Quantum Machine Learning

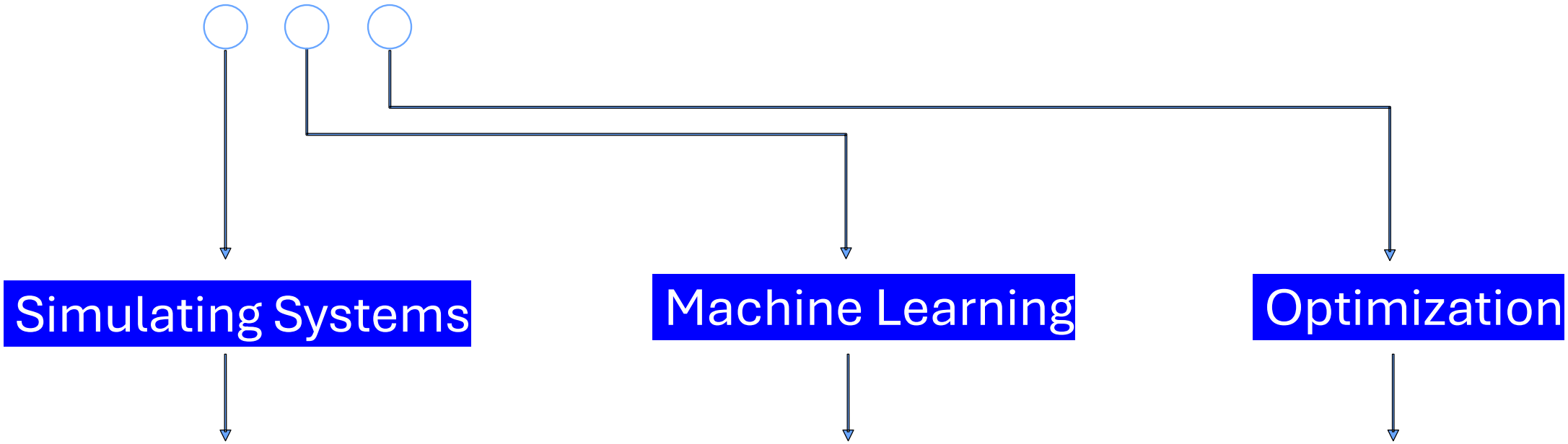
Chapter 8

Agenda

- Quantum Computing Application Areas
- Machine Learning, a Recap
- Quantum Machine Learning
- Data Encoding and Mapping
- Near-term methods: QSVM, QNN, QGAN
- Fault-tolerant methods
- Qiskit Machine Learning
- Summary

Quantum Application Areas

Quantum Applications span 3 Areas



Simulating Systems

- Quantum chemistry
- Material science
- High energy physics

Machine Learning

- Better model training
- Pattern recognition
- Fraud detection

Optimization

- Portfolio optimization
- Risk analysis
- Loans & credit scoring
- Monte Carlo-like applications

Quantum Applications span 3 Areas (Cont.)

Simulating Quantum Systems

- Improved battery materials
- Manufacturing defect identification
- Semiconductor materials
- Chemical property prediction
- Drug Discovery
- Protein Structure Predictions
- Disease Risk Predictions

Artificial Intelligence

- Accelerated Diagnosis
- Genomic Analysis
- Chemical product design
- Catalyst discovery
- Chemical process optimization
- High energy physics classification
- Transaction classification
- Product recommendation

- Fraud detection
- Risk analysis
- Options pricing
- Derivatives Pricing
- Investment Risk Analysis
- Portfolio Management
- Transaction Settlement
- Finance Offer Recommender
- Credit/Asset Scoring
- Airline Scheduling

