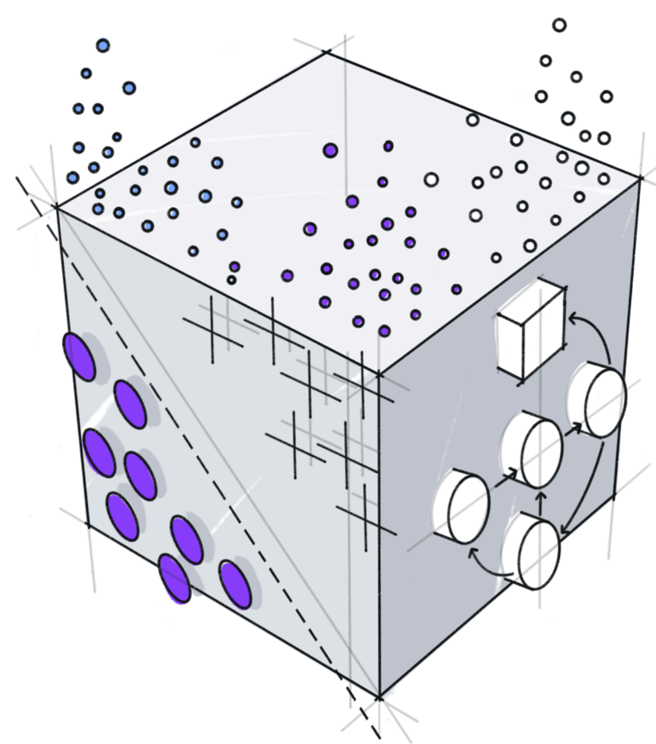
Introduction to (Qiskit) Quantum Machine Learning

Ruihao Li

Qiskit Fall Fest 22 @ CQC

10/14/2022

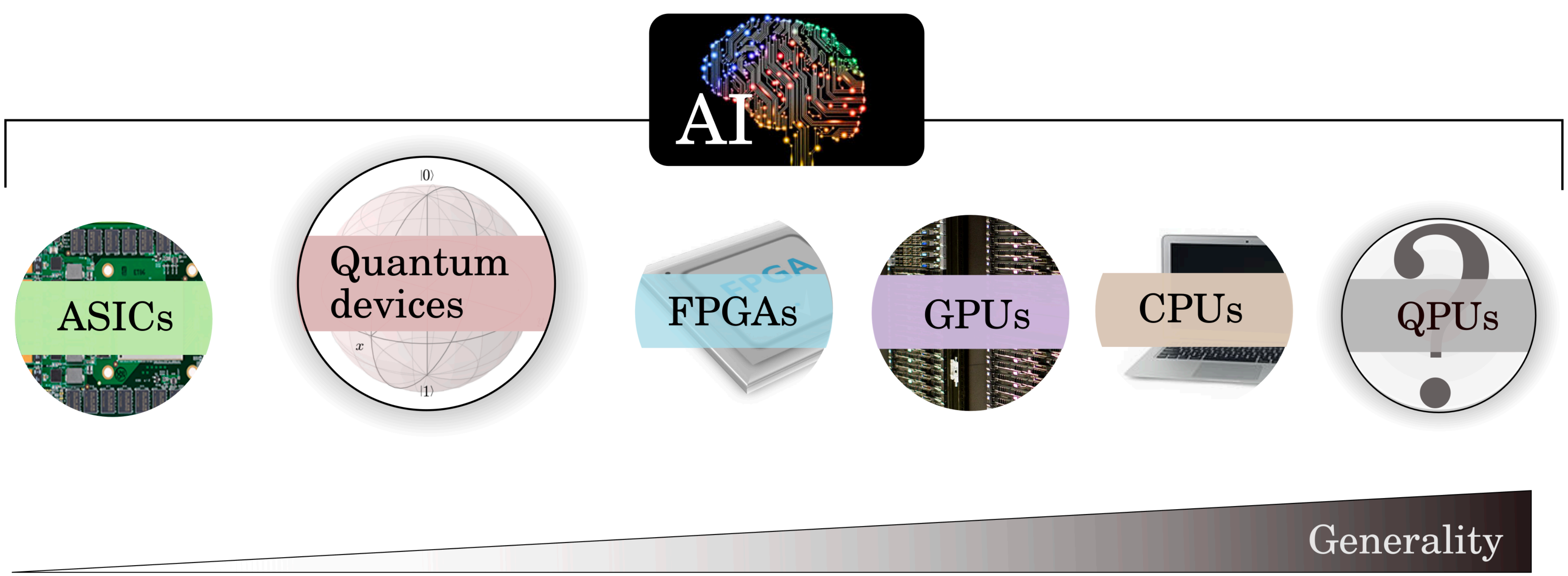


A bit about myself

- PhD student studying theoretical condensed matter physics
- Qiskit Advocate (https://qiskit.org/advocates/)
- Quantum Algorithms Research Intern @ Agnostiq in Summer 2022
- Currently working on quantum optimization & quantum error correction
- I occasionally write some blog posts about QC (https://ruihao-li.github.io/blog/)

