Computing quantum systems

Pre-colloquium

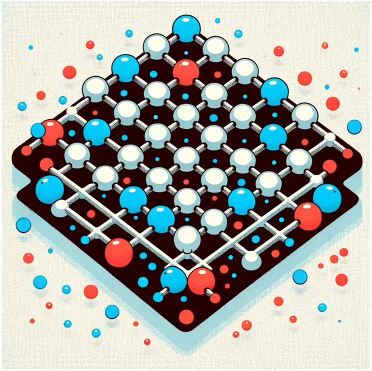
Matteo Robbiati

February 2024

Compute quantum mechanics

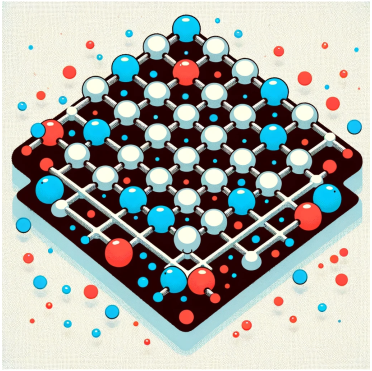
Compute quantum mechanics

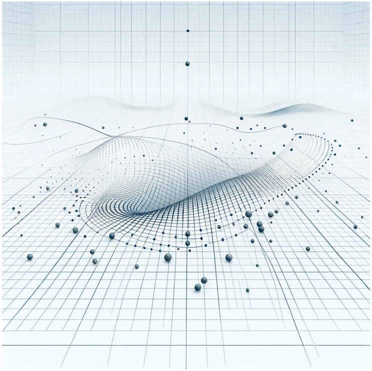
- Representing N particles is difficult;



Compute quantum mechanics

- Representing N particles is difficult;
- considering N spins $(\uparrow, \downarrow)$, we deal with a 2^N dimensional Hilbert space!





What can we do?

What can we do?

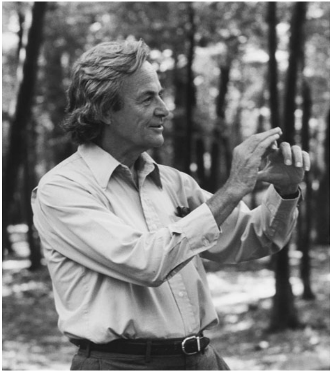
1. we can try to use classical methods to represent the system;

What can we do?

1. we can try to use classical methods to represent the system;
2. we can build a quantum computer.

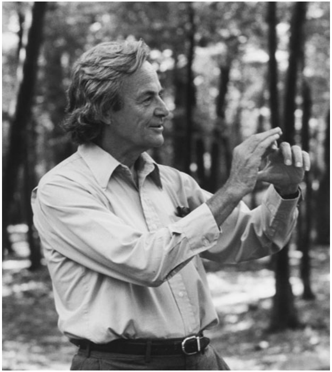
What can we do?

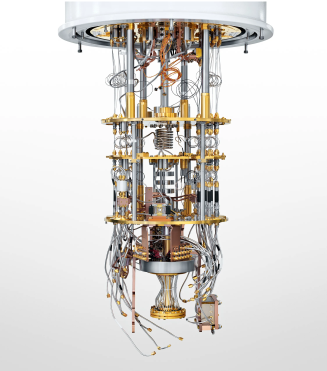
1. we can try to use classical methods to represent the system;
2. we can build a quantum computer.



What can we do?

1. we can try to use classical methods to represent the system;
2. we can build a quantum computer.



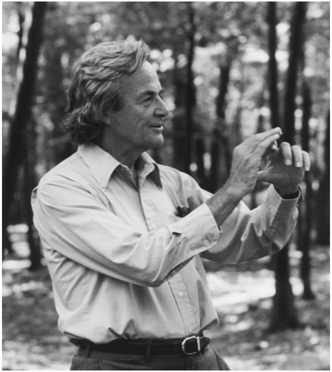


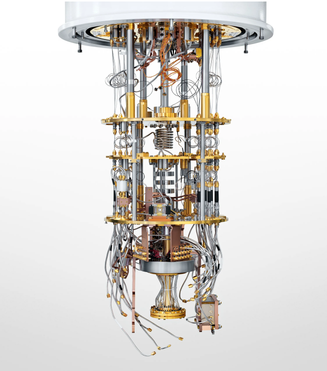
Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy.

Richard Feynman, 1982, Simulating Physics with Computers

What can we do?

1. **we can try to use classical methods to represent the system**;
2. we can build a quantum computer.





Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy.

Richard Feynman, 1982, Simulating Physics with Computers

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;
2. each `complex 128` requires 16 bytes of memory;

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;
2. each `complex 128` requires 16 bytes of memory;
3. let's take a nice 32Gb of RAM: it can store up to 2 billions of `complex 128`.

