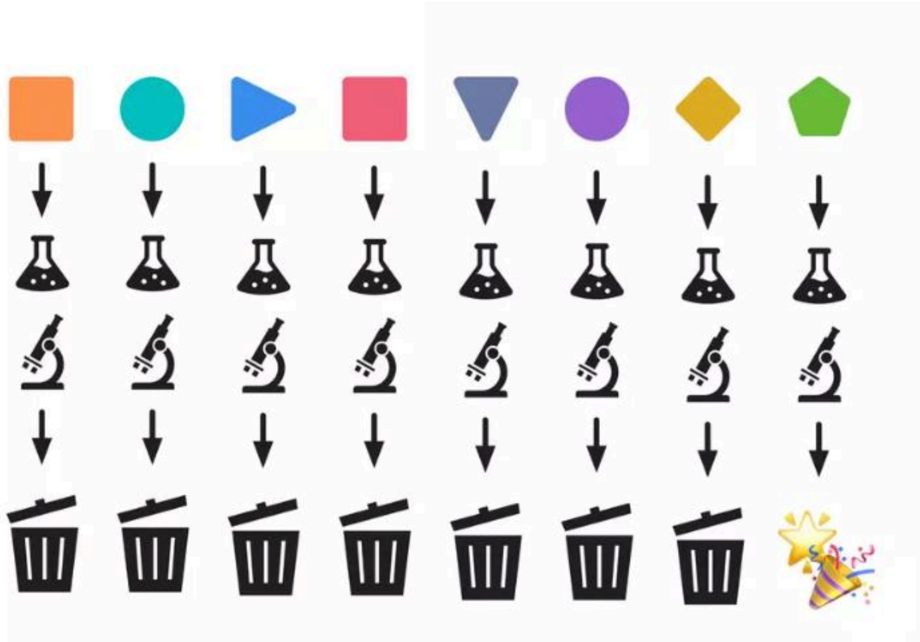
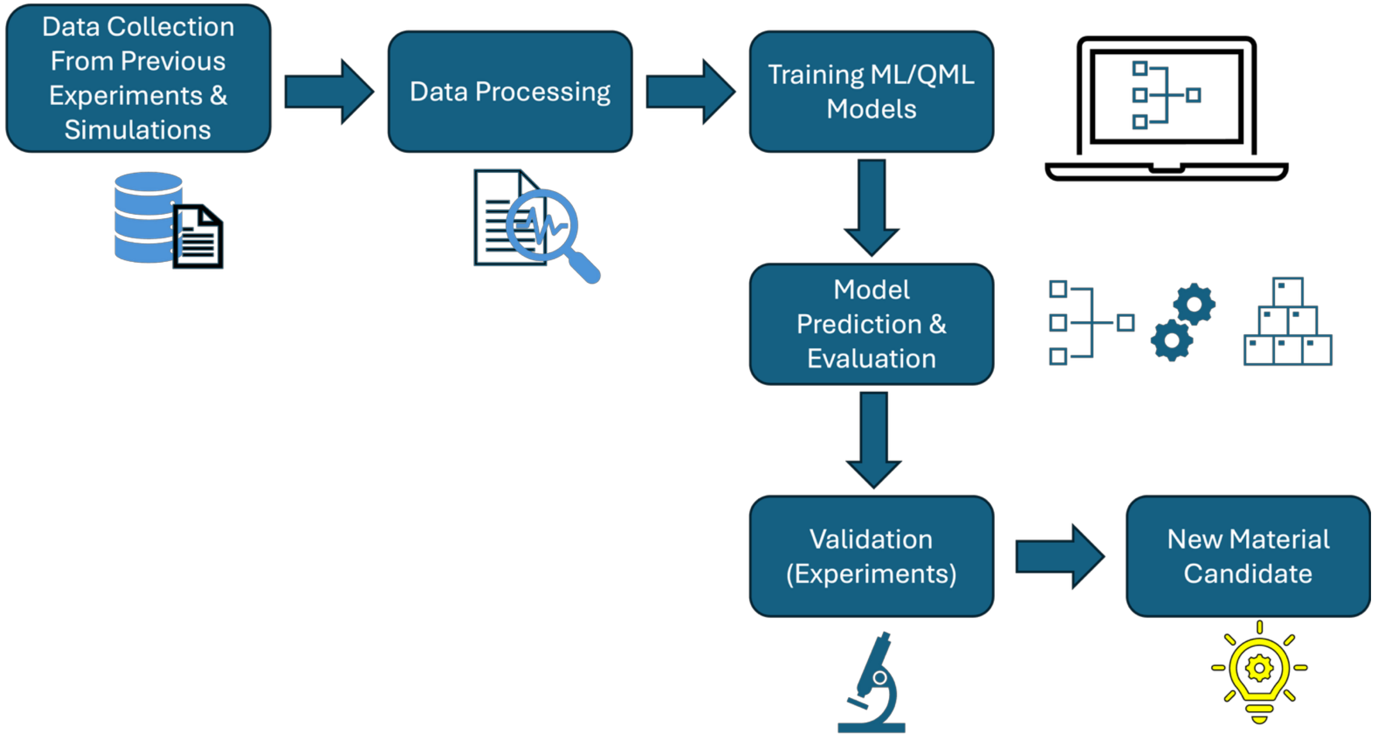
Quantum Machine Learning Applications in Material Science