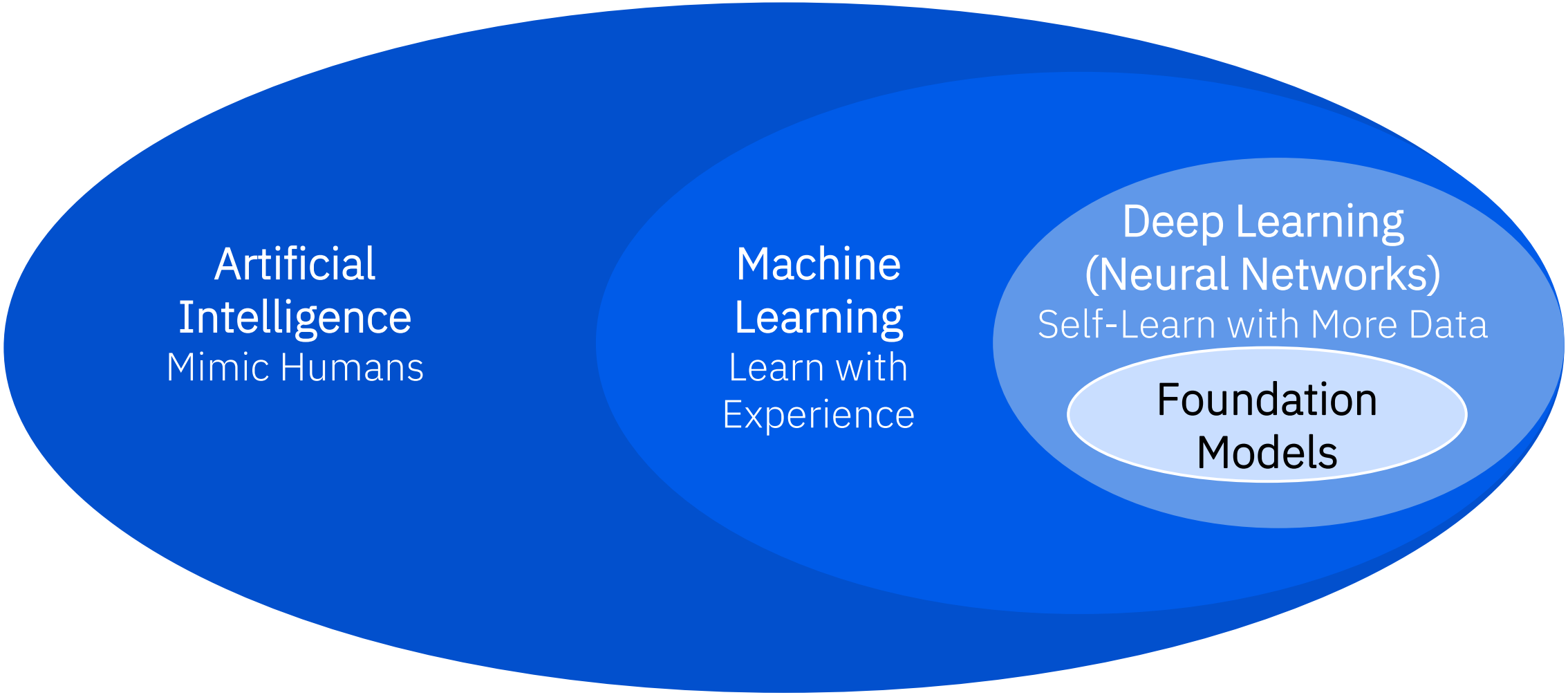
Optimization / Monte Carlo

- Irregular Operations
- Network Optimization
- Product Portfolio Optimization Process Planning
- Quality Control
- Vehicle Routing
- Raw materials shipping
- Refining Processes
- Seismic imaging
- Disruption Management

