Predicting unkown observables on ground states

Applied Quantum Algorithms Project

César Hernando de la Fuente

Leiden University

May 19, 2025

Part 1: Theory questions

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.

a) Fermionic multi-particle states must be antisymmetric under particle exchange, according to the Spin-statistics theorem. Given the wave functions of individual orbitals, the multi-particle state can be obtained by the Slater determinant. For M orbitals and N_e fermions, a state in the first quantization can be represented as:

$$\Phi(x_1, .., x_{N_e}) = |\psi_{i_1} ... \psi_{i_{N_e}}|, \quad where \quad i_k \in \{0, ..., M\}, \tag{1}$$

which must satisfy that $\Phi(x_1, .., x_i, ..., x_j, ...x_{N_e}) = -\Phi(x_1, .., x_j, ..., x_i, ...x_{N_e})$.

In the second quantization states are represented by occupation number vectors that form a basis for the Fock space: $|f\rangle = |f_1, ..., f_M\rangle$, where f_p is 1 if the orbital p is occupied and 0 if it is unoccupied. The vaccum state is defined as $|\Omega\rangle = |0...0\rangle$ and arbitrary states can be built by applying creation operators:

$$|f\rangle = |f_1 ... f_M\rangle = {a_1}^{\dagger f_1} ... {a_M}^{\dagger f_M} |\Omega\rangle \tag{2}$$

Moreover, the annihilation operators remove a fermion from its orbital:

$$a_p |f_1 ... f_p = 1 ... f_M\rangle = |f_1 ... f_p = 0 ... f_M\rangle , \tag{3}$$

and yield 0 if there was no electron in that orbital before.

From the antisymmetry fermionic states we can learn the following anticommutation property for creation operators:

$$|\psi_p \psi_q| = -|\psi_q \psi_p| \implies {a_p}^\dagger {a_q}^\dagger |\Omega\rangle = -{a_q}^\dagger {a_p}^\dagger |\Omega\rangle \implies \{{a_p}^\dagger, {a_q}^\dagger\} = 0, \tag{4}$$

that leads to the Pauli exclusion principle for $p = q$: ${a_p}^\dagger {a_p}^\dagger = 0$. Applying the property that $(ab)^\dagger = b^\dagger a^\dagger$ to Equation (4), we obtain an anticommutation relation for annihilation operators:

$$\langle\Omega| a_q a_p = -\langle\Omega| a_p a_q \implies \{a_p, a_q\} = 0 \tag{5}$$

Finally, from the rule that you can only annihilate what has already been created, the following can be deduced:

$$a_p a_q^\dagger = -a_q^\dagger a_p + \delta_{pq} \implies \{a_p, a_q^\dagger\} = 0 \tag{6}$$

If we applied LHS and RHS of Equation (6) to $|\Omega\rangle$, we get 0 in both sides if $p \neq q$ and $|\Omega\rangle$ if $p = q$. If we apply both sides to non-vacuum states for cases $p = q$ and $p \neq q$, we see Equation (6) always holds.

In summary, the Fermionic anticommutation relations are stated in Equations (4), (5) and (6).

b) The transformation from Fermionic creation and annihilation operators to Majorana operation is defined in the following way:

$$c_{i,0} = a_i + a_i^\dagger \quad and \quad c_{i,1} = iI(a_i - a_i^\dagger), \tag{7}$$

where I is the identity matrix.

c) First of all, it is immediate to see that the Majorana operators are Hermitian: $c_{i,\alpha} = c_{i,\alpha}^\dagger$. Furthermore, we can evaluate its anticommutation relations:

$$\{c_{i,0}, c_{j,0}\} = \{a_i, a_j\} + \{a_i, a_j^\dagger\} + \{a_i^\dagger, a_j\} + \{a_i^\dagger, a_j^\dagger\} = 0 + \delta_{ij} + \delta_{ij} + 0 = 2\delta_{ij} \tag{8}$$

$$\{c_{i,1}, c_{j,1}\} = \{i(a_i - a_i^\dagger), i(a_j - a_j^\dagger)\} = -(\{a_i, a_j\} - \{a_i, a_j^\dagger\} - \{a_i^\dagger, a_j\} + \{a_i^\dagger, a_j^\dagger\}) = 2\delta_{ij} \tag{9}$$

$$\{c_{i,0}, c_{j,1}\} = \{a_i + a_i^\dagger, i(a_j - a_j^\dagger)\} = i(0 - \delta_{ij} + \delta_{ij} + 0) = 0 \quad (analogue \quad for \quad \{c_{i,1}, c_{j,0}\}) \tag{10}$$

We can combine the previous equations in a more compact way:

$$\{c_{i,\alpha}, c_{j,\beta}\} = 2\delta_{ij}\delta_{\alpha\beta} \tag{11}$$

Since the Majorana operators are Hermitian, we can also state that $\{c_{i,\alpha}^\dagger, c_{j,\beta}^\dagger\} = 2\delta_{ij}\delta_{\alpha\beta}$ and $\{c_{i,\alpha}, c_{j,\beta}^\dagger\} = 2\delta_{ij}\delta_{\alpha\beta}$

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.

Let us first consider the case of a single fermionic creation operator, as the case of an annihilation operators is almost identical. On the one hand, the creation operator acts in the following way on an occupation number vector:

$$a_p^\dagger |f_1 ... f_p ... f_M\rangle = \delta_{f_p 0} (-1)^{\sum_{i=0}^{p-1} f_i} |f_1 ... f_p \oplus 1 ... f_M\rangle \tag{12}$$

On the other hand, the action of the creation operator can be expressed in terms of Pauli gates as follows:

$$a^\dagger = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \frac{1}{2}(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}) = \frac{1}{2}(X - iY) \tag{13}$$

Combining the last two equations and taking into account that creation operators acting on different orbitals anticommute and remembering that Z gates generate a -1 phase when applied to state $|1\rangle$, we obtain the Jordan-Wigner transformation:

$$a_p^\dagger \to Z_1 \otimes ... \otimes Z_{p-1} \otimes \frac{X_p - iY_p}{2} \otimes 1_{p+1} \otimes ... \otimes 1_M \tag{14}$$

and similarly for the annihilation operator:

$$a_p \to Z_1 \otimes ... \otimes Z_{p-1} \otimes \frac{X_p + iY_p}{2} \otimes 1_{p+1} \otimes ... \otimes 1_M \tag{15}$$

Hence, for a creation/annihilation operator acting of the p^{th} orbital/qubit, the locality is p. Consequently, the average locality will be:

$$Average \;\; locality \;\; = \;\; (1 + 2 + ... + M)/M = \frac{\frac{M}{2}(M+1)}{M} = \frac{M+1}{2} \tag{16}$$

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

The Hubbard model is a very simple model that can describe interacting fermions moving in a solid. Its Hamiltonian contains two parts: a single particle part, associated with the kinetic energy, describes particles hopping on a lattice with a single or a few energy bands; and a two-particle part, associated with an on-site Coulomb interaction, which is the shortest possible range of interaction. Despite its simplicity, the Hubbard model exhibits almost all interesting phenomena one observes in nature, such as magnetic ordering of any kind, metal-insulator transition or superconductivity [1].