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;
2. each `complex 128` requires 16 bytes of memory;
3. let's take a nice 32Gb of RAM: it can store up to 2 billions of `complex 128`.
4. a 30 qubits state requires $\sim$ 1 billion of complex numbers;

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;
2. each `complex 128` requires 16 bytes of memory;
3. let's take a nice 32Gb of RAM: it can store up to 2 billions of `complex 128`.
4. a 30 qubits state requires ~ 1 billion of complex numbers;
5. a 31 qubits state cannot be represented by my PC;

Can we represent a state with a classical computer?

Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;
2. each `complex 128` requires 16 bytes of memory;
3. let's take a nice 32Gb of RAM: it can store up to 2 billions of `complex 128`.
4. a 30 qubits state requires ~ 1 billion of complex numbers;
5. a 31 qubits state cannot be represented by my PC;
6. no problem. Let's get serious: `Fugaku`!

Can we represent a state with a classical computer?

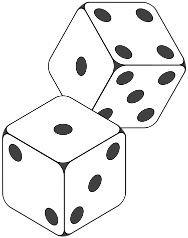
Let's suppose we want to represent a system of **qubits** $(\uparrow, \downarrow)$.

1. Each `float 64` requires 8 bytes of memory to be stored;
2. each `complex 128` requires 16 bytes of memory;
3. let's take a nice 32Gb of RAM: it can store up to 2 billions of `complex 128`.
4. a 30 qubits state requires ~ 1 billion of complex numbers;
5. a 31 qubits state cannot be represented by my PC;
6. no problem. Let's get serious: `Fugaku`!

Some smart strategies

1. Variational Monte Carlo (VMC): given a wave function $\Psi(\boldsymbol{x}|\boldsymbol{\theta})$ and a target H, MC methods are used to minimize:

$$\frac{\int \mathrm{d}\boldsymbol{x}\, \Psi^*(\boldsymbol{x}|\boldsymbol{\theta})\, H\, \Psi(\boldsymbol{x}|\boldsymbol{\theta})}{\int \mathrm{d}\boldsymbol{x}\, |\Psi(\boldsymbol{x}|\boldsymbol{\theta})|^2} \geq E_0;$$



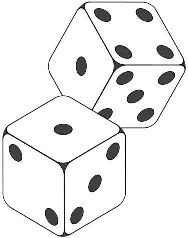
arXiv:1508.02989

Some smart strategies

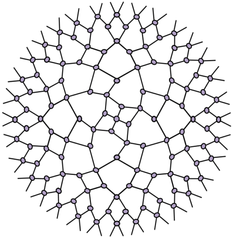
1. Variational Monte Carlo (VMC): given a wave function $\Psi(\boldsymbol{x}|\boldsymbol{\theta})$ and a target H, MC methods are used to minimize:

$$\frac{\int \mathrm{d}\boldsymbol{x}\, \Psi^*(\boldsymbol{x}|\boldsymbol{\theta})\, H\, \Psi(\boldsymbol{x}|\boldsymbol{\theta})}{\int \mathrm{d}\boldsymbol{x}\, |\Psi(\boldsymbol{x}|\boldsymbol{\theta})|^2} \geq E_0;$$

2. Tensor Networks (TNs): contraction of complex systems into simpler structures;



arXiv:1508.02989



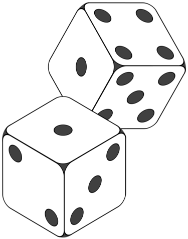
arXiv:1708.00006

Some smart strategies

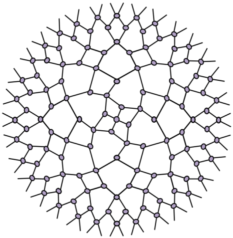
1. Variational Monte Carlo (VMC): given a wave function $\Psi(\boldsymbol{x}|\boldsymbol{\theta})$ and a target H, MC methods are used to minimize:

$$\frac{\int \mathrm{d}\boldsymbol{x}\, \Psi^*(\boldsymbol{x}|\boldsymbol{\theta})\, H\, \Psi(\boldsymbol{x}|\boldsymbol{\theta})}{\int \mathrm{d}\boldsymbol{x}\, |\Psi(\boldsymbol{x}|\boldsymbol{\theta})|^2} \geq E_0;$$

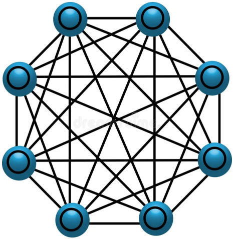
2. Tensor Networks (TNs): contraction of complex systems into simpler structures;
3. Neural Network Quantum States: use complex ANNs to represent the state.



arXiv:1508.02989



arXiv:1708.00006

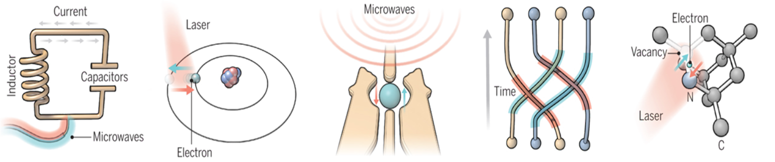


arXiv:1606.02318

A snapshot of quantum computing

Qubits

1. classical bits are replaced by **qubits** $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.



