Introduction to Quantum Machine Learning

EVA: Quantum Machine Learning – Lecture 1

Pavel Sulimov

Spring 2026

ZHAW Zürcher Hochschule für Angewandte Wissenschaften

Course logistics

- 14 classes: 1.5 h lecture + 45 min practice
- **Tools:** Qiskit, PennyLane, PyTorch, NumPy
- **Assessment:** final project presentation (pass/fail)
- All materials distributed as Jupyter notebooks and slides

A note on AI coding assistants

Using AI tools (ChatGPT, Claude, Cursor, Copilot, ...) is **welcome**. Quantum computing is hard enough without pretending the tools do not exist.

But be careful: Qiskit and PennyLane APIs change fast.

Common AI mistake	**What changed**
`execute(circuit, backend)`	Removed in Qiskit 1.0
`BasicAer`, `qiskit.opflow`	Replaced by primitives API

Rules of engagement:

1. Tell the AI which versions you use (e.g. Qiskit 2.x, PennyLane 0.40+)
2. **Run every snippet** before trusting it
3. Cross-check: `docs.quantum.ibm.com`, `docs.pennylane.ai`

Prerequisites

Rate yourself 1–5:

1. Linear algebra (vectors, matrices, eigenvalues)
2. Probability and statistics
3. Python programming
4. Machine learning basics (supervised learning, gradient descent)
5. Quantum computing (any exposure?)

No prior quantum knowledge required. We start from scratch.

What is QML? A five-way taxonomy

QML is a family of approaches, not one field.

c

Approach	Core idea	Hardware
Quantum-inspired	Tensor networks, dequantized alg.	Classical
Quantum-enhanced	Variational circuits + classical optimizer	NISQ
Quantum-native	Fully quantum alg. (HHL, quantum PCA)	Fault-tol.
Physics-informed	Quantum circuit as PDE solver (QPINN)	NISQ
Quantum-informed	Classical NN shaped by quantum calc.	Classical

NISQ = Noisy Intermediate-Scale Quantum. HHL = Harrow-Hassidim-Lloyd. PDE = Partial Differential Equation.

The QML landscape

	Classical processing	Quantum processing
Classical data	Classical ML (our baseline)	Hybrid QML, kernels QPINNs (classical PDEs)
Quantum data	Quantum-informed Quantum-inspired	Quantum simulation error correction

QPINNs can appear in multiple quadrants depending on whether the PDE describes classical or quantum physics.

Why quantum for data? State-space bottleneck

An n-qubit pure state has 2^n complex amplitudes:

$$|\psi\rangle = \sum_{x=0}^{2^n-1} \alpha_x |x\rangle , \qquad \sum_x |\alpha_x|^2 = 1$$

Exact classical simulation stores all amplitudes explicitly (about 16 bytes each in double precision):

Qubits	**Amplitudes**	**Statevector memory**
30	$2^{30} \approx 1.07 \times 10^9$	$\sim$ 16 GB
40	$2^{40} \approx 1.10 \times 10^{12}$	$\sim$ 16 TB
50	$2^{50} \approx 1.13 \times 10^{15}$	$\sim$ 16 PB

Quantum hardware evolves this state physically with gates, while a classical statevector simulator must materialize the amplitudes in memory.

Where are we? The NISQ era and quantum utility

Current hardware (as of 2025–2026):

- IBM Eagle/Heron: 100 to 1000+ qubits
- Google Willow, IonQ, Quantinuum
- Error rates around 10^{-3}, coherence times around $100\,\mu$s

Landmark result: a 127-qubit IBM Eagle processor simulating an Ising model matched exact solutions in a regime where classical brute force was not feasible within comparable resources.

This is “quantum utility,” not “supremacy”: tensor-network methods have since matched the result, but quantum hardware is now competing with classical supercomputers on scientific problems.

Kim et al., *Nature* 618 (2023).

