Machine Learning Guest Lecture
A Glimpse of Quantum Machine Learning

Hao-Chung Cheng (鄭皓中)
haochung@ntu.edu.tw
Quantum Information Lab (量子資訊處理實驗室)
Department of Electrical Engineering
National Taiwan University
May 28, 2021
Outline

1. Part I – Basics of Quantum Information Processing
2. Part II – Various Models of Quantum ML
3. Part III – ML with Quantum Algorithmic Speed-up
4. Part IV – Learning with Quantum Machines
5. Conclusions and Outlooks
Quantum Machine Learning?
Prologue

• *Quantum information science* have demonstrated advantages in several fields:
  - **Computing**: speed-ups certain computational tasks where no classical methods can do.
  - **Communication**: entanglement-assisted communication increases channel capacity.
  - **Cryptography**: extending classical key with information-theoretic security.
  - **Sensing**: more accurate estimation, positioning, and synchronization.
  - **Simulation**: simulating complex reactions that are formidable for classical computers.

• Any other applications/advantages of quantum information technologies?

• *Artificial intelligence* and *machine learning* tasks are essentially implemented on a physical device. How about doing the job on a quantum computational device?

• Can quantum information science revolutionize the way of learning from data?
Part I – Basics of Quantum Information Processing
Brief History of Quantum Computation (1/2)

• Paul Benioff (1979): “The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines.

• Feynman (1981): “Why don’t we store information on individual particles that already follow the very rules of quantum mechanics that we try to simulate?

“Nature Isn’t classical, dammit, and if you want to make a simulation of nature, you’d better make it quantum mechanical.”


• Umesh Vazirani and Ethan Bernstein (1993) pushed it forward (bounded error).

• Daniel Simon (1994) demonstrated an exponential speedup.
Brief History of Quantum Computation (2/2)

• Seth Lloyd (1993) described a method of building a working quantum computer.
• Peter Shor (1994) invented a polynomial-time quantum algorithm for factoring.
• David DiVincenzo (1996) outlined the key criteria of a quantum computer.
• Isaac Chuang et al. (2001) implemented Shor’s algorithm on a nuclear magnetic resonance (NMR) system to factor the number 15 as a demonstration.

→ A variety of interdisciplinary fields such as Quantum Computation, Quantum Communication, Quantum Simulation, Quantum Sensing, Quantum Chemistry, etc.
Industry – Tech Giants

- **IBM**
  - IBM Q System One Computer Center
  - 53, 65-qubit processor for IBM Q Network

- **Google AI**
  - 54-qubit processor “Sycamore”
  - 72-qubit processor “Bristlecone”

- **Microsoft**
  - Quantum Development Kit
  - Q# Programming Language
  - Azure Quantum – cloud service

- **Intel**
  - 49-qubit processor
Other Quantum Processors

- Trapped Ion
- Photonics
  - NV Center-in-Diamond
  - Superconducting
- Silicon-Based Spin
  - Honeywell
  - IonQ
  - Xanadu
  - PsiQuantum
  - Origin Quantum
- Photonics
  - Rigetti

(32 qubits)
Photonics is the only way to deliver

1,000,000 qubits

A useful quantum computer requires at least a million qubits.

Our quantum computer will be built using the same industrial tools that produce your laptop.

Error correction is at the centre of everything we do. It is the only known way to ensure that such a complex device can function reliably.

Thirty years ago, photonic quantum computing was believed impossible. Twenty years ago, it was proved possible but dismissed as impractical. Today, after numerous architectural breakthroughs and advances in silicon photonics, PsiQuantum uniquely has a clear path to a useful quantum computer.
Google Aims for Commercial-Grade Quantum Computer by 2029

Tech giant is one of many companies racing to build a business around the nascent technology.
EXHIBIT 1 | Companies Assume Four Roles Across Layers of the Stack in the Quantum Computing Ecosystem

**SERVICES**
- IBM
- Google
- Rigetti Computing
- Microsoft

**APPLICATIONS LAYER**
- Potential expansion
  - Zapata Computing
  - Cambridge Quantum Computing
  - Telus Matrix Group
  - Entanglement Partners
  - Quantum Consultants
  - Quantum Benchmark
  - Strangeworks
  - Q-CTRL
  - Qiskit
  - ProteinQuake
  - QubitLogic

**SYSTEM SOFTWARE LAYER**
- Microsoft
- Alibaba Group
- IonQ
- QuTech
- Intel
- Silicon Quantum Computing
- PsiQ
- Alpine Quantum Technologies

**SYSTEMS**
- D-Wave Systems
- Emerging: Honeywell Xanadu Qilimanjaro

**QUANTUM COMPUTER HARDWARE**
- AWS is invested in IonQ
- QuTech was founded by TU Delft and TNO, and has collaborations with Intel and Microsoft.
- Quantum Circuits (qci) is a spinoff from Yale University.
- See QC