For a 1-dimensional chain, the Hamiltonian in the second quantization can be written as [2]:

$$H = -t \sum_{i=1}^{L-1} \sum_{\sigma=\uparrow,\downarrow} (a_{i\sigma}^\dagger a_{i+1\ \sigma} + a_{i+1\sigma}^\dagger a_{i\sigma}) + U \sum_{i=1}^{L} n_{i\uparrow} n_{i\downarrow}, \tag{17}$$

where t is the hopping amplitude between neighboring sites, U is the on-site Coulomb repulsion, $n_{i\sigma} = a_{i\sigma}^\dagger a_{i\sigma}$ is the number operator, and L is the number of lattice sites.

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)?

First, we focus on transforming the term $a^\dagger_{i\sigma}a_{i+1\ \sigma}$ of Equation (17), for which we use Equations (14) and (15) (without loss of generality, we assume that $\sigma =\uparrow$):

$$a^\dagger_{i\sigma}a_{i+1\ \sigma} \rightarrow (Z_{1\uparrow}\otimes Z_{1\downarrow}\otimes ...\otimes \frac{X_{i\sigma}-iY_{i\sigma}}{2}\otimes ...\otimes 1_{L\downarrow})(Z_{1\uparrow}\otimes Z_{1\downarrow}\otimes ...\otimes Z_{i\downarrow}\otimes \frac{X_{i+1\ \sigma}+iY_{i+1\ \sigma}}{2}\otimes 1_{i+1\ \downarrow}\otimes ...\otimes 1_{L\downarrow}) \tag{18}$$

Multiplying term by term the previous equation we obtain:

$$a^\dagger_{i\sigma}a_{i+1\ \sigma} \rightarrow 1_{1\uparrow}\otimes ...\otimes \frac{X_{i\sigma}Z_{i\sigma}-iY_{i\sigma}Z_{i\sigma}}{2}\otimes Z_{i\downarrow}\otimes \frac{X_{i+1\ \sigma}+iY_{i+1\ \sigma}}{2}\otimes 1_{i+1\ \downarrow}\otimes ...\otimes 1_{L\downarrow} \tag{19}$$

The second term in the Equation (17) is simply the adjoint, so we can write the Jordan-Wigner transformation immediately:

$$a^\dagger_{i+1\sigma}a_{i\ \sigma} \rightarrow 1_{1\uparrow}\otimes ...\otimes \frac{X_{i\sigma}Z_{i\sigma}+iY_{i\sigma}Z_{i\sigma}}{2}\otimes Z_{i\downarrow}\otimes \frac{X_{i+1\ \sigma}-iY_{i+1\ \sigma}}{2}\otimes 1_{i+1\ \downarrow}\otimes ...\otimes 1_{L\downarrow} \tag{20}$$

Finally, we transform the term $a^\dagger_{i\uparrow}a_{i\uparrow}a^\dagger_{i\downarrow}a_{i\downarrow}$. We first consider only $a^\dagger_{i\uparrow}a_{i\uparrow}$:

$$a^\dagger_{i\uparrow}a_{i\uparrow} \rightarrow (Z_{1\uparrow}\otimes Z_{1\downarrow}\otimes ...\otimes \frac{X_{i\uparrow}-iY_{i\uparrow}}{2}\otimes ...\otimes 1_{L\downarrow})(Z_{1\uparrow}\otimes Z_{1\downarrow}\otimes ...\otimes \frac{X_{i\uparrow}+iY_{i\uparrow}}{2}\otimes ...\otimes 1_{L\downarrow}) = 1_{1\uparrow}\otimes ...\otimes \frac{1_{i\uparrow}-Z_{i\uparrow}}{2}\otimes ...\otimes 1_{L\downarrow} \tag{21}$$

where I have used that $(1_{i\sigma}+iX_{i\sigma}Y_{i\sigma}-iY_{i\sigma}X_{i\sigma}+1_{i\sigma})/4 = (1_{i\sigma}-Z_{i\sigma})/2$ since $i[X,Y] = i(2iZ) = -2Z$.

We consider now the term $a^\dagger_{i\downarrow}a_{i\downarrow}$, whose Jordan-Wigner transformation is almost identical similar:

$$a^\dagger_{i\downarrow}a_{i\downarrow} \rightarrow 1_{1\uparrow}\otimes ...\otimes \frac{1_{i\downarrow}-Z_{i\downarrow}}{2}\otimes ...\otimes 1_{L\downarrow} \tag{22}$$

Multiplying Equations (21) and (22), we obtain the transformation of the final term of the Hubbard Hamiltonian:

$$a^\dagger_{i\uparrow}a_{i\uparrow}a^\dagger_{i\downarrow}a_{i\downarrow} \rightarrow 1_{1\uparrow}\otimes ...\otimes \frac{1_{i\uparrow}-Z_{i\uparrow}}{2}\otimes \frac{1_{i\downarrow}-Z_{i\downarrow}}{2}\otimes ...\otimes 1_{L\downarrow} \tag{23}$$

We have reached to the final step, which consists of combining all the transformed terms and take into account the sum of the lattice sites. Thus, the Jordan-Wigner transformed Hubbard Hamiltonian is:

$$H_{J-W} = -\frac{t}{2}\sum_{i=1}^{L-1}(X_{i\uparrow}Z_{i\uparrow}\otimes Z_{i\downarrow}\otimes X_{i+1\uparrow} + X_{i\downarrow}Z_{i\downarrow}\otimes Z_{i+1\uparrow}\otimes X_{i+1\downarrow}) + \frac{U}{4}\sum_{i=1}^{L}(1_{i\uparrow}-Z_{i\uparrow})\otimes(1_{i\downarrow}-Z_{i\downarrow}), \tag{24}$$

where I have calculated the second term in the first sum in a similar way as the first one but using $\sigma = \downarrow$ in Equations (18) and (20). Furthermore, it it worth noting that the identity acts on the rest of the qubits where a Pauli gate is not applied.