Superconducting loops
A resistance-free current oscillates back and forth around a circuit loop. An injected microwave signal excites the current into super-position states.

Trapped ions
Electrically charged atoms, or ions, have quantum energies that depend on the location of electrons. Tuned lasers cool and trap the ions, and put them in superposition states.

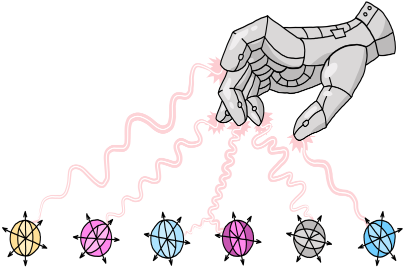
Silicon quantum dots
These "artificial atoms" are made by adding an electron to a small piece of pure silicon. Microwaves control the electron's quantum state.

Topological qubits
Quasiparticles can be seen in the behavior of electrons channeled through semi-conductor structures. Their braided paths can encode quantum information.

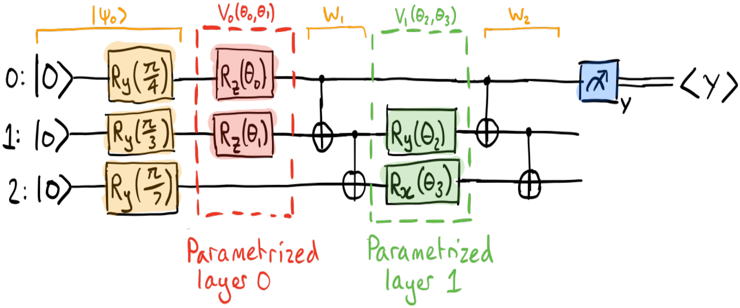
Diamond vacancies
A nitrogen atom and a vacancy add an electron to a diamond lattice. Its quantum spin state, along with those of nearby carbon nuclei, can be controlled with light.

Qubits

1. classical bits are replaced by **qubits** $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.
2. we can manipulate the qubit state applying **gates**: $|\psi'\rangle = \mathcal{U}(\boldsymbol{\theta})\,|\psi\rangle$.
   Typically we use 1-qubit and 2-qubits gates!

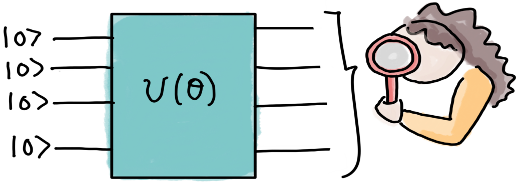


1. classical bits are replaced by **qubits** $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.
2. we can manipulate the qubit state applying **gates**: $|\psi'\rangle = \mathcal{U}(\boldsymbol{\theta})\,|\psi\rangle$.
   Typically we use 1-qubit and 2-qubits gates!
3. combine together gates to build **quantum circuits**;



Qubits

1. classical bits are replaced by **qubits** $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.
2. we can manipulate the qubit state applying **gates**: $|\psi'\rangle = \mathcal{U}(\boldsymbol{\theta})\,|\psi\rangle$.
   Typically we use 1-qubit and 2-qubits gates!
3. combine together gates to build **quantum circuits**;
4. to access the information we need to measure the system.



New computational power

New computational power

With quantum computing, we introduce new tools.

New computational power

With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;

New computational power

With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;
- we can prepare superposition:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

New computational power

With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;
- we can prepare superposition:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{with} \quad H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},\ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix};$$

New computational power

With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;
- we can prepare superposition:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{with} \quad H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},\ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix};$$

- let's apply a controlled-NOT (CNOT) gate on a second qubit prepared in $|0\rangle$:

$$\text{CNOT}\left(\underbrace{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}_{\text{control}} \otimes |0\rangle\right) =$$

New computational power

With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;
- we can prepare superposition:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{with} \quad H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},\ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix};$$

- let's apply a controlled-NOT (CNOT) gate on a second qubit prepared in $|0\rangle$:

$$\mathrm{CNOT}\left(\underbrace{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}_{\text{control}} \otimes |0\rangle\right) = \frac{1}{\sqrt{2}}(|{\color{blue}0}0\rangle + \mathrm{NOT}_{\mathrm{targ}}\,|{\color{red}1}0\rangle) =$$

New computational power

With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;
- we can prepare superposition:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{with} \quad H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},\ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix};$$

- let's apply a controlled-NOT (CNOT) gate on a second qubit prepared in $|0\rangle$:

$$\text{CNOT}\left(\underbrace{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}_{\text{control}} \otimes |0\rangle\right) = \frac{1}{\sqrt{2}}(|00\rangle + \text{NOT}_{\text{targ}}|10\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

New computational power

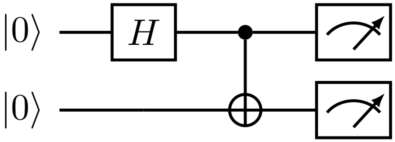
With quantum computing, we introduce new tools.

