# Intro to quantum computing

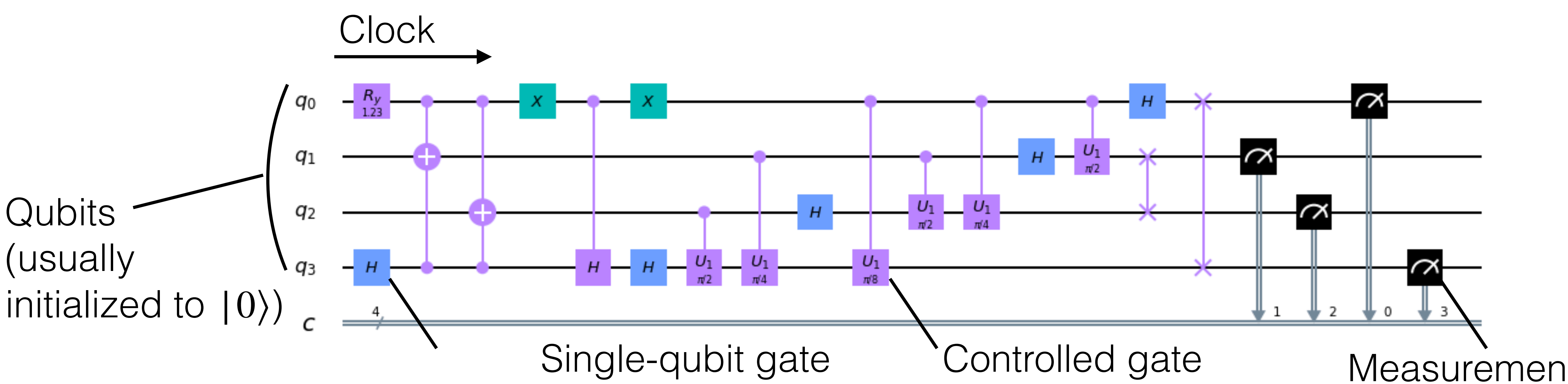
Yutaro Iiyama

# Preamble

- I'm skipping a lot of caveats, knowingly and unknowingly
- Will talk only about "digital" QC (aka quantum circuit) based on qubits
  - No annealing, no qutrits, no continuous-variable, etc.

# How it works

- A standard quantum computer consists of
  - Qubits: two-state system (SU(2) representation space)
  - Single-qubit gates: classically steerable units that act on individual qubits (SU(2) rotations)
    - Controlled gates: gates that act on one qubit depending on the state of (an)other qubit(s)
  - Measurements: project qubits to one of two states
- A quantum circuit can be represented by a circuit diagram:



# How it works

- N-qubit system starts with $|0\rangle^{\otimes N} =: |\mathbf{0}\rangle$
- Single-qubit gates rotate single qubits
  - Generically, $|0\rangle \to \alpha|0\rangle + \beta|1\rangle \quad (\alpha, \beta \in \mathbb{C};\ |\alpha|^2 + |\beta|^2 = 1)$
  - For a tensor product of N qubits,

$$|0\rangle^{\otimes N} \to \bigotimes_{i=0}^{N-1} \left(\alpha_i|0\rangle_i + \beta_i|1\rangle_i\right) = \alpha_0 \ldots \alpha_{N-1}|0\rangle_0 \ldots |0\rangle_{N-1} + \ldots + \beta_0 \ldots \beta_{N-1}|1\rangle_0 \ldots |1\rangle_{N-1} = \sum_{j=0}^{2^N-1} c_j|\mathbf{j}\rangle$$

where $|\mathbf{j}\rangle := \bigotimes_{i=0}^{N-1} |s_i^j\rangle_i$ for $j = \left(s_0^j s_1^j \ldots s_{N-1}^j\right)_2$ $\quad$ (e.g. $|\mathbf{3}\rangle = |0\rangle_0|1\rangle_1|1\rangle_2$)

and $c_j = \prod_{\{i|s_i^j=0\}} \alpha_i \prod_{\{i|s_i^j=1\}} \beta_i$

- So we get a superposition of $|\mathbf{j}\rangle$s, but this is actually a "separable" state - not interesting
  - Full state can be expressed by $\{\alpha_i, \beta_i\}_i$ where each pair has 4 - 1 - 1 = 2 real parameters → 2N-dim. parameter space