Software ecosystem

Framework	**Strengths**	**Backend**
Qiskit (IBM)	Hardware access, primitives API	IBM Quantum
PennyLane (Xanadu)	Autodiff, PyTorch/JAX integration	Multiple
Cirq (Google)	Google hardware	Google QCS
Amazon Braket	Multi-hardware marketplace	AWS

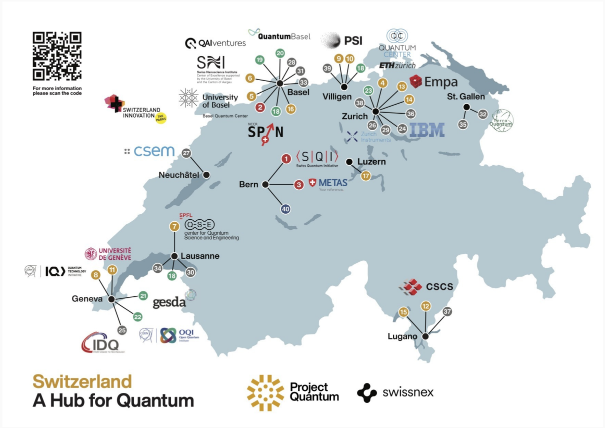
This course: **Qiskit** + **PennyLane** + **PyTorch**.

Why QML now?

Several developments are converging:

1. Hardware reaching **utility scale** (IBM 127+ qubits)
2. The **variational/hybrid** paradigm works on real devices
3. Open-source tooling (Qiskit, PennyLane) makes experiments easy
4. Theoretical results on **expressivity** and quantum kernels
5. Growing **physics-informed ML** community (PINNs, QPINNs)
6. Industry adoption: QuantumBasel, IBM Quantum Network, D-Wave

The Swiss quantum ecosystem



Source: Swissnex Quantum Mapping, October 2025.

Course roadmap

Block	Theme	Classes
I	Foundations	1–3
II	Data & Encoding	4–5
III	Variational QML	6–9
IV	Advanced Topics	10–12
V	Frontiers & Projects	13–14

I: Found.
II: Encoding
III: Variat.
IV: Adv.
V: Frontiers

What you will be able to do by the end

- Build and train a **hybrid quantum-classical classifier**
- Understand **quantum kernel methods** and when they help
- Implement and analyze **variational quantum circuits**
- Know when **QPINNs** are appropriate for scientific computing
- Critically evaluate **quantum advantage claims**
- Use Qiskit and PennyLane fluently

Classical ML in 10 minutes

Machine learning paradigms

Before we enter the quantum world, we need a shared vocabulary for the classical ML concepts that QML builds on.

Supervised Given pairs (x_i, y_i), learn $f : x \to y$.
Classification, regression.

Unsupervised Given $\{x_i\}$ only, find structure.
Clustering, dimensionality reduction.

Reinforcement Agent interacts with environment, maximizes reward.

This course focuses mostly on **supervised learning**. Think of the quantum model as a black box with tunable parameters, similar to a neural network.

Supervised learning as function approximation

Given dataset $\{(x_i, y_i)\}_{i=1}^N$, find parameters θ:

$$f_\theta(x) \approx y$$

The parameters θ are learned by minimizing a **loss function**:

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\big(f_\theta(x_i),\ y_i\big)$$

The model f_θ can be a neural network, a support vector machine (SVM), a decision tree, or, as we will see, a quantum circuit.

The role of linear algebra

Almost everything in QML reduces to linear algebra:

- Data points are **vectors** $x \in \mathbb{R}^d$
- Transformations are **matrices**: $y = Wx + b$
- Similarity is an **inner product**: $\langle x, x' \rangle$

In quantum computing:

- States are vectors in $\mathbb{C}^{2^n}$
- Gates are **unitary** matrices: $U^\dagger U = I$
- Overlap is $\langle \psi | \phi \rangle$

Recommended: 3Blue1Brown, "Essence of Linear Algebra" (YouTube playlist).

