KMI School 2024

Quantum Computing for Particle Physics and Astrophysics
Nagoya University, March 6, 2024

Variational Quantum Algorithm and Quantum Machine Learning

ICEPP, The University of Tokyo
Koji Terashi

Ground State of Physical System

Charactering fundamental properties of a physical system often requires to know the ground state and its energy eigenvalue of the system

For an arbitrary state $|\psi\rangle$, the expectation value of a given Hamiltonian H is bounded by the ground state energy $\lambda_{\min}$:

$$\langle\psi|H|\psi\rangle \geq \lambda_{\min}$$

➡ Variational Method in Quantum Mechanics

Capability of quantum computer for calculating the expectation value of a Hamiltonian enables us to use the variational method to find the ground state

Variational Method

Let us consider a problem of approximating the energy eigenvalue for a certain system expressed by a matrix

For an eigenstate $|\psi_i\rangle$ and the eigenvalue λ_i of a given operator A:

$$A|\psi_i\rangle = \lambda_i|\psi_i\rangle$$

$\lambda_i = \lambda_i^*$ is real if A is a Hermitian observable ($A = A^\dagger$)

Considering the Hamiltonian H of a given physical system, the Hamiltonian can be expressed with eigenvalues λ_i and eigenvectors $|\psi_i\rangle$ in a diagonal form:

$$\Rightarrow H = \sum_{i=1}^{N} \lambda_i|\psi_i\rangle\langle\psi_i| \qquad \langle\psi_i|\psi_j\rangle = 0 \quad \forall i \neq j$$

Variational Method

For Hamiltonian $H = \sum_{i=1}^{N} \lambda_i |\psi_i\rangle\langle\psi_i|$, the expectation value of the Hamiltonian for an arbitrary state $|\psi\rangle$ is $\langle\psi|H|\psi\rangle =: \langle H\rangle_\psi$

Variational Method

For Hamiltonian $H = \sum_{i=1}^{N} \lambda_i |\psi_i\rangle\langle\psi_i|$, the expectation value of the Hamiltonian for an arbitrary state $|\psi\rangle$ is $\langle\psi|H|\psi\rangle =: \langle H\rangle_\psi$

$$\langle H\rangle_\psi = \langle\psi| \left(\sum_{i=1}^{N} \lambda_i |\psi_i\rangle\langle\psi_i| \right) |\psi\rangle$$

$$= \sum_{i=1}^{N} \lambda_i \langle\psi|\psi_i\rangle\langle\psi_i|\psi\rangle$$

$$= \sum_{i=1}^{N} \lambda_i |\langle\psi_i|\psi\rangle|^2$$

Variational Method

For Hamiltonian $H = \sum_{i=1}^{N} \lambda_i |\psi_i\rangle\langle\psi_i|$, the expectation value of the Hamiltonian for an arbitrary state $|\psi\rangle$ is $\langle\psi|H|\psi\rangle =: \langle H\rangle_\psi$

$$\langle H\rangle_\psi = \langle\psi| \left(\sum_{i=1}^{N} \lambda_i |\psi_i\rangle\langle\psi_i| \right) |\psi\rangle$$

$$= \sum_{i=1}^{N} \lambda_i \langle\psi|\psi_i\rangle\langle\psi_i|\psi\rangle$$

$$= \sum_{i=1}^{N} \lambda_i |\langle\psi_i|\psi\rangle|^2$$

Since $|\langle\psi_i|\psi\rangle|^2 \geq 0$

$$\langle H\rangle_\psi = \sum_{i=1}^{N} \lambda_i |\langle\psi_i|\psi\rangle|^2 \geq \lambda_{min}$$

Equality holds if $|\psi\rangle = |\psi_{\min}\rangle$

Variational Method

$$\langle H \rangle_\psi = \sum_{i=1}^{N} \lambda_i |\langle \psi_i | \psi \rangle|^2 \geq \lambda_{min} \qquad \text{Equality holds if } |\psi\rangle = |\psi_{\min}\rangle$$

If one can take $|\psi\rangle$ to be as close as possible to $|\psi_{\min}\rangle$,
$\lambda_{\min}$ can be well approximated by $\langle H \rangle_\psi$

Variational Method

$$\langle H \rangle_\psi = \sum_{i=1}^{N} \lambda_i |\langle \psi_i | \psi \rangle|^2 \geq \lambda_{min}$$

Equality holds if $|\psi\rangle = |\psi_{\mathrm{min}}\rangle$

If one can take $|\psi\rangle$ to be as close as possible to $|\psi_{\mathrm{min}}\rangle$,
λ_{min} can be well approximated by $\langle H \rangle_\psi$

But how can we do that?

We cannot take $|\psi\rangle$ in an arbitrary manner because the overlap with $|\psi_{\mathrm{min}}\rangle$ is exponentially small with increasing system size

State Preparation in Variational Method

How can we take $|\psi\rangle$ to be close to $|\psi_{\min}\rangle$?

<u>Strategy:</u>

- Consider a certain initial state $|\psi_0\rangle$
- Generate a trial state $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_0\rangle$ by applying a unitary $U(\boldsymbol{\theta})$ (called **Ansatz**) to $|\psi_0\rangle$
- Calculate $\lambda(\boldsymbol{\theta}) := \langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle$
- Find the smallest value of $\lambda(\boldsymbol{\theta})$ by varying the parameter $\boldsymbol{\theta}$

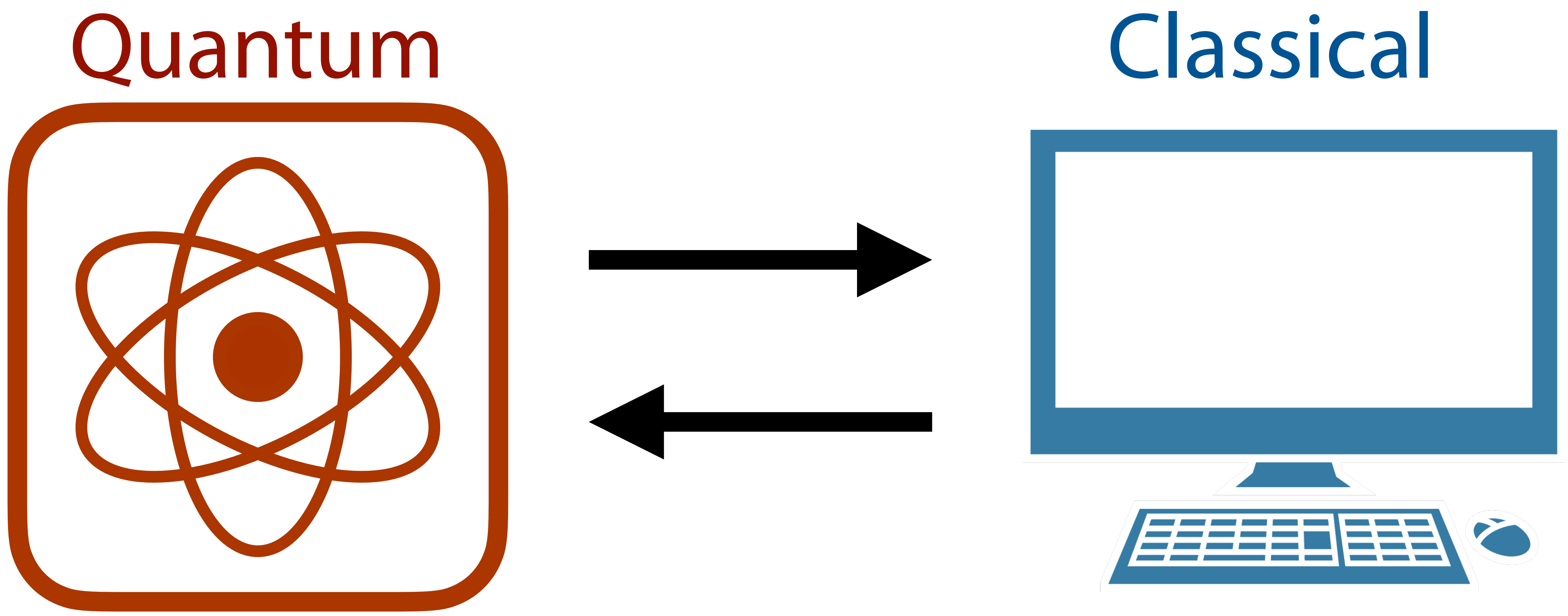
$$\boldsymbol{\theta} \to \boldsymbol{\theta}^* \Longrightarrow \lambda(\boldsymbol{\theta}^*) = \langle\psi(\boldsymbol{\theta}^*)|H|\psi(\boldsymbol{\theta}^*)\rangle \sim \langle\psi_{\min}|H|\psi_{\min}\rangle = \lambda_{\min}$$

State Preparation in Variational Method

$$\boldsymbol{\theta} \to \boldsymbol{\theta}^* \;\Rightarrow\; \lambda(\boldsymbol{\theta}^*) = \langle \psi(\boldsymbol{\theta}^*) \,|\, H \,|\, \psi(\boldsymbol{\theta}^*) \rangle \sim \langle \psi_{\mathrm{min}} \,|\, H \,|\, \psi_{\mathrm{min}} \rangle = \lambda_{\mathrm{min}}$$

⇒ Variational Quantum Eigensolver (VQE)

Find optimized parameter $\boldsymbol{\theta}^*$ by repeating the calculation of $\lambda(\boldsymbol{\theta})$ using quantum computer and parameter update using classical computer

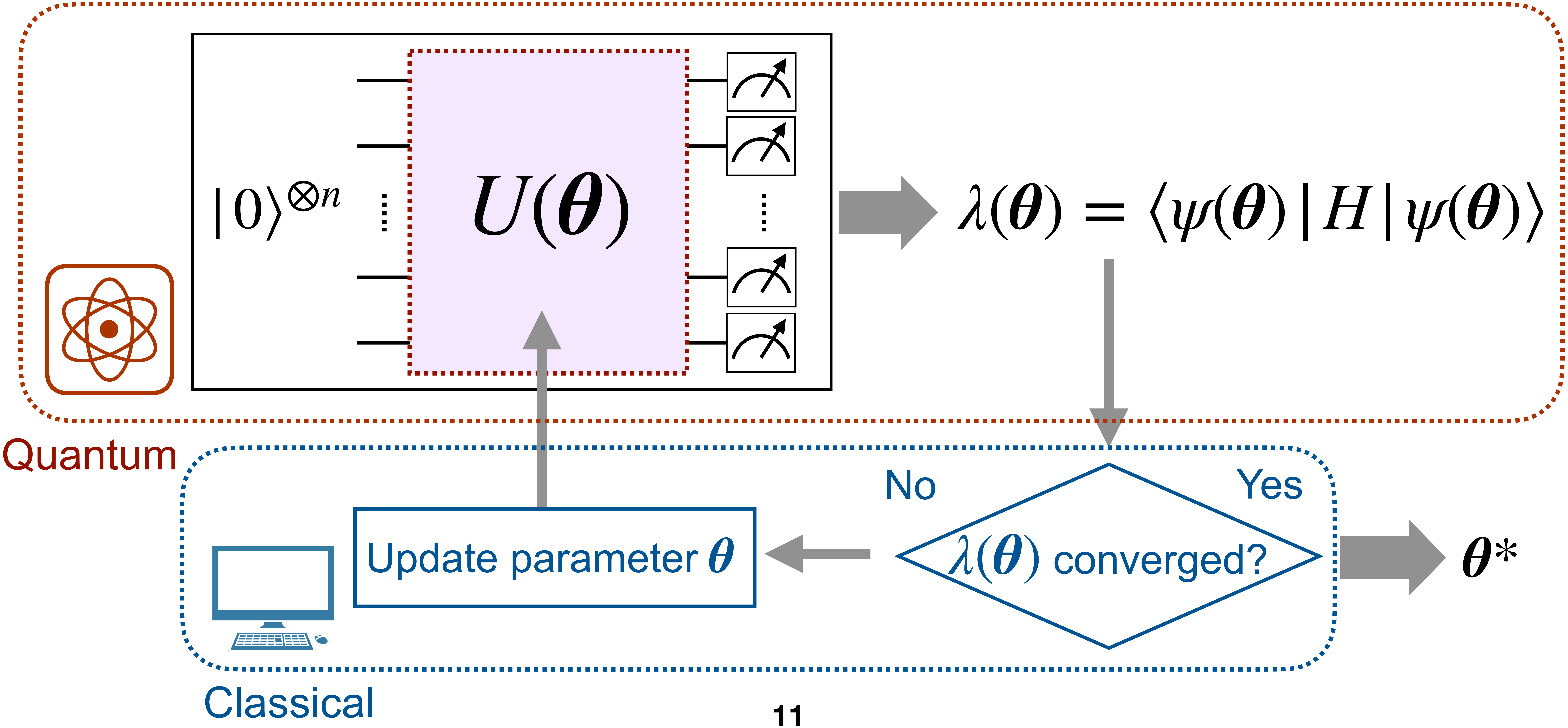


Quantum-Classical hybrid algorithm

Relatively robust against hardware noise in the present quantum computer

Variational Quantum Eigensover

Find optimized parameter $\boldsymbol{\theta}^*$ by repeating the calculation of $\lambda(\boldsymbol{\theta})$ using quantum computer and parameter update using classical computer



Parameter Optimization

Optimize parameter $\boldsymbol{\theta}$ so that $\lambda(\boldsymbol{\theta}) := \langle \psi(\boldsymbol{\theta}) \,|\, H \,|\, \psi(\boldsymbol{\theta}) \rangle$ becomes the smallest

$$\boldsymbol{\theta} \rightarrow \boldsymbol{\theta}^* \quad \Rightarrow \quad \lambda(\boldsymbol{\theta}^*) = \langle \psi(\boldsymbol{\theta}^*) \,|\, H \,|\, \psi(\boldsymbol{\theta}^*) \rangle \sim \langle \psi_{\mathrm{min}} \,|\, H \,|\, \psi_{\mathrm{min}} \rangle = \lambda_{\mathrm{min}}$$