**Full-Stack (End-to-End)**
- Google
- IBM
- Microsoft
- Amazon
- Honeywell
- Rigetti

**Software Applications**
- RIVERLANE
- Menten AI
- ZAPATA
- QC Ware
- QubitLogic
- QRITHM
- QULAB

**Cloud Computing**
- agnostiq
- 1QBit
- BraneCell
- Aliro

**Quantum Encryption and AI**
- ISARA
- agnostiq
- Post Quantum

**Systems & Firmware**
- 1QBit
- Q-CTRL
- Aliro
- QINDOM
- STRANGE WORKS

**Quantum Hardware**
- Silicon Quantum Computing
- PsiQ
- IONQ
- IonQ
- XANADU
- QM
- QUANDELIA
- QUANTUM MACHINE

**Sources:** Quantum Computing Report (quantumcomputingreport.com); BCG analysis.

1Based on player’s ambition with varying levels of maturity and service activities.
2Multiple technologies in the labs with focus on topological qubits.
3Qilimanjaro is a spinoff from the University of Barcelona.
4AWS is invested in IonQ.
5QuTech was founded by TU Delft and TNO, and has collaborations with Intel and Microsoft.
6Quantum Circuits (qci) is a spinoff from Yale University.
7See QC is a subsidiary of Hypers.
8Vision to become end-to-end provider.
9Alpine Quantum Technologies (AQT) is a spinoff from University of Innsbruck.
Quantum Computer System Stacks

Application Layer
- Quantum Algorithm
  - Quantum
  - Classical

Systems Software Layer
- Quantum DSL, Compilation, Unitary Synthesis, Pulse Control, Noise Mitigation, Error Correction

Hardware Layer
- Quantum Hardware
  - System Controller
  - Qubits

Logical layer
- Logical quantum processor
  - Controls
  - Readout
  - Logical operations and magic states

Physical layer
- Quantum error correction
  - Lattice of superconducting qubits and resonators
  - Readout
  - Quantum limited amplifiers
  - Microwave pulses

Building logical qubits in a superconducting quantum computing system

Jay M. Gambetta, Jerry M. Chow & Matthias Steffen

npj Quantum Information 3, Article number: 2 (2017)
Quantum Simulation

Quantum Machine Learning

Quantum Computing

Algorithm

Complexity

Architecture

Quantum Error Correction

Quantum Communication

Quantum Internet

Quantum Information Processing

Quantum Sensing

Quantum Cryptography

Quantum Information Theory
A Statistical Framework of Quantum Theory

- **Preparation**: A preparation procedure determines the *state* of a system.

- **Evolution**: How is a quantum state *evolving*?

- **Measurement**: A measurement procedure produces some random outcomes.
The Quantum Bit (Qubit)

• **Definition**: A *qubit* is the fundamental unit of quantum information. It is a *superposition* state represented by a linear combination of $|0\rangle$ and $|1\rangle$ in $\mathbb{C}^2$:

$$|\psi\rangle = a|0\rangle + b|1\rangle, \ a, b \in \mathbb{C}, \ |a|^2 + |b|^2 = 1$$

• Physically, a qubit can be realized by a two-state (two-level) quantum-mechanical system e.g. a spinning electron or polarized light.

• **A quantum register** (a quantum system) is a collection of qubits we use for computation.
Vector Representation (Dirac Notation) for a Qubit

- \( |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow |\psi\rangle = a|0\rangle + b|1\rangle = \begin{bmatrix} a \\ b \end{bmatrix} \in \mathbb{C}^2, \quad |a|^2 + |b|^2 = 1 \)

- Bra–Ket notation: the Ket vectors \( |\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix}, |\phi\rangle = \begin{bmatrix} c \\ d \end{bmatrix} \)

- Inner product: \( \langle \phi | := |\phi\rangle^\dagger = [c^*, \ d^*] \)

\[ \Rightarrow \langle \phi | \psi \rangle = [c^*, \ d^*] \begin{bmatrix} a \\ b \end{bmatrix} = c^*a + d^*b \in \mathbb{C} \]

- \( |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \)

- \( |+\rangle := \frac{|0\rangle + |1\rangle}{\sqrt{2}} \)

- Paul Dirac (1902-1984)
$\Rightarrow |\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)e^{i\varphi}|1\rangle, \ \theta \in [0, \pi], \ \varphi \in [0, 2\pi]$
Quantum Measurement

- A quantum measurement (with respect to the computational basis \(\{|0\rangle, |1\rangle\}\)) gives you the readout of ‘0’ or ‘1’ with certain probability.

- The Born rule:

\[
|\psi\rangle = a|0\rangle + b|1\rangle
\]

\[
\Pr(0) = |\langle 0|\psi\rangle|^2 = \left| \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right|^2 = |a|^2
\]

\[
\Pr(1) = |\langle 1|\psi\rangle|^2 = \left| \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right|^2 = |b|^2
\]

- After measurement, a qubit is forced to collapse (irreversibly) through projection to one of the basis.