Arizona State University

Data-driven discovery workflow



Traditional Design Approach

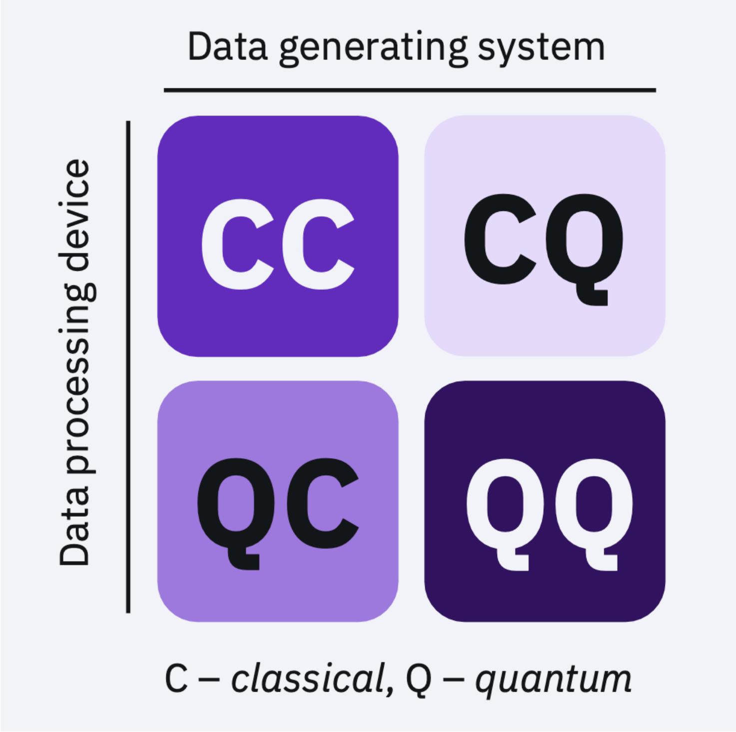


Data-driven Approach

Common Machine Learning tasks in Material Science

- Property Prediction
  - Bulk Modulus
  - Band - gap energy
  - Thermo-electric properties (Thermal and Electrical Conductivities)
- Classification Task
  - Any screening tasks to identify the category of a material
- Generative Tasks
  - Create compositions, structures of a certain type of materials based on desirable properties

A General ML Workflow



Training Data: X_{Train}

Model: $f(X_{train} : \theta) = \hat{y}$

Cost Function: $C(f(X_{train} : \theta), y)$

Compute gradient: $\dfrac{\partial C(f(X_{train} : \theta), y)}{\partial \theta}$