➡ **Gradient descent** to find parameter $\boldsymbol{\theta}^*$ that minimizes $\lambda(\boldsymbol{\theta})$

Calculate the gradient $\partial \lambda(\boldsymbol{\theta}) / \partial \theta_j$ with respect to each θ_j and update the parameter θ_j so that $\lambda(\boldsymbol{\theta})$ decreases

$$\theta_j \; \rightarrow \; \theta_j' = \theta_j - \epsilon \frac{\partial \lambda(\boldsymbol{\theta})}{\partial \theta_j}$$

$\epsilon(>0)$ = Learning rate

Parameter Optimization

So-called "**Parameter Shift Rule**" used to obtain the gradient for certain type of unitaries

Consider a unitary $U(\boldsymbol{\theta}) = \prod_{j=1}^{L} U_j(\theta_j)$ with $U_j(\theta_j) = e^{-i\theta_j P_j/2}$ $\quad P_j \in \{X, Y, Z\}$

Parameter Optimization

So-called "**Parameter Shift Rule**" used to obtain the gradient for certain type of unitaries

Consider a unitary $U(\boldsymbol{\theta}) = \prod_{j=1}^{L} U_j(\theta_j)$ with $U_j(\theta_j) = e^{-i\theta_j P_j/2}$ $\quad P_j \in \{X, Y, Z\}$

Expectation value of observable M

$$\langle M(\boldsymbol{\theta})\rangle = \mathrm{Tr}\left[MU(\boldsymbol{\theta})\rho U(\boldsymbol{\theta})^\dagger\right]$$

$$\Rightarrow \frac{\partial}{\partial\theta_j}\langle M(\boldsymbol{\theta})\rangle = \frac{1}{2}\left[\left\langle M\left(\boldsymbol{\theta} + \frac{\pi}{2}\boldsymbol{e}_j\right)\right\rangle - M\left\langle\left(\boldsymbol{\theta} - \frac{\pi}{2}\boldsymbol{e}_j\right)\right\rangle\right]$$

$\boldsymbol{e}_j$ = unit vector with 1 in j-th element, 0 otherwise

Gradient of the expectation value can be obtained as a difference between two expectation values with $\theta_j \pm \pi/2$

Caveat:

Need to calculate expectation values twice per parameter

Variational Quantum Algorithm

VQE is a typical example of Variational Quantum Algorithm (VQA)

Variational Quantum Algorithm

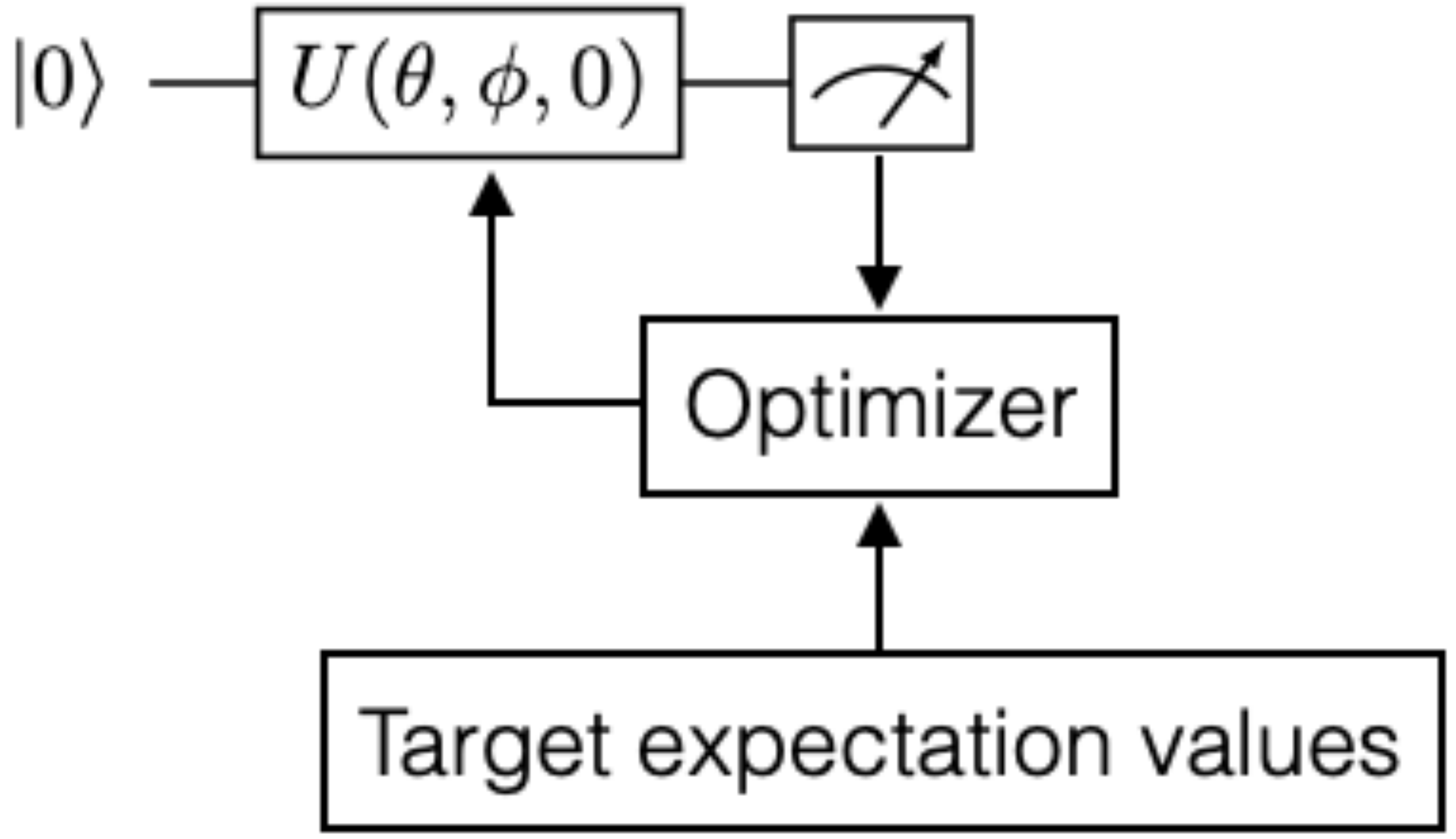
- Implement unitary operator $U(\boldsymbol{\theta})$ with parameterized quantum circuit (called *variational quantum circuit* or *variational form*)
- Generate output state $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_0\rangle$ by applying $U(\boldsymbol{\theta})$ to an initial state $|\psi_0\rangle$
- Optimize parameter $\boldsymbol{\theta}$ so that the output state approximates the desired state

Exercise of VQA

Let us try to approximate randomly chosen quantum state using VQA

- Generate a random single-qubit state $|\psi_0\rangle$
- Approximate $|\psi_0\rangle$ with a trial state $|\psi(\theta,\phi)\rangle = U(\theta,\phi,0)|0\rangle$
- Use $U(\theta,\phi,\lambda=0)$ gate as a single-qubit variational form:

$$U(\theta,\phi,\lambda) = \begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i\lambda+i\phi}\cos\frac{\theta}{2} \end{pmatrix}$$



Single-qubit state fully determined by the expectation values of Pauli X, Y and Z operators

➡ Optimize (θ,ϕ) so that

$$\langle P\rangle_{|\psi_0\rangle} \approx \langle P\rangle_{|\psi(\theta,\phi)\rangle}\ {}^\forall P \in \{X,Y,Z\}$$

Exercise of VQE

Consider a problem of minimizing the expectation value of an observable using parameter shift rule

- Generate a trial state $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|0\rangle$ with

$$U(\boldsymbol{\theta}) = \prod_l \left(\prod_j R_j^Z(\theta_{j2}^l) R_j^Y(\theta_{j1}^l) \right)$$

- Calculate the expectation value $\langle\psi(\boldsymbol{\theta})|O|\psi(\boldsymbol{\theta})\rangle$ with $O = ZXY$
- Minimize it in a quantum-classical optimization loop

We will try another VQE exercise in a later session

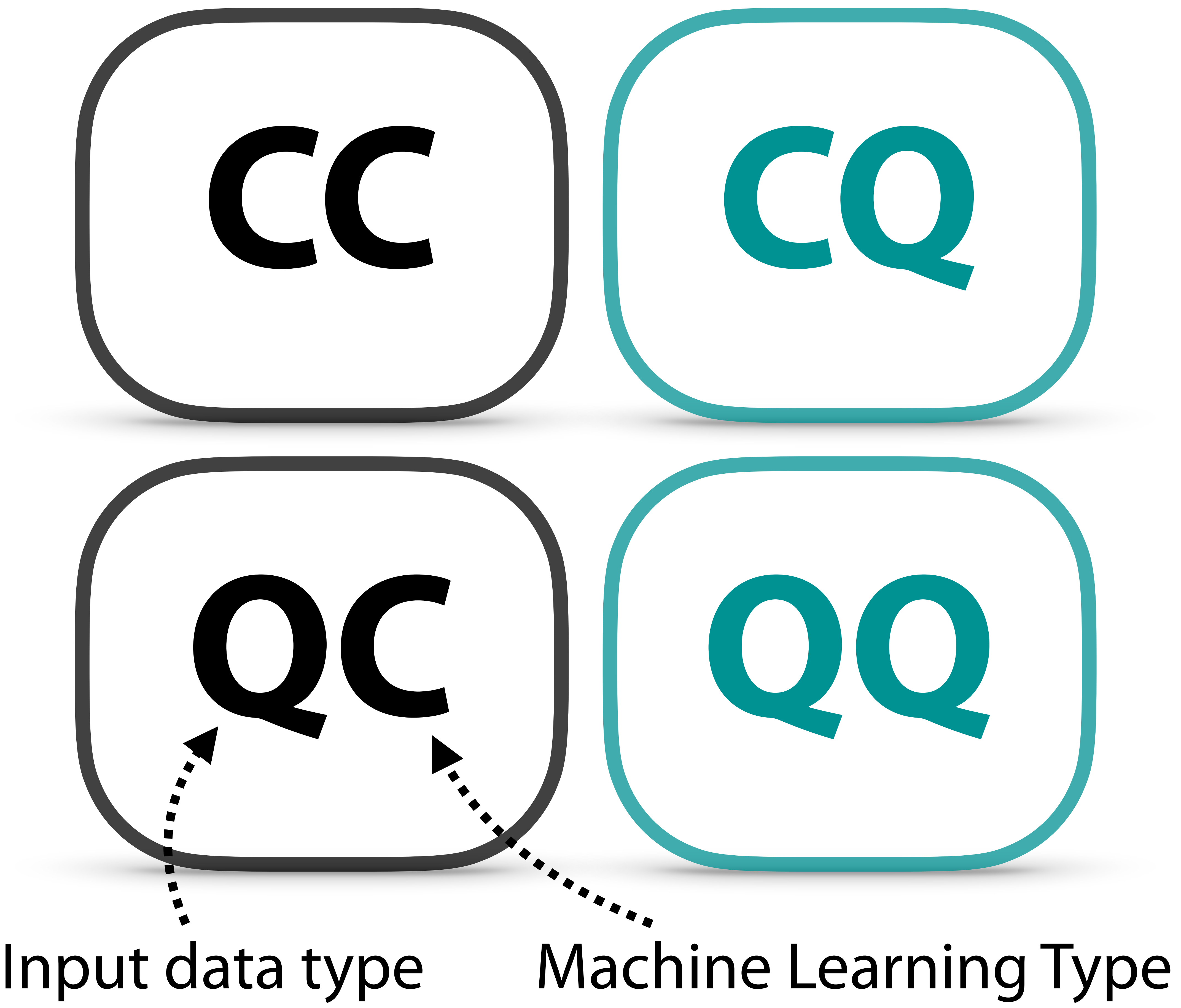
Hands-on Exercise (I)

- Variations Quantum Algorithm :
  - Approximate randomly chosen quantum state
- Variational Quantum Eigensolver :
  - Minimize expectation value of an observable

Quantum Machine Learning

Performing machine learning task by including quantum computing technology in the training and/or inference processes