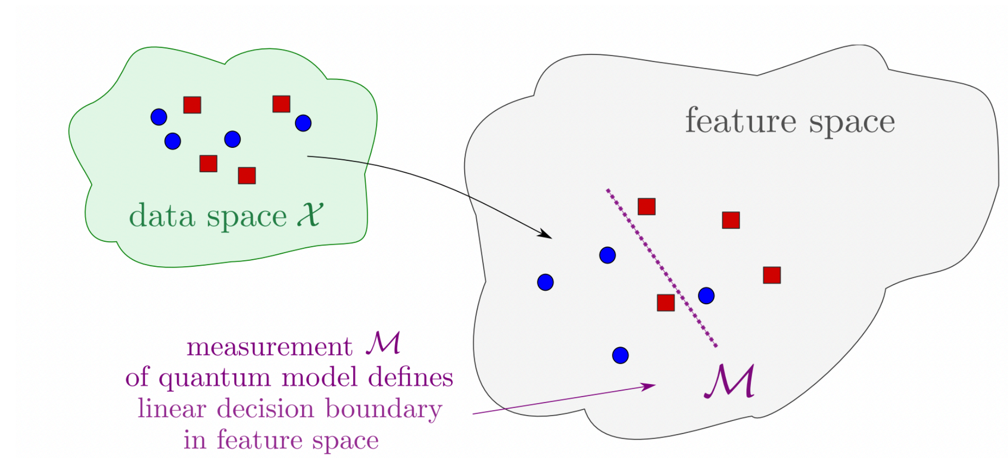
Why QML

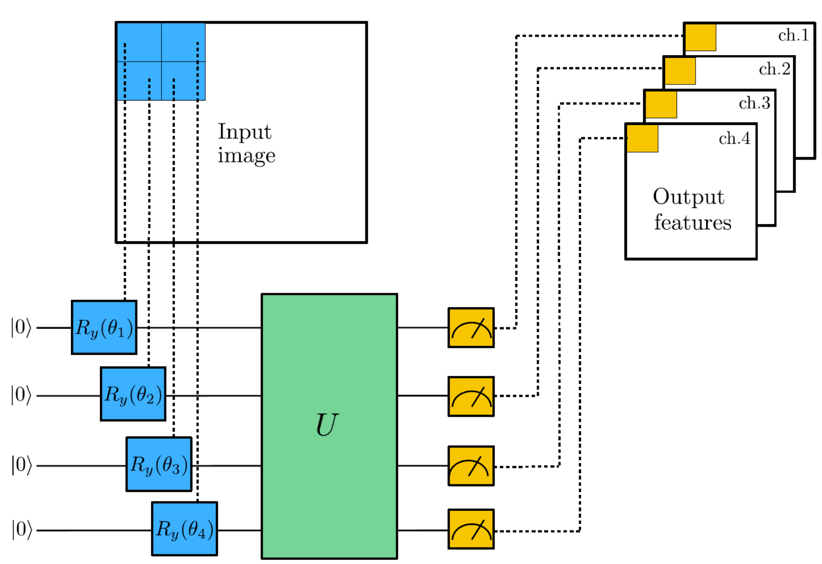
- Machine learning (ML) has proven to be super useful in everyday life
- ML today already uses different processors: CPUs, GPUs, TPUs, etc.
- Quantum computers (QPUs) could be used as special-purpose ML accelerators
- May enable training of previously intractable models by leveraging the power of quantum mechanics

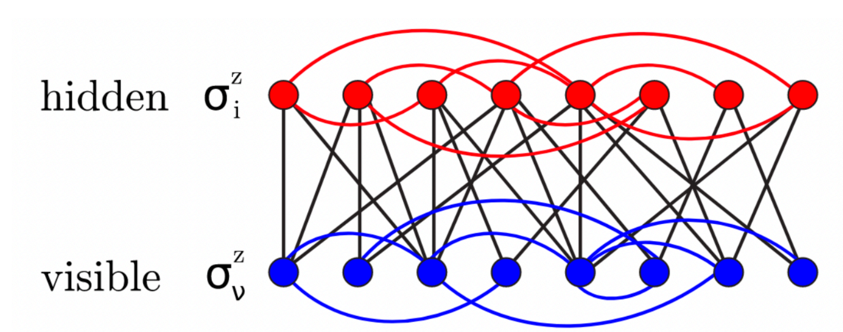


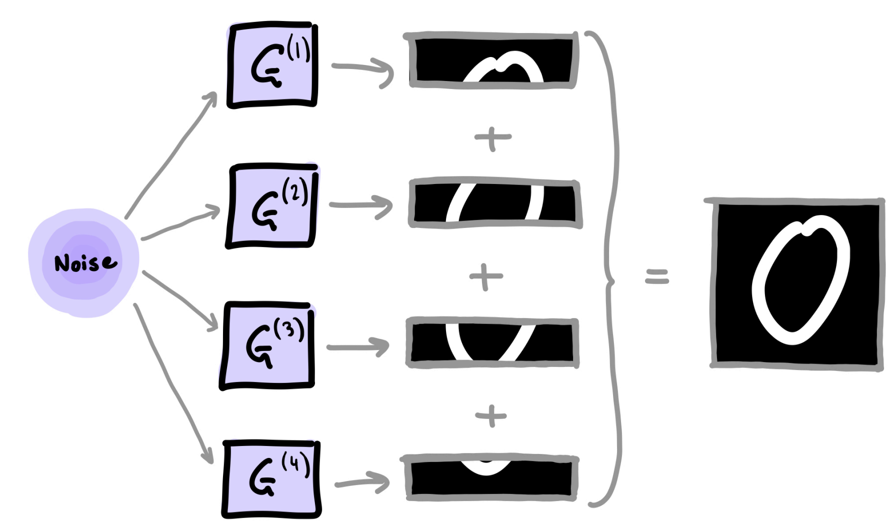
Why QML

- Quantum computing could also lead to new machine learning models
- Examples:
  - **Quantum kernel methods**
  - **Quantum neural networks (QNNs)**
  - Quanvolutional neural networks
  - Quantum Boltzmann machines
  - Quantum generative adversarial networks (qGANs)
  - ……