Update: θ

Arizona State University

Variational Circuit as classifier

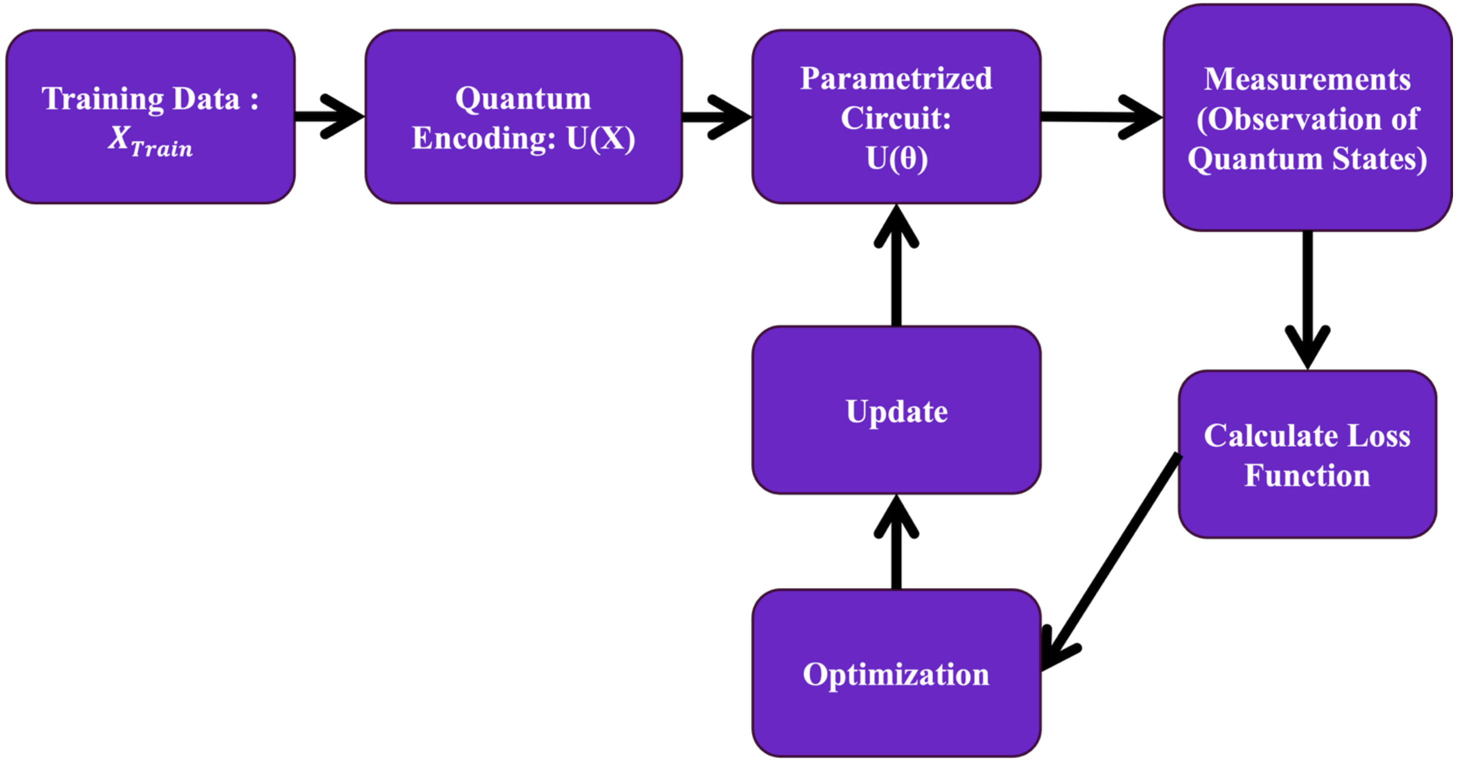
Objective: Train a quantum circuit on labeled samples in order to predict labels for new data

Step 1: Encode the classical data into a quantum state

Step 2: Apply a parameterized model

Step 3: Measure the circuit to extract labels

Step 4: Use optimization techniques to update model parameters



Step 1: Encoding Data

Types of encoding:

- Basis Encoding
- Amplitude Encoding
- Angle Encoding
- Higher Order Encoding

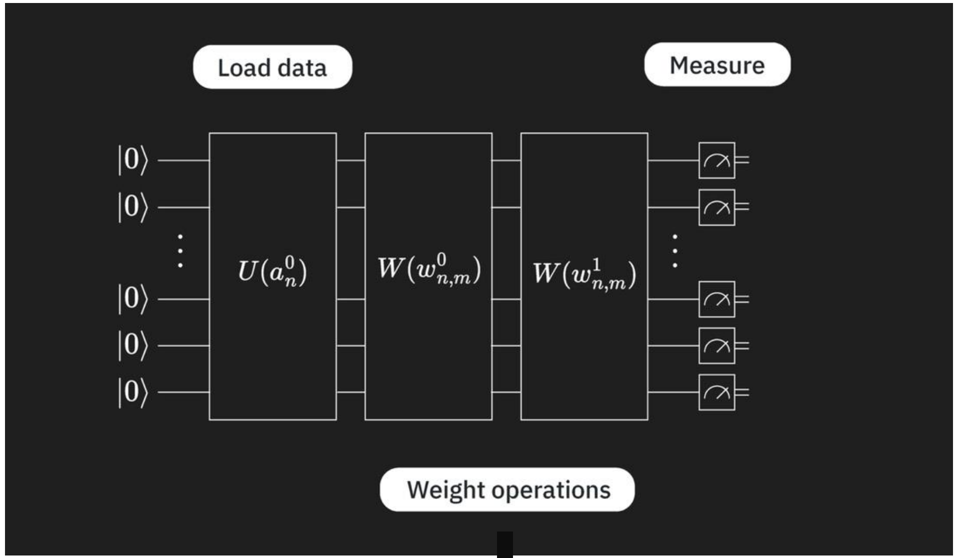
Qiskit also provides feature maps, which combine encoding with entanglement, to make more efficient use of qubits to represent a feature in a quantum state.

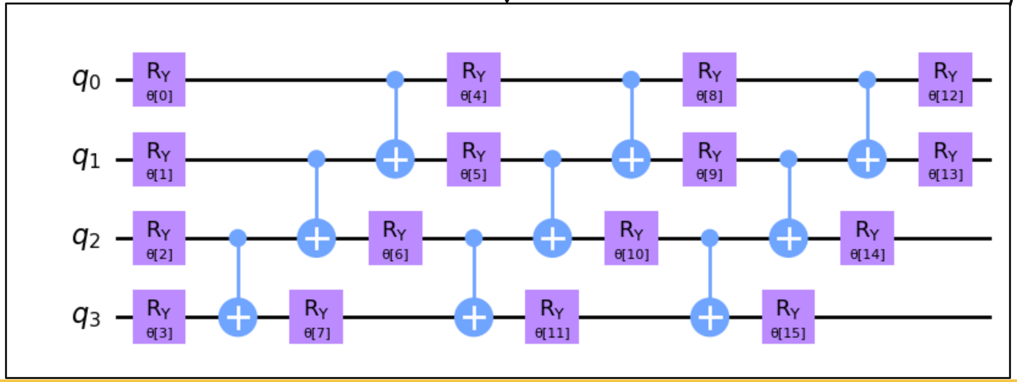
List of built-in feature maps provided by Qiskit:

- Z feature map
- ZZ feature map
- Pauli feature map

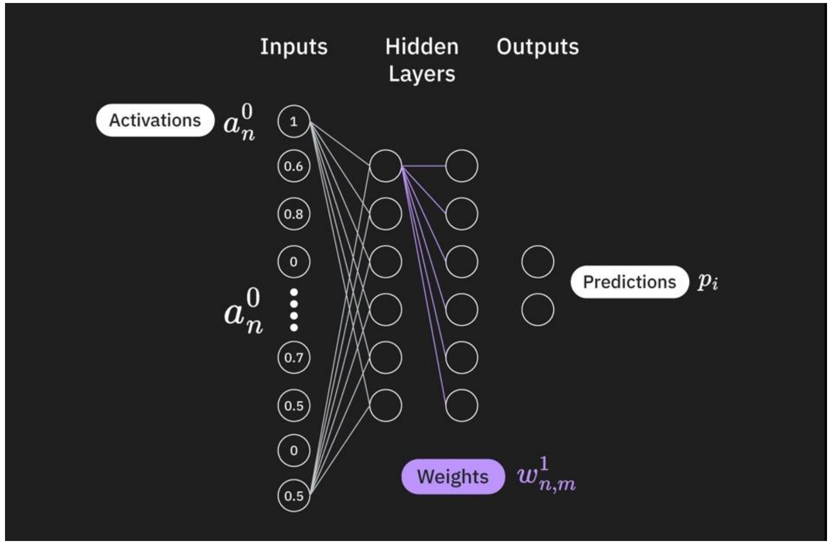
Details about Data Encoding in QC

Step 2: Applying variational circuit/ Ansatz





The picture shown is the variational circuit used in a QML classifier circuit, the encoded data is passed through Y rotation, and then entangled (partial/ full), and repeated multiple times/ layers (mimics neural network layers for classical counterpart)



Arizona State University

Step 3: Taking Measurements (Samplers and Estimators)

In QML, the final quantum state is measured using one of two primitives:

•**Sampler**: Measures the probabilities of bitstring outcomes → used for classification.

•**Estimator**: Measures expectation value of an observable (like Z) → used for regression.

These primitives let us extract classical information from a quantum circuit efficiently. They run on both simulators and hardware.

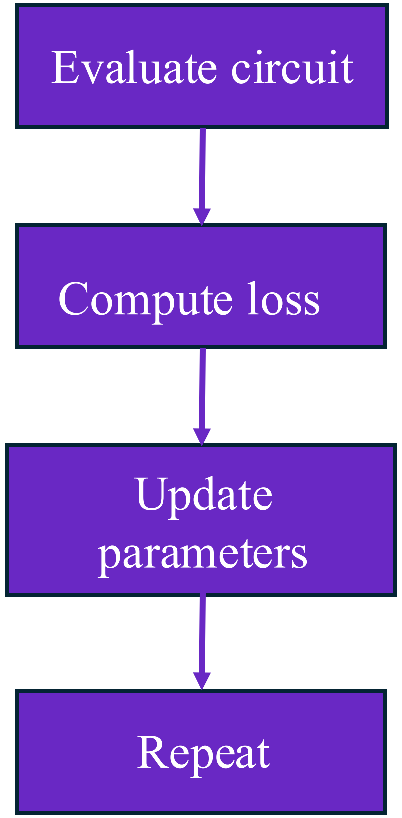
Step 4: Optimization

In QML, we use: (i) Classical Optimizers (ii) Quantum optimizers (Quantum Annealers, QAOA, etc.)

For this presentation we discuss and use classical optimizers with Quantum circuits.
Some classical optimizers: **COBYLA (Gradient-free), L-BFGS-B (Gradient based), SPSA, etc.**

If gradients are needed → uses **Parameter Shift Rule**

More about optimizers



Let's move to the Jupiter Notebook

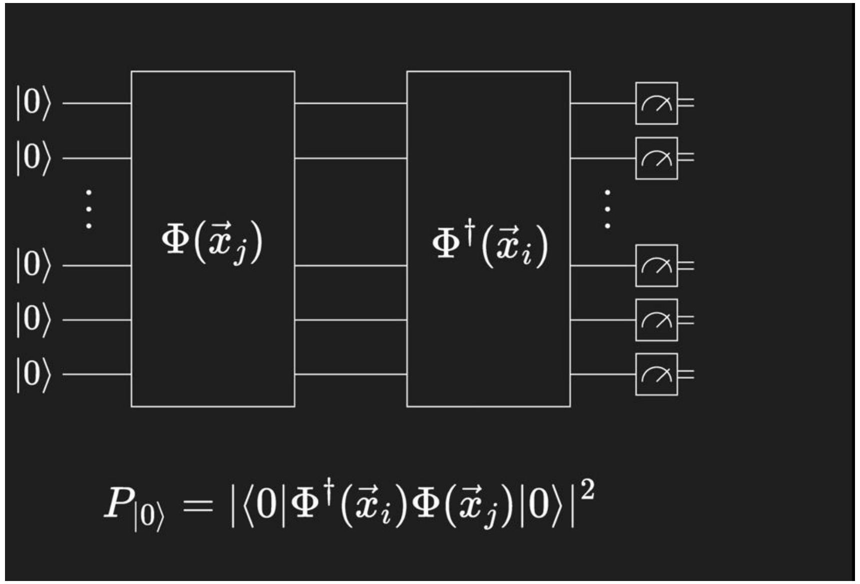
Things to remember while using Real Quantum Hardware