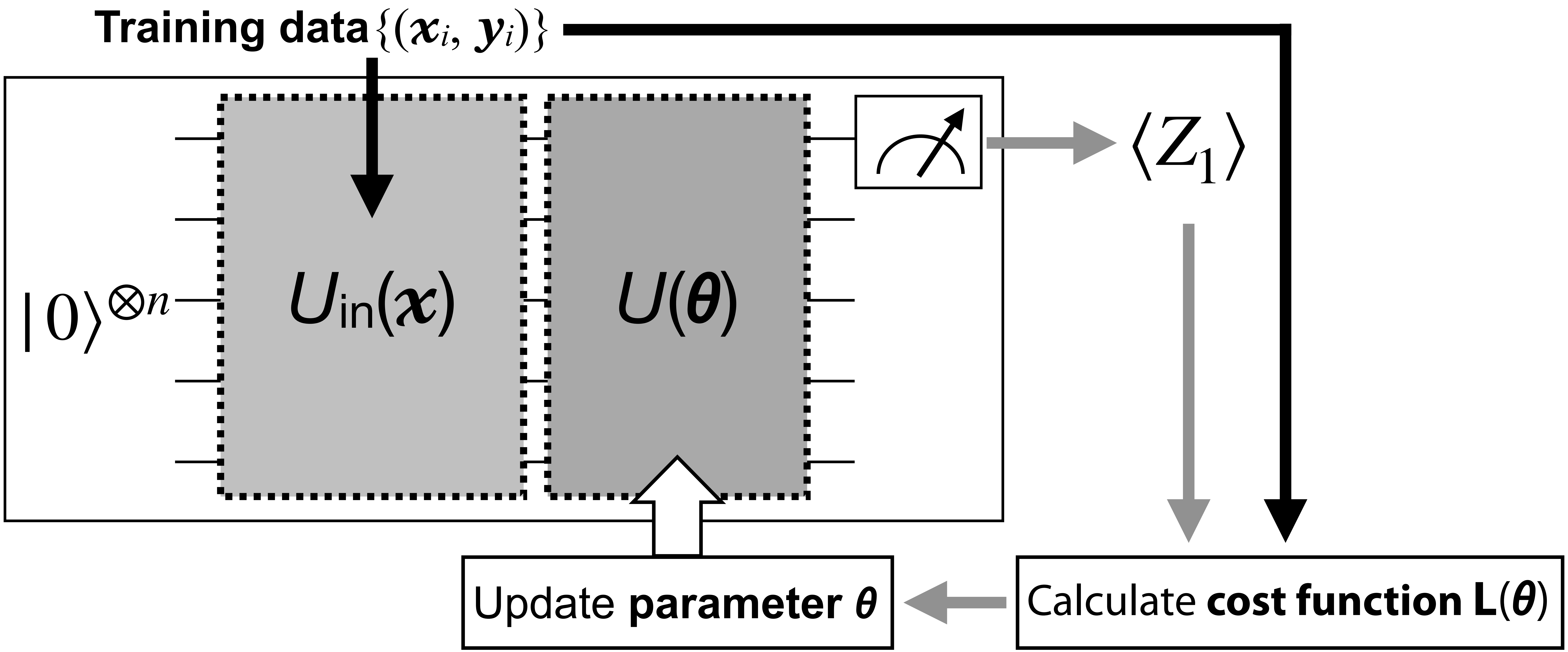
➡ **Quantum Machine Learning**



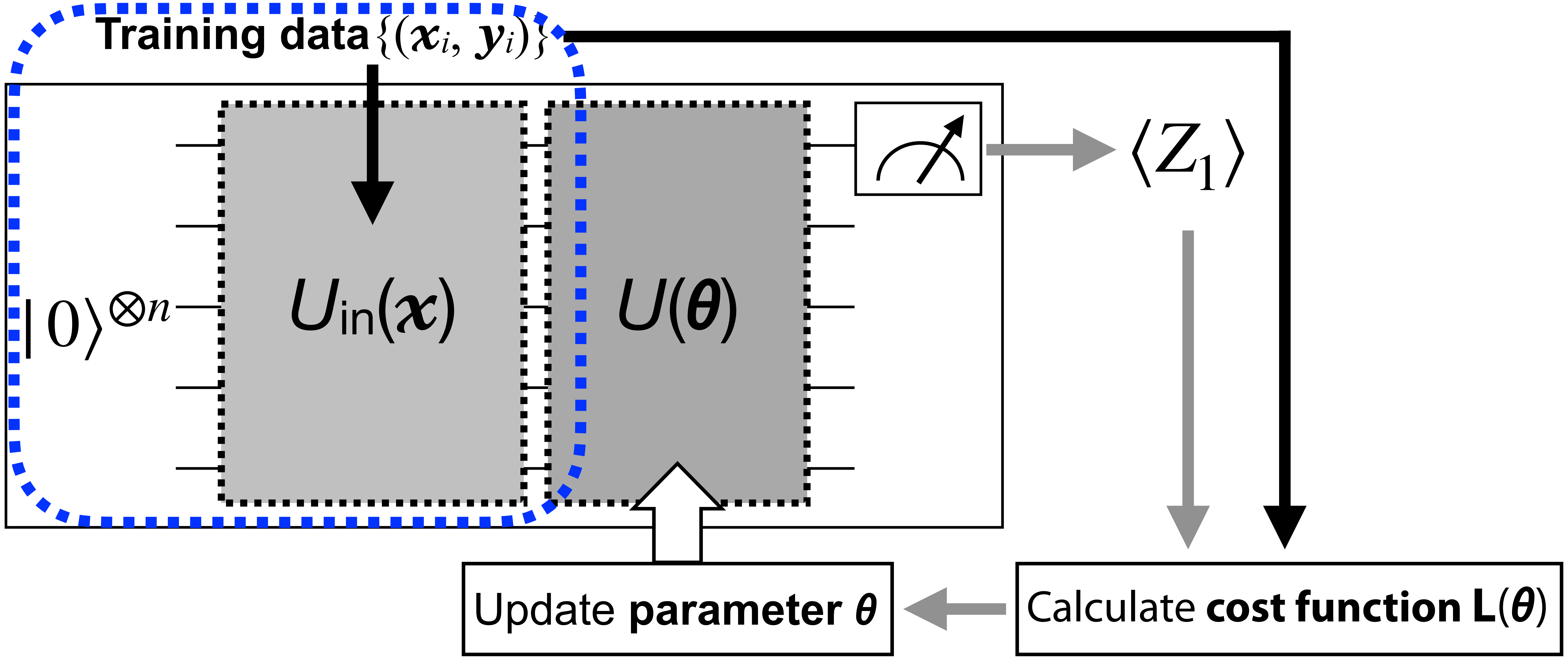
Classical = **C**
Quantum = **Q**

In the hands-on exercise, we will try QML in both CQ and QQ settings

Conventional **Quantum Neural Network (QNN)** model for supervised machine learning task



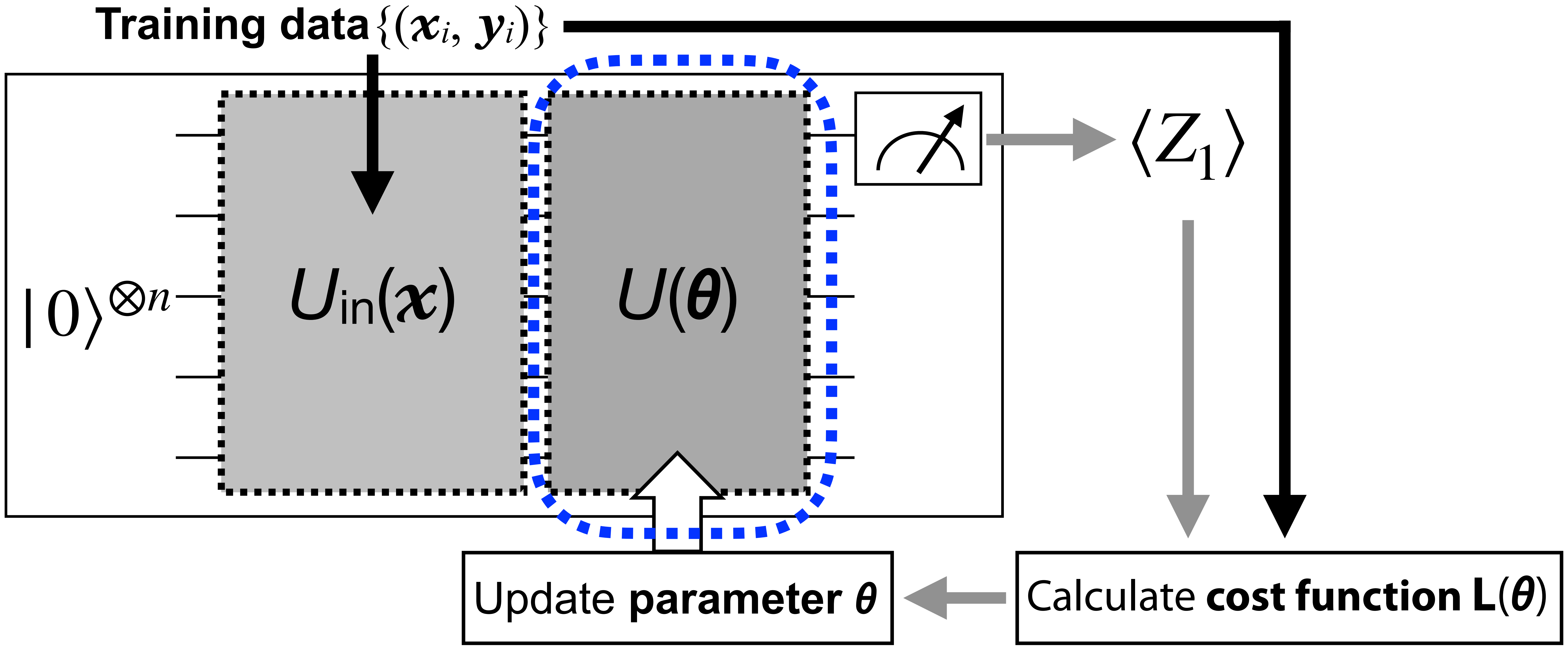
QML with Quantum Neural Networks



1. Prepare $|\psi_{\text{in}}(\boldsymbol{x}_i)\rangle = U_{\text{in}}(\boldsymbol{x}_i)|0\rangle^{\otimes n}$ using unitary $U_{\text{in}}(\boldsymbol{x})$ with input data $\boldsymbol{x}_i$ as parameter

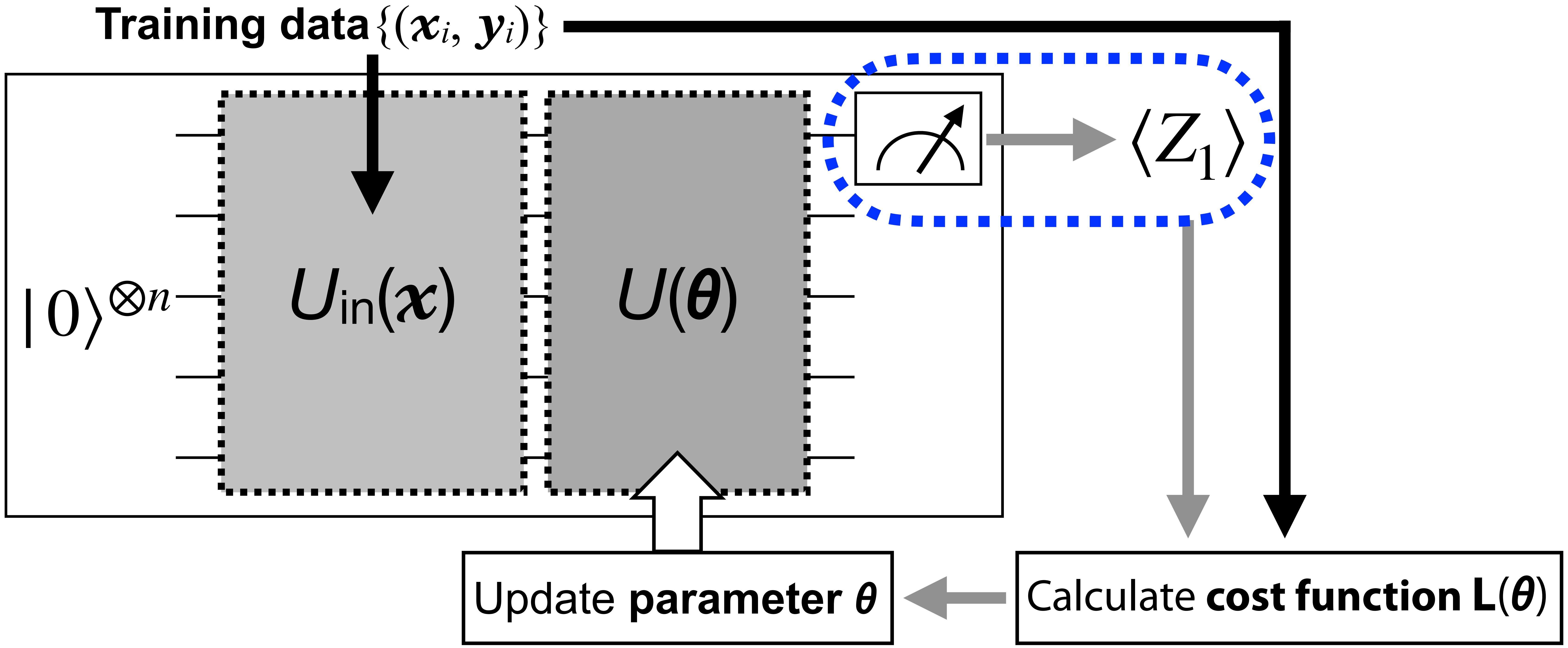
$U_{\text{in}}(\boldsymbol{x})$ called feature map

QML with Quantum Neural Networks



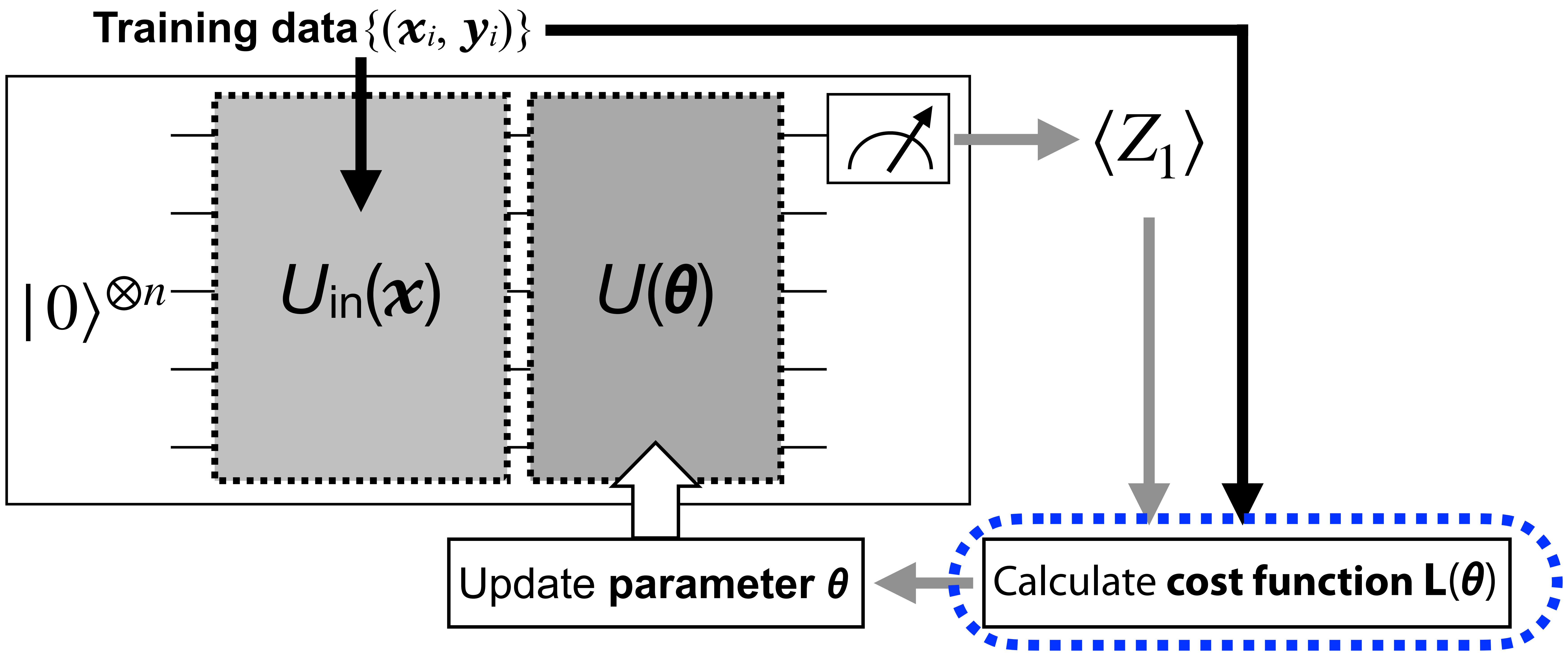
1. Prepare $|\psi_{\rm in}(\boldsymbol{x}_i)\rangle = U_{\rm in}(\boldsymbol{x}_i)|0\rangle^{\otimes n}$ using unitary $U_{\rm in}(\boldsymbol{x})$ with input data $\boldsymbol{x}_i$ as parameter
2. Generate output state $|\psi_{\rm out}(\boldsymbol{x}_i, \boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_{\rm in}(\boldsymbol{x}_i)\rangle$ by processing $|\psi_{\rm in}(\boldsymbol{x}_i)\rangle$ with $U(\boldsymbol{\theta})$