- prepare a quantum state in the computational zero $|0\rangle$;
- we can prepare superposition:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{with} \quad H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},\ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix};$$

- let's apply a controlled-NOT (CNOT) gate on a second qubit prepared in $|0\rangle$:

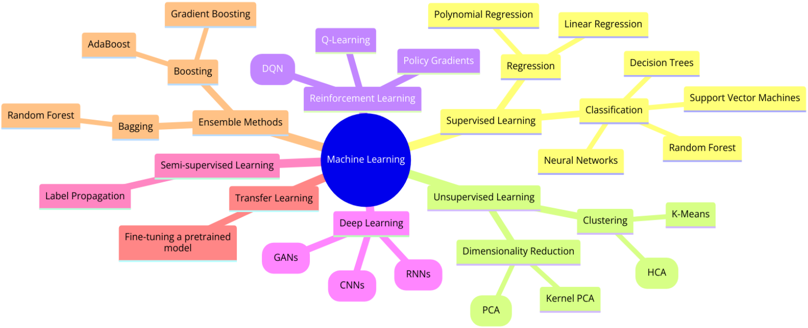
$$\mathrm{CNOT}\left(\underbrace{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)}_{\text{control}} \otimes |0\rangle\right) = \frac{1}{\sqrt{2}}(|00\rangle + \mathrm{NOT}_{\mathrm{targ}}|10\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$



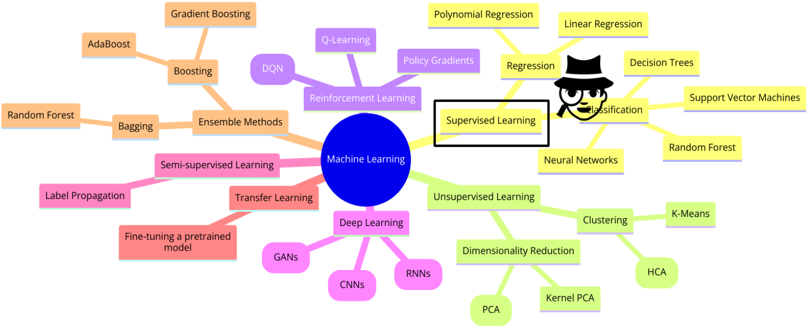
Quantum Machine Learning

Classical Machine Learning

I asked ChatGPT to give me a comprehensive diagram of Machine Learning (ML) models.



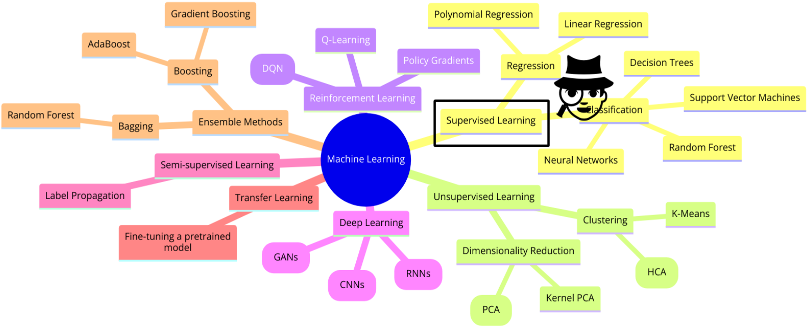
I asked ChatGPT to give me a comprehensive diagram of Machine Learning (ML) models.



Focusing on the supervised ML!

Classical Machine Learning

I asked ChatGPT to give me a comprehensive diagram of Machine Learning (ML) models.

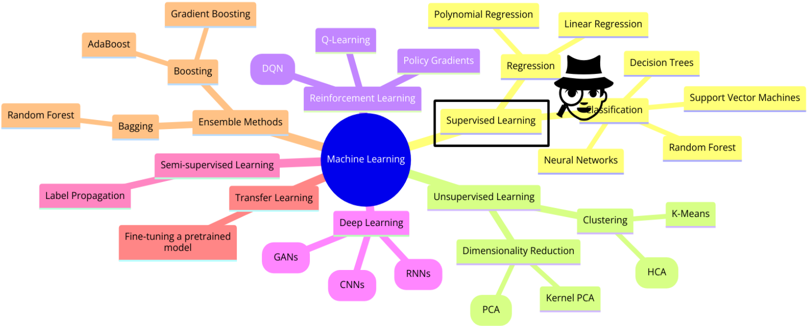


Focusing on the supervised ML!

- we aim to know some hidden law between two variables: $\boldsymbol{y} = f(\boldsymbol{x})$;

Classical Machine Learning

I asked ChatGPT to give me a comprehensive diagram of Machine Learning (ML) models.