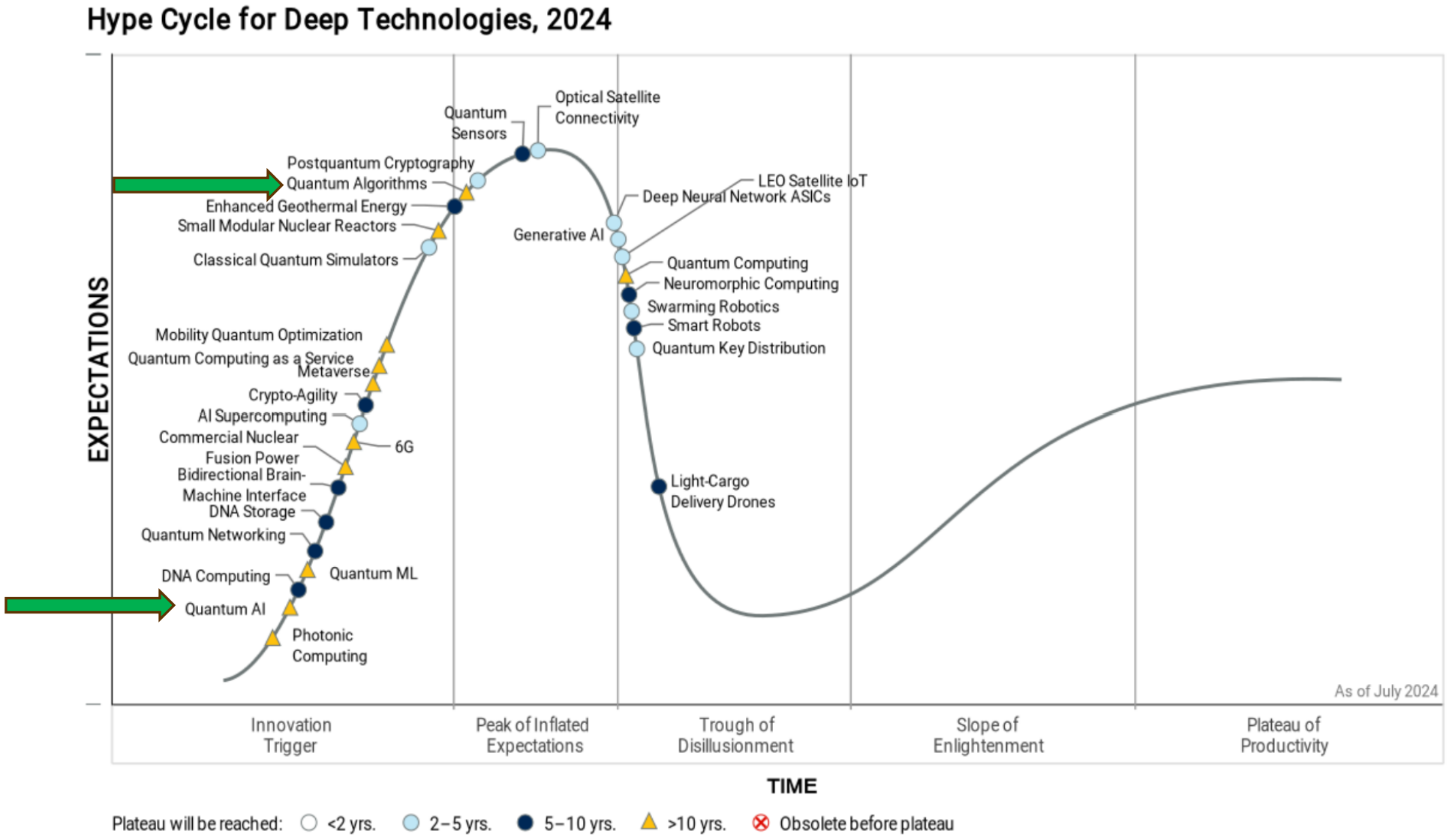
- Freight Forecasting
- Irregular Operations
- Fabrication Optimization
- Manufacturing Supply Chain
- Fluid Dynamics

and many more …

Artificial Intelligence



We will discuss HOT TRENDS today !