QML with Quantum Neural Networks



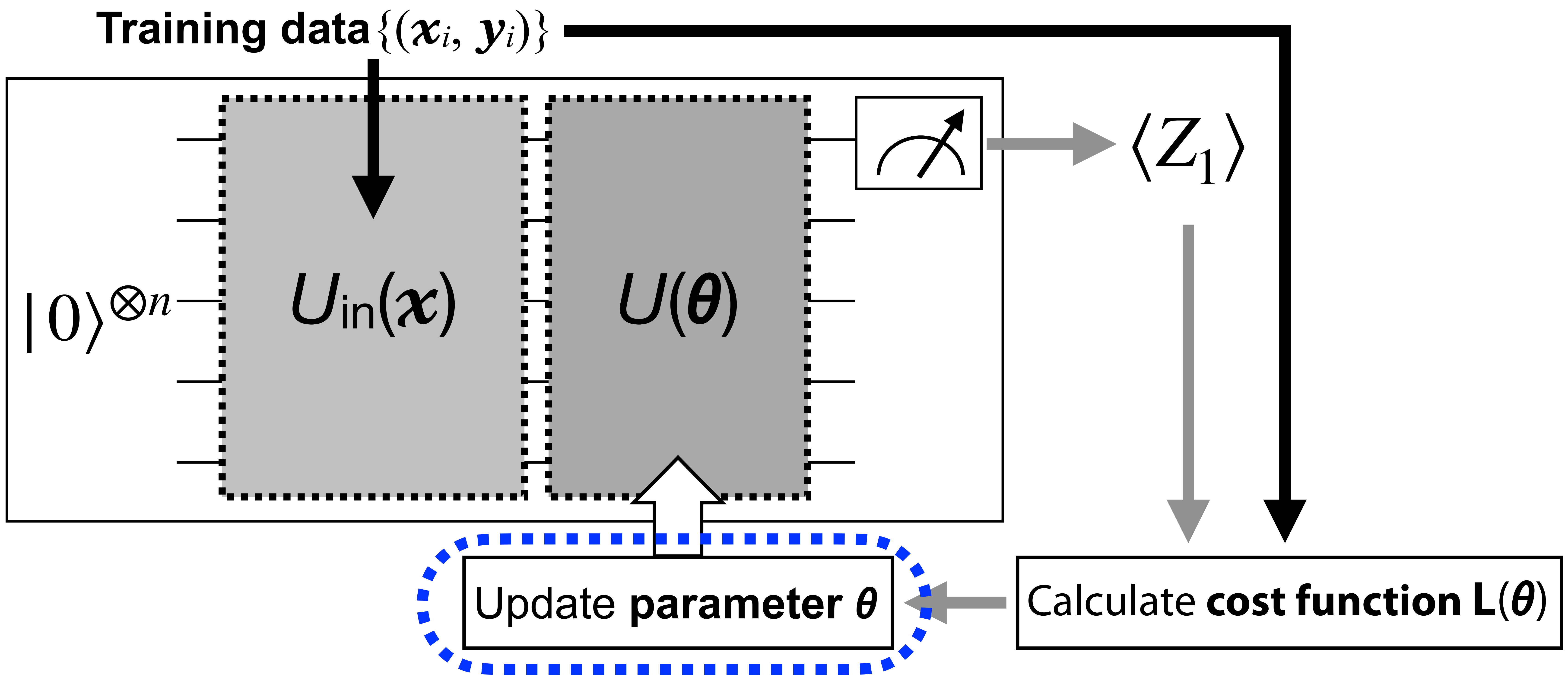
1. Prepare $|\psi_{\mathrm{in}}(\boldsymbol{x}_i)\rangle = U_{\mathrm{in}}(\boldsymbol{x}_i)|0\rangle^{\otimes n}$ using unitary $U_{\mathrm{in}}(\boldsymbol{x})$ with input data $\boldsymbol{x}_i$ as parameter
2. Generate output state $|\psi_{\mathrm{out}}(\boldsymbol{x}_i, \boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_{\mathrm{in}}(\boldsymbol{x}_i)\rangle$ by processing $|\psi_{\mathrm{in}}(\boldsymbol{x}_i)\rangle$ with $U(\boldsymbol{\theta})$
3. Measure observable O (e.g, Pauli Z operator on the first qubit) under the state $|\psi_{\mathrm{out}}\rangle$

QML with Quantum Neural Networks



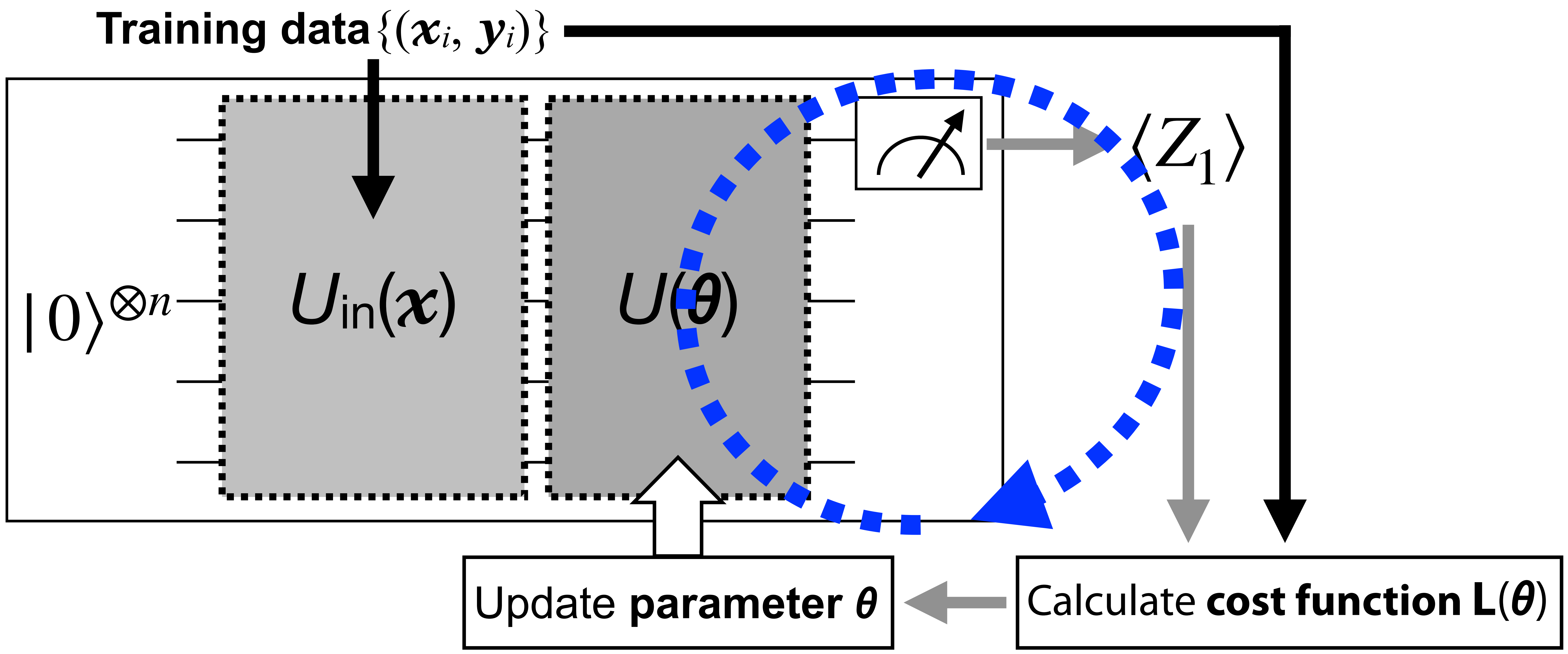
1. Prepare $|\psi_{\mathrm{in}}(\boldsymbol{x}_i)\rangle = U_{\mathrm{in}}(\boldsymbol{x}_i)|0\rangle^{\otimes n}$ using unitary $U_{\mathrm{in}}(\boldsymbol{x})$ with input data $\boldsymbol{x}_i$ as parameter
2. Generate output state $|\psi_{\mathrm{out}}(\boldsymbol{x}_i, \boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_{\mathrm{in}}(\boldsymbol{x}_i)\rangle$ by processing $|\psi_{\mathrm{in}}(\boldsymbol{x}_i)\rangle$ with $U(\boldsymbol{\theta})$
3. Measure observable O (e.g, Pauli Z operator on the first qubit) under the state $|\psi_{\mathrm{out}}\rangle$
4. Calculate cost function $L(\boldsymbol{\theta})$ from the model output $F(\langle O \rangle_{\boldsymbol{x}_i, \boldsymbol{\theta}})$ and input label $\boldsymbol{y}_i$

QML with Quantum Neural Networks



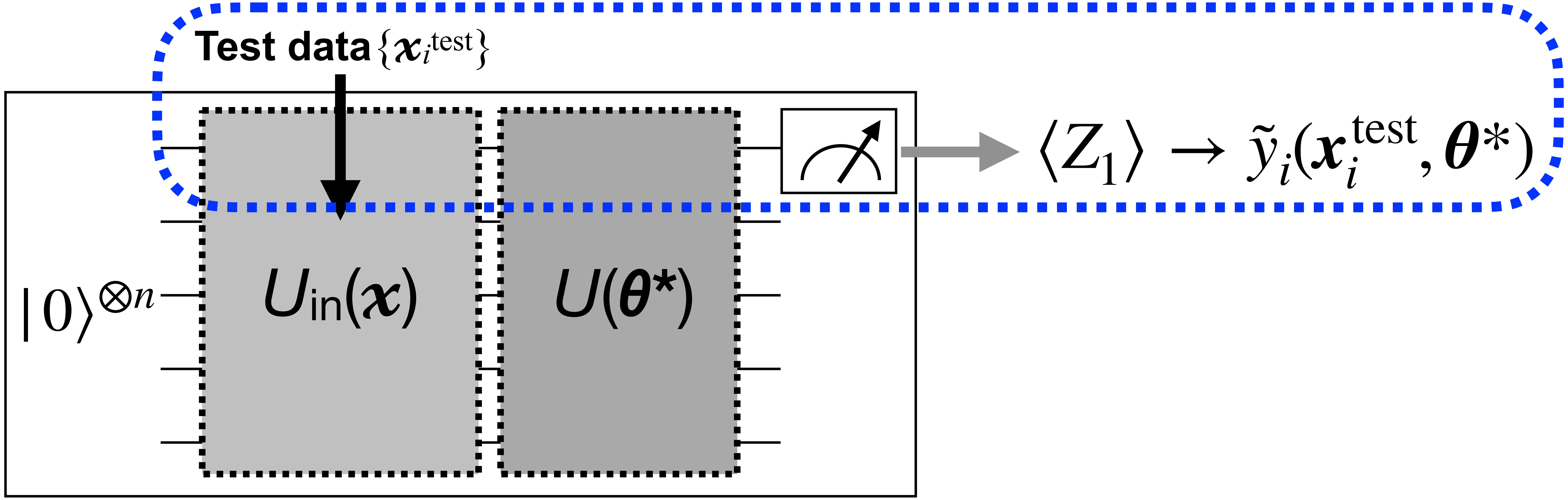
1. Prepare $|\psi_{\mathrm{in}}(\boldsymbol{x}_i)\rangle = U_{\mathrm{in}}(\boldsymbol{x}_i)|0\rangle^{\otimes n}$ using unitary $U_{\mathrm{in}}(\boldsymbol{x})$ with input data $\boldsymbol{x}_i$ as parameter
2. Generate output state $|\psi_{\mathrm{out}}(\boldsymbol{x}_i, \boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_{\mathrm{in}}(\boldsymbol{x}_i)\rangle$ by processing $|\psi_{\mathrm{in}}(\boldsymbol{x}_i)\rangle$ with $U(\boldsymbol{\theta})$
3. Measure observable O (e.g, Pauli Z operator on the first qubit) under the state $|\psi_{\mathrm{out}}\rangle$
4. Calculate cost function $L(\boldsymbol{\theta})$ from the model output $F(\langle O \rangle_{\boldsymbol{x}_i, \boldsymbol{\theta}})$ and input label $\boldsymbol{y}_i$
5. Update parameter $\boldsymbol{\theta}$ so that $L(\boldsymbol{\theta})$ becomes smaller