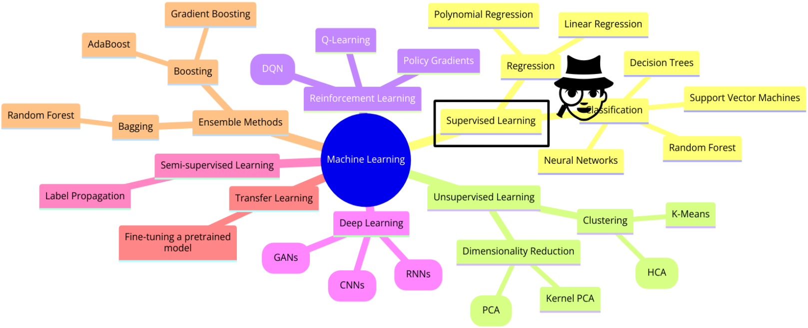


Focusing on the supervised ML!

- we aim to know some hidden law between two variables: $\boldsymbol{y} = f(\boldsymbol{x})$;
- we define a parameteric model which returns $\boldsymbol{y}_{\text{est}} = f_{\text{est}}(\boldsymbol{x}; \boldsymbol{\theta})$;

Classical Machine Learning

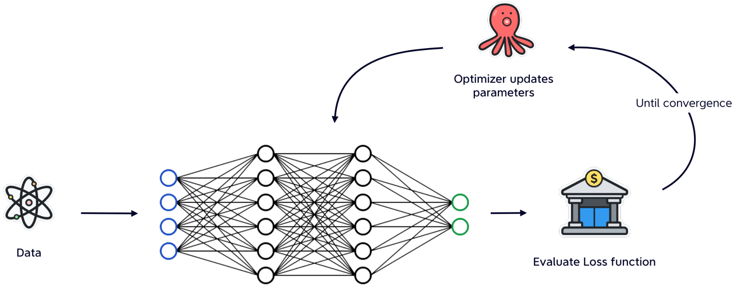
I asked ChatGPT to give me a comprehensive diagram of Machine Learning (ML) models.



Focusing on the supervised ML!

- we aim to know some hidden law between two variables: $\boldsymbol{y} = f(\boldsymbol{x})$;
- we define a parameteric model which returns $\boldsymbol{y}_{\text{est}} = f_{\text{est}}(\boldsymbol{x}; \boldsymbol{\theta})$;
- we define an optimizer, which task is to compute $\text{argmin}_{\boldsymbol{\theta}} \left[J(\boldsymbol{y}_{\text{meas}}, \boldsymbol{y}_{\text{est}}) \right]$.

Classical Machine Learning



Parametric gates

Parametric gates

- Among the gates, parametric ones can be useful!

Parametric gates

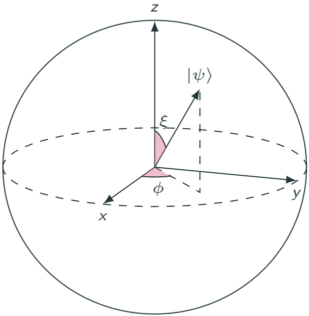
- Among the gates, parametric ones can be useful!
- Let's consider a single qubit system:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

Parametric gates

- Among the gates, parametric ones can be useful!
- Let's consider a single qubit system:

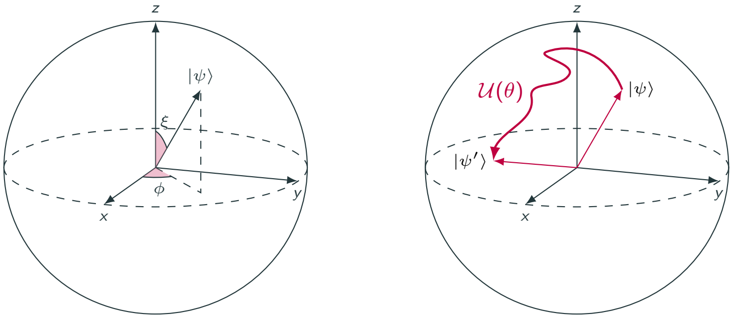
$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad \text{with} \qquad \alpha = \cos\frac{\theta}{2}, \quad \beta = e^{i\phi}\sin\frac{\theta}{2}.$$



Parametric gates

- Among the gates, parametric ones can be useful!
- Let's consider a single qubit system:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad \text{with} \qquad \alpha = \cos\frac{\theta}{2}, \quad \beta = e^{i\phi}\sin\frac{\theta}{2}.$$



We can use as parametric gates the rotation around the axis of the block sphere:

$$R_k(\theta) = \exp\big[-i\theta\sigma_k\big], \qquad \text{with} \qquad \sigma_k \in \{I, \sigma_x, \sigma_y, \sigma_z\}.$$

Quantum Machine Learning

Machine Learning

$\mathcal{M}$: model;

$\mathcal{O}$: optimizer;

$\mathcal{J}$: loss function.

(x, y): data

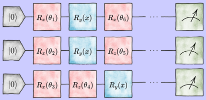
Quantum Computation

$\mathcal{Q}$: qubits;

$\mathcal{S}$: superposition;

$\mathcal{E}$: entanglement.

Circuit execution



Expected values

$y_{est} \equiv \langle \psi' | \hat{O} | \psi' \rangle$

Machine Learning

$\mathcal{M}$: model;

$\mathcal{O}$: optimizer;

$\mathcal{J}$: loss function.