Machine Learning, a Recap

Artificial Intelligence (AI)

Human intelligence exhibited by machines

AI can be defined as a technique that enables machines to mimic cognitive functions associated with human minds – cognitive functions include all aspects of learning, reasoning, perceiving, and problem solving.

Machine Learning (ML)

Systems that learn from historical data

ML-based systems are trained on historical data to uncover patterns. Users provide inputs to the ML system, which then applies these inputs to the discovered patterns and generates corresponding outputs.

Deep Learning (DL)

ML technique that mimics human brain function

DL is a subset of ML, using **multiple layers of neural networks**, which are interconnected nodes, which work together to process information. DL is well suited to complex applications, like image and speech recognition.

Foundation Model

Generative AI systems

AI model built using a **specific kind of neural network architecture**, called a **transformer**, which is designed to generate sequences of related data elements (for example, like a sentence).

1950's 1980's 2010's 2020's

Machine learning (ML) types

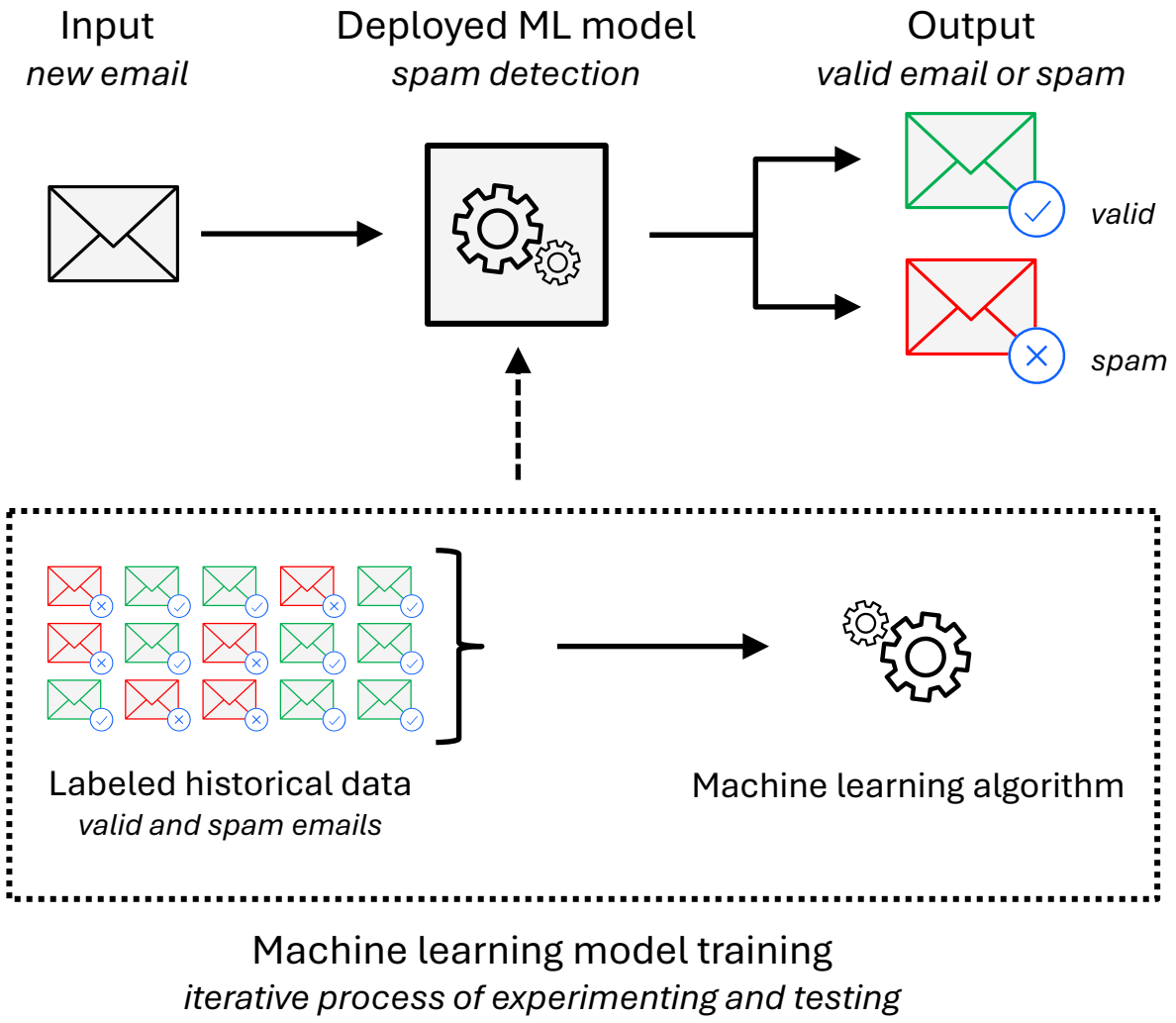
Classification models

Classification models assign labels to model inputs or assign them to specific categories.

Common use cases include:

- **Fraud detection**: predict whether a transaction is fraudulent based on patterns in the data
- **Sentiment analysis**: classify text as positive, negative, or neutral
- **Medical diagnosis**: assign a disease label to a patient's case, based on symptoms and medical history
- **Image recognition**: recognize objects or identify people based on visible features and characteristics