QML with Quantum Neural Networks



1. Prepare $|\psi_{\text{in}}(\boldsymbol{x}_i)\rangle = U_{\text{in}}(\boldsymbol{x}_i)|0\rangle^{\otimes n}$ using unitary $U_{\text{in}}(\boldsymbol{x})$ with input data $\boldsymbol{x}_i$ as parameter
2. Generate output state $|\psi_{\text{out}}(\boldsymbol{x}_i, \boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_{\text{in}}(\boldsymbol{x}_i)\rangle$ by processing $|\psi_{\text{in}}(\boldsymbol{x}_i)\rangle$ with $U(\boldsymbol{\theta})$
3. Measure observable O (e.g, Pauli Z operator on the first qubit) under the state $|\psi_{\text{out}}\rangle$
4. Calculate cost function $L(\boldsymbol{\theta})$ from the model output $F(\langle O \rangle_{\boldsymbol{x}_i, \boldsymbol{\theta}})$ and input label $\boldsymbol{y}_i$
5. Update parameter $\boldsymbol{\theta}$ so that $L(\boldsymbol{\theta})$ becomes smaller
6. Determine optimized parameter $\boldsymbol{\theta}^*$ that minimizes $L(\boldsymbol{\theta})$ by iterating the 2-5 steps

QML with Quantum Neural Networks



1. Prepare $|\psi_{\text{in}}(\boldsymbol{x}_i)\rangle = U_{\text{in}}(\boldsymbol{x}_i)|0\rangle^{\otimes n}$ using unitary $U_{\text{in}}(\boldsymbol{x})$ with input data $\boldsymbol{x}_i$ as parameter
2. Generate output state $|\psi_{\text{out}}(\boldsymbol{x}_i, \boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_{\text{in}}(\boldsymbol{x}_i)\rangle$ by processing $|\psi_{\text{in}}(\boldsymbol{x}_i)\rangle$ with $U(\boldsymbol{\theta})$
3. Measure observable O (e.g, Pauli Z operator on the first qubit) under the state $|\psi_{\text{out}}\rangle$
4. Calculate cost function $L(\boldsymbol{\theta})$ from the model output $F(\langle O \rangle_{\boldsymbol{x}_i, \boldsymbol{\theta}})$ and input label $\boldsymbol{y}_i$
5. Update parameter $\boldsymbol{\theta}$ so that $L(\boldsymbol{\theta})$ becomes smaller
6. Determine optimized parameter $\boldsymbol{\theta}^*$ that minimizes $L(\boldsymbol{\theta})$ by iterating the 2-5 steps
7. Predict the label $\tilde{y}_i(\boldsymbol{x}_i^{\text{test}}, \boldsymbol{\theta}^*)$ for unseen test data $\boldsymbol{x}_i^{\text{test}}$ using the trained model

Simple Example of QML

Let us consider an *inverse problem* to find a function f when only input data x_i and the output $y_i = f(x_i)$ are given

- Feature map: $U_{\text{in}}(x_i) = \prod_j R_j^Z(\cos^{-1}(x^2))R_j^Y(\sin^{-1}(x))$
- Variational form: $U(\boldsymbol{\theta}) : U(\{\theta_j^l\}) = \prod_{l=1}^{d}\left(\left(\prod_{j=1}^{n} U_{\text{rot}}(\theta_j^l)\right)\cdot U_{\text{ent}}\right)\cdot \prod_{j=1}^{n} U_{\text{rot}}(\theta_j^0)$

  Apply U_{rot} first, then use d times a pair of U_{ent} and U_{rot}

  $U_{\text{rot}}(\theta_j^l) = R_j^Y(\theta_{j3}^l)R_j^Z(\theta_{j2}^l)R_j^Y(\theta_{j1}^l)$: Single-qubit rotation gates with $\theta's$ as parameters

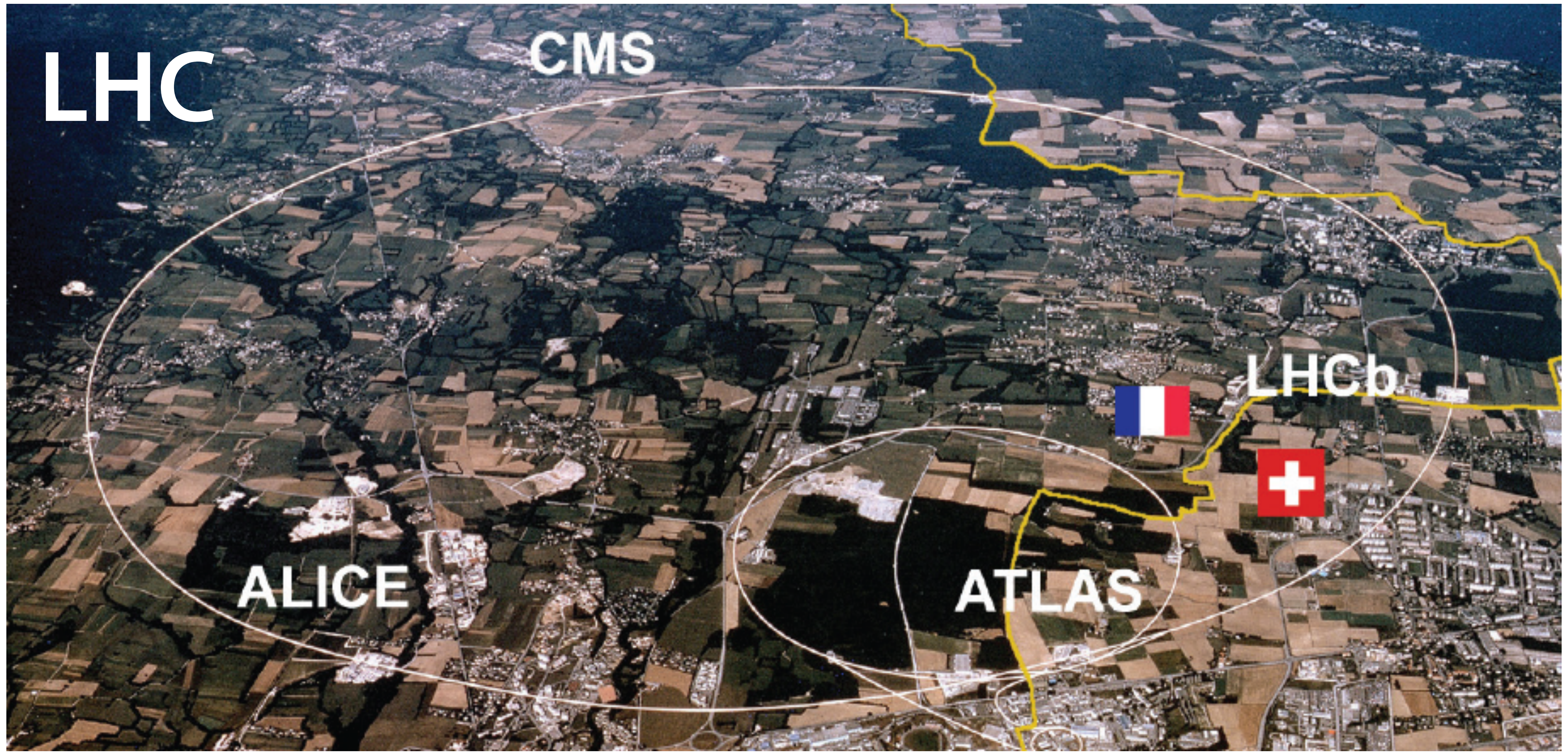
  $U_{\text{ent}} = \prod_{j=1}^{n} C_{j\%n+1}^{j}[Z]$: Controlled-Z gate as an entangling gate

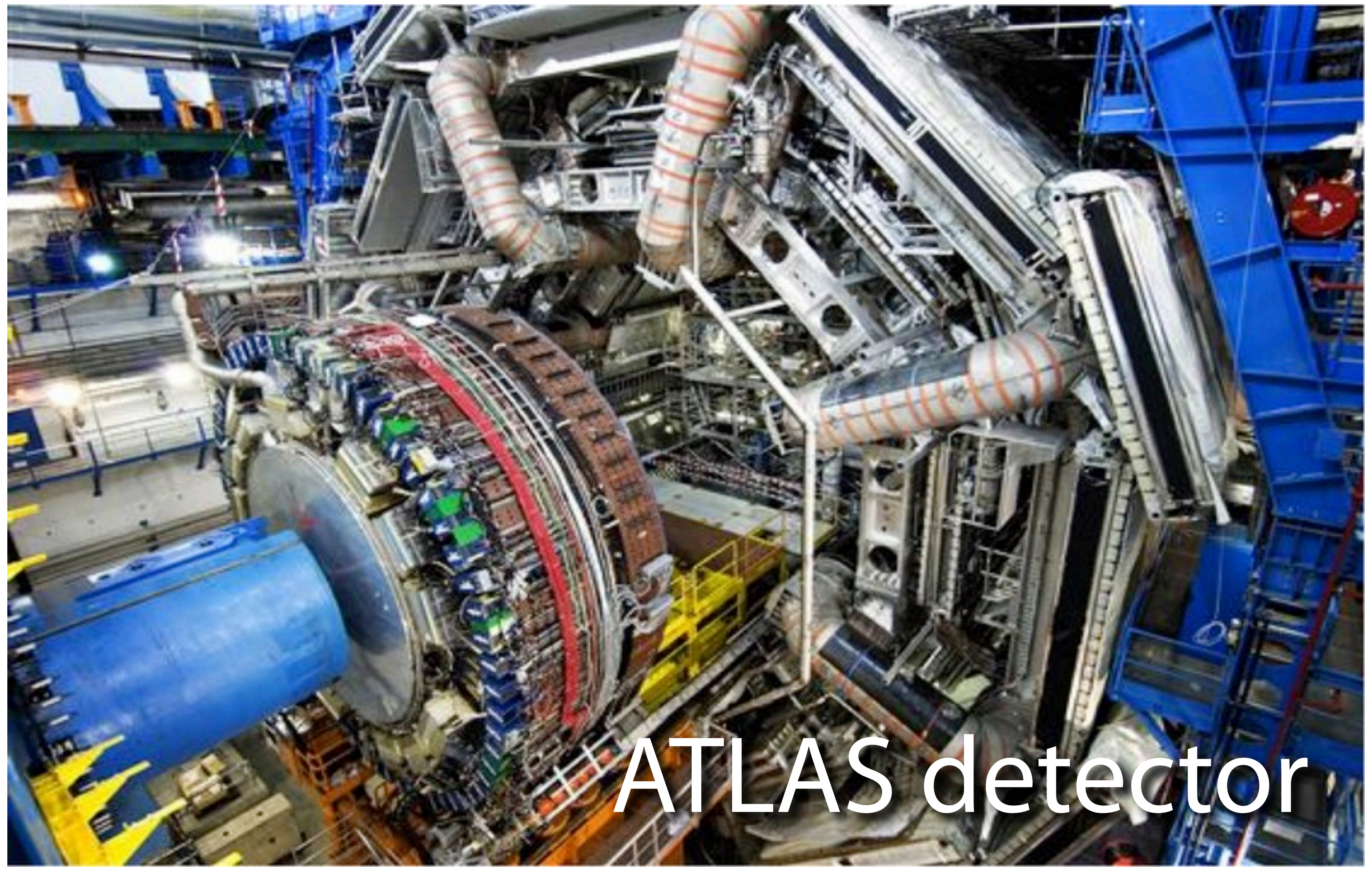
Hands-on Exercise (II)

- Quantum Machine Learning :
  - Learn a function from input and output data from the function

Application to High-Energy Physics

Learn more complex data from high-energy physics (HEP) experiment
(though actually "truth-level" simulation data)





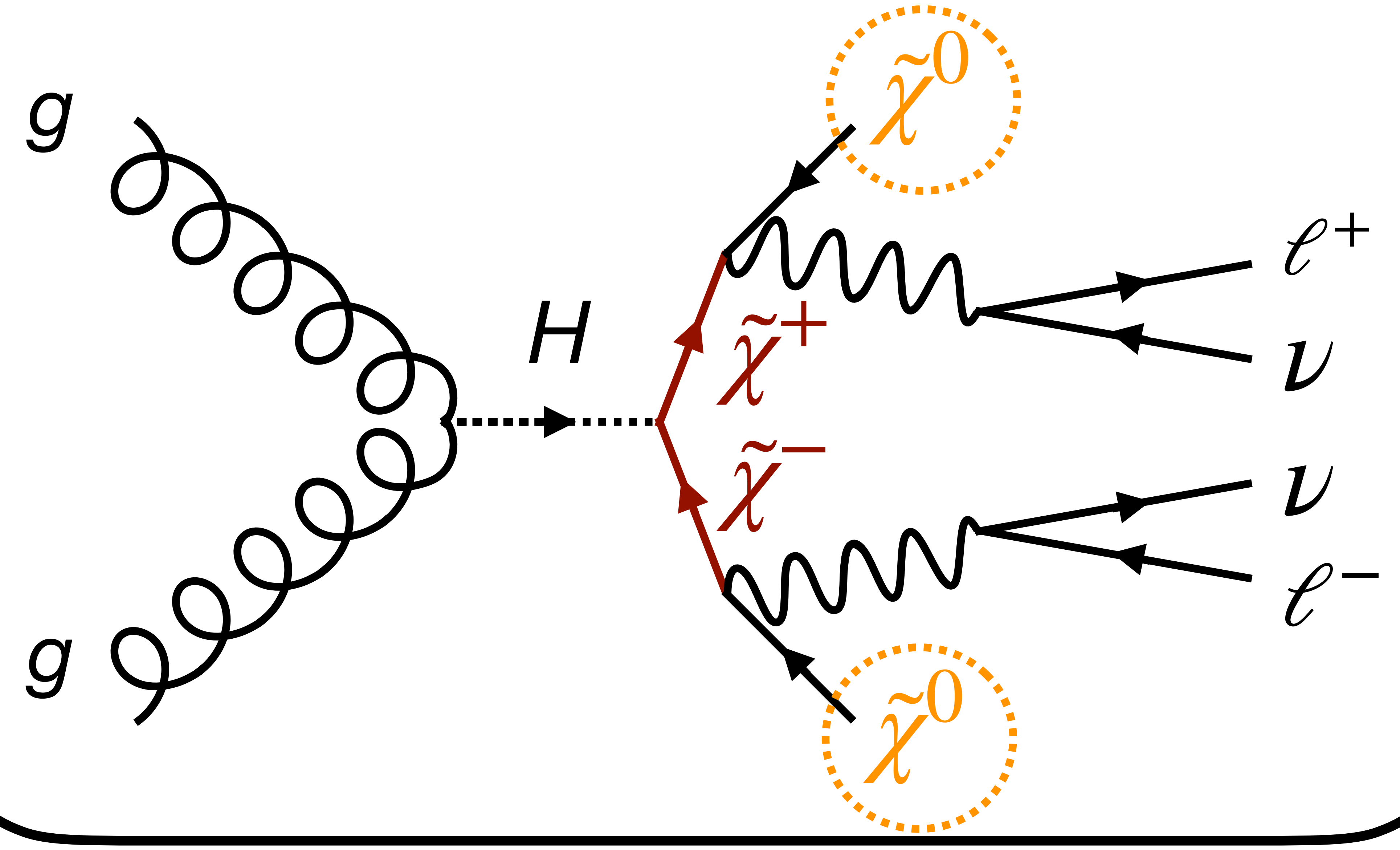
Machine learning technique is ubiquitous in HEP experiment