Max Born (1882-1970)
Single-Qubit Gates (1/2)

• In gate-based quantum computers, a quantum operation is a *unitary operation*.

• The quantum $X$ gate is given by the Pauli $X$ matrix $|0\rangle \xrightarrow{X} |1\rangle$

  → It is called the *NOT* gate or the “*bit flip*” gate since it rotates $\pi$ around the $x$-axis.

  $X := |0\rangle\langle 1| + |1\rangle\langle 0| = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

  $\Rightarrow X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$

• The quantum $Z$ gate rotates $\pi$ around the $z$-axis $Z := |0\rangle\langle 0| - |1\rangle\langle 1|$

  → It is called the “*phase flip*” gate $\Rightarrow Z|\psi\rangle = r_1|0\rangle + r_2 e^{i(\pi+\psi)}|1\rangle$

• The quantum $Y$ gate rotates $\pi$ around the $y$-axis.

  → It does the bit flip and phase flip at the same time.
Single-Qubit Gates (2/2)

• The Hadamard gate changes the basis from \{ |0\rangle, |1\rangle \} to \{ |+\rangle, |-\rangle \} → It creates superposition; and it is self-inverse: \( HH = I \)

\[
|b\rangle \quad \begin{array}{c}
\text{H}
\end{array} \quad \frac{1}{\sqrt{2}} \left( |0\rangle + (-1)^b |1\rangle \right), \; b \in \{0, 1\}
\]

Jacques Hadamard (1865-1963)

• The quantum \( R_Z^\phi \) gate rotates \( \phi \) around the z-axis

\[
R_Z^\phi := \begin{pmatrix}
1 & 0 \\
0 & e^{i\phi}
\end{pmatrix}
\]

• The quantum \( R_X^\phi \) gate rotates \( \phi \) around the x-axis

\[
R_X^\phi := \begin{pmatrix}
\cos(\frac{\phi}{2}) & -i\sin(\frac{\phi}{2}) \\
-i\sin(\frac{\phi}{2}) & \cos(\frac{\phi}{2})
\end{pmatrix}
\]

• The quantum \( R_Y^\phi \) gate rotates \( \phi \) around the y-axis

\[
R_Y^\phi := \begin{pmatrix}
\cos(\frac{\phi}{2}) & -\sin(\frac{\phi}{2}) \\
\sin(\frac{\phi}{2}) & \cos(\frac{\phi}{2})
\end{pmatrix}
\]
Just a bit math...

- Definition. **Tensor product** of matrices $A := \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$, $B := \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}$

$$A \otimes B = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \otimes B$$

$$:= \begin{pmatrix} a_{1,1}B & a_{1,2}B \\ a_{2,1}B & a_{2,2}B \end{pmatrix} = \begin{pmatrix} a_{1,1} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} \\ a_{2,1} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix} = \begin{pmatrix} a_{1,1}b_{1,1} & a_{1,1}b_{1,2} & a_{1,2}b_{1,1} & a_{1,2}b_{1,2} \\ a_{1,1}b_{2,1} & a_{1,1}b_{2,2} & a_{1,2}b_{2,1} & a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} & a_{2,1}b_{1,2} & a_{2,2}b_{1,1} & a_{2,2}b_{1,2} \\ a_{2,1}b_{2,1} & a_{2,1}b_{2,2} & a_{2,2}b_{2,1} & a_{2,2}b_{2,2} \end{pmatrix}$$
More Qubits – Quantum Entanglement (1/2)

• General 2-qubit state:

\[ |\psi\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle \in \mathbb{C}^4 \quad \sum_{i,j\in\{0,1\}} |a_{ij}|^2 = 1 \]

• A two-qubit is represented by a unit vector in \textit{four}-dimensional linear space \( \mathbb{C}^{2\times 2} \)

-> The computational basis:

\[
|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}
\]

• Tensor product: \( |01\rangle \equiv |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \times 0 \\ 1 \times 1 \\ 0 \times 0 \\ 0 \times 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \)
• Entangled state: there are states that cannot be expressed as the product form, i.e.

$$\not\exists |q_1\rangle, |q_2\rangle \in \mathbb{C}^2, \text{ s.t. } |\psi\rangle = |q_1\rangle \otimes |q_2\rangle$$

with $\frac{1}{2}$ probability,

\[
\begin{array}{c}
\uparrow \\
\downarrow
\end{array}
\]

with $\frac{1}{2}$ probability,

\[
\begin{array}{c}
\downarrow \\
\uparrow
\end{array}
\]

or

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

$$= \frac{|++\rangle + |--\rangle}{\sqrt{2}}$$
Multi-Qubit Gates (1/3)

- A qubit gate has a 2 by 2 unitary matrix in a given basis.
  - For an $n$-qubit gate, the matrix is $2^n$ by $2^n$ (tensor product of matrices).

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

$$\Rightarrow H \otimes H |00\rangle = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = |+\rangle \otimes |+\rangle$$

$(A \otimes B)a \otimes b = (Aa) \otimes (Bb)$
Multi-Qubit Gates (2/3)

- The previous example is a 2-qubit gate of the product form.