(x, y): data

Expected values

$y_{est} \equiv \boxed{\langle\psi'|\hat{O}|\psi'\rangle}$

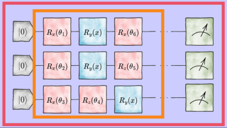
Quantum Computation

$\mathcal{Q}$: qubits;

$\mathcal{S}$: superposition;

$\mathcal{E}$: entanglement.

Circuit execution



Quantum Machine Learning!

Optimizer $\mathcal{O}$
Hybrid-strategy

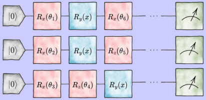
Machine Learning
$\mathcal{M}$: model;
$\mathcal{O}$: optimizer;
$\mathcal{J}$: loss function.
(x, y): data

loss function $\mathcal{J}$
$\mathcal{J}(y_{meas}, y_{est})$

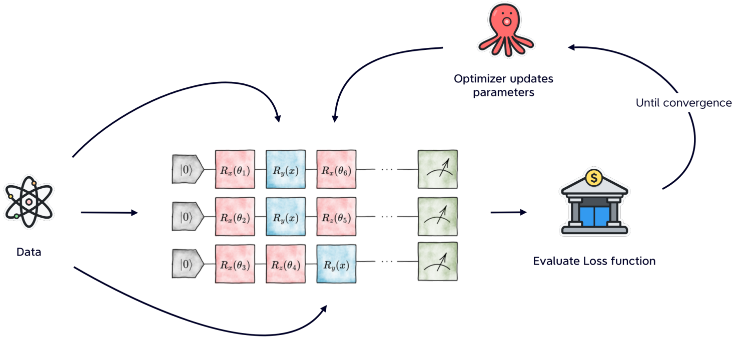
Expected values
$y_{est} \equiv \langle\psi'|\hat{O}|\psi'\rangle$

Quantum Computation
$\mathcal{Q}$: qubits;
$\mathcal{S}$: superposition;
$\mathcal{E}$: entanglement.

Circuit execution



From ML to QML



But can this work?

But can this work?

1. Depend on the problem

But can this work?

1. Depend on the problem, but what better of playing in the Hilbert space when tackling a many body system?

But can this work?

1. Depend on the problem, but what better of playing in the Hilbert space when tackling a many body system?
2. We can take some more rational proof of utility.

But can this work?

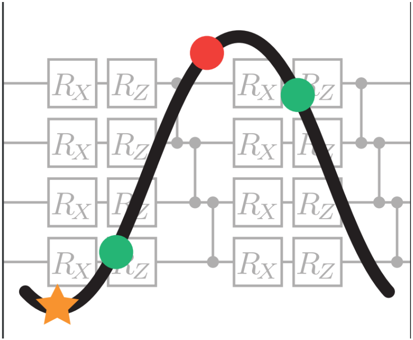
1. Depend on the problem, but what better of playing in the Hilbert space when tackling a many body system?
2. We can take some more rational proof of utility.

Using Variational Quantum Circuits we define a Variational Quantum Computer!

But can this work?

1. Depend on the problem, but what better of playing in the Hilbert space when tackling a many body system?
2. We can take some more rational proof of utility.

Using Variational Quantum Circuits we define a Variational Quantum Computer!

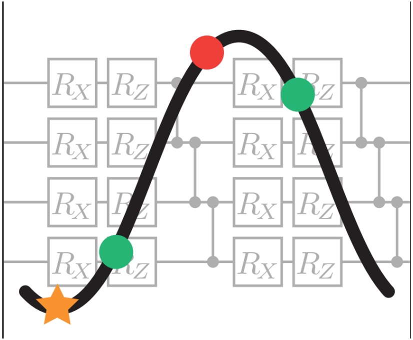


But can this work?

1. Depend on the problem, but what better of playing in the Hilbert space when tackling a many body system?
2. We can take some more rational proof of utility.

Using Variational Quantum Circuits we define a Variational Quantum Computer!

1. we want a quantum circuit $\mathcal{U}(\boldsymbol{\theta})$ to approximates some law V;

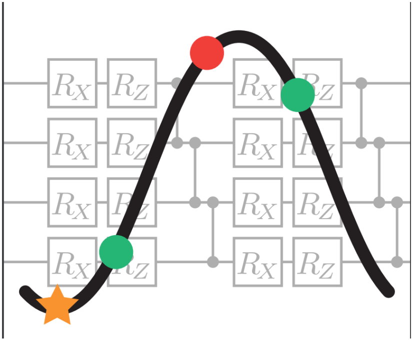


But can this work?

1. Depend on the problem, but what better of playing in the Hilbert space when tackling a many body system?
2. We can take some more rational proof of utility.

Using Variational Quantum Circuits we define a Variational Quantum Computer!