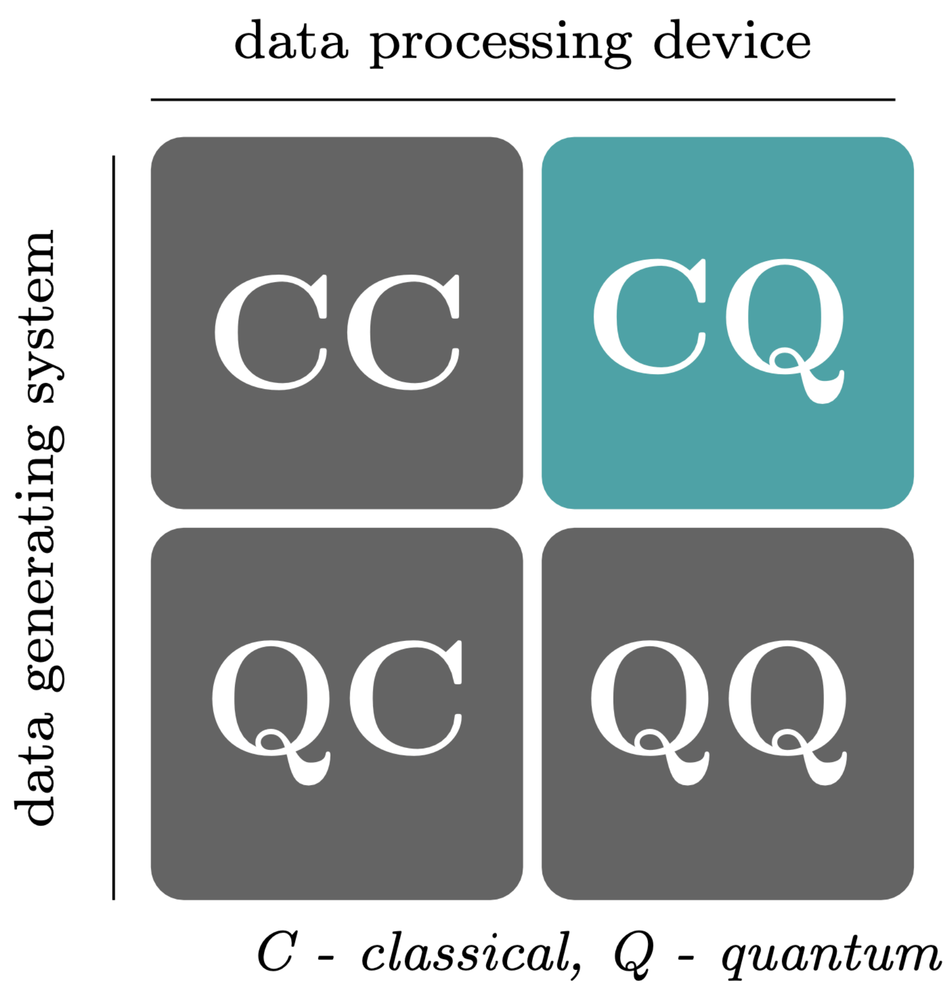








QML approaches



CC: quantum-inspired ML models, e.g., tensor networks

QC: classical ML to help understand quantum systems

CQ: typically a synonym for “QML”