- Controlled-NOT gate:

  Ex. \[ |1\rangle \quad |1\rangle \quad |0\rangle \quad |1\rangle \]

- Used to prepare the Bell state:

\[ |0\rangle \quad H \quad |\Phi^+\rangle \]

\[
|00\rangle \quad \overset{H \otimes I}{\Rightarrow} \quad \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \otimes |0\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |10\rangle) \quad \overset{CNOT}{\leftrightarrow} \quad \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle)
\]
Multi-Qubit Gates (3/3)

- Toffoli gate (CCNOT gate)
  → *universal* classically

\[
|000\rangle \leftrightarrow |000\rangle \\
|001\rangle \leftrightarrow |001\rangle \\
|010\rangle \leftrightarrow |010\rangle \\
|011\rangle \leftrightarrow |011\rangle \\
|100\rangle \leftrightarrow |100\rangle \\
|101\rangle \leftrightarrow |101\rangle \\
|110\rangle \leftrightarrow |111\rangle \\
|111\rangle \leftrightarrow |110\rangle \\
\]

- Swap gate

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

- *Universal* quantum gates sets: \{CNOT, T, H\} or \{CCNOT, H\}
Elementary Quantum Gates

• Pauli gates
  \[ X := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y := \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \]

• Rotation gates
  \[ R_x(\phi) := e^{-i \frac{\phi}{2} X} = \begin{pmatrix} \cos(\frac{\phi}{2}) & -i \sin(\frac{\phi}{2}) \\ -i \sin(\frac{\phi}{2}) & \cos(\frac{\phi}{2}) \end{pmatrix} \]
  \[ R_y(\phi) := e^{-i \frac{\phi}{2} Y} = \begin{pmatrix} \cos(\frac{\phi}{2}) & -\sin(\frac{\phi}{2}) \\ \sin(\frac{\phi}{2}) & \cos(\frac{\phi}{2}) \end{pmatrix} \]
  \[ R_z(\phi) := e^{-i \frac{\phi}{2} Z} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \]

• Controlled-Z gates
  \[ S := R_z(\frac{\pi}{2}) \quad T := R_z(\frac{\pi}{4}) \]

• CNOT gate
  \[ |c\rangle \quad |c\rangle \quad \begin{pmatrix} I_2 & 0 \\ 0 & X \end{pmatrix} \]

• Toffoli (CCNOT) gate
  \[ |c_1\rangle \quad |c_2\rangle \quad |c_1\rangle \quad |c_2\rangle \quad |c_1\rangle \quad |c_2\rangle \quad \begin{pmatrix} I_3 & 0 & 0 \\ 0 & I_3 & 0 \\ 0 & 0 & X \end{pmatrix} \]

• Hadamard gates
  \[ H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \]

• Swap gate \(|\psi\rangle|\phi\rangle \leftrightarrow |\phi\rangle|\psi\rangle\)
Gate-Based Quantum Computation
Efficiently Preparing Superposition States

- We can create a superposition of \textit{exponentially many} terms with only a \textit{linear} number of the Hadamard gates.

\[
\left( \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \right)^{\otimes n}
\]

\[
H^{\otimes n} |0\rangle^{\otimes n} = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle
\]
The Quantum Oracle & Quantum Parallelism

• The *quantum oracle* for any Boolean function $f: \{0,1\}^n \rightarrow \{0,1\}^m$ is given by the quantum gate $U_f: |x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$, $\forall x \in \{0,1\}^n$, $\forall y \in \{0,1\}^m$.

• We set the input register to an equal superposition of all $2^n$ possible $n$-bit strings:

$$U_f : \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|0\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|f(x)\rangle$$

• In *one* run of the protocol, we obtain a final state with depends on all of the function values. On the other hand, we would need *exponentially* many queries to have full access to the function $f$. 
Quantum Computing – Unstructured Search

- Searching in an unstructured list with size $N$

The best classical algorithm requires number of queries proportional to $N$

→ Lov Grover (1996) proposed a quantum algorithm requires $\approx \sqrt{N}$ queries
Quantum Computing – Factorization

• Integer Factorization used in the RSA cryptography system

Example: $463570199875051 = 27644437\times16769023$

$\approx 2^{49}$

The computational complexity of the best known classical algorithm scales \textit{exponentially} in the number of bits of the integer.

→ Peter Shor (1994) invented a \textit{polynomial-time} quantum algorithm

• Other cryptosystem such as the \textit{Diffie–Hellman key exchange security} (based on the hardness of the \textit{discrete logarithm problem}) and the \textit{Elliptic curve cryptography} can be broke in polytime by applying Shor’s idea.
Relations – A Glimpse of The Complexity Zoo

- Efficiently solvable by classical computer
- Efficiently solvable by quantum computer

- $n \times n$ Chess/Go
- SAT, $n \times n$ Sudoku
- Travelling salseman
- Graph Isomorphism
- Integer factorization, Discrete logarithm
- Multiplication, Primality, etc.
Ground-to-satellite quantum teleportation

Ji-Gang Ren, Ping Xu, [...] Jian-Wei Pan

Nature 549, 70–73(2017) | Cite this article

Satellite-to-ground quantum key distribution

Sheng-Kai Liao, Wen-Qi Cai, [...] Jian-Wei Pan

Nature 549, 43–47(2017) | Cite this article

Satellite-based entanglement distribution over 1200 kilometers

Juan Yin, [...]