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 and (c) the Electronic structure problem?

a) The one dimensional Hubbard hamiltonian was written in Equation 17. We observe that the number of kinetic energy terms depends of the number of sites as $2\times 2(L-1) = O(L)$, while the number of interaction temrs is equal to the number of sites. Thus, the total number of terms scales linearly with the number of sites. For the two dimensional case, considering a rectangular lattice, each site has maximum 4 neighbouring sites, and the number of edges is given by $n_x(n_y-1)+(n_x-1)n_y$, where n_x and n_y are the number of sites in each axis. Thus, the number of edges scales linearly with the number of sites and consequently, the number of kinetic terms too. For the three dimensional model the number of neighbouring sites increase, but the number of edges and therefore kinetic terms also increase linearly.

b) For the Heisenberg hamiltonian, studied in my miniproject extensively (see Equation 32, the number of terms is 3 times the number of edges of the spin lattice considered. Since the number of edges in a chain, recatangular and cubic lattice scales linearly with the number of spins, the number of terms will scale linearly too.

c) The second-quantized hamiltonian of the electronic structure problem is:

$$H_{E.S.} = \sum_{pq} h_{pq}a_p^\dagger a_q + \frac{1}{2}\sum_{pqrs} h_{pqrs}a_p^\dagger a_q^\dagger a_r a_s \tag{25}$$

The first term coefficcients h_pq represent one-electron integrals accounting for the kinetic energy and the nuclear attraction, while the second term coefficients h_{pqrs} represent two-electron integrals, and specify how any electron pair in orbitals p, q interacts with a pair in r, s. In other words, the indices encodee the initial and final orbitals of each of the electron pairs. Thus, the number of terms for the electronic structure hamiltonian scales as N^4, where N is the number of orbitals.

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.

Implicit QSVM

In order to explain implicit Quantum Support Vector Machines (QSVM), also called quantum kernel estimators, we need to describe classical support vector machines (SVMs) first. Let us consider a problem where we want to learn how to classify data, which can be expressed as a high-dimensional vector $x \in \mathbb{R}^d$, into two classes: $y \in \{-1, 1\}$. We are given a training set of N data points $x_j \in \mathbb{R}^d$ together with their class y_j. In general, there does not exist a hyperplane $H = \{x \in \mathbb{R}^d : wx + b = 0\}$ that separates the data points with $y = 1$ from those with $y = -1$. However, let us consider for now a particular case where H exists. Since the data points live in a real space, there will actually exist infinite hyperplanes that separate the training data points. The goal is to classify correctly data points outside of the training set and consequently (i.e. it does not only "fit" but also generalizes), it would be ideal if the hyperplane not only separates the data points of different classes but also maximizes its distance to the closest x_j. This is the intuitive idea behind SVMs [3].

Moving towards a mathematical description of SVMs, the hyperplane parallel to H and that contains the closest x_j will be parametrized by: $\omega x + b = C$, and its reflection over H is parametrized by: $\omega x + b = -C$, where C depends on the distance to the closest x_j. Dividing by C and keeping the original names for the normalized ω and b, we see that the separation and margin hyperplanes are parametrized by: $wx + b = 0$, $wx + b = 1$ and $wx + b = -1$. The larger the margin, the larger the constant C and the smaller the norm of ω will be. Thus, we can express the optimization of SVMs in the following way [3]:

$$Minimize \quad \frac{1}{2}||w||^2 \quad subject\ to \quad y_j(\omega x + b) \geq 1 \tag{26}$$

If we relax the constraint and introduce slack variables ξ_j that control the tolerance with which we allow misclassification, we can recast the problem:

$$Minimize \quad \frac{1}{2}||w||^2 + \sum_j \xi_j \quad subject\ to \quad y_j(\omega x + b) \geq 1 - \xi_j \quad and \quad \xi_j \geq 0, \tag{27}$$

where C is a hyperparameter. However, there is an equivalent formulation that is the one used in practice which is known as the Lagrangian dual of the optimization problem (proof in [4]):

$$Maximize \quad \sum_j \alpha_j - \frac{1}{2}\sum_{j,k} y_j y_k \alpha_j \alpha_k (x_j \cdot x_k) \quad subject\ to \quad 0 \leq \alpha_j \leq C \quad and \quad \sum_j \alpha_j y_j = 0, \tag{28}$$

where $\omega = \sum_j \alpha_j y_j x_j$. Therefore, first we obtain the value of the optimal α_j and calculate the optimal support vector ω. We can also obtain the value of b by finding some x_j that lies at the boundary of the

margin and solving a simple equation [4]. Next, we can classify data points outside the training data set by calculating the sign of $wx + b$. Many classification problems that one would encounter are not linearly separable, but there is a trick that consists of mapping the data points to a higher dimensional space (feature space) through a function φ known as a feature map. It may happen that the problem is converted to linearly separable in this feature space. The mathematical description of the optimization task is then as follows:

$$Maximize \quad \sum_j \alpha_j - \frac{1}{2}\sum_{j,k} y_j y_k \alpha_j \alpha_k (\varphi(x_j)\cdot\varphi(x_k)) \quad subject\ to \quad 0 \leq \alpha_j \leq C \quad and \quad \sum_j \alpha_j y_j = 0 \quad (29)$$

Now we have the required background to explain the implicit QSVM, which is hybrid ML classification technique that consists of performing SVM where the feature map is a quantum circuit (and the feature space is a Hilbert space). The only thing for which we use a quantum computer is to calculate the following [5]:

$$K(x_j, x_k) = |\langle\varphi(x_j)|\varphi(x_k)\rangle|^2 = |\langle 0|\phi^\dagger(x_j)\phi(x_k)|0\rangle|^2, \quad (30)$$

which is nothing else than the fidelities of all the pairs of feature vectors corresponding to the training data points. These values can be interpreted as the probabilities of measuring $+1(|0\rangle)$ after initializing a quantum circuit in state $|0\rangle$ and applying first $\phi(x_k)$ and then $\phi^\dagger(x_j)$. The latter can be implemented by reversing the order of the gates in the unitary ϕ and subsituting each gate by its inverse. Thus, we could measure the qubits in the Z basis M times and calculate the estimator of the probability of measuring all qubits in state $|0\rangle$.