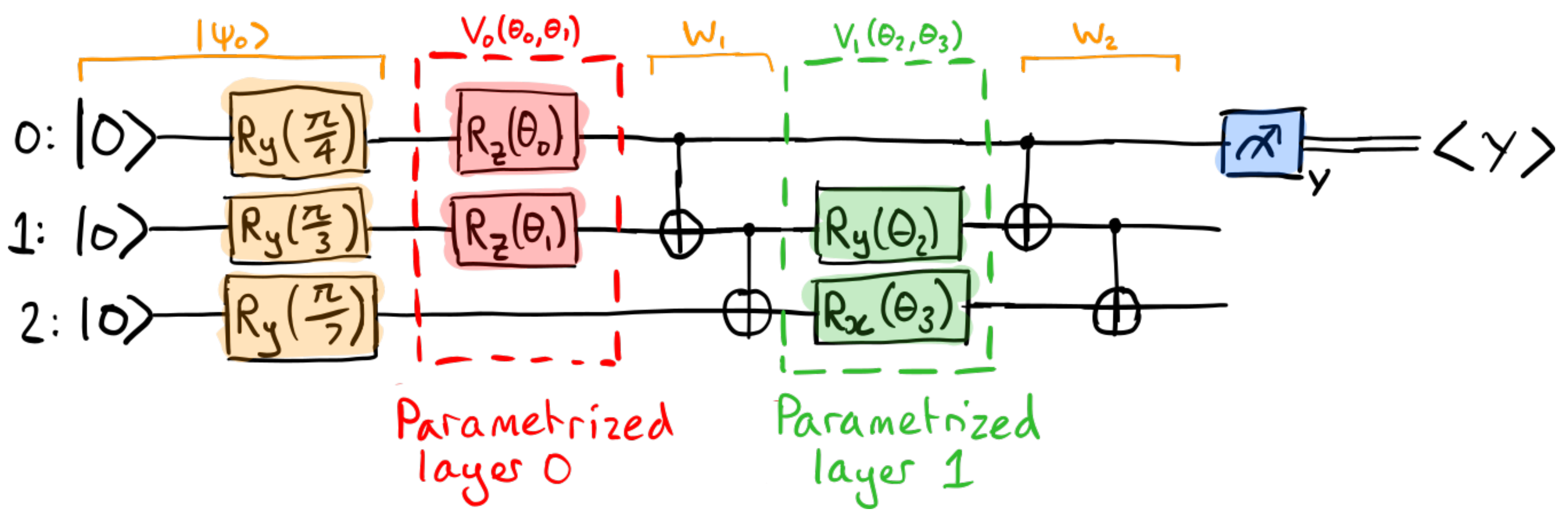
QQ: data derived from measuring a quantum system or data is made up of quantum states

Schuld & Petruccione, Springer, 2nd ed. (2021)

Key concepts of QML

Variational quantum circuits (VQCs)

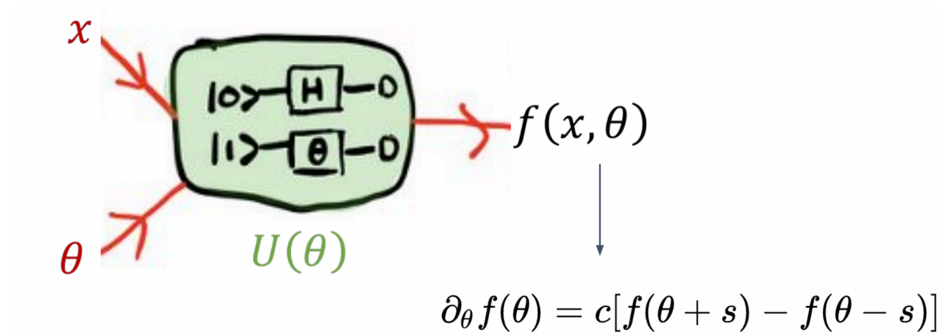
- Main QML method for noisy intermediate-scale quantum (NISQ) devices
- Structure similar to other modern quantum algorithms: e.g. *variational quantum eigensolver (VQE)*, *quantum approximate optimization algorithm (QAOA)*
- General steps:
    1. Preparation of a fixed initial state
    2. Encode classical data into a quantum state (encoding/embedding layer)
    3. Apply a parameterized model (processing layer)
    4. Perform measurements to extract observables



Key concepts of QML

Quantum circuit training

- How to train variational quantum circuits like we train neural networks?
- Most widely used method: **gradient descent** - SGD, Adam, natural gradient, etc; all of them require one important ingredient: the gradient of a circuit's output with respect to its input parameters
- **Backpropagation:** powers modern deep learning models
  - Pros: nice scaling properties w.r.t. the number of parameters
  - Cons: increased memory usage to store all intermediate values; $\implies$ can't be used directly on quantum computers