Example: Spam detection for email



Machine learning (ML) types

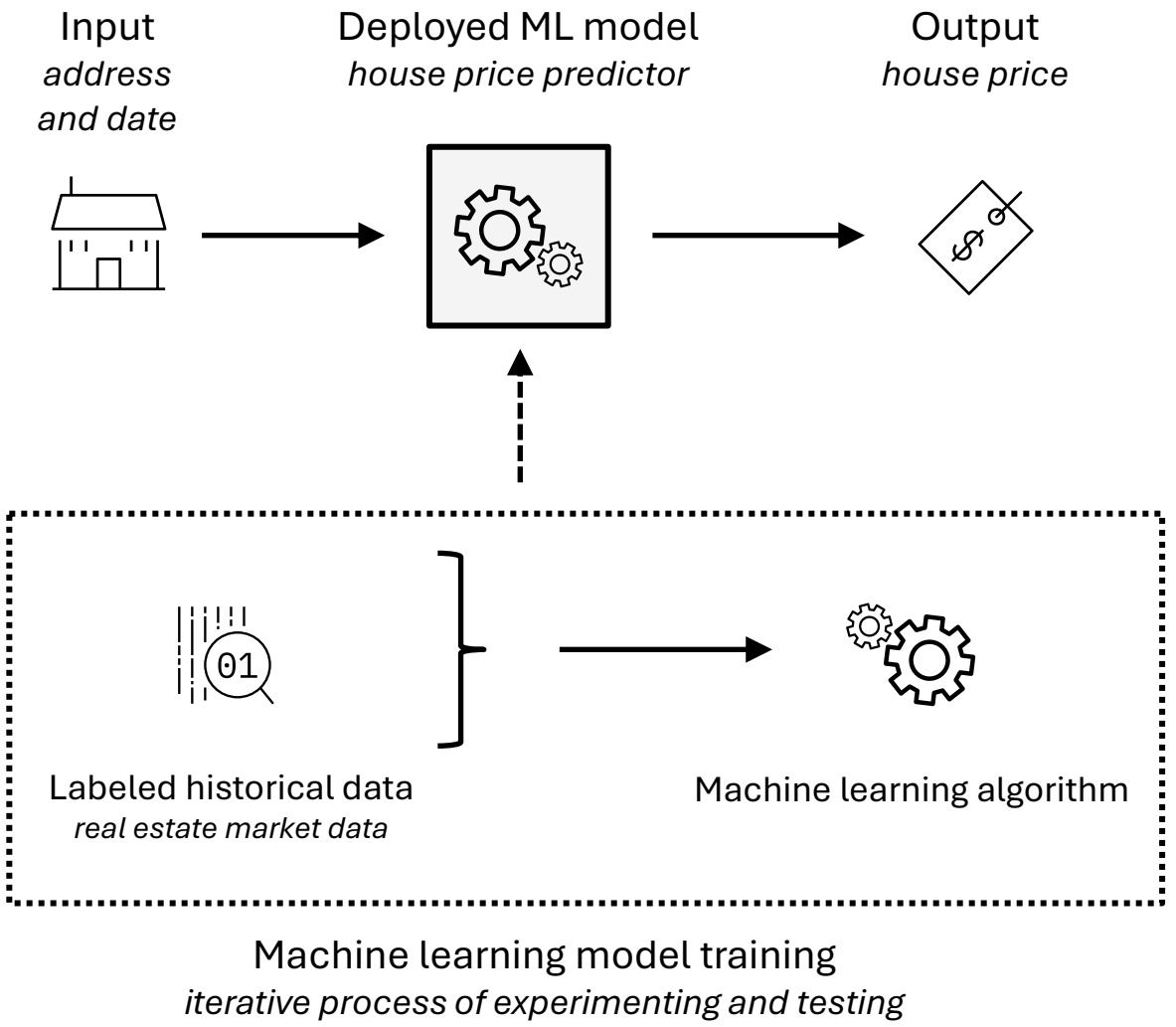
Regression models

Regression models make predictions based on the model input.

Common use cases include:

- **Stock market analysis**: securities price prediction based on historical data or news events
- **Sales**: forecasting based on historical data or market trends
- **Healthcare:** predict patient outcomes based on factors such as age, gender, medical history, or treatment plans
- **Customer behavior analysis**: predict future customer purchasing patterns based on demographic data, past purchase history, and advertising campaigns

Example: House price prediction



Machine learning (ML) types

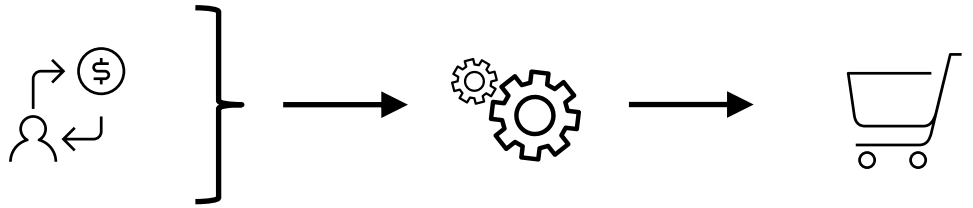
Clustering models

Clustering models identifies distinct groupings of individual data points that share common characteristics within a larger data set.

Common use cases include:

- **Customer segmentation**: group customers based on similar preference, behaviors, and demographics
- **Genetic analysis**: group genes with similar functions or processes
- **Social network analysis**: identify communities or groups within a social network
- **Market basket analysis**: identify items that are commonly purchased together