2 complex

normalization

overall phase equivalence

# How it works

- Controlled gates create inseparable states. For example,

$$\frac{1}{\sqrt{2}}\left(|0\rangle_0 + |1\rangle_0\right) \cdot |0\rangle_1 \xrightarrow{\text{controlled NOT}} \frac{1}{\sqrt{2}}\left(|0\rangle_0|0\rangle_1 + |1\rangle_0|1\rangle_1\right)$$

Qubit 1 is inverted only when qubit 0 is $|1\rangle$

- Combination of controlled and single-qubit gates create state $\sum_{j=0}^{2^N-1} c_j |\mathbf{j}\rangle$ where $c_j$ are independent parameters.
  - 2 real param × $2^N$ terms - 1 norm. constraint - 1 overall phase → $2^{N+1}$ - 2 dimensional parameter space

# Output = a distribution

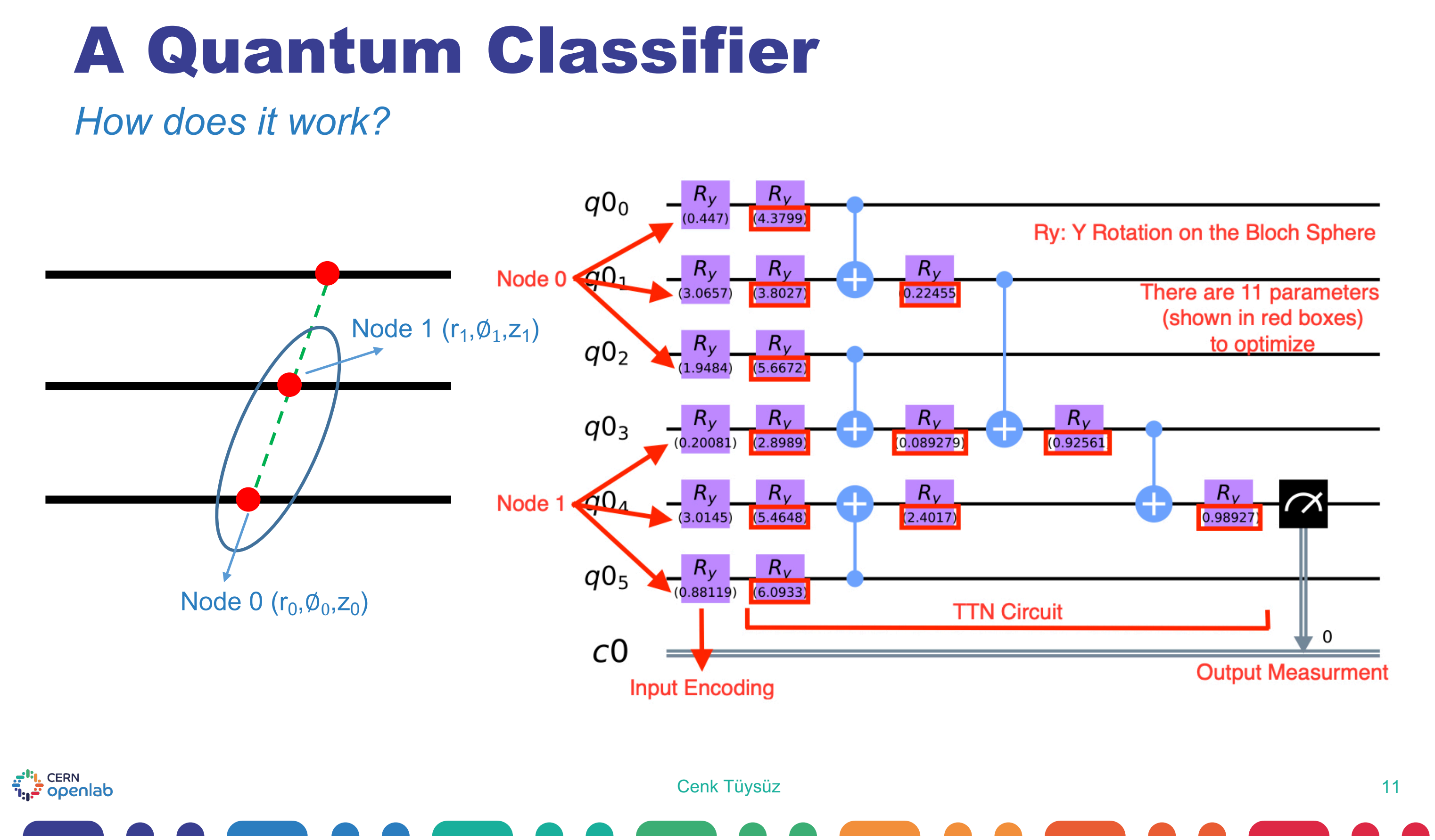
- The circuit is one giant SU($2^N$) operator that "rotates" $|\mathbf{0}\rangle$ to a final state

$$\text{(circuit)}|\mathbf{0}\rangle = \sum_{j=0}^{2^N-1} c_j |\mathbf{j}\rangle$$