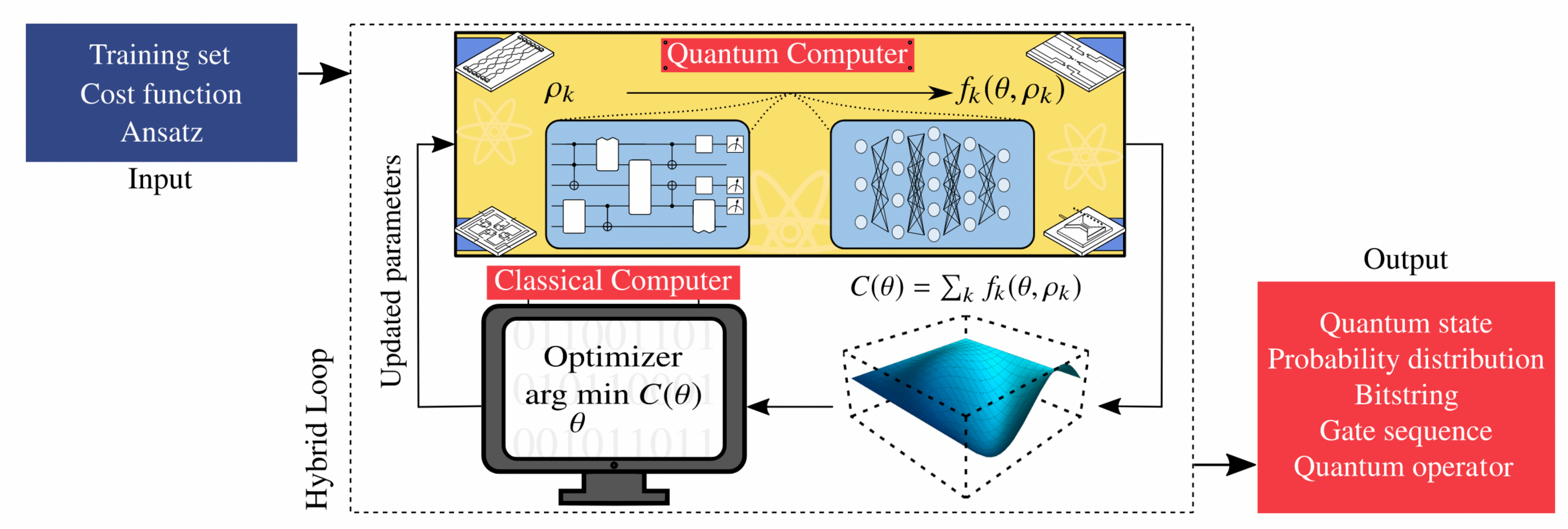


- **Parameter-shift rule:**
  - Pros: allows us to compute the function and its gradient on the same quantum device; gives *exact* gradients
  - Cons: scales roughly linearly with the number of parameters

Key concepts of QML

Hybrid computation

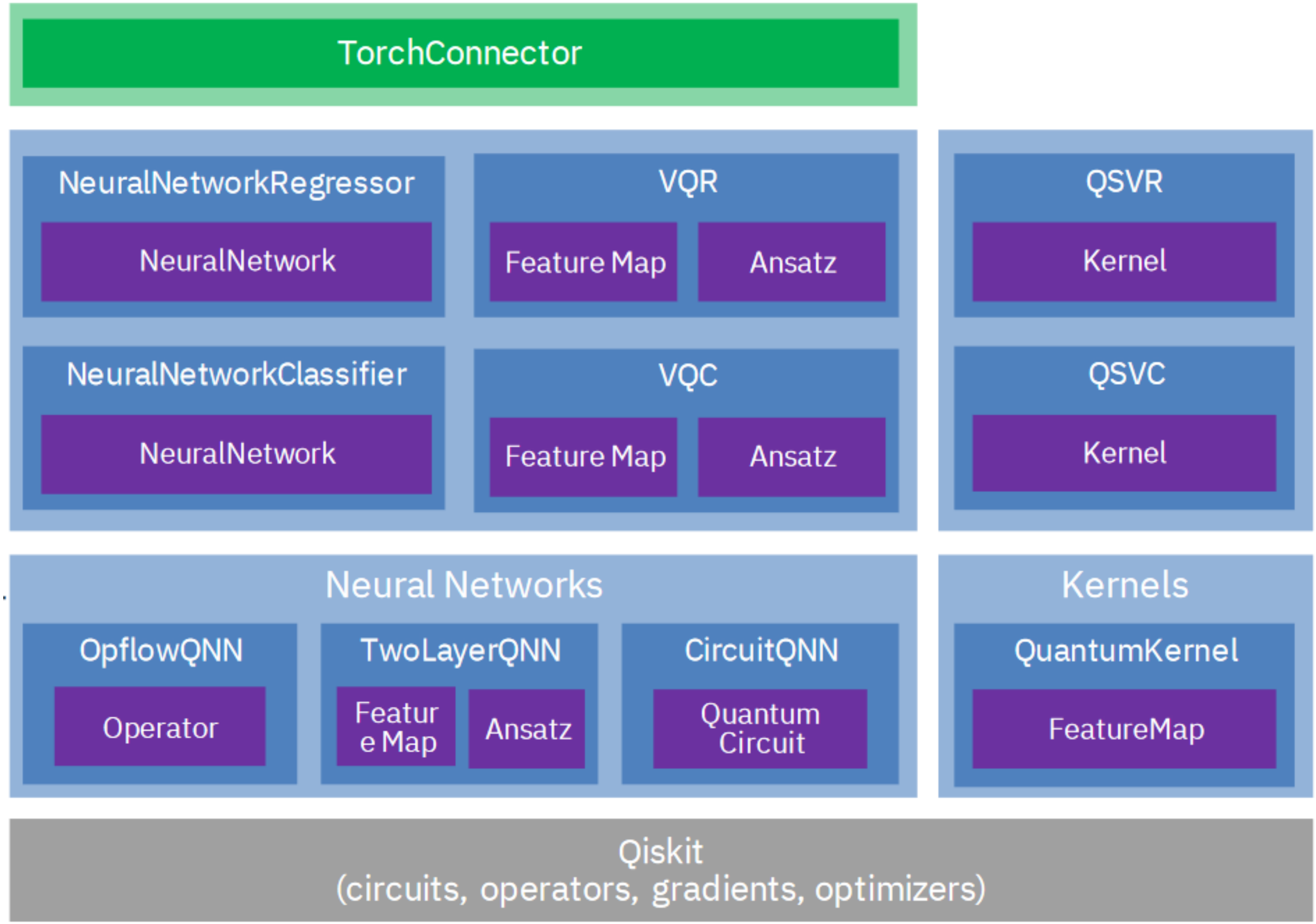
- Use quantum computers together with classical processors (CPUs, GPUs)
  - Classical optimization loop
  - Pre-/post-process quantum circuits outputs
  - Arbitrarily structured hybrid computations



Cerezo et al., Nat. Rev. Phys. 2021.