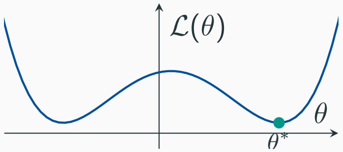
Loss functions

Regression: Mean Squared Error

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{N}\sum_{i=1}^{N}\big(f_\theta(x_i) - y_i\big)^2$$

Classification: Cross-entropy

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log f_\theta(x_i)$$



Gradient descent

Update rule:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t)$$

where η is the learning rate.

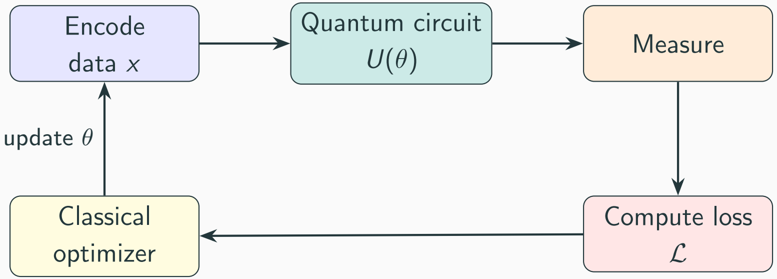
Variants: stochastic gradient descent (SGD), Adam, L-BFGS, ...

Key question for QML: How do we compute $\nabla_\theta \mathcal{L}$ when f_θ is a quantum circuit?

$\rightarrow$ Parameter-shift rule (Class 7)

The hybrid quantum-classical loop (teaser)

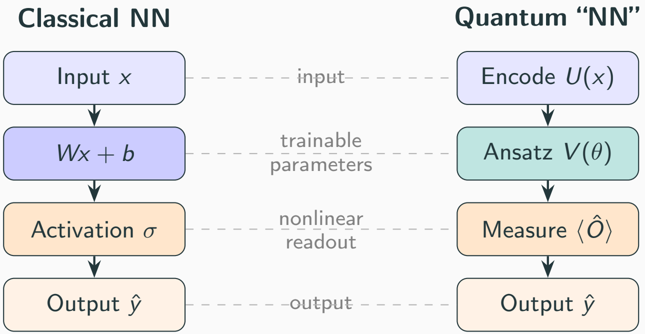
The same optimize-and-update logic applies when the model is a quantum circuit:



The "model" is a quantum circuit. The optimizer is classical.
We will make this precise in Part 3 and in Class 6.

`lecture_01_demo.ipynb`, Demo 5: sweep $R_y(\theta)$ and watch how a single gate parameter controls the output probability.

Classical neural network vs quantum “neural network”



Weights W	$\longleftrightarrow$	gate angles θ
Activation σ	$\longleftrightarrow$	measurement
Network depth	$\longleftrightarrow$	circuit depth

Physics-Informed Neural Networks (PINNs)

A neural network $f_\theta(x, t)$ approximates the solution of a partial differential equation (PDE):

$$\mathcal{L} = \underbrace{\sum_i \left(f_\theta(x_i) - u_i\right)^2}_{\text{data mismatch}} + \lambda \underbrace{\sum_j \left|\mathcal{N}[f_\theta](x_j)\right|^2}_{\text{PDE residual}}$$

where $\mathcal{N}$ is the differential operator.

No labeled data needed if the physics is known.

Key idea: domain knowledge as a regularizer.

Raissi, Perdikaris, Karniadakis (2019), *J. Comput. Phys.*

From PINNs to QPINNs (teaser)

Replace the classical NN with a **parameterized quantum circuit** (PQC):

$$f_\theta(x) \;\rightarrow\; \langle U(\theta, x)\, 0 \cdots 0 | O | U(\theta, x)\, 0 \cdots 0 \rangle$$

The loss function stays the same (PDE residual), but the function approximator is quantum.

Potential advantages:

- Expressivity in exponentially large Hilbert space
- Natural encoding of certain symmetries

Active research: Schrödinger equation, Maxwell's equations, fluid dynamics.

$\rightarrow$ Covered in depth in Class 12.