We have not given details about the different options of implementing the feature map unitary ϕ yet, since we will discuss them when explaining explicit QSVMs.

Explicit QSVM

Explicit QSVMs (or quantum variational classifiers) resemble more the structure of parametrized quantum circuits. The idea is to build a quantum circuit that has two blocks. The first one is a feature map unitary $\phi(x)$ and the second one is a variational ansatz $V(\theta)$. Finally we choose an observable O and we execute the circuit M times to estimate its expectation value $tr[\rho(x)V(\theta)OV^\dagger(\theta)]$. The training of the model consists of finding the θ (vector) that maximizes the margin:

$$\theta^* = argmax \quad min(x \in Data)[(-1)^{label(x)} Tr[\rho(x)V(\theta)OV^\dagger(\theta)]] \quad (31)$$

Finally, we can classify data points outside the training data set by looking at the sign of $Tr[\rho(x)V(\theta^*)OV^\dagger(\theta^*)$.

Having explained the workflow of explicit QSVMs, we can now describe some possible implementations of variational anstatze and feature maps. A very common variational ansatz is the hardware efficient, described in [6]. Regarding feature maps, there are different types of encoding, including angle encoding (data points

are normalized between in the interval between 0 and 2π and used as angles of rotation gates. In this case, they are called Fourier feature maps. There are other possibilities, where we first transform the x by a non-linear function and then we use the output as the rotation angle. Some examples of these are product and Chebyshev feature maps [6], which have been used for QPINNs. Furthermore, there are other possibilities like amplitude encoding or ZZ feature map [3].

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?

a) Feature maps, as explained in the previous section, are simply a transfomation (function) that takes a data point $x \in \mathbb{R}^d$ as input and outputs a vector in a higher dimensional space $\mathbb{R}^D$, where $D > d$. This feature map can make the classification problem linearly separable in the feature space, i.e., there exists a hyperplane that separates correctly the data points according to their binary labels (-1 and +1). A particular example is a mapping to a quantum Hilbert space, which is isomorph to $\mathbb{R}^D$ for a specific D depending on the number of qubits used in the quantum circuit.

b+c) Kernels are simply scalar functions of two variables. A kernel has a corresponding feature map if it can be expressed as an inner product of two vectors of a feature space (generated by a feature map). According to Mercel's theorem [4], a sufficient condition for the existence of a feature map given a kernel is that the kernel function is positive semidefinite, i.e., it is symmetric and $K(x_j, x_k) \geq 0$ for all $x_j, x_k \in Dom(K)$. Consequently, using the Lagrangian dual formulation of SVMs, we only need to evaluate the kernel function instead of calculating transforming the data points with the feature map and then calculating their inner product, which sometimes can provide an efficiency advantage.

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

Since every quantum circuit can be simulated classically, although possibly with an exponential overhead, there are no learning problems which quantum learners can learn but classical learners cannot. Rather, the more interesting question is whether there are learning problems that quantum learning algorithms can learn much faster than any classical learning algorithm. Some examples of quantum learning advantages have been demonstrated for artificial cryptography-inspired datasets when dealing with classical data. Other learning separations have been proved for concept classes based on the discrete logarithm problem and for concept classes based on the discrete cube root [7].

Part 2: Report of the project on predicting unknown observables on ground states (2.F.B)

Table of Contents

1 Introduction **11**

1.1 Description of the system 11

1.2 Related work 11

1.3 Problem statement 12

2 Methods **13**

2.1 Quantum Lasso model 14

2.1.1 Hardware-efficient VQE 14

2.1.2 Implementation of Lasso regression 15

2.2 Classical random Fourier feature map Lasso model 16

2.3 Deep Neural network 17

3 Results **17**

3.1 Analysis of performance of VQE for different lattice sizes 17

3.2 Comparison of ML models 18

3.3 Comparison of performance for the Ising model 19

4 Conclusions **19**

1 Introduction

Finding ground states of hamiltonians of many-body quantum systems is a very hard task, but it has profound implications for physics, chemistry and materials science. There exists several powerful classical methods that have enabled solutions for certain restricted instances, such as density functional theory, quantum Montecarlo [8] or density-matrix renormalization group. However, many general classes of problems remain outside the reach of even the most advanced classical algorithms [9].

Quantum computers were proposed by Feynmann to solve these types of problems[10], but realising scalable and fault-tolerant quantum computers faces a huge number of challenges, and are unlikely to be available in the near future. However, in recent years different machine learning (ML) techniques have been applied to many-body physics problems and have been shown to be effective [11] [12]. In this project, we investigate different ML methods to predict the expectation value of an unknown observable in the ground state for a 2D anti-ferromagnetic random system of spins.

1.1 Description of the system

The physical system studied in this project is a two dimensional rectangular lattice of spins whose interaction is modeled by the Heisenberg hamiltonian:

$$H_{Heis} = \sum_{\langle i,j \rangle} J_{ij}(X_i X_j + Y_i Y_j + Z_i Z_j), \tag{32}$$

where the coefficients J_{ij} capture the coupling strength between the spins, and X, Y, Z are the Pauli gates. Although the focus of this project is in this hamiltonian, we will explore how the performance of the different machine learning models change when the Heisenberg hamiltonian is replaced by the Ising hamiltonian:

$$H_{Ising} = \sum_{\langle i,j \rangle} J_{ij} Z_i Z_j \tag{33}$$

It is worth mentioning the fact that the Ising model is invariant under exchanging $|\uparrow\rangle$ and $|\downarrow\rangle$ states and thus its eigenvalues come in pairs. Consequently, it is a gapless hamiltonian.

1.2 Related work

In this section we will relate how this project fits and relates with some papers in the field of Quantum Machine learning for many-body quantum physics. Additionally, we will answer to the theoretical questions of the project (2.F).