- Hybrid quantum-classical neural networks (we will see an example of this)

Qiskit Machine Learning



Example 1: Kernel method

Support vector machine (SVM)

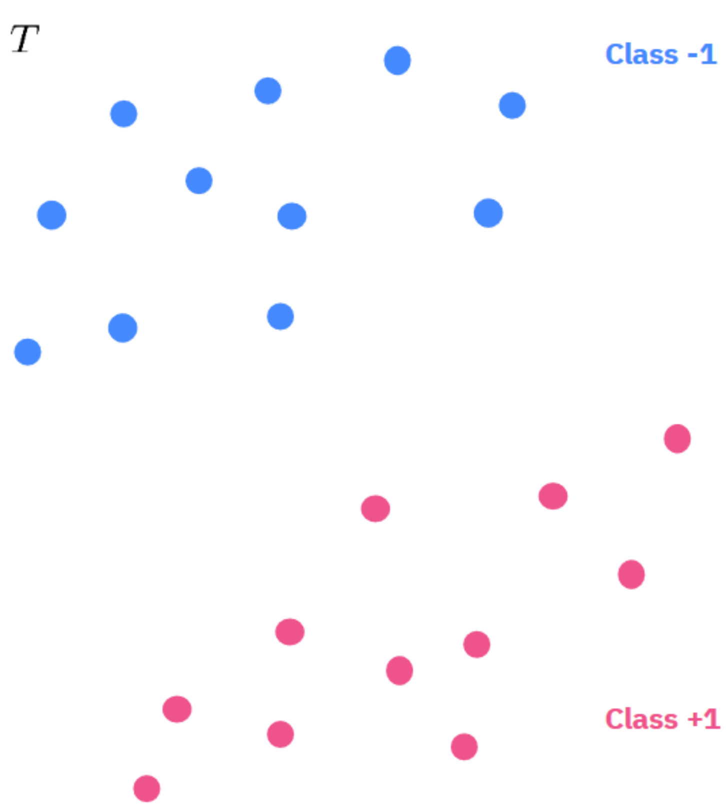
- Linear decision function:

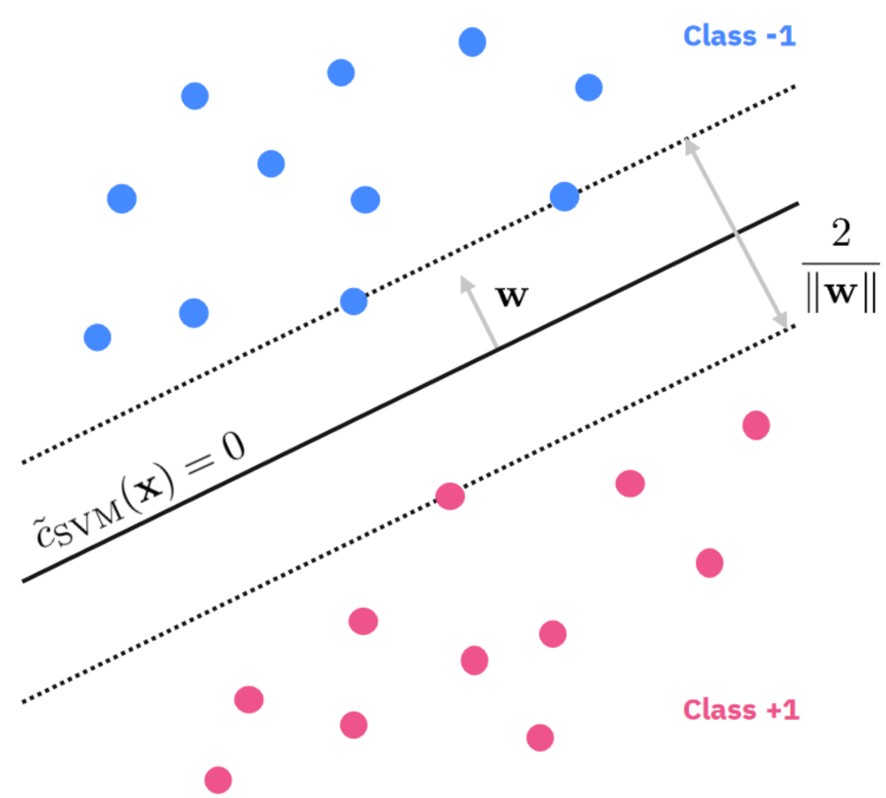
$$\tilde{c}_{SVM}(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^T\mathbf{x} - b)$$

- Objective: maximize margin

$$\min_{\mathbf{w}\in\mathbb{R}^s,\ b\in\mathbb{R}} \|\mathbf{w}\|$$

under constraint: $y_i \cdot (\mathbf{w}^T\mathbf{x}_i - b) \geq 1,\ \forall i\,.$