- Experiment different feature maps and choose the best fit for your dataset (there is no size fits all option)
- Same goes for building the variational circuit. Most of the part in building variational circuit is heuristic.
- Selection of backend (tend to choose the least busy options)
- Use a pass manager to transpile both the circuit and the observables to the architecture of the chosen backend.

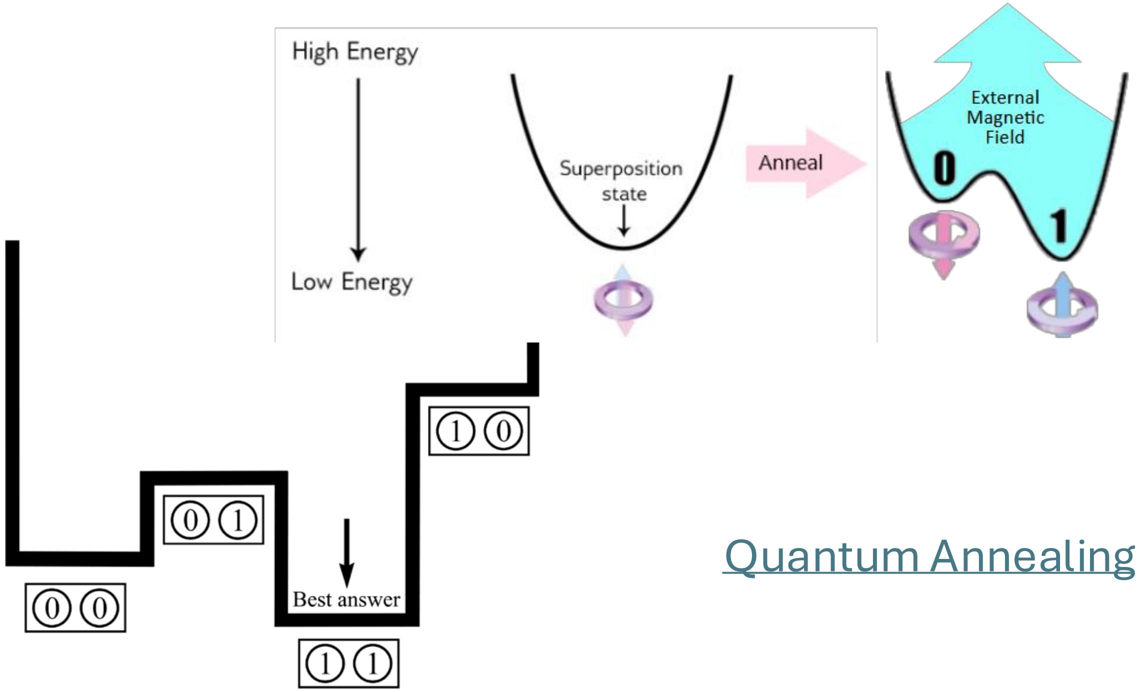
Generate a basic preset pass manager

Different QML Techniques

- Quantum Kernels technique: Mapping the data to a higher dimension, used for Clustering, and classification applications
- Quantum Optimization: Using Classical ML techniques with a quantum optimizer



Quantum Kernels



Quantum Annealing

Arizona State University
NSF

Acknowledgements

IBM Quantum:

- Darren Kwee
- Dr. Meltem Tolunay

My PhD advisor: Dr. Houlong Zhuang

Thank You!