Example: Market basket analysis



Unlabeled historical data
transaction data

Clustering algorithm

Output
items commonly purchased together

Machine learning (ML) types

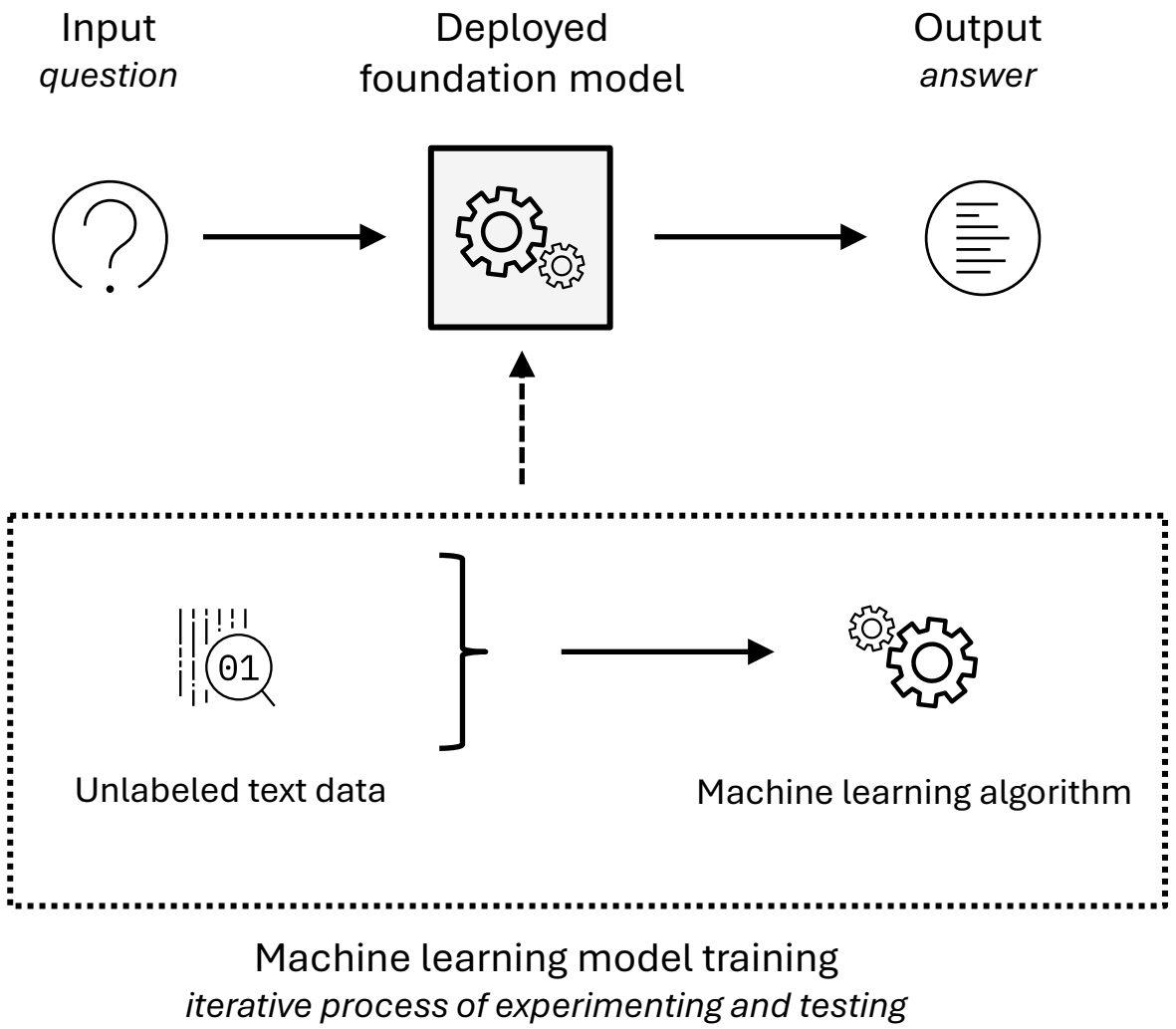
Generative models

Generative models **create new data** in response to the input request.

Common use cases include:

- **Text generation**: write new text in response to the request, which could be simply answering a question, summarizing text, or writing a lengthy essay
- **Code generation**: generate computer code based on a textual description of the proposed program
- **Image generation**: create images based on the textual request