QPINN = Quantum Physics-Informed Neural Network. PQC = Parameterized Quantum Circuit.

Quantum-informed NN vs. quantum physics-informed NN

These two acronyms sound alike but differ in architecture:

	QINN (quantum-informed)	**QPINN** (quantum physics-informed)
Model	**Classical** neural network	**Quantum** circuit (PQC)
Quantum role	Quantum calculations supply training data, symmetry constraints, or regularization terms	The PQC *is* the function approximator
Loss function	Standard ML loss, possibly with a quantum-derived regularizer	$\mathcal{L}_{\text{data}} + \lambda \mathcal{L}_{\text{PDE}}$ (physics residual)
Hardware	Runs on classical hardware	Requires quantum processor or simulator
Example	Classical NN trained on quantum chemistry potentials	PQC solving the Schrödinger equation

In short: QINN uses quantum *knowledge* to improve a classical model. QPINN uses a quantum

Kernel methods

Data not linearly separable $\to$ map to higher-dimensional space $\to$ linear separator in that space.

Feature map: $\phi : \mathbb{R}^d \to \mathbb{R}^D, \quad D \gg d$

Kernel trick: compute similarity without mapping explicitly:

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

Examples: polynomial kernel, radial basis function (RBF) kernel.

Quantum kernels: the punchline

Quantum circuits produce states in an **exponentially large** Hilbert space $\mathbb{C}^{2^n}$.

If we can compute the overlap efficiently:

$$K_Q(x_i, x_j) = \left| \langle 0 | \, U^\dagger(x_j) \, U(x_i) \, | 0 \rangle \right|^2$$

we get a **quantum kernel**: an implicit feature map in 2^n dimensions.

We will build this from scratch in Classes 4–5 and 8.

But first, we need to understand the building blocks: qubits, gates, and measurement.

From bits to qubits

Classical bit

A two-state system: 0 or 1.

Physical realization: voltage levels, magnetic domains.

The state is **deterministic**: reading the bit gives a definite answer, and you can read it as many times as you like without changing it.

Qubit

Also a two-state system, but can be in **superposition**:

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

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.

- α, β are **probability amplitudes** (complex numbers!)
- The state encodes more than one bit of information, but measurement extracts only 1 bit
- Measurement **destroys** the superposition

Dirac notation

Ket (column vector):

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Bra (conjugate-transpose row vector): $\langle\psi| = |\psi\rangle^\dagger$

Inner product: $\langle\phi|\psi\rangle \in \mathbb{C}$

Orthonormality: $\langle 0|1\rangle = 0, \quad \langle 0|0\rangle = \langle 1|1\rangle = 1.$

Why complex numbers?

Quick reminder:

$$
\begin{aligned}
z &= a + bi, \quad i^2 = -1 \\
|z| &= \sqrt{a^2 + b^2} \\
z^* &= a - bi \\
e^{i\theta} &= \cos\theta + i \sin\theta \quad \text{(Euler's formula)}
\end{aligned}
$$

Quantum mechanics *requires* complex numbers. This is not a mathematical convenience; it is physics.

Recommended: 3Blue1Brown, "$e^{i\pi}$ in 3.14 minutes, using dynamics" (`https://youtu.be/v0YEaeIClKY`).

Born rule

For $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, measurement in the computational basis gives:

$$P(0) = |\alpha|^2, \qquad P(1) = |\beta|^2$$

Normalization: $P(0) + P(1) = |\alpha|^2 + |\beta|^2 = 1$

Measurement is destructive: after measuring, the state **collapses** to the observed outcome.

You cannot "peek" at a quantum state without changing it.

Worked example

$$|\psi\rangle = \frac{3}{5}|0\rangle + \frac{4i}{5}|1\rangle$$

$$P(0) = \left|\frac{3}{5}\right|^2 = \frac{9}{25} = 0.36$$

$$P(1) = \left|\frac{4i}{5}\right|^2 = \frac{16}{25} = 0.64$$

Check: $\frac{9}{25} + \frac{16}{25} = 1$ ✓

Note: the *i* in front of $\frac{4}{5}$ does not affect the probability. $|4i/5|^2 = (4/5)^2 = 16/25$. But that *i* *does* matter for interference, as we will see shortly.