Despite the effort of the quantum machine learning community in the last years, only a few examples of quantum learning separations have been demonstrated. Some of these involve artificial cryptography-inspired datasets when dealing with classical data. Moreover, learning separations have also been proved for concept classes based on the discrete logarithm problem and for concept classes based on the discrete cube root

[7]. However, how is learning separation defined and what are the differences with respect to separation in computing?

Learning separation is defined as a demonstrable difference in the efficiency of learning from data between two computational models, like classical and quantum algorithms. In particular, if a concept class can be efficiently learned by one model but not by the other, under the same learning framework, e.g., PAC learning. Some categories of learning problems are: [13] (CC, CQ,QC,QQ). The first letter refers to whether a concept class is classically or quantumly efficiently learnable, whereas the second refers whether it is classically or quantumly polynomially evaluatable. In contrast, computational separation refers to the difficulty of evaluating a function, instead of learning it from data. Complexity classes such as P, NP, BPP or BQP represent different computational complexities.

These ideas of learnability have also been studied in the field of many-body physics. For instance, in [9] they consider a family of gapped local quantum hamiltonians, where the hamiltonian $H(x)$ depends smoothly on m parameters (in our project, the parameters are the coupling strength between spins). The training data consists of sampled values of x, accompanied by a classical representation (classical shadow) of the ground state of $H(x)$. After learning the ground state, the model can predict ground state properties. Specifically, products of local observables. A very similar machine learning model is explained in [14], where in this case the targets are not classical representations of ground states but expectation values of observables in the ground state (like in this project). However, it also assumed gapped local hamiltonians. A high level explanation of the ML algorithm of [14] is given in section 2.2.

In spite of their similarity, the ML algorithms of the two papers mentioned have some differences. First, [9] is guaranteed to work only if x are sampled from a uniform distribution, whereas [14] works for any distribution. Secondly, the sample complexity of [9] is $N = n^{O(1/\epsilon)}$, whereas for [14] it is $N = \log(n)2^{polylog(1/\epsilon)}$, where n is the number of qubits and ϵ is the prediction error. Regarding computational complexity, it is $O(nN)$ for both (N is different).

Having made a brief digression to comment some of the papers related to the topic of this work, we now explain the problem statement of this project.

1.3 Problem statement

The goal of this project is to develop a machine learning (ML) model that predicts the expectation value of an unknown k-local observable in the ground state for a certain combination of coupling strengths. The ML method belongs to the category of supervised learning, where the input features are the set of coupling strengths J_{ij}. Since the lattice is rectangular and the interaction is between nearest neighbours, we divide the couplings in two groups: the horizontal and vertical coupling strengths, where each of them can be organised in a matrix. However, we linearize each matrix and concatenate them so that they form a vector of features, which we denote as $x \in [-1, 1]^m$, where m is the total number of couplings. It can be easily found analitically that $m = n_x(n_y - 1) + (n_x - 1)n_y = 2n_xn_y - n_x - n_y$, where n_x and n_y are the number of rows and columns of spins, respectively. The targets of the ML model are the expectation values of the

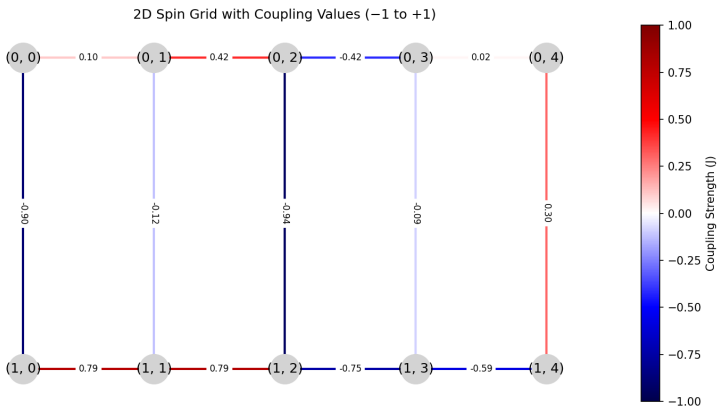

Figure 1: Random initialization of the couplings between the spins in a rectangular 2x5 lattice.

unknown observable in the ground state (for a given set of couplings x), which we denote as y. The ML algorithms will have to learn from this training dataset and learn to predict the ground state expectation value of an unknown observable.

In the Methods section we discuss the three different ML methods utilised, two based on LASSO regression, while the other one is a deep neural network (DNN). One of the LASSO models is quantum, as it uses the Variational Quantum Eigensolver as a subroutine; whereas the other is purely classical, and was proposed in [14]. Finally, the DNN is also classical. Following the description of the algorithms, we present the results obtained in different experiments, where we compare the performance of the different ML models, using different lattice sizes, hamiltonians and hyperparameters. Finally, we outline the main findings of the project and propose some ideas for future work.

2 Methods

In this section we describe three different machine learning methods used in this project. Before explaining how each of them work, we define some common aspects that we use for all the project.

First of all, we generate 100 examples, which provides a good balance between accuracy and runtime. Secondly, we consider lattices of 2x2 and 2x5 spins (see Figure 1. The executions of the 2x2 grids were performed during the day, but the ones of 2x5 required running them overnight. Thirdly, we generate random couplings sampled from the uniform distribution between -1 and 1. Finally, we choose the same observable as in [9] and [14], which is the correlation between the qubits in the coordinates (0,0) and (0,1):

$$C_{01} = \frac{1}{3}(X_0X_1 + Y_0Y_1 + Z_0Z_1), \tag{34}$$

which a 2-local observable. In the following, we explain the machine learning models utilized for this project.

2.1 Quantum Lasso model

The main model studied in this project is quantum, in the sense that it uses the Variational Quantum Eigensolver (VQE) to calculate the ground state, ground state energy and ground state expectation value of the Pauli strings in which the observable is decomposed. Having the input features x_l and targets y_l for N examples, the model applies a Lasso regression [15], described as:

$$\min_{\substack{w\in\mathbb{R}^m \\ \|w\|_1\leq B}} \frac{1}{N}\sum_{l=1}^{N} |w\cdot\phi(x_l) - y_l|^2 \tag{35}$$

which can be rewritten equivalently as:

$$\min_{w\in\mathbb{R}^m} \frac{1}{N}\sum_{l=1}^{N} |w\cdot\phi(x_l) - y_l|^2 + \alpha||\omega||_1, \tag{36}$$

where the constraint is recast as a penalty or regularization term. The intuition of why Lasso regression includes this regularizing L1 norm, and not L2 norm (Ridge regression), is because we want the model to favor parameter solutions with low number of non-zero values (sparse).