1. we want a quantum circuit $\mathcal{U}(\boldsymbol{\theta})$ to approximates some law V;
2. executing $\mathcal{U}(\boldsymbol{\theta})$ we use a variational quantum state to reach the solution;

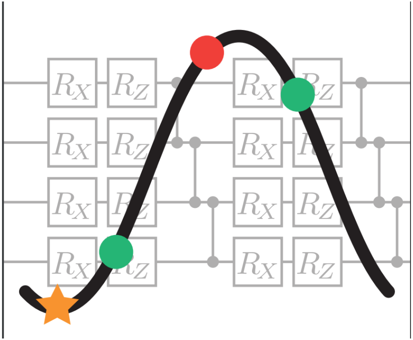


But can this work?

1. Depend on the problem, but what better of playing in the Hilbert space when tackling a many body system?
2. We can take some more rational proof of utility.

Using Variational Quantum Circuits we define a Variational Quantum Computer!

1. we want a quantum circuit $\mathcal{U}(\boldsymbol{\theta})$ to approximates some law V;
2. executing $\mathcal{U}(\boldsymbol{\theta})$ we use a variational quantum state to reach the solution;
3. **Solovay-Kitaev theorem**: the number of gates needed by $\mathcal{U}$ to represent V with precision δ is $\mathcal{O}(\log^c \delta^{-1})$, where $c < 4$.



A practical example

Approximating ground states - Adiabatic Computing

Approximating ground states - Adiabatic Computing

What is a quantum annealer?

Approximating ground states - Adiabatic Computing

What is a quantum annealer?

- A QC with many spins arranged on a lattice connected to nearest neighbors;

Approximating ground states - Adiabatic Computing

What is a quantum annealer?

- A QC with many spins arranged on a lattice connected to nearest neighbors;
- typically Ising models can be encoded into a quantum annealer!

Approximating ground states - Adiabatic Computing

What is a quantum annealer?

- A QC with many spins arranged on a lattice connected to nearest neighbors;
- typically Ising models can be encoded into a quantum annealer!

The game is to slowly (*adiabatically*) move an H_0 to a target Hamiltonian H_1, in which we encode the problem:

Approximating ground states - Adiabatic Computing

What is a quantum annealer?

- A QC with many spins arranged on a lattice connected to nearest neighbors;
- typically Ising models can be encoded into a quantum annealer!

The game is to slowly (*adiabatically*) move an H_0 to a target Hamiltonian H_1, in which we encode the problem:

$$H_{ad} = H_0 \big[1 - s(\tau)\big] + s(\tau) H_1 .$$

Approximating ground states - Adiabatic Computing

What is a quantum annealer?

- A QC with many spins arranged on a lattice connected to nearest neighbors;
- typically Ising models can be encoded into a quantum annealer!

The game is to slowly (*adiabatically*) move an H_0 to a target Hamiltonian H_1, in which we encode the problem:

$$H_{ad} = H_0 \big[1 - s(\tau)\big] + s(\tau) H_1.$$

If adiabatic enough, you'll remain in the ground state.

Approximating ground states - Adiabatic Computing

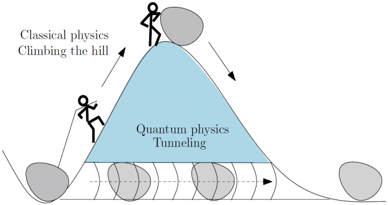
What is a quantum annealer?

- A QC with many spins arranged on a lattice connected to nearest neighbors;
- typically Ising models can be encoded into a quantum annealer!

The game is to slowly (*adiabatically*) move an H_0 to a target Hamiltonian H_1, in which we encode the problem:

$$H_{ad} = H_0\big[1 - s(\tau)\big] + s(\tau)H_1.$$

If adiabatic enough, you'll remain in the ground state.



arXiv:2207.01827

Approximating ground states - Adiabatic Computing

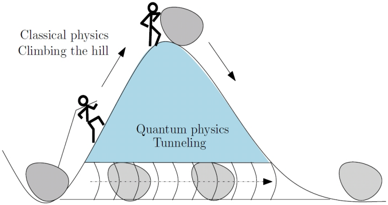
What is a quantum annealer?

- A QC with many spins arranged on a lattice connected to nearest neighbors;
- typically Ising models can be encoded into a quantum annealer!

The game is to slowly (*adiabatically*) move an H_0 to a target Hamiltonian H_1, in which we encode the problem:

$$H_{ad} = H_0 \big[1 - s(\tau)\big] + s(\tau) H_1 .$$

If adiabatic enough, you'll remain in the ground state.



arXiv:2207.01827

https://www.dwavesys.com/

This method is known as Variational Quantum Eigensolver (VQE).

This method is known as Variational Quantum Eigensolver (VQE).

1. consider a target Hamiltonian H_{target}, whose ground state is $|\psi_0\rangle$;

This method is known as Variational Quantum Eigensolver (VQE).