Example 1: Kernel method

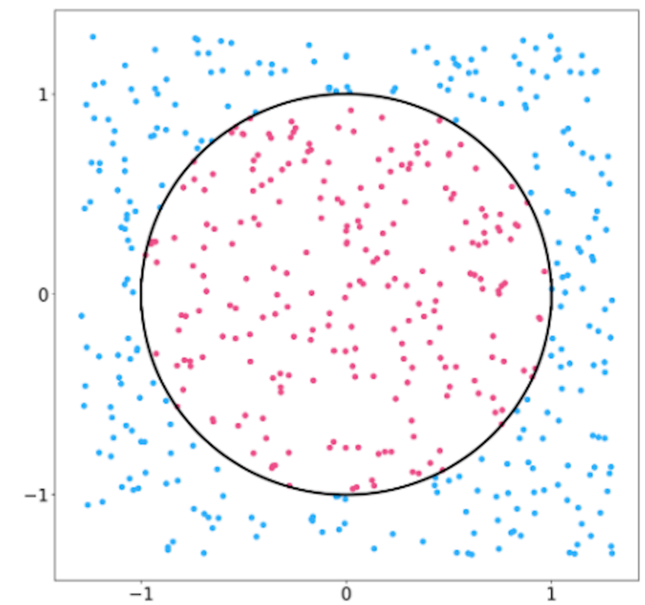
Kernelized SVM

- Vanilla SVM works only for linearly separable data
- Introduce a nonlinear feature transformation (i.e., **feature map**):

$$\phi : \mathbb{R}^s \to \mathscr{V}$$

$$\tilde{c}_{SVM} = \mathrm{sign}(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle_\mathcal{V} - b)$$

  s.t. data becomes linearly separable in feature space.



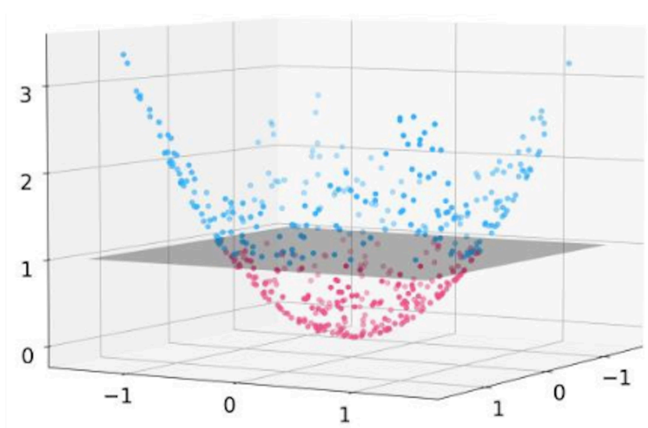
original space

- **Kernel trick** is to rewrite the SVM problem to only explicitly depend on the kernels

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_\mathcal{V} \ ,$$

  not on the feature vectors $\phi(\mathbf{x})$.

- Example: feature map $\phi(\mathbf{x}) = (x_1, x_2, x_1^2 + x_2^2) \in \mathbb{R}^3, \quad \mathbf{x} \in \mathbb{R}^2$.