Special states

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Both give 50/50 measurement in the Z-basis.

But they are **orthogonal**: $\langle + | - \rangle = 0$.

Different quantum states can give the *same* statistics in one measurement basis but are perfectly distinguishable in another.

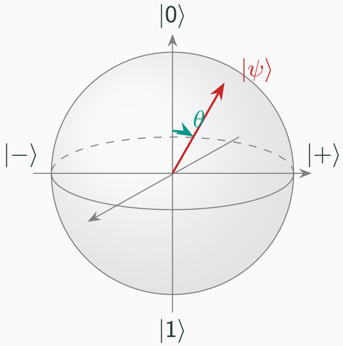
This is the first hint that there is more to a quantum state than just probabilities.

The Bloch sphere

Any single-qubit pure state:

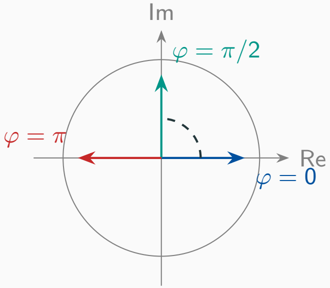
$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$$

with $\theta \in [0,\pi]$, $\varphi \in [0,2\pi)$.



Phase: what it is and why it matters

Think of a complex amplitude as an **arrow in the plane**: its length is the probability amplitude, its angle is the phase.



All three arrows have the **same length**, so they give the same measurement probability. But when two amplitudes *add up* (interference), their angles determine whether they reinforce or cancel. This is how quantum computing gets its power.

Global vs. relative phase

Global phase: $e^{i\gamma}|\psi\rangle$ and $|\psi\rangle$ produce identical measurement outcomes for any experiment. Global phase is **unobservable**.

Analogy: translating every clock in the room by the same amount does not change the time differences between them.

Relative phase: in $\alpha|0\rangle + e^{i\varphi}\beta|1\rangle$, the angle φ between the two amplitudes is physical. It changes the point on the Bloch sphere.

Analogy: two pendulums swinging with the same amplitude but different starting positions will produce different interference patterns when they meet.

Phase example: $|+\rangle$ vs $|-\rangle$

$|+\rangle$ and $|-\rangle$ differ only by a relative phase of π:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \qquad |-\rangle = \frac{|0\rangle + e^{i\pi}|1\rangle}{\sqrt{2}} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Measure in the Z-basis (standard $|0\rangle$ / $|1\rangle$): both give 50/50. **Indistinguishable.**

Measure in the X-basis ($|+\rangle$ / $|-\rangle$): $|+\rangle$ always gives $+1$, $|-\rangle$ always gives -1. **Perfectly distinguishable.**

The right measurement basis reveals information that was hidden in the phase.

`lecture_01_demo.ipynb`, Demo 4: verify this with Qiskit.

Quick check

Is $\frac{1}{\sqrt{2}}\big(|0\rangle + i\,|1\rangle\big)$ the same state as $\frac{i}{\sqrt{2}}\big(|0\rangle + i\,|1\rangle\big)$?

Quick check

Is $\frac{1}{\sqrt{2}}\big(|0\rangle + i\,|1\rangle\big)$ the same state as $\frac{i}{\sqrt{2}}\big(|0\rangle + i\,|1\rangle\big)$?

Yes. The overall factor *i* is a global phase.

But the *i* multiplying $|1\rangle$ in the first expression is a *relative* phase between $|0\rangle$ and $|1\rangle$.
That one is physical and puts the state on the equator of the Bloch sphere at $\varphi = \pi/2$.