Physical Review Letters 120, 030501 (2018)

Entanglement-based secure quantum cryptography over 1,120 kilometres

Juan Yin, Yu-Huai Li, Sheng-Kai Liao, Meng Yang, Yuan Cao, Liang Zhang, Ji-Gang Ren, Wen-Qi Cai, Wei-Yue Liu, Shuang-Lin Li, Rong Shu, Yong-Mei Huang, Lei Deng, Li Li, Qiang Zhang, Nai-Le Liu, Yu-Ao Chen, Chao-Yang Lu, Xiang-Bin Wang, Feihu Xu, Jian-Yu Wang, Cheng-Zhi Peng, Artur K. Ekert & Jian-Wei Pan

Nature 582, 501–505 (2020) | Cite this article
Quantum computational advantage using photons

Han-Sen Zhong\textsuperscript{1,2,*}, Hui Wang\textsuperscript{1,2,*}, Yu-Hao Deng\textsuperscript{1,2,*}, Ming-Cheng Chen\textsuperscript{1,2,*}, Li-Chao Peng\textsuperscript{1,2}, Yi-Han Luo\textsuperscript{1,2}, Jian Qin\textsuperscript{1,2}, Dian Wu\textsuperscript{1,2}, Xing Ding\textsuperscript{1,2}, Yi Hu\textsuperscript{1,2}, Peng Hu\textsuperscript{3}, Xiao-Yan Yang\textsuperscript{3}, Wei-Jun Zhang\textsuperscript{3}, Hao Li\textsuperscript{3}, Yuxuan Li\textsuperscript{4}, Xiao Jiang\textsuperscript{1,2}, Lin Gan\textsuperscript{4}, Guangwen Yang\textsuperscript{4}, Lixing You\textsuperscript{3}, Zhen Wang\textsuperscript{3}, Li Li\textsuperscript{1,2}, Nai-Le Liu\textsuperscript{1,2}, Chao-Yang Lu\textsuperscript{1,2}, Jian-Wei Pan\textsuperscript{1,2,†}

\textsuperscript{1}Hefei National Laboratory for Physical Sciences at Microscale and Department of Modern Physics, University of Science and Technology of China, Hefei, Anhui 230026, China.

\textsuperscript{2}CAS Centre for Excellence and Synergetic Innovation Centre in Quantum Information and Quantum Physics, University of Science and Technology of China, Shanghai 201315, China.

\textsuperscript{3}State Key Laboratory of Functional Materials for Informatics, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China.

\textsuperscript{4}Department of Computer Science and Technology and Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China.

*†Corresponding author. Email: pan@ustc.edu.cn

* These authors contributed equally to this work.

Hide authors and affiliations

Science 03 Dec 2020:
eabe8770
DOI: 10.1126/science.abe8770
Part II – Various Models of Quantum ML
Brief History of Quantum Machine Learning (1/4)

• In 2000’s – early explorations; mostly on detection & estimation instead of learning.
• The term ‘quantum machine learning’ was coined around 2013 (arXiv:1307.0411).
  → HHL algorithm for approximately solve linear equations with a quantum RAM.
Brief History of Quantum Machine Learning (2/4)

• Some critiques about quantum speeding-up ML since 2015.
→ Not just about practically building a quantum compute but caveats of using it.

• Two types of quantum neural networks (QNNs) in 2018.
  - Unitary feed-forward network and Quantum Boltzmann machine
Brief History of Quantum Machine Learning (3/4)

- Finding applications, advantages, and preliminary analysis of QNNs.
Brief History of Quantum Machine Learning (4/4)

• Barren plateaus (vanishing gradients), quantum No-Free-Lunch theorem, etc…
Four Categories

- **CC**: Learning **classical data with classical machine** – Classical ML.
- **QC**: Learning **quantum objects with classical ML** (e.g. matrix concentrations).
  - S. Aaronson’07 – Learning quantum states.
- **CQ**: Processing **classical datasets using quantum computing** – Lloyd *et al.*
- **QQ**: Processing **“quantum data” using quantum machine** – largely open.