feature space

Example 1: Kernel method

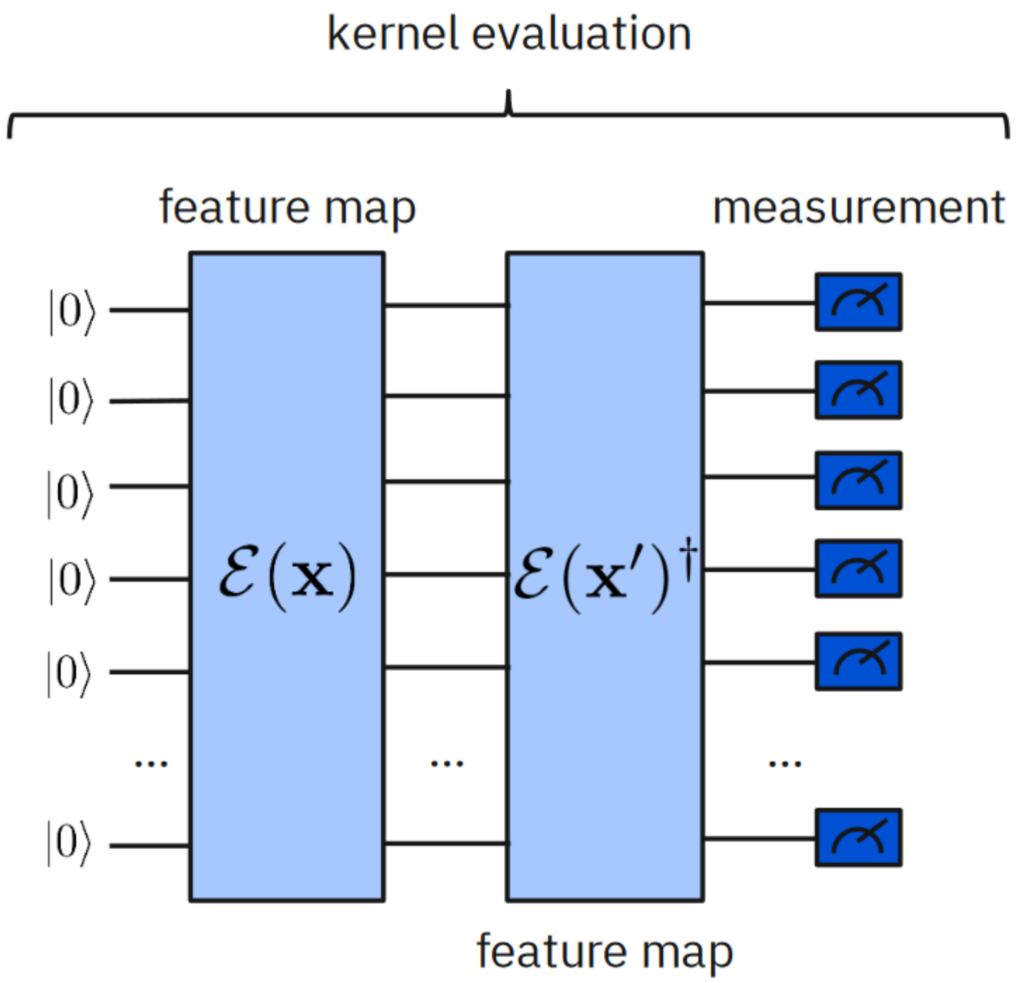
Quantum SVM

- Feature map is defined as a quantum circuit $\mathscr{E}(\mathbf{x})$:

$$\begin{aligned} \mathscr{E} : \mathbb{R}^s &\rightarrow \mathcal{S}(2^q) \\ \mathbf{x} &\mapsto |\psi(\mathbf{x})\rangle\langle\psi(\mathbf{x})| \end{aligned}$$

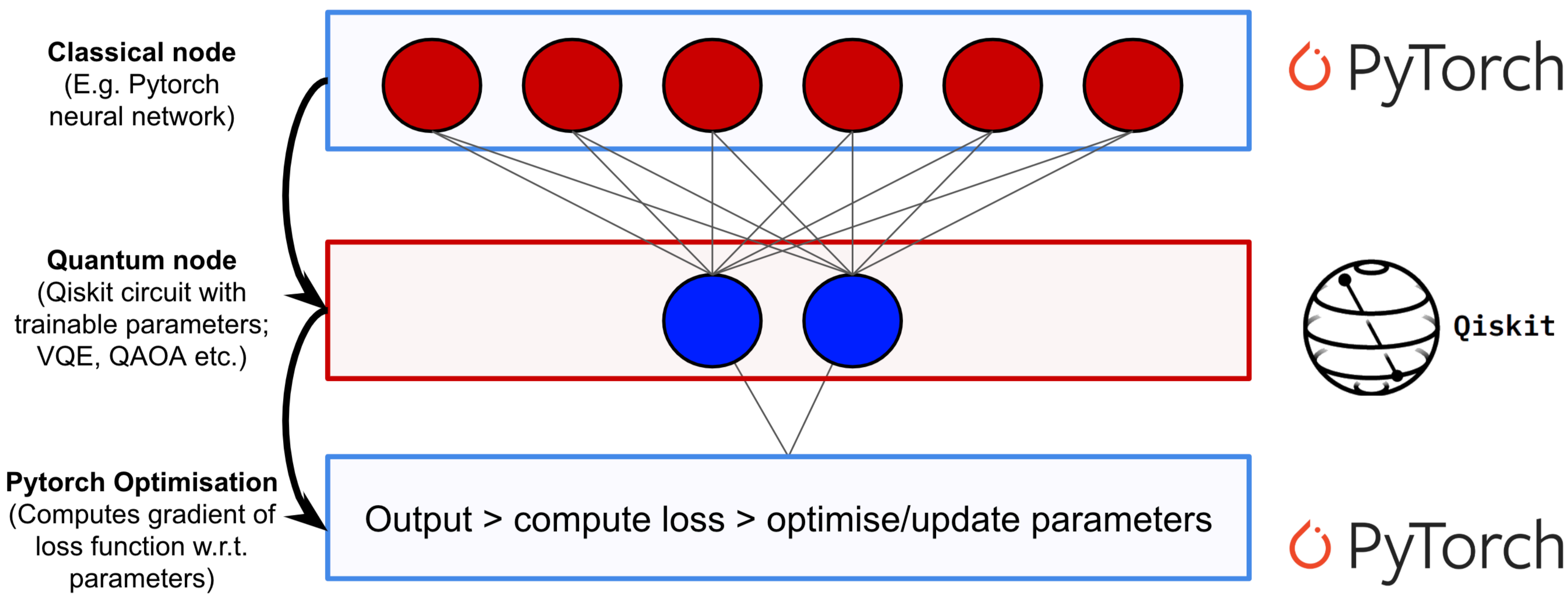
- Quantum kernel as a Hilbert-Schmidt inner product:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \mathrm{tr}[\,|\psi(\mathbf{x}')\rangle\langle\psi(\mathbf{x}')|\,|\psi(\mathbf{x})\rangle\langle\psi(\mathbf{x})|\,] \\ &= |\langle\psi(\mathbf{x}')|\psi(\mathbf{x})\rangle|^2 \\ &= |\langle 0|\mathscr{E}^\dagger(\mathbf{x}')\mathscr{E}(\mathbf{x})|0\rangle|^2 . \end{aligned}$$



Example 2: Hybrid NNs

- Based on: https://qiskit.org/textbook/ch-machine-learning/machine-learning-qiskit-pytorch.html
- Classical neural network with a quantum component



Example 2: Hybrid NNs