.2 only works for data coming from the uniform distribution.
4. Construct the quantum model and train it using the LASSO regression algorithm:
   (a) For every training point in $T^{\alpha} = \{(x_\ell, y_\ell)\}_{i=1}^N$ the quantum algorithm prepares poly(n) copies of the state $\rho_{\mathcal{H}}(x_\ell)$ and computes the estimates of the expectation values $\langle P_j \rangle_\ell = \mathrm{Tr}[\rho_{\mathcal{H}}(x_\ell)P_j] \quad \forall j = 1, ..., m$ up to a certain precision ϵ_1.
   Note that m scales at most polynomially in n as $\{P_j\}_{j=1}^m$ are local observables.
   (b) To prepare the states $\rho_{\mathcal{H}}(x_\ell)$ you can use the variational algorithm in Ref. 4-5
   (c) Define the model $h(x) = w \cdot \phi(x)$, where $\phi(x)$ is the vector of the Pauli string expectation values $\phi(x) = [\mathrm{Tr}[\rho_{\mathcal{H}}(x)P_1], ..., \mathrm{Tr}[\rho_{\mathcal{H}}(x)P_m]]$ computed at Step 1. Set the right hyperparameter $B \geq 0$ (motivate the choice!)

and run the LASSO regression finds an optimal w^* from the following optimization process:

$$\min_{\substack{w \in \mathbb{R}^m \\ ||w||_1 \leq B}} \frac{1}{N} \sum_{l=1}^{N} |w \cdot \phi(x_l) - y_l|^2 \tag{6}$$

with $\{(x_l, y_l = \mathrm{Tr}[\rho(x_l) O(\alpha)])\}_{l=1}^{N}$ being the training data.

Importantly, to meet the learning condition the optimization does not need to be solved exactly, i.e. $w^* = \alpha$. As reported in Ref. 3 it is sufficient to obtain a w^* whose training error is ϵ_2 larger than the optimal one.

5. Construct the classical model as described in Ref.3

The report should contain:

1. Answer to the questions above
2. Report of the ML experiment and comparison of the results between the classical and quantum method.
3. **8+** Try using different families of local Hamiltonians and compare the results, specifically you should look at how the performance of the quantum model vs the classical one varies with respect to the Hamiltonian family considered.
4. **9+** Discussion with formal arguments on why there should exist a family of Hamiltonians which give rise to a learning separation for this kind of task. Identify an example of such family of Hamiltonians providing proofs of your claim.