Comparing bits and qubits

	Classical bit	**Qubit**
State	deterministic: 0 or 1	probabilistic: $\alpha \|0\rangle + \beta \|1\rangle$
Information	1 bit	continuous (θ, φ)
Copying	trivial	no-cloning theorem
Reading	identity operation	measurement = collapse
Math	$\{0, 1\}$	$\mathbb{C}^2$, unit sphere

Physical realizations of qubits

Platform	Companies	Qubit type
Superconducting	IBM, Google	Transmon
Trapped ions	IonQ, Quantinuum	Atomic ions
Photonic	Xanadu	Squeezed light
Neutral atoms	QuEra	Rydberg atoms

Different platforms, same linear algebra underneath.

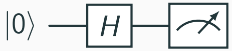
The Hadamard gate

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Action:

$$H\left|0\right\rangle = \left|+\right\rangle, \qquad H\left|1\right\rangle = \left|-\right\rangle$$

Creates equal superposition from a basis state.

As a circuit: 

Live demo: Hadamard in Qiskit

```
from qiskit import QuantumCircuit
from qiskit.primitives import StatevectorSampler

qc = QuantumCircuit(1)
qc.h(0)
qc.measure_all()

sampler = StatevectorSampler()
result = sampler.run([qc], shots=1024).result()
counts = result[0].data.meas.get_counts()
print(counts)  # {’0’: ~512, ’1’: ~512}
```

Full runnable code in `lecture_01_demo.ipynb`.

Same circuit in PennyLane

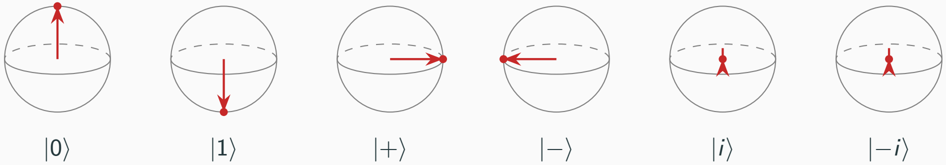
```
import pennylane as qml

dev = qml.device("default.qubit", wires=1)

@qml.qnode(dev)
def circuit():
    qml.Hadamard(wires=0)
    return qml.probs(wires=0)

print(circuit())  # [0.5, 0.5]
```

Same physics, different framework.
PennyLane returns exact probabilities (no shot noise by default).

Bloch sphere visualization

States to visualize in the demo notebook and practice:

$$|0\rangle,\ |1\rangle,\ |+\rangle,\ |-\rangle,\ |i\rangle = \frac{|0\rangle+i|1\rangle}{\sqrt{2}},\ |-i\rangle = \frac{|0\rangle-i|1\rangle}{\sqrt{2}}$$



`lecture_01_demo.ipynb`, Demo 3: interactive 3D Bloch spheres.

Recap: from bits to qubits

- Qubit = unit vector in $\mathbb{C}^2$
- Measurement = projection (probabilistic, destructive)
- Bloch sphere = geometric picture of single-qubit states
- Global phase is unobservable; relative phase produces interference
- Hadamard gate creates superposition from basis states

With these tools in hand, we can ask: does any of this actually help with machine learning?

Can quantum help ML?

Speed-up claims

Grover's search: $O(\sqrt{N})$ vs $O(N)$, a quadratic speed-up.

Harrow-Hassidim-Lloyd (HHL): exponential speed-up for solving linear systems $Ax = b$, under specific conditions on the matrix A.

Caveats:

- Data loading bottleneck (no practical quantum RAM exists)
- Dequantization: classical algorithms inspired by quantum ones can sometimes match the speed-up (Tang, 2018)
- Readout: extracting the full solution may negate the speed-up

Expressivity argument

An n-qubit circuit acts in a 2^n-dimensional space.

A parameterized quantum circuit with $O(n)$ parameters can represent functions that would need $O(2^n)$ parameters classically.

When does this matter?

- When the target function has structure aligned with the quantum circuit architecture
- When the data lives in a space where quantum features are naturally useful

Not always true, but sometimes genuinely useful.