- Measurement projects each qubit into 0 or 1 state
→ Overall measurement projects the final state to one $|\mathbf{j}\rangle$
- Probability of obtaining j = $|c_j|^2$
- Run the circuit many times and repeat the measurement
→ Get a histogram over j with bin contents prop. to $|c_j|^2$
- <u>Output of a quantum circuit is a distribution over {0, …, $2^N$-1}</u>

# Input = gate parameters

- Usually, the initial state of a quantum circuit is fixed to $|\mathbf{0}\rangle$
  - We don't know a way to prepare arbitrary states
    (If we do, we have mastered quantum computing)
- QC becomes all about setting up the gates so that the output distribution gives the answer to the problem at hand



# Applications

- Is the big(gest?) question.
- QC calculates $O(2^N)$ numbers in parallel using O(M) gates
  - M should be $\ll 2^N$ for it to make sense
- We only get 1 out of $O(2^N)$ numbers per "shot"
  - We must be OK with getting answers statistically
  - Or we engineer a clever circuit whose final state is dominantly a single $|\mathbf{j}\rangle$ that is the desired answer (cf. Grover's algorithm)
- Quantum memory is not a thing yet
  - All computations must go from initial to final state in one shot

# Hint for applications

- We know how to use QC to do a really Fast Fourier Transform

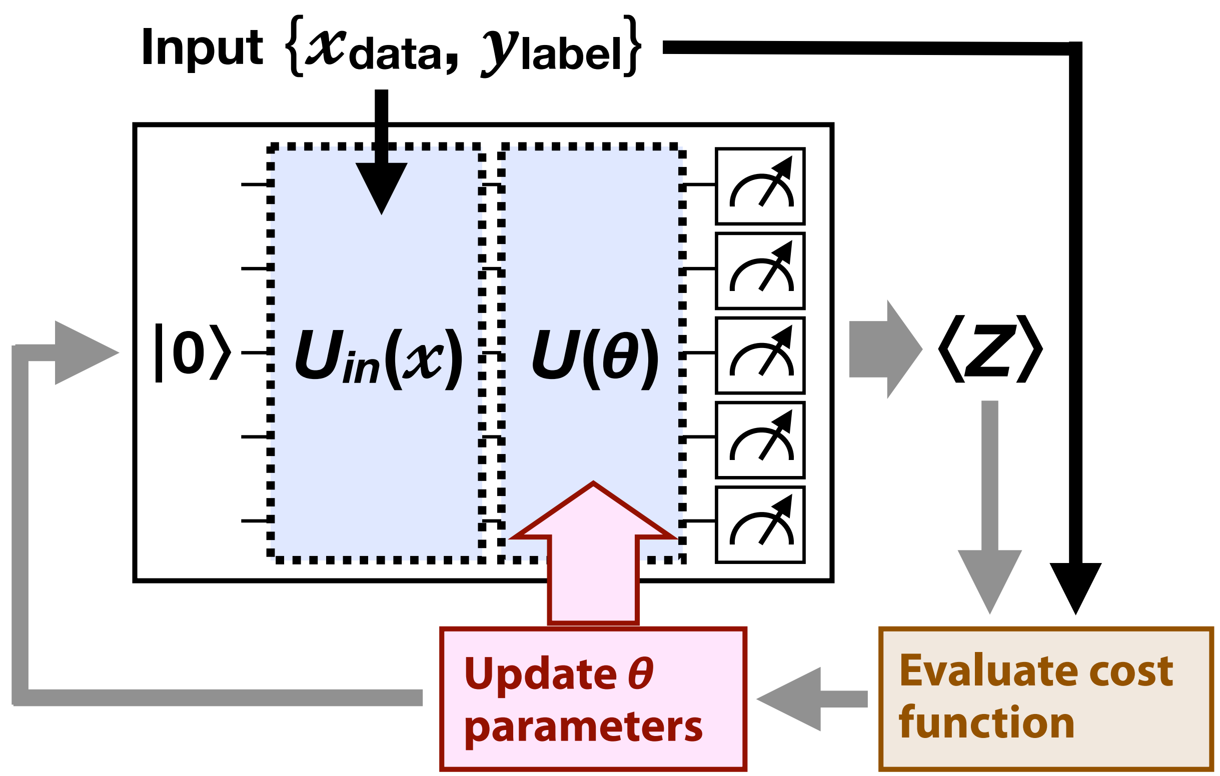
$$|\mathbf{j}\rangle \to \frac{1}{\sqrt{\mathcal{N}}} \sum_{k=0}^{\mathcal{N}-1} \mathrm{e}^{2\pi \mathrm{i} \frac{jk}{\mathcal{N}}} |\mathbf{k}\rangle \quad (\mathcal{N} = 2^N)$$