Detector reconstruction, simulation, data analysis, trigger, …

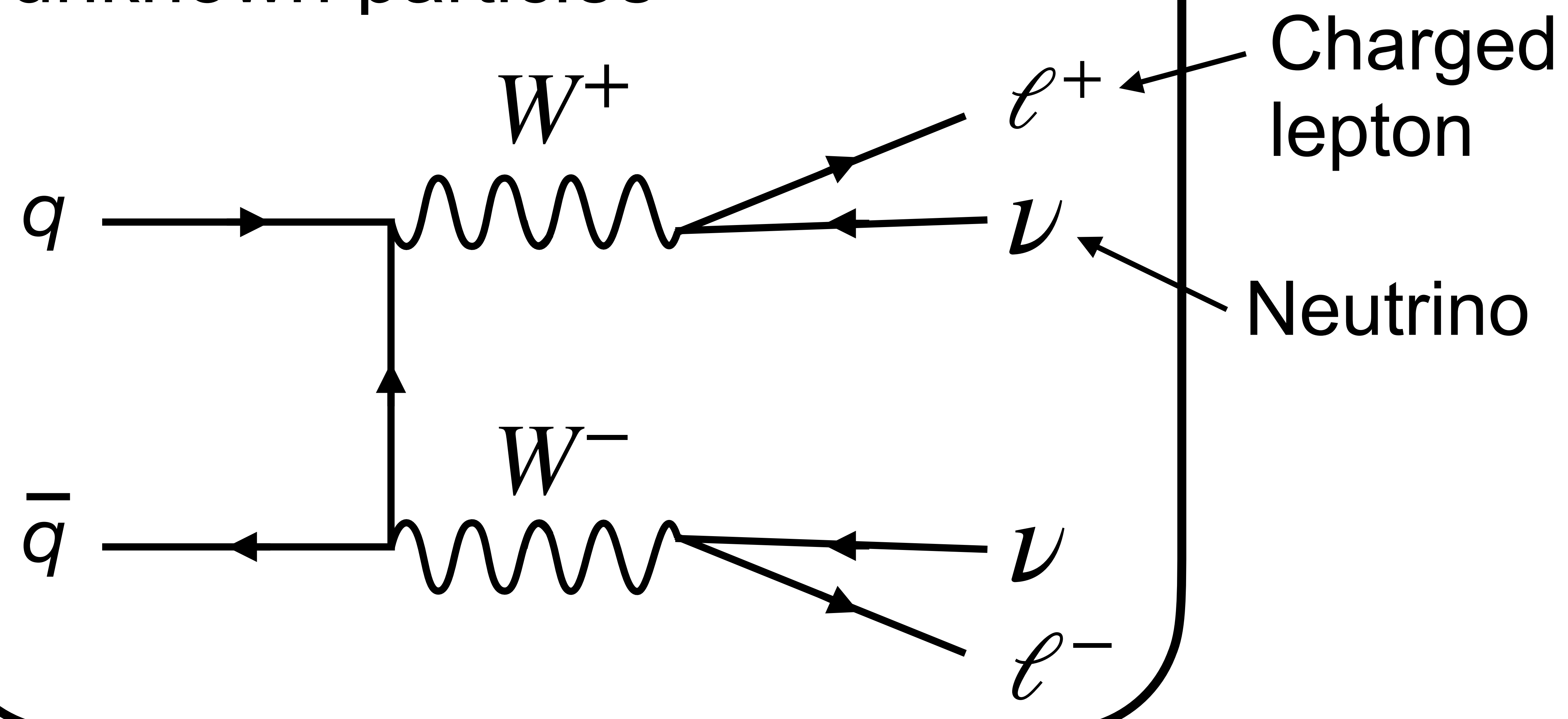
QML Application to Event Classification

Classify events that contain new physics signal from background events

Signal event with unknown particles



Background event without unknown particles



Neutralino ➡ Not observed by the detector, like SM neutrino

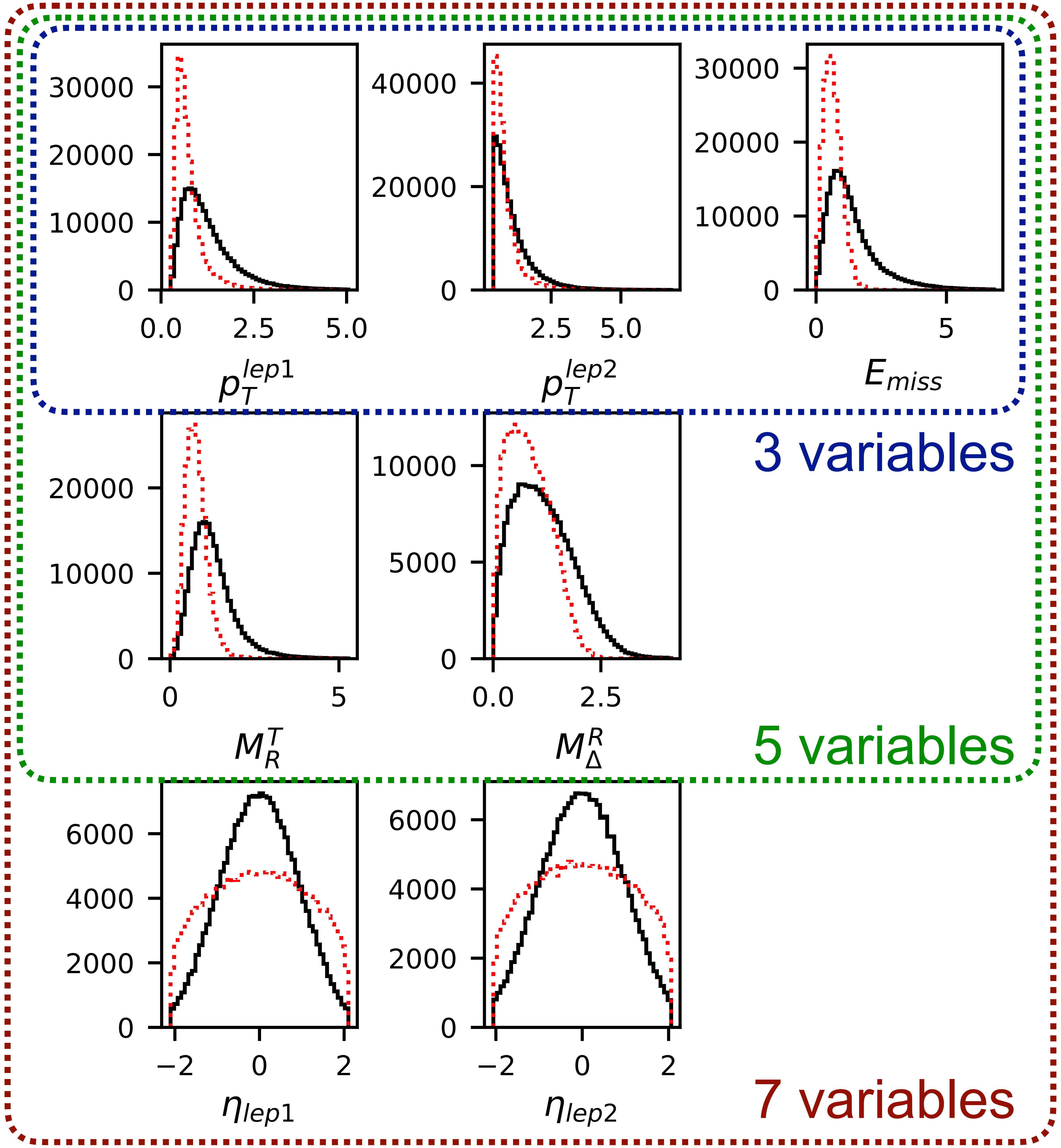
Use differences in kinematical properties due to the presence of *H* and $\tilde{\chi}^{\pm}/\tilde{\chi}^0$ to classify signal from background

Training Data

Use SUSY dataset in machine learning repository by University of California, Irvine

Signal (red histogram) and **background** (black histogram) differ in kinematical features of final state particles

Use these features as input data and classify signal and background events



Quantum Neural Network Model

- Feature map: $U_{\text{in}}(\boldsymbol{x}_i) = U_\phi(\boldsymbol{x}_i)H^{\otimes n}$

$$U_\phi(\boldsymbol{x}_i) = \exp\left(i \sum_{k=1}^{n} \phi_{\{k,k+1\}}(\boldsymbol{x}_i) Z_k Z_{k\%n+1} + i \sum_{k=1}^{n} \phi_{\{k\}}(\boldsymbol{x}_i) Z_k \right)$$

$$\phi_{\{k\}}(\boldsymbol{x}_i) = x_i^{\{k\}},\ \phi_{\{k,k+1\}}(\boldsymbol{x}_i) = (\pi - x_i^{\{k\}})(\pi - x_i^{\{k\%n+1\}})$$

➡ called ZZ feature map

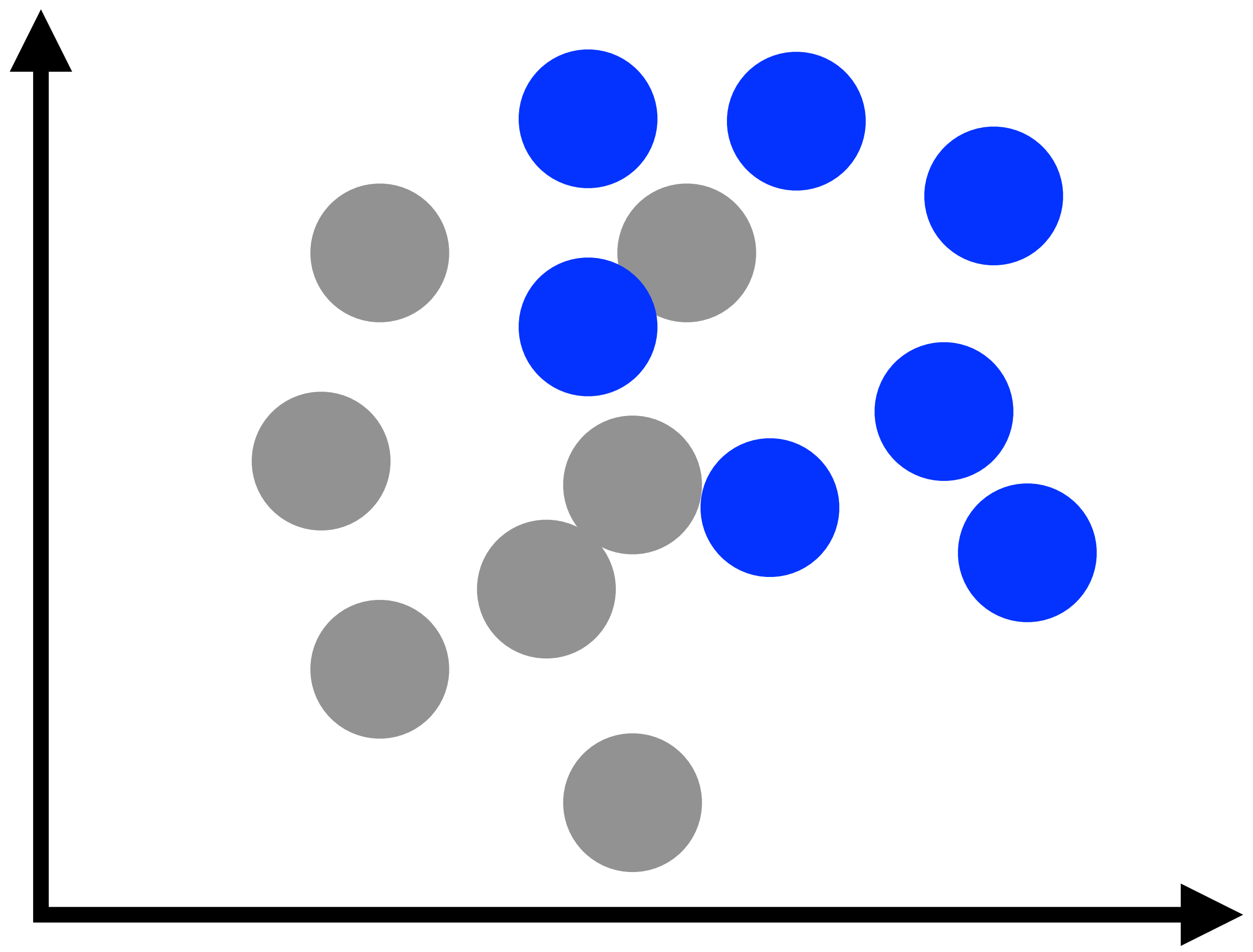
- Variational form:

$$U(\{\theta_j^l\}) = \prod_{l=1}^{d} \left(\left(\prod_{j=1}^{n} U_{\text{rot}}(\theta_j^l) \right) \cdot U_{\text{ent}} \right) \cdot \prod_{j=1}^{n} U_{\text{rot}}(\theta_j^0)$$

$$U_{\text{rot}}(\theta_j^l) = R_j^Z(\theta_{j2}^l) R_j^Y(\theta_{j1}^l) \qquad U_{\text{ent}} = \prod_{j=1}^{n} C_{j\%n+1}^{j}[Z]$$