Regarding the feature map $\phi(x)$, for the quantum model we use the following:

$$\phi(x) = [Tr[\rho(x_l)P_1],\ Tr[\rho(x_l)P_2,\ ...,\ Tr[\rho(x_l)P_p]], \tag{37}$$

where p is the number of Pauli strings in which the observable is decomposed in. Having found the optimal ω^*, the model can predict the ground state expectation value of the observable for unseen combinations of coupling strengths x.

In the following, we explain the details of the implementations of the VQE and the LASSO regression.

2.1.1 Hardware-efficient VQE

The variational quantum algorithm is a hybrid algorithm that is used to estimate the ground state energy of a hamiltonian. It is hybrid in the sense that it uses a parametrized quantum circuit, also known as anstatz, to evaluate the expectation value of the hamiltonian (throgh the measurement of the Pauli gates in which it can be decomposed). Thus, we can treat this expectation value as a function with respect to the variational parameters, and tweak these to minimize the expectation value of the hamiltonian. According to the variational principle, the minimum value obtained constitutes an uppper bound for the ground state of the hamiltonian [16].

There is a wide variety of optimization algorithms that can be used in VQE, some required the computation of gradients, and others gradient-free. In this project, we use Adam optimizer [17]. Regarding the quantum gates used for the ansatz, there exist many different possibilities [18], but inspired by [6], we implement a hardware-efficient ansatz. This ansatz is especially suitable for NISQ quantum computers it consists of

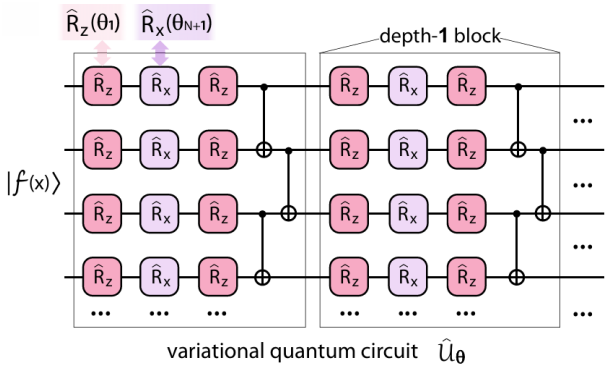

Figure 2: A variational ansatz in the hardware efficient form. It consists of a parametrized rotation layer, such that an arbitrary single qubit rotation can be implemented. Variational angles θ are set for each rotation individually. The rotation layer is then followed by an entangling layer chosen as $CNOT$ operations between nearest neighbours. The blocks of "rotations-plus-entangler" are repeated d times. This figure is taken from [6].

different layers, where in each layer we apply the rotation gates R_X, R_Z, R_Y (with different variational angles) on each qubit, and then apply $CNOT$ gates between neighbouring qubits in a cascade way (see Figure 2) The number of layers in which this sequence is repeated is known as the depth of the anstatz.

The VQE algorithm depends on different hyperparameters, such as the depth d of the ansatz, the number of epochs or optimization steps of the Adam optimizer, and its initial learning rate. After tuning both of them, we found that a good set of hyperparameters that balance runtime and accuracy is: $d = 3$, 500 optimization steps and initial learning rate of 0.01. Finally, it is important to mention that the VQE was implemented and executed using Pennylane.

In the Results section we will test the accuracy of our VQE, comparing the estimated ground state energies of different hamiltonians and lattice sizes with their exact ground state obtained by diagonalization (using Numpy).

2.1.2 Implementation of Lasso regression

As it is common in ML, we split the generated dataset in two: a training and a test dataset. This ensures that the ML model is not memorizing but actually learning from the data, and can generalize and predict correctly for unseen data. As we explained previously (see Equation 36), the Lasso regression cost function includes a regularization term which is multiplied by the hyperparameters α. In order to choose the optimal α that minimizes the error, we use of Lasso regression called cross-validation Lasso.

It works in the following way [19]: first, in splits the data in k folds (5 in our implementation); then, we choose a range of α $(10^{-4}, 10^{-3}, ..., 1)$; next, for each α, we train Lasso using $k-1$ folds, and calculate and save the validation error using the remaining fold. After that, we rotate which folds are used for training and validation, and we calculate the average validation loss. Having performed these steps for all α values, we select the α with the lower average validation loss. Finally, for that value of α, we find the optimum

parameters of the model training with all the folds, and then use the test data to assess its performance. This algorithm is already implemented in the method *LassoCV*, which is one of the linear models that the Python package *sklearn* offers.

One of the metrics commonly used to assess the learning performance of linear regression models are the mean squared error, which is the average of the squares of the difference between the predicted and labeled targets across the test dataset. Another metric is the R^2 score, or coefficient of determination, which represents how well the model explains the variability of the target variable. On the one hand, a value of R^2 close to 1 corresponds to a model that predicts the targets perfectly. On the other hand, if R^2 is close to 0, the model predictions are no better than using the average, and if it is less than 0, the model performs even worse than if it just used the average.

2.2 Classical random Fourier feature map Lasso model

In this project we compare the quantum model described previously with a purely classical machine learning model, described in [14]. In the paper they generate the training data targets, i.e. the ground state expectation values of the observable, using the density-matrix renormalization group based on matrix product states. However, in our project we use a maximum of 10 qubits and therefore, diagonalizing the hamiltonian to find the ground state is done very fast using the *Numpy linalg.eigh* method (for Hermitian matrices).

Regarding the ML model, they also use Lasso regression, but with a very different feature map. The feature map consists of using only those coupling strenghts associated with qubits that are close to the qubits where the observable acts. For instance, in this project we use as the observable the correlation between the qubits 0 and 1: $C_{01} = \frac{1}{3}(X_0X_1 + Y_0Y_1 + Z_0Z_1)$. Thus, for the feature map we only select those couplings associated with edges that are at a Manhattan distance of 1 to the (0,0)-(0,1) edge. This includes the horizontal couplings connecting qubits (0,0) and (0,1), (0,1) and (0,2), and the edge between (1,0) and (1,1); as well as the vertical edges connecting qubits (0,0) and (1,0) and (0,1) and (1,1). The general algorithm to find the local edges can be found in the code.