**Statistical Learning Framework**

- **unknown target function** \( g : \mathcal{X} \rightarrow \mathcal{Y} \)
- **unknown distribution** \( \mu \)
- **training samples** \( \mathcal{D} : (x_1, y_1), \cdots, (x_n, y_n) \)
- **hypothesis set** \( \mathcal{H} \)
- **final hypothesis** \( h \approx g \)

### Different Output Space

- **binary classification**: \( \mathcal{Y} = \{-1, +1\} \)
- **multiclass classification**: \( \mathcal{Y} = \{1, 2, \cdots, K\} \)
- **regression**: \( \mathcal{Y} = \mathbb{R} \)
- **unsupervised**: \( \mathcal{Y} = \emptyset \)

- **training**
- **testing**

#### statistical learning framework

\[ \mathbb{E}_{\mathcal{D}} (\text{good } \mathcal{D}) \]

- known \( E_{\text{empirical}}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i) \)

- testing \( \mathcal{D} \) (good \( \mathcal{D} \) & good \( \mathcal{H} \))

- unknown \( E_{\text{ensemble}}(h) = \mathbb{E}_\mu \ell(h(x), g(x)) \)
Categories from The Learning Framework (1/2)

- **Data & Goal**
  - classical or quantum
  - hybrid

- **Learning Mechanism**
  - classical ML algorithms
  - quantum variational circuits

- **Physical device for implementation**
  - classical CPU/GPU/TPU
  - quantum processors
<table>
<thead>
<tr>
<th>Data &amp; goal</th>
<th>Learning mechanism</th>
<th>Physical device</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>classical</td>
<td>classical</td>
<td>classical</td>
<td>classical ML</td>
</tr>
<tr>
<td>classical</td>
<td>classical</td>
<td>quantum</td>
<td>speedup running time</td>
</tr>
<tr>
<td>classical</td>
<td>quantum</td>
<td>classical</td>
<td>VQC &amp; simulator</td>
</tr>
<tr>
<td>classical</td>
<td>quantum</td>
<td>quantum</td>
<td>VQC &amp; QPU</td>
</tr>
<tr>
<td>quantum</td>
<td>classical</td>
<td>classical</td>
<td>tomography &amp; simulator</td>
</tr>
<tr>
<td>quantum</td>
<td>classical</td>
<td>quantum</td>
<td>tomography &amp; QPU</td>
</tr>
<tr>
<td>quantum</td>
<td>quantum</td>
<td>classical</td>
<td>quantum simulation VQC? &amp; simulator</td>
</tr>
<tr>
<td>quantum</td>
<td>quantum</td>
<td>quantum</td>
<td>quantum Boltzmann machine?</td>
</tr>
</tbody>
</table>
Part III – ML with Quantum Algorithmic Speeding-up

<table>
<thead>
<tr>
<th>Data &amp; goal</th>
<th>Learning mechanism</th>
<th>Physical device</th>
</tr>
</thead>
<tbody>
<tr>
<td>classical</td>
<td>classical</td>
<td>quantum</td>
</tr>
</tbody>
</table>
Flowchart of Classical ML with Quantum Algorithm

Classical

Data set $\mathcal{D}$: input $x$

Classical machine learning algorithm

Prediction $y$

Quantum

Data set $\mathcal{D}$: input $x$

Encoding

Quantum machine learning algorithm

Read out

Prediction $y$

Quantum system

State preparation

Quantum evolution

measurement
Information Encoding

• To learn from classical data, we need to load data from classical memory into the quantum computer; this process is called \textit{state preparation} in QML.

• In the training phase, we consider data set $\mathcal{D} = \{x_1, ..., x_m\}$ of $N$–dimensional real feature vectors.

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Number of qubits</th>
<th>Runtime of state preparation</th>
<th>Input feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basis</td>
<td>$N$</td>
<td>$O(MN)$</td>
<td>Binary</td>
</tr>
<tr>
<td>Amplitude</td>
<td>$\log N$</td>
<td>$O(MN)/O(\log MN)$</td>
<td>Continuous</td>
</tr>
<tr>
<td>Qsample</td>
<td>$N$</td>
<td>$O(MN)/O(\log MN)$</td>
<td>Binary</td>
</tr>
<tr>
<td>Hamiltonian</td>
<td>$\log N$</td>
<td>$O(MN)/O(\log MN)$</td>
<td>Continuous</td>
</tr>
</tbody>
</table>
Basis Encoding (1/2)

• Assume each data is $N$-dimensional bit string, i.e. $x_m \in \{0,1\}^N$ for $x_m \in \mathcal{D}$:

$$x_m \mapsto |x_m\rangle \quad \mathcal{D} \mapsto |\mathcal{D}\rangle := \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x_m\rangle$$

• For example, given $\mathcal{D} = \{x_1, x_2\} = \{0101, 1110\}$, then $|\mathcal{D}\rangle = \frac{1}{\sqrt{2}} (|0101\rangle + |1110\rangle)$.

$$|0101\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = (0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T$$

$$|1110\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)^T$$

$$\Rightarrow |\mathcal{D}\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1/\sqrt{2} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1/\sqrt{2} \\ 0 \end{pmatrix}^T$$
Basis Encoding (2/2)

- Preparation in time $O(MN)$ by Ventura–Martinez, and Trugenberger.
- Approach by the Quantum random access memory, in time $O(\log N)$.

\[
\text{QRAM: } \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |m\rangle |0 \cdots 0\rangle \mapsto \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |m\rangle |x_{m}\rangle
\]

- However, an efficient hardware implementation is still an open challenge.
Computing in Basis Encoding

• Suppose we have a Boolean logic gate \( f: \{0,1\}^N \rightarrow \{0,1\} \) giving binary label to each data \( x_m \). Then, this can be implemented by (universal) quantum Toffoli gates. This is called a \textit{quantum oracle} \( U_f: |x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle \).