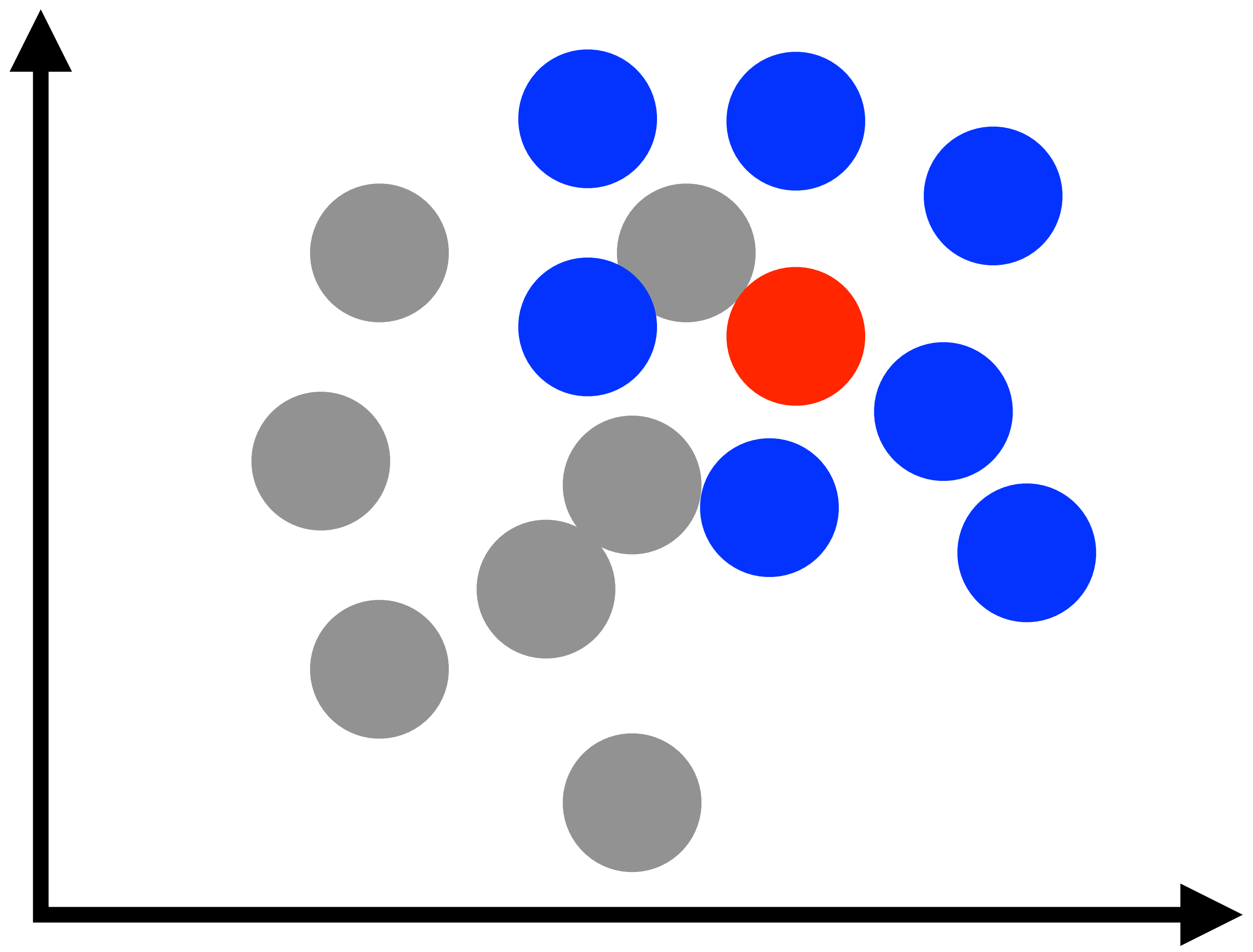
Kernel Method for Machine Learning

Assuming there are two classes of data: and



There is a new data

➡ Which class of or does the new data belong to?



Likely

➡ *Distance* or *Similarity* between data can be used to judge

Kernel Method for Machine Learning

Function to quantify the distance measure $\kappa : D \times D \to \mathbb{R}$

Consider a matrix with elements $K_{m,m'} = \kappa(\boldsymbol{x}^m, \boldsymbol{x}^{m'})$ constructed from two different training points $\boldsymbol{x}^m, \boldsymbol{x}^{m'}$ in the training sample D

Kernel Matrix

The closer the data points are, the larger the matrix elements

Representative kernel function is

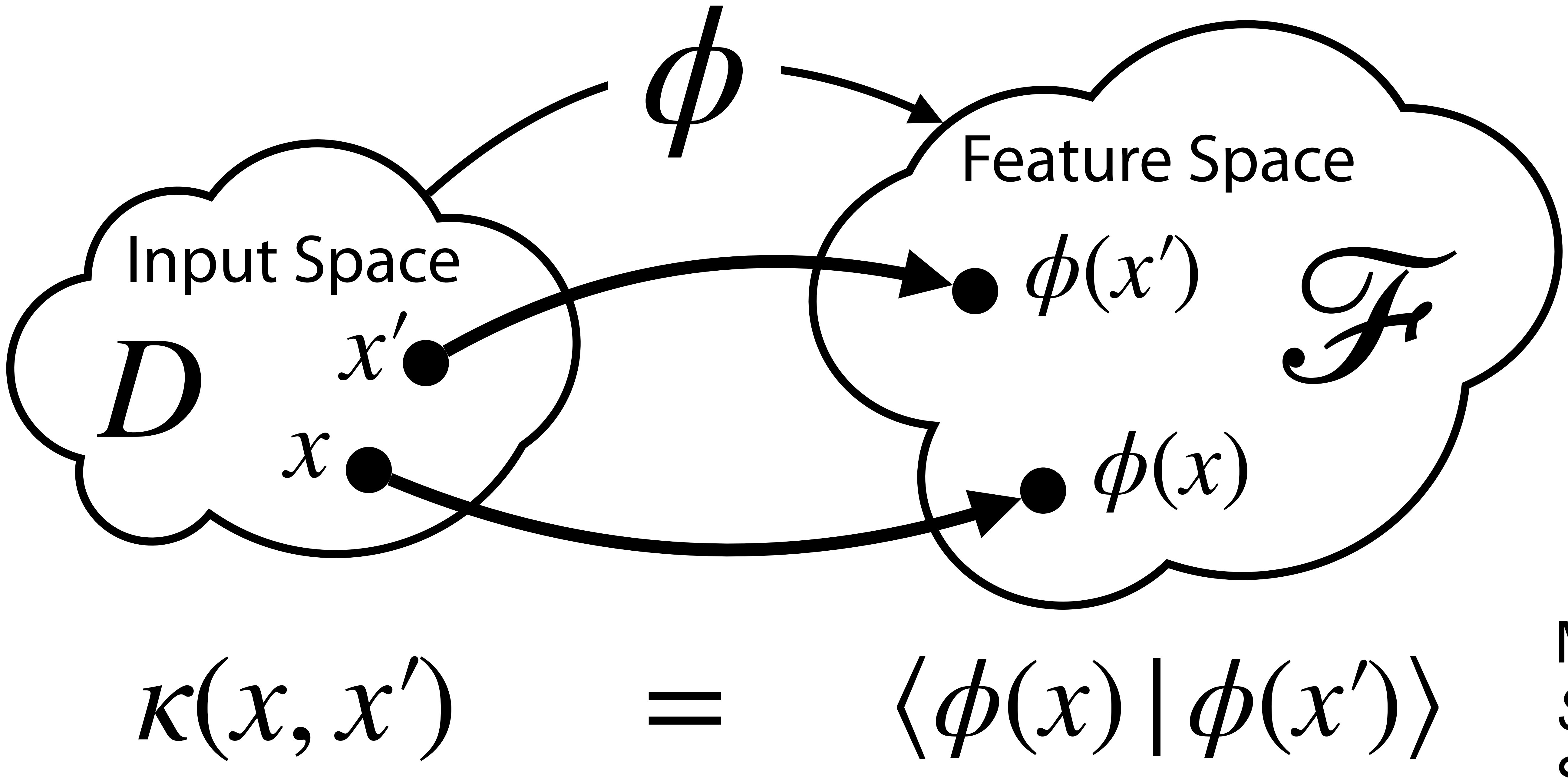
- Linear kernel : $\boldsymbol{x}^T \boldsymbol{x}'$
- Gaussian kernel : $e^{-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|^2}$

Kernel Method for Machine Learning

Input data encoded in high-dimensional Hilbert space using feature map in QML

➡ Inner products of encoded states can be used as a kernel function

$\kappa(\boldsymbol{x}^m, \boldsymbol{x}^{m'}) = \langle \phi(\boldsymbol{x}) \,|\, \phi(\boldsymbol{x}') \rangle$ for a feature map $\phi : D \to \mathscr{F}$



For positive definite kernel, there is a feature space $\mathscr{F}$ created by ϕ such that the kernel κ in D is equal to inner product in $\mathscr{F}$

M. Schuld, F. Petruccione, *Supervised Learning with Quantum Computers*, Springer Nature

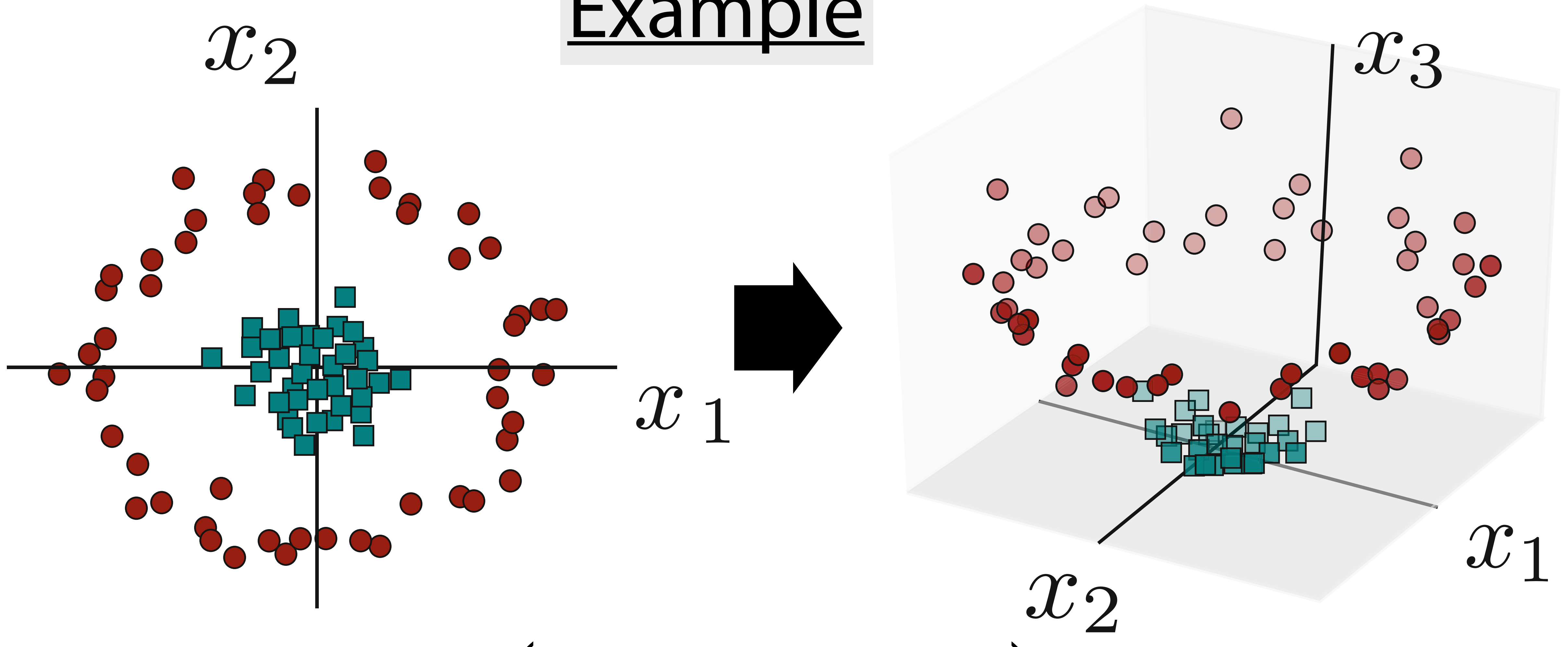
Kernel Method for Machine Learning

Input data encoded in high-dimensional Hilbert space using feature map in QML

➡ Inner products of encoded states can be used as a kernel function

$$\kappa(\boldsymbol{x}^m, \boldsymbol{x}^{m'}) = \langle \phi(\boldsymbol{x}) \,|\, \phi(\boldsymbol{x}') \rangle \text{ for a feature map } \phi : D \to \mathscr{F}$$

Example



$$\phi(x_1, x_2) = \left(x_1, x_2, \frac{(x_1^2 + x_2^2)}{2} \right)$$

Data classification can often improve when the input data are encoded into higher-dimensional feature space

M. Schuld, F. Petruccione, *Supervised Learning with Quantum Computers*, Springer Nature

How can we find a good feature map?

Kernel Method for Machine Learning

Input data encoded in high-dimensional Hilbert space using feature map in QML

➡ Inner products of encoded states can be used as a kernel function

$\kappa(\boldsymbol{x}^m, \boldsymbol{x}^{m'}) = \langle \phi(\boldsymbol{x}) \,|\, \phi(\boldsymbol{x}') \rangle$ for a feature map $\phi : D \rightarrow \mathscr{F}$

Quantum Kernel

How can we find a good feature map?