We denote the vector containing the local couplings as z. The feature map implemented in the paper is known as the random Fourier feature map, and is defined as follows:

$$\phi : z \to \begin{pmatrix} \cos(\frac{\gamma}{\sqrt{l}}(\omega_1 z)) \\ \cos(\frac{\gamma}{\sqrt{l}}(\omega_1 z)) \\ ... \\ \cos(\frac{\gamma}{\sqrt{l}}(\omega_R z)) \\ \sin(\frac{\gamma}{\sqrt{l}}(\omega_R z)) \end{pmatrix}, \tag{38}$$

where l is the number of local couplings (length of z), R (determines dimensioin of feature map) and γ are hyperparameters, and the ω_i are l-dimensional vectors sampled from a multivariate standard normal distribution.

Using $\phi(z)$ as inputs and the ground state expectation value of the observable (correlation of qubits 0 and

1), we apply Cross-validated Lasso model, as described in section 2.1.2

2.3 Deep Neural network

The last method that we used in this project was a Deep Neural Network (DNN), which is the most popular ML algorithm in Supervised learning nowadays. For this method, we consider the set of strength couplings (linearised) as inputs and the ground state expectation value as outputs. The architecture of the DNN consists of 4 hidden layers with 8,16,16 and 8 neurons, which apply the ReLU activation function. The output layer has obviously one neuron, and no activation function. The loss function used is the mean squared error, and we use Adam optimizer with 10 batches.

3 Results

This section presents the results obtained in different simulations. As a preliminary, we show how the Hardware-efficient performs for different lattice sizes. Then, we will compare the different machine learning models. Finally, we will compare how the performance of the models change when we use other hamiltonians such as the the Ising and Transverse field Ising model.

3.1 Analysis of performance of VQE for different lattice sizes

Let us first start analysing the performance of the VQE. In Table 3.1 we show the estimated ground state energies by VQE compared to the true value, for 2x2 and 2x5 lattices, for different random couplings. We observe that for both lattice sizes the relative error depends on the specific random couplings configuration.

System	$\mathbf{E_{exact}}$	$\mathbf{E_{VQE}}$	**Relative error**
2×2 Spin Lattice	-3.11	-2.98	4.2%
2×2 Spin Lattice	-1.71	-1.55	9.4%
2×5 Spin Lattice	-11.16	-10.75	3.7%
2×5 Spin Lattice	-6.31	-5.32	15.7%

Table 1: Comparison of VQE and exact Ground State Energies for 2×2 and 2×5 spin lattice systems.

In order to show that the VQE implementation is correct, even though it does not give exact results, in Figure 3 we show two maps of the correlation between all the qubits in a 2x2 grid obtained by VQE and by exact diagonalization. By exact diagonalization we mean that the hamiltonian is diagonalised, then we select the lowest eigenvalue and its corresponding eigenvector, and obtain the expectation value of the observable by matrix multiplications. We observe that the VQE estimations are very similar to the exact ones, validating our implementation.

In Figure 4 we plot the same maps for a 2x5 lattice. The VQE correlation values are noisier in this case, but still are strongly correlated to the exact values, meaning that the most ad least correlated pairs of spins

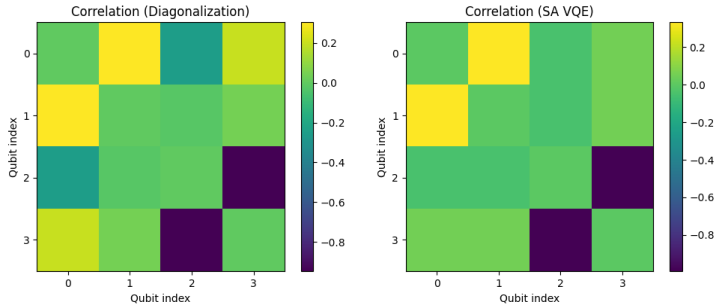

Figure 3: Maps of correlations between all pairs of qubits for a 2x2 spin lattice with random couplings. The map of the left was obtained by diagonalization, while the one in the right was obtained using VQE

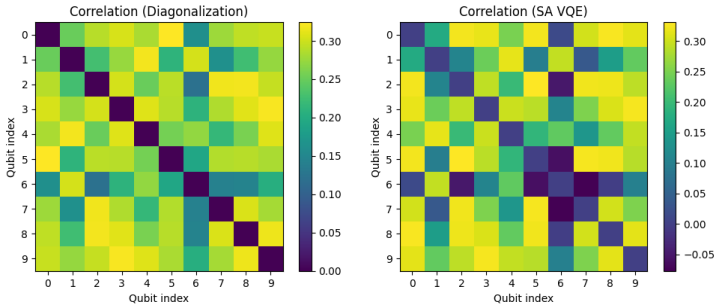

Figure 4: Maps of correlations between all pairs of qubits for a 2x5 spin lattice with random couplings. The map of the left was obtained by diagonalization, while the one in the right was obtained using VQE

(without taking into account the diagonal) for the exact calculation are also the most and least correlated pairs for the VQE estimation.

3.2 Comparison of ML models

In this section we compare the performance of the quantum Lasso model, the random Fourier feature map model and the neural network for the Heisenberg hamiltonian and a lattice grid of 2x5. The obtained R^2 scores and mean squared error on the test data of each of the models is shown in Table 2. For the random Fourier feature map model, we considered the hyperparameters $\gamma = 0.6$ and $R = 10$

Model	$\mathbf{R^2}$	**MSE**
Quantum Lasso	0.9999997	7.8×10^{-8}
Fourier feature Lasso	0.78	0.05
Neural Network	0.70	0.073

Table 2: Performance metrics (R^2 and MSE) for three machine learning models considering the Heisenberg model.

Additionally, for the Quantum model we see that the parameters that Lasso learns are [0.333, 0.333, 0.333], which agree with the coefficients of each of the Pauli strings of the two-spins correlation operator. In the Table 2 we observe that the Quantum model predicts the observable almost perfectly, whereas the Fourier

feature Lasso model from [14] achieves a R^2 of 0.78, which is not perfect but considerably good, taking it into account that it outperforms the deep neural network.