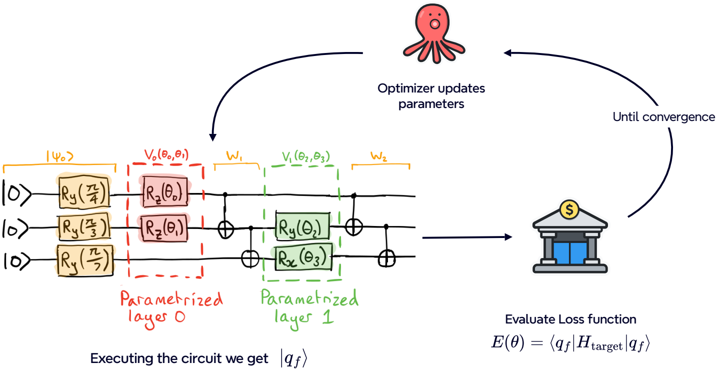
1. consider a target Hamiltonian H_{target}, whose ground state is $|\psi_0\rangle$;
2. we take a parametrized quantum circuit $\mathcal{U}(\boldsymbol{\theta})$, with action $|q_f\rangle = \mathcal{U}(\boldsymbol{\theta})\,|0\rangle$

This method is known as Variational Quantum Eigensolver (VQE).

1. consider a target Hamiltonian H_{target}, whose ground state is $|\psi_0\rangle$;
2. we take a parametrized quantum circuit $\mathcal{U}(\boldsymbol{\theta})$, with action $|q_f\rangle = \mathcal{U}(\boldsymbol{\theta})\,|0\rangle$
3. the goal is to train $\mathcal{U}(\boldsymbol{\theta})$ to return a $|q_f\rangle = |\tilde{\psi}_0\rangle$.

This method is known as Variational Quantum Eigensolver (VQE).

1. consider a target Hamiltonian H_{target}, whose ground state is $|\psi_0\rangle$;
2. we take a parametrized quantum circuit $\mathcal{U}(\boldsymbol{\theta})$, with action $|q_f\rangle = \mathcal{U}(\boldsymbol{\theta})\,|0\rangle$
3. the goal is to train $\mathcal{U}(\boldsymbol{\theta})$ to return a $|q_f\rangle = |\tilde{\psi}_0\rangle$.

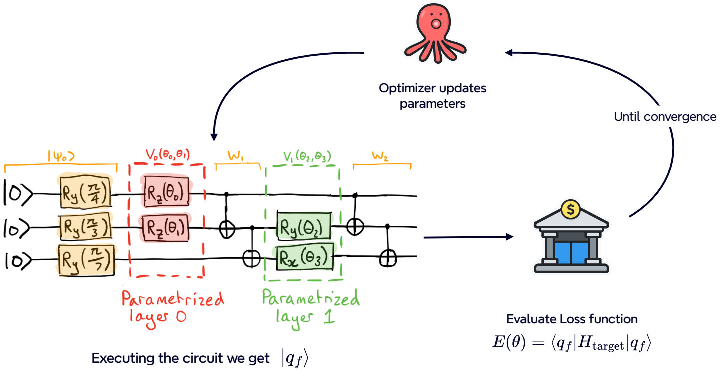


Approximating ground states - VQE

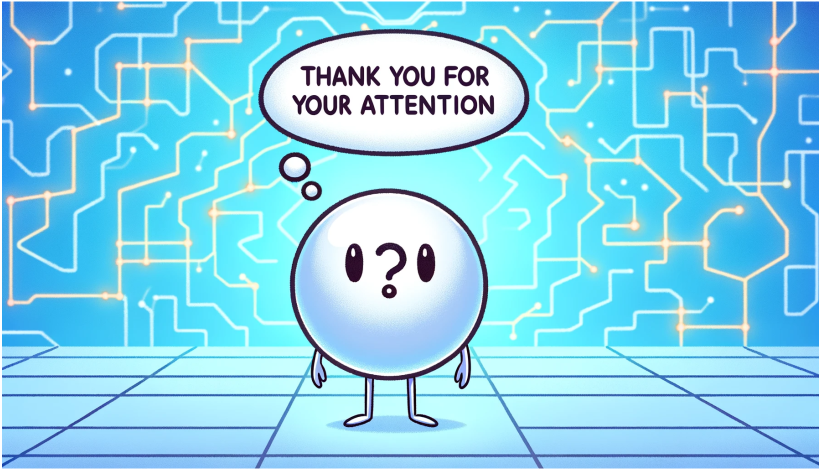
arXiv:1304.3061

This method is known as Variational Quantum Eigensolver (VQE).

1. consider a target Hamiltonian H_{target}, whose ground state is $|\psi_0\rangle$;
2. we take a parametrized quantum circuit $\mathcal{U}(\boldsymbol{\theta})$, with action $|q_f\rangle = \mathcal{U}(\boldsymbol{\theta})\,|0\rangle$
3. the goal is to train $\mathcal{U}(\boldsymbol{\theta})$ to return a $|q_f\rangle = |\tilde{\psi}_0\rangle$.



>_ Let's code it!


THANK YOU FOR
YOUR ATTENTION