Kernel argument

Quantum feature maps produce kernel matrices that are **classically hard to compute**:

$$K_Q(x, x') = \left| \langle 0^n | \, U^\dagger(x') \, U(x) \, | 0^n \rangle \right|^2$$

If the data structure *aligns* with the quantum kernel, there is a genuine advantage.

Recent result (2025): mitigating exponential concentration in covariant quantum kernels.

Agliardi et al., *npj Quantum Information* (2025).

Fewer parameters

Empirical evidence: quantum models with $\sim$10 parameters matching classical neural networks with hundreds of parameters on certain datasets.

Interpretation: the quantum circuit architecture encodes assumptions about the data (a “structural bias”) that happen to be useful for the task.

Example: a quantum circuit naturally operates in a unitary (norm-preserving) space. For problems with conservation laws or symmetries, this can be a good match.

This is neither proven in general nor universally true, but it is a recurring pattern in experiments.

The QPINN argument

For problems governed by known physical laws:

- Schrödinger equation (quantum chemistry)
- Maxwell's equations (electromagnetics)
- Navier-Stokes (fluid dynamics)

QPINNs offer a path where the quantum circuit's expressivity is directly useful for representing solutions in **high-dimensional spaces**.

Challenges: barren plateaus, circuit depth, noise.

Active research area. Not yet mature, but promising.

Reality check

- Most current QML results use **small datasets with few qubits**
- **Barren plateaus** make training hard at scale (Class 13)
- Hardware **noise** limits practical circuit depth
- No proven end-to-end quantum advantage for practical ML yet
- Classical methods keep improving (tensor networks, dequantization)

Honest framing matters. Hype without evidence harms the field.

The pragmatic view

Even without exponential speed-up, QML is valuable because:

1. Quantum circuit architectures encode **structural assumptions** (unitarity, symmetry) that classical NNs do not
2. Quantum-inspired classical methods (tensor networks) are already useful in production
3. QPINNs connect QML to the broader **scientific computing community**
4. Quantum reservoir computing avoids barren plateaus entirely
5. The field is young. We are training the researchers who will build what comes next

Summary and what comes next

Today:

- QML = family of approaches (five-way taxonomy)
- Qubits, superposition, Bloch sphere, Born rule
- Phase is the key resource for quantum computing
- Quantum advantage: real potential, honest caveats

Next class (Class 2):

- Quantum gates: Pauli, Hadamard, rotations
- Multi-qubit systems, tensor product
- Entanglement: the resource that makes quantum *quantum*

Now: 45-min practice session.

References i

1. Biamonte et al., "Quantum machine learning," *Nature* 549 (2017).
2. Schuld & Petruccione, *Supervised Learning with Quantum Computers*, Springer (2018).
3. Raissi, Perdikaris, Karniadakis, "Physics-informed neural networks," *J. Comput. Phys.* 378 (2019).
4. Kyriienko et al., "Solving nonlinear differential equations with differentiable quantum circuits," *Phys. Rev. A* 103, 052416 (2021).
5. Kim et al., "Evidence for the utility of quantum computing before fault tolerance," *Nature* 618 (2023).
6. Agliardi et al., "Mitigating exponential concentration in covariant quantum kernels," *npj Quantum Information* (2025).
7. Tang, "A quantum-inspired classical algorithm for recommendation systems," STOC (2019).

Useful links

Linear algebra 3Blue1Brown, “Essence of Linear Algebra”
`https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab`

Complex numbers / Euler’s formula 3Blue1Brown, “$e^{i\pi}$ in 3.14 minutes”
`https://youtu.be/v0YEaeIClKY`

Quantum computing intro IBM Quantum Learning
`https://learning.quantum.ibm.com/`

PINNs “Physics Informed Neural Networks” (overview)
`https://benmoseley.blog/my-research/so-what-is-a-physics-informed-neural-network/`

QML demos PennyLane QML demonstrations
`https://pennylane.ai/qml/demos/`