• \textit{Quantum parallelism}:

\[
U_f: \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x_m\rangle|0\cdots0\rangle \mapsto \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x_m\rangle|f(x_m)\rangle
\]

• To read-out the state of the qubits, we have to measure it (quantum tomography). → \textit{Frequentist estimator} via multiple shots of measurements.

• \textit{Caveats}: (i) large amounts of qubits needed; (ii) realizing a quantum RAM.
Amplitude Encoding

• For simplicity, suppose each feature vector $x_m = (x_{m,1}, \ldots, x_{m,N})$ is normalized, i.e. $\sum_{i=1}^{N} x_{m,i}^2 = 1$. (This condition can be removed by non-expansive manipulations.)

• Then, we can prepare the quantum state $|x_m\rangle$ by using $n = \log N$ quantum bits:

$$x_m \mapsto |x_m\rangle := \sum_{i=1}^{N} x_{m,i} |i\rangle \quad \mathcal{D} \mapsto |\mathcal{D}\rangle := \frac{1}{\sqrt{M}} \sum_{m=1}^{M} \sum_{i=1}^{N} x_{m,i} |m\rangle |i\rangle$$

• State preparation in linear time: $O(MN)/O(\log MN)$.

• For sufficiently uniform vectors, the preparation may be efficiently done via qRAM.
Computing in Amplitude Encoding (1/2)

• Clustering: Assigning a vector \( \vec{u} \in \mathbb{C}^N \) to two groups: \( \{\vec{v}_m\}_{m=1}^M \) and \( \{\vec{w}_m\}_{m=1}^M \).

• Classical approach: to compare the distance \( \left| \vec{u} - \frac{1}{M} \sum_{m=1}^{M} \vec{v}_m \right|^2 \).

  Takes time \( O(\text{poly}(MN)) \).

  \[ \text{involving evaluating inner product } \langle \hat{u}, \vec{v}_m \rangle \]

• The \textit{swap test} for computing the inner product.

\[ \Pr(0) = \frac{1}{2} + \frac{1}{2} |\langle \psi | \phi \rangle|^2 \]
Computing in Amplitude Encoding (2/2)

• The clustering can be done in $O(\log MN)$ in a quantum computer.

• Caveats:
  
  - If wanted to read-out the value of any specific entry $x_i$ of $|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$, then in general would require repeating the algorithm roughly $O(N)$ times, this then kills the exponential speed-up.
  
  - Once again, if preparing the amplitude encoding in $|x\rangle$ requires super-logarithmic time, then the speed-up unfortunately vanishes.
Support Vector Machine with Quantum RAM

• The Support Vector Machine (SVM) can be formulated as a quadratic programming problem, which can be solved in time $O(\text{poly}(M, N))$, i.e.
  \[ \max_{\vec{\alpha}} \left\{ L(\vec{\alpha}) = \langle \vec{\alpha}, \vec{y} \rangle - \frac{1}{2} \langle \vec{\alpha}, K \vec{\alpha} \rangle : \sum_{j=1}^{M} \alpha_j = 0, y_j \alpha_j \geq 0 \right\}, \]
  \[ [K]_{ij} := k(\vec{x}_i, \vec{x}_j). \]

• Efficient inner product evaluation via quantum algorithm takes $O(\text{poly}(M) \log N)$.

• Using a non-sparse matrix exponentiation technique for approximating the kernel matrix inverse requires $O(\log MN)$.
  ← Exponential speed-up!

• Caveats:
  - Works with non-sparse kernel matrix, which has a small condition number (as in the HHL).
  - Again, the quantum RAM is indispensable.
**Support Vector Machine without Quantum RAM (1/3)**

- **Rotation encoding**: \( \Phi: \vec{x} \mapsto |\Phi(\vec{x})\rangle \) mapping the data to the *quantum feature space*.
- To implement the feature map, a rotation \( Z \) gate \( U_\Phi(\vec{x}) \) is used in the *ansatz* \( U_\Phi(\vec{x})H \) e.g. \( U_\Phi(x) = e^{-ixZ} \) for 1-bit \( x \); \( U_\Phi(x) = e^{i(x_1Z_1 + x_2Z_2 + (\pi - x_1)(\pi - x_2)Z_1Z_2)} \) for 2-bit \( \vec{x} \).

• With the quantum feature map, the classical kernel is $k(\tilde{x}_i, \tilde{x}_j) = |\langle \Phi(\tilde{x}_i) | \Phi(\tilde{x}_j) \rangle|^2$.