This is of course a problem and data dependent

Modifying kernel function would result in *non-trivial* changes in feature space

➡ called **Kernel Trick**

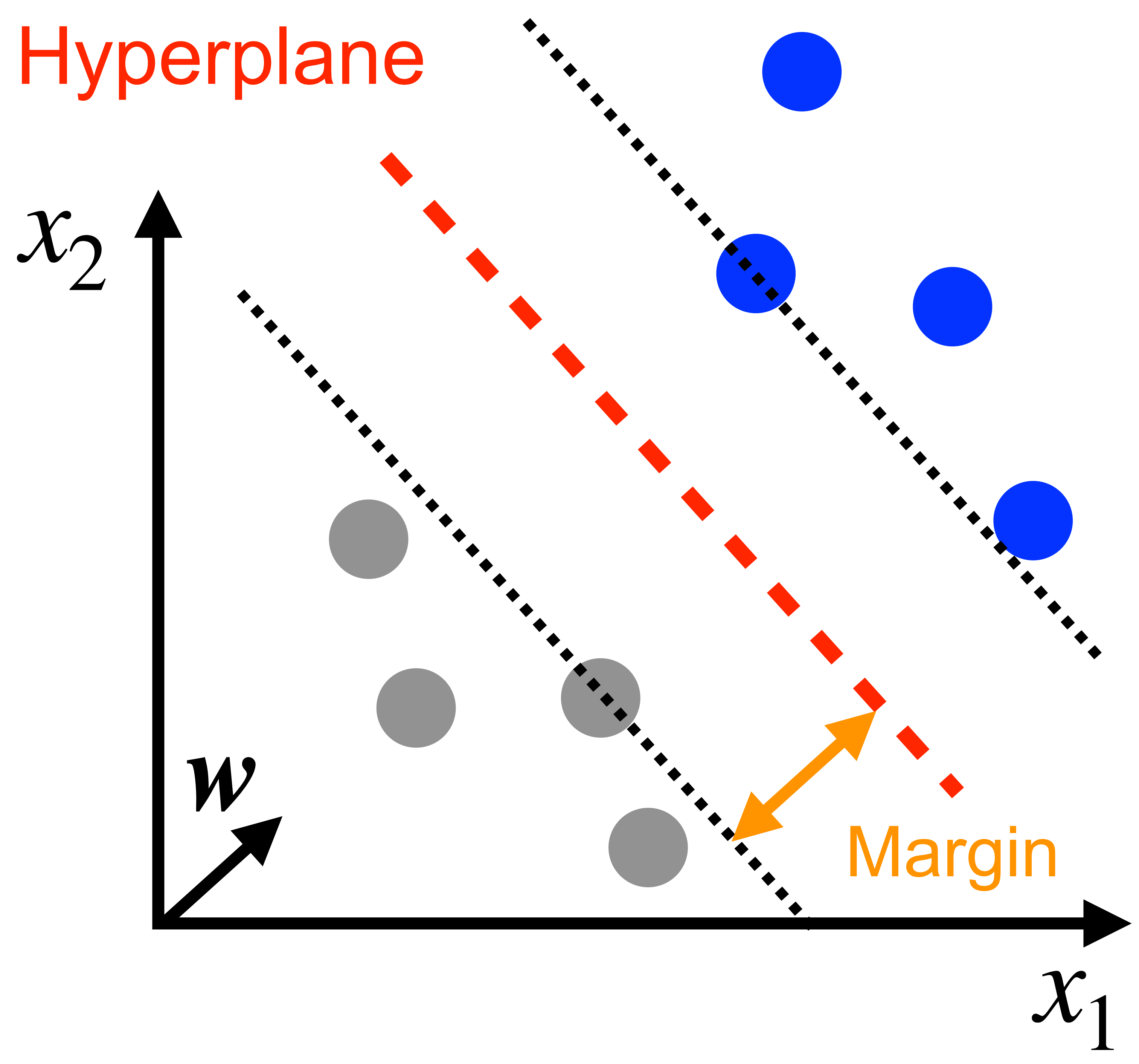
ML Classification in Feature Space

Consider a 2-class classification problem:

$$\{(\boldsymbol{x}_i, y_i)\}(i = 1, \cdots, N)\ \ \boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \{+1, -1\}$$

Define ***hyperplane*** as the points $\{\boldsymbol{x} \,|\, \boldsymbol{x} \in \mathbb{R}^d, \boldsymbol{w} \cdot \boldsymbol{x} + b = 0\}$

➡ Separate the feature space linearly into two data regions with different labels by a hyperplane



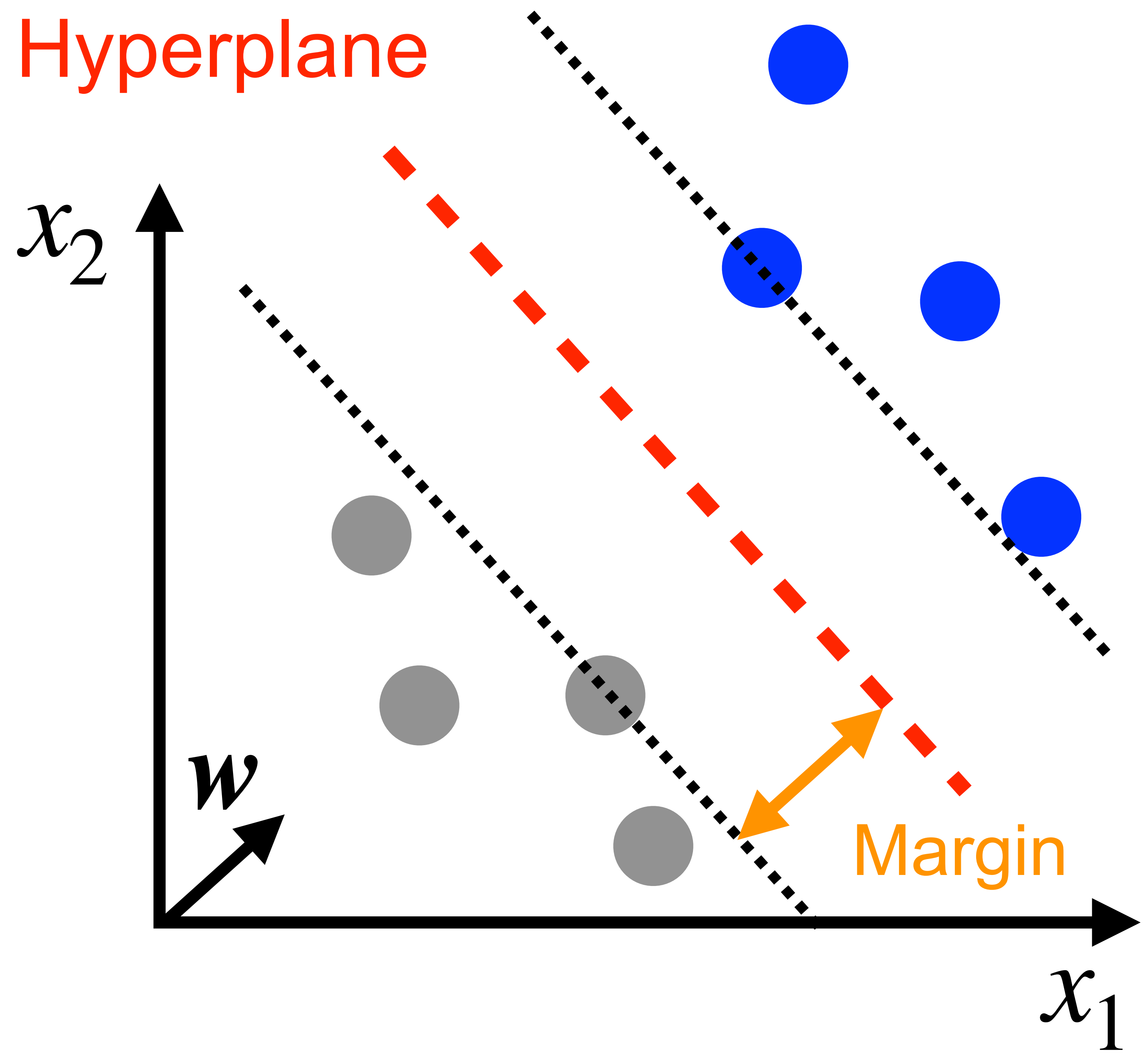
ML Classification in Feature Space

Consider a 2-class classification problem:

$$\{(\boldsymbol{x}_i, y_i)\}(i = 1,\cdots,N) \;\; \boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \{+1, -1\}$$

Define ***hyperplane*** as the points $\{\boldsymbol{x} \,|\, \boldsymbol{x} \in \mathbb{R}^d, \boldsymbol{w} \cdot \boldsymbol{x} + b = 0\}$

➡ Separate the feature space linearly into two data regions with different labels by a hyperplane



<u>Machine Learning task:</u>

- Training : Find a hyperplane $(\boldsymbol{w}, b)$ that satisfies $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \;\; \forall i$
- Inference : Sign of $\boldsymbol{w} \cdot \boldsymbol{x}_i' + b$ gives a label prediction for unseen new data $\boldsymbol{x}_i'$

Support Vector Machine

Find a hyperplane that maximizes the margin ($\propto 1/ \|\boldsymbol{w}\|$) between hyperplane and the nearest data point

Quantum SVM

Linear separation of data points in Hilbert space ➡ **Finding the minimum of** $\| \boldsymbol{w} \|^2$

➡ Further translated to a problem of finding parameters $\{\alpha_i\}$ that minimize

$$L(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

under the conditions $\sum_{i=1}^{N} \alpha_i y_i = 0, \alpha_i \geq 0$ for parameters $\{\alpha_i\}(i = 1, \cdots, N)$

More details in [QML workbook]

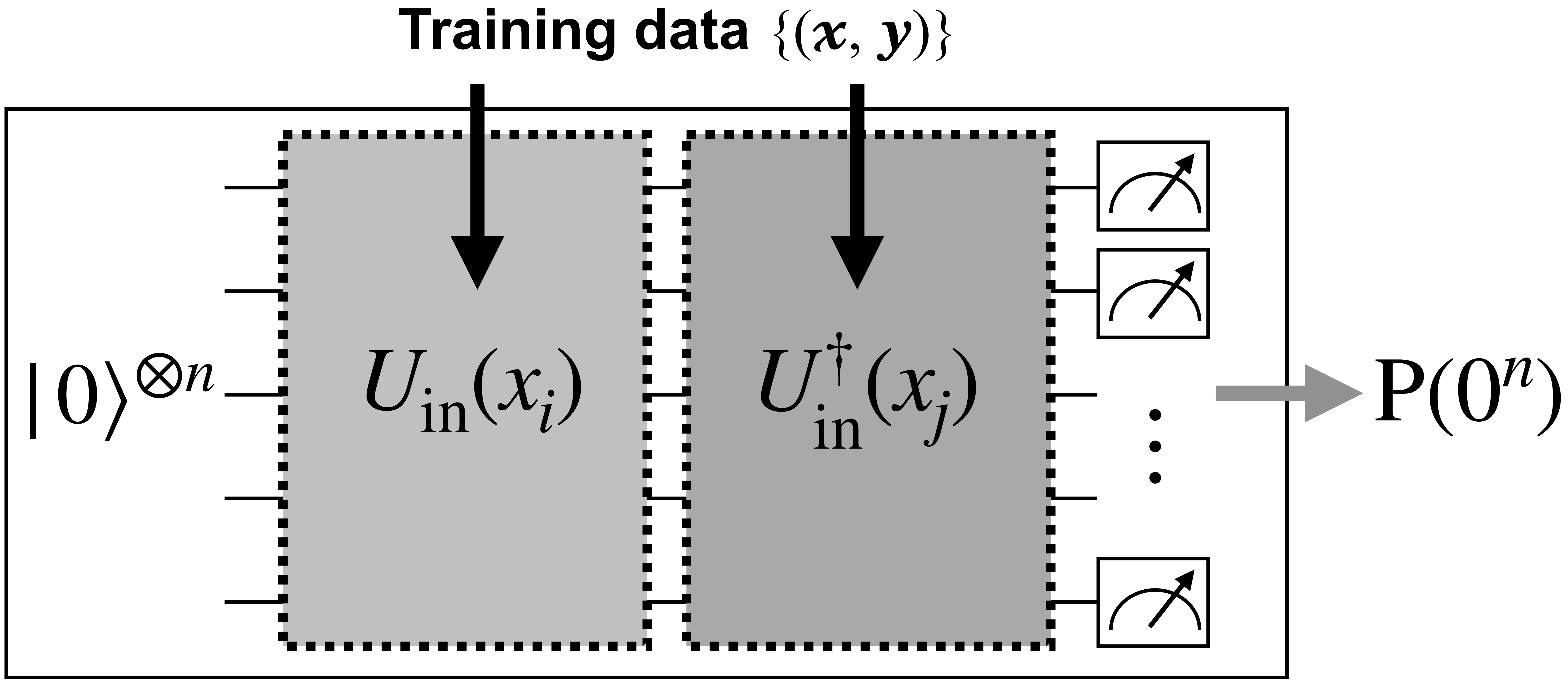
Calculate ***Quantum Kernel*** $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ using quantum computer

Sign of $\sum_{i=1}^{N} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}') + b^*$ with optimized parameters $\{\alpha_i^*, b^*\}$

➡ Label prediction for test data $\boldsymbol{x}'$

Quantum Circuit for QSVM

Quantum kernel can be obtained as inner product for a given feature map $U_{\text{in}}(x)$



Quantum Kernel $K(x_i, x_j) := |\langle\phi(x_j)|\phi(x_i)\rangle|^2 = |\langle 0^{\otimes n}|U^{\dagger}_{\text{in}}(x_j)|U_{\text{in}}(x_i)|0^{\otimes n}\rangle|^2$

➡ Probability of measuring 0 in all n-qubits with the initial state $|0\rangle^{\otimes n}$

Hands-on Exercise (III)

- Quantum Machine Learning :
  - Event classification using quantum neural network model
  - Event classification using quantum kernel method