→ Basis for algorithms like factoring (Shor)

- We also know search algorithms
- Recently, more attention given to using the $2^N$-dimensional vector space as the latent space of certain algorithms
  - $2^N$ numbers can't be accessed, so use them as a hidden layer
  - Inner product (= distance) in the space is a natural concept
    → Good synergy with kernel methods
  - Large dimensionality → possibility of universal approximation

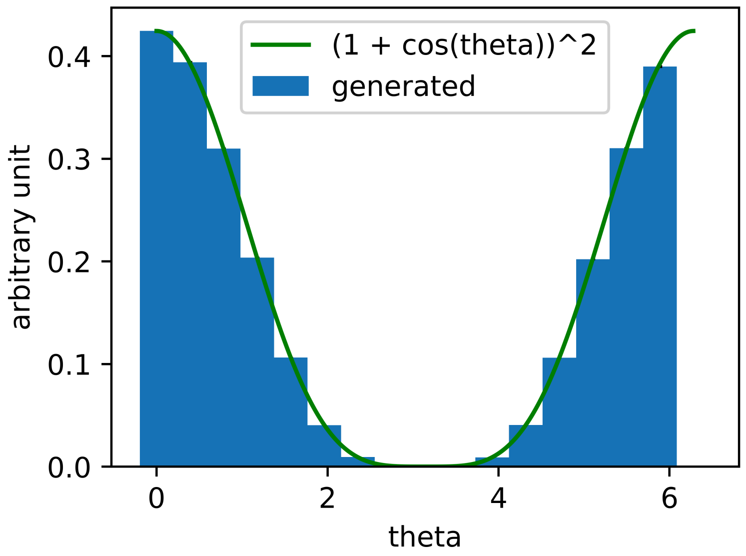
# Quantum ML with variational circuits

- Fix a gate representation (encoding) of input data
- Apply a set of gates with learnable params to the "data state"
- Run the circuit for a large number of iterations
  - Compute the cost function from an aggregate of the output (e.g. mean)
- Update the learnable params



# Side track: my dream application

- Getting 1 point per shot with a probability distribution
  → An event generator!
- Label $2^N$ points in the event phase space and create a state $|\psi\rangle = \sum_{i=0}^{\mathcal{N}-1} \psi(\mathbf{x}_i)|i\rangle$ where $\psi(\mathbf{x})$ is the scattering amplitude to $\mathbf{x}$
- Each circuit run produces one unweighted event
- Benefit: no need to integrate (aka gridpack generation)
  - $T_{integration}$ ~ exp(number of final state particles)
  - In the QC generator, $N_{qubit}$ ~ number of final state particles




Proof of concept: $\psi(\theta) = (1+\cos\theta)^2$
(LO amplitude of $e^+e^- \to \mu^+\mu^-$) implemented on IBM circuit simulator

# Existing ecosystem

- Google and IBM have the real machines
- Google: Cirq and now TensorFlow Quantum
- IBM: Qiskit and "IBM Quantum Experience"
  - Lets anyone with a (free) account use their QC on cloud
- Xanadu (Canadian startup): Pennylane (abstract layer for QML. Can do automatic differentiation of quantum circuits)
- Likely there are many more to check out

# NISQ reality

- We are in the "noisy intermediate scale quantum" regime
- Machines (at least the IBM ones) are really noisy!
    - Very low fidelity for any realistic number of gates
- Even though QC is a reality now, it's still time to focus on algorithm & tools development
- QC simulators are still important

# QFT

- There is already work on FPGA impl. of QC simulators
- As far as we know, there is no generic framework for translating circuits in e.g. Cirq into RTL
    - Actually the link above doesn't even use HLS
- Can we make an HLS4QML?
    - But we are calling it QFT