• Such a kernel (constructed from the above ansatz) is efficiently implementable via short-depth quantum circuits, but believed classically hard.
Support Vector Machine without Quantum RAM (3/3)

Qiskit

Getting started  Tutorials  Providers  Applications  Resources  Github

Docs > Qiskit Aqua API Reference > Aqua (Algorithms for QUantum Applications) (qiskit.aqua) > Algorithms (qiskit.aqua.algorithms) > qiskit.aqua.algorithms.QSVM

qiskit.aqua.algorithms.QSVM

CLASS QSVM(Feature_map, training_dataset=None, test_dataset=None, datapoints=None, multiclass_extension=None, Lambda2=0.001, quantum_instance=None)

Quantum SVM algorithm.
### Part IV – Learning with Quantum Machines

<table>
<thead>
<tr>
<th>Data &amp; goal</th>
<th>Learning mechanism</th>
<th>Physical device</th>
</tr>
</thead>
<tbody>
<tr>
<td>classical</td>
<td>quantum</td>
<td>classical/quantum</td>
</tr>
</tbody>
</table>
Variational (Parametric) Quantum Circuits

• The QNN is characterized by a unitary ansatz $U(\theta) := e^{-iH_n\theta_n} \cdots e^{-iH_n\theta_n}$, where $\theta = \{\theta_1, \ldots, \theta_n\}$ are the parameters we aim to learn and $H_i$ are Hermitian operators.

• Goal: find a good assignment of parameters (e.g. via gradient descent) in terms of an objective function $f_\theta(x) := \langle \psi_{\theta,x} | \Pi \otimes I | \psi_{\theta,x} \rangle$, where $|\psi_{\theta,x}\rangle := U(\theta)V(x)|0\rangle$. 
Hybrid Training for Variational Algorithms

- Idea: Use the quantum machine to compute the objective function $f(\theta^{(t)})$, and then use a classical device to compute better circuit parameters $\theta^{(t+1)}$ with respect to the objective. Iterate the routine until the objective is optimized.
Optimization Methods (1/2)

- Quantum Approximate Optimization Algorithm (QAOA) for the MaxCut Problem. → heuristic, highly problem-specific, could be computationally expensive.
- Iterative derivative-free methods (e.g. the PSO).
- Numerical gradient (finite difference).
- Analytical gradient (**parameter shift rule**).
  - In general, $\partial U(\theta)$ might not be unitary.
  - By rotation gate, $\partial U(\theta)$ can be expressed as linear combinations of unitaries; hence computable by circuits.
  - Other gradient methods.
Solving combinatorial optimization problems using QAOA

In this tutorial, we introduce combinatorial optimization problems, explain approximate optimization algorithms, explain how the Quantum Approximate Optimization Algorithm (QAOA) works and present the implementation of an example that can be run on a simulator or on a 5 qubit quantum chip.

Hybrid quantum-classical Neural Networks with PyTorch and Qiskit

Quantum machine learning
We’re entering an exciting time in quantum physics and quantum computation. Near-term quantum devices are rapidly becoming a reality, accessible to everyone over the internet. This, in turn, is driving the development of quantum machine learning and variational quantum circuits.
Various Ansatzs for QNNs (1/2)

- The dimension of the parameter space is typically much smaller than the space of all unitary operators (e.g. $O(2^{2n})$ for $n$-qubits).

- **Goal**: designing an ansatz for the QNN such that it is rich enough to allow the parameterized circuit to approximate the solution with as few parameters as possible.

- It is desirable that the qubit-efficient circuits is hard for classical simulation – the objective function cannot be efficiently computed on a classical computer. (But it does not rule out the possibility that there are other efficient classical ways.)
Caveat: Barren Plateaus

• On one hand, we desire *entanglement* between qubits to exploit full *quantumness*.

• On the other hand, excess of entanglement between visible and hidden layers may cause barren plateaus; hence both gradient descent and gradient methods fail.

• Any trade-off? How to quantify/characterize them?
Various Quantum Software Packages

FIG. 8. Times for simulating random quantum circuits with a single thread using several libraries.
Epilogue
Conclusions and Outlooks

• There are other QML models that we haven’t discussed in the lecture:
  - Quantum Boltzmann machines $e^{-\sum_i \theta_i H_i / Z(\theta)}$.
  - Quantum example oracle $|0^n\rangle \mapsto \sum_x \sqrt{p(x)} |x\rangle |f(x)\rangle$.
  - Quantum PAC-learning model.

• QML is still at its very early stage of research. How to demonstrate quantum advantages, to provide theoretical evidence, and to identify the field it can apply? On the other hand, can one prove that the quantum circuit learning is just a lure?

• Conversely, existing ML techniques could help quantum information development.

• Even if QML does not help classical problems, it might help quantum problems.

• My ultimate goal: to invent a truly quantumly-meaning learning paradigm.

Hao-Chung Cheng (鄭皓中)  
haochung@ntu.edu.tw