3.3 Comparison of performance for the Ising model

We now apply the same ML methods to the Ising hamiltonian in order to see whether they are able to predict the ground state correlation with a similar performance. In table X we show the performance metrics R^2 and MSE of the ML methods for lattice sizes 2x2 and 2x5. In this case we have used the same hyperparameters as before for the random Fourier feature map.

Lattice Size	**Model**	**R^2**	**MSE**
2×2	Quantum Lasso	0.9999998	1.1×10^{-8}
2×2	Fourier feature Lasso	0.57	0.04
2×2	Neural Network	0.84	0.011
2×5	Quantum Lasso	0.999996	4.5×10^{-7}
2×5	Fourier feature Lasso	0.42	0.058
2×5	Neural Network	-0.41	0.10

Table 3: Performance metrics (R^2 and MSE) for three machine learning models and two different lattice dimensions considering the Ising model.

Table 3 shows very interesting results. First of all, we observe that the quantum model still predicts almost perfectly. Additionally, we note that the learned parameters were [0, 0, 0.333], which is logical since the correlations of the X and Y projections are 0 in the ground state. This is because the ground state must be an eigenstate of the hamiltonian, and thus all spins must point either up or down.

Furthermore, for a lattice size of 2x2 the performance of the random Fourier feature map decreases significantly, and for the lattice of 2x5 it predicts even worse. This decrease of performance makes complete sense, since as explained in [14], the model is only guaranteed to work if the hamiltonian has a gap. However, the Ising hamiltonian does not have a gap since its eigenvalues come in pairs due to the symmetry of flipping all the spins.

We also observe that the neural network can predict well the correlation of qubits 0 and 1 in the ground state for a small lattice size, but it gives an extremely bad prediction when the lattice grows. This also makes sense, since the systems grows in complexity while the complexity of the neural network stays the same.

4 Conclusions

To sum up, we have implemented three different machine learning models to predict the expectation value of an unknown observable in the ground state for a 2D anti-ferromagnetic system of random spins. We have shown that for the Heisenberg model and a lattice of 2x5 spins, the quantum model is able to predict the observable almost perfectly, whereas the neural network and the random Fourier feature map Lasso model can predict the observable reasonably well.

We also analysed how the performance of the models changes if we study if we replace the Heisenberg hamiltonian by an Ising model. We obtained that the quantum model maintains its performance, whether both the neural network and random Fourier feature map Lasso model reduce drastically their prediction accuracy. The decrease for the classical Lasso model agrees with [14], where it is mentioned that this model is only guaranteed to work for gapped hamiltonians.

Regarding some ideas for future work, one could try other hamiltonians such as the transverse field Ising model, the Rydberg atom chain, or the bond-alternating XXZ model. Moreover, other observables with higher locality could be tested, since this would affect the dimensionality of the Fourier feature map, as more couplings would be inside the local region. Finally, instead of using the hardware-efficient ansatz, a symmetry-adapted VQE could be utilised [18].

References

[1] Andreas Mielke. "11 The Hubbard Model and its Properties". In: *Autumn School organized by the Forschungszentrum Jülich and the German Research School for Simulation Sciences at Forschungszentrum Jülich* ().

[2] Jan-Michael Reiner et al. "Emulating the one-dimensional Fermi-Hubbard model by a double chain of qubits". In: *Physical Review A* 94.3 (2016), p. 032338.

[3] Elias F Combarro, Samuel González-Castillo, and Alberto Di Meglio. *A practical guide to quantum machine learning and quantum optimization: Hands-on approach to modern quantum algorithms*. Packt Publishing Ltd, 2023.

[4] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*. Vol. 4. AML-Book New York, 2012.

[5] Vojtěch Havlíček et al. "Supervised learning with quantum-enhanced feature spaces". In: *Nature* 567.7747 (2019), pp. 209–212.

[6] Oleksandr Kyriienko, Annie E Paine, and Vincent E Elfving. "Solving nonlinear differential equations with differentiable quantum circuits". In: *Physical Review A* 103.5 (2021), p. 052416.

[7] Casper Gyurik and Vedran Dunjko. "Exponential separations between classical and quantum learners". In: *arXiv preprint arXiv:2306.16028* (2023).

[8] J.M. Thijssen. *Computational Physics*. 2nd ed. Cambridge University Press, 2007. ISBN: 9781139171397. DOI: https://doi.org/10.1017/CBO9781139171397.

[9] Hsin-Yuan Huang et al. "Provably efficient machine learning for quantum many-body problems". In: *Science* 377.6613 (2022), eabk3333.

[10] Richard P Feynman. "Simulating physics with computers". In: *Feynman and computation*. cRc Press, 2018, pp. 133–153.

[11] Evert PL Van Nieuwenburg, Ye-Hua Liu, and Sebastian D Huber. "Learning phase transitions by confusion". In: *Nature Physics* 13.5 (2017), pp. 435–439.

[12] Sebastian J Wetzel and Manuel Scherzer. "Machine learning of explicit order parameters: From the Ising model to SU (2) lattice gauge theory". In: *Physical Review B* 96.18 (2017), p. 184410.

[13] Casper Gyurik and Vedran Dunjko. "On establishing learning separations between classical and quantum machine learning with classical data". In: *arXiv preprint arXiv:2208.06339* (2022).

[14] Laura Lewis et al. "Improved machine learning algorithm for predicting ground state properties". In: *nature communications* 15.1 (2024), p. 895.

[15] Jonas Ranstam and Jonathan A Cook. "LASSO regression". In: *Journal of British Surgery* 105.10 (2018), pp. 1348–1348.

[16] Jun John Sakurai and Jim Napolitano. *Modern quantum mechanics*. Cambridge University Press, 2020.

[17] Sebastian Bock and Martin Weiß. "A proof of local convergence for the Adam optimizer". In: *2019 international joint conference on neural networks (IJCNN)*. IEEE. 2019, pp. 1–8.

[18] Kazuhiro Seki, Tomonori Shirakawa, and Seiji Yunoki. “Symmetry-adapted variational quantum eigensolver”. In: *Physical Review A* 101.5 (2020), p. 052340.

[19] Tomoyuki Obuchi and Yoshiyuki Kabashima. “Cross validation in LASSO and its acceleration”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2016.5 (2016), p. 053304.