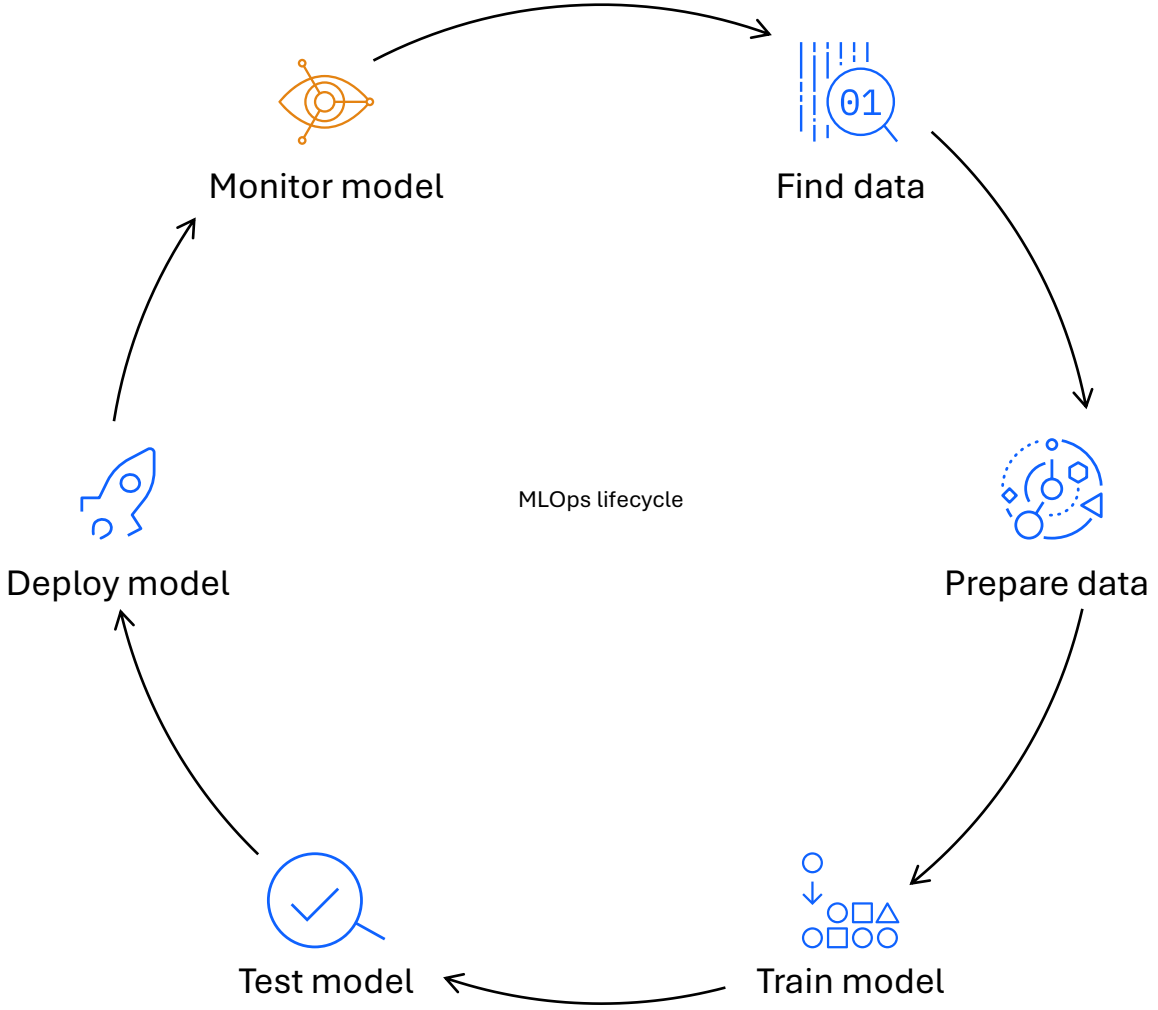
Example: Text generation



Use mathematics, statistics, specialized programming, and analytics to extract meaningful insights from data.

Designed to build, deploy, manage, and monitor ML models.

Machine Learning Operations (MLOps) lifecycle



Machine Learning (ML) methods

Supervised learning

- An operator provides the ML algorithm with a **known dataset** that includes desired inputs and outputs.

Semi-supervised learning

- By using a **combination of labeled and unlabeled data**, ML algorithms can learn to label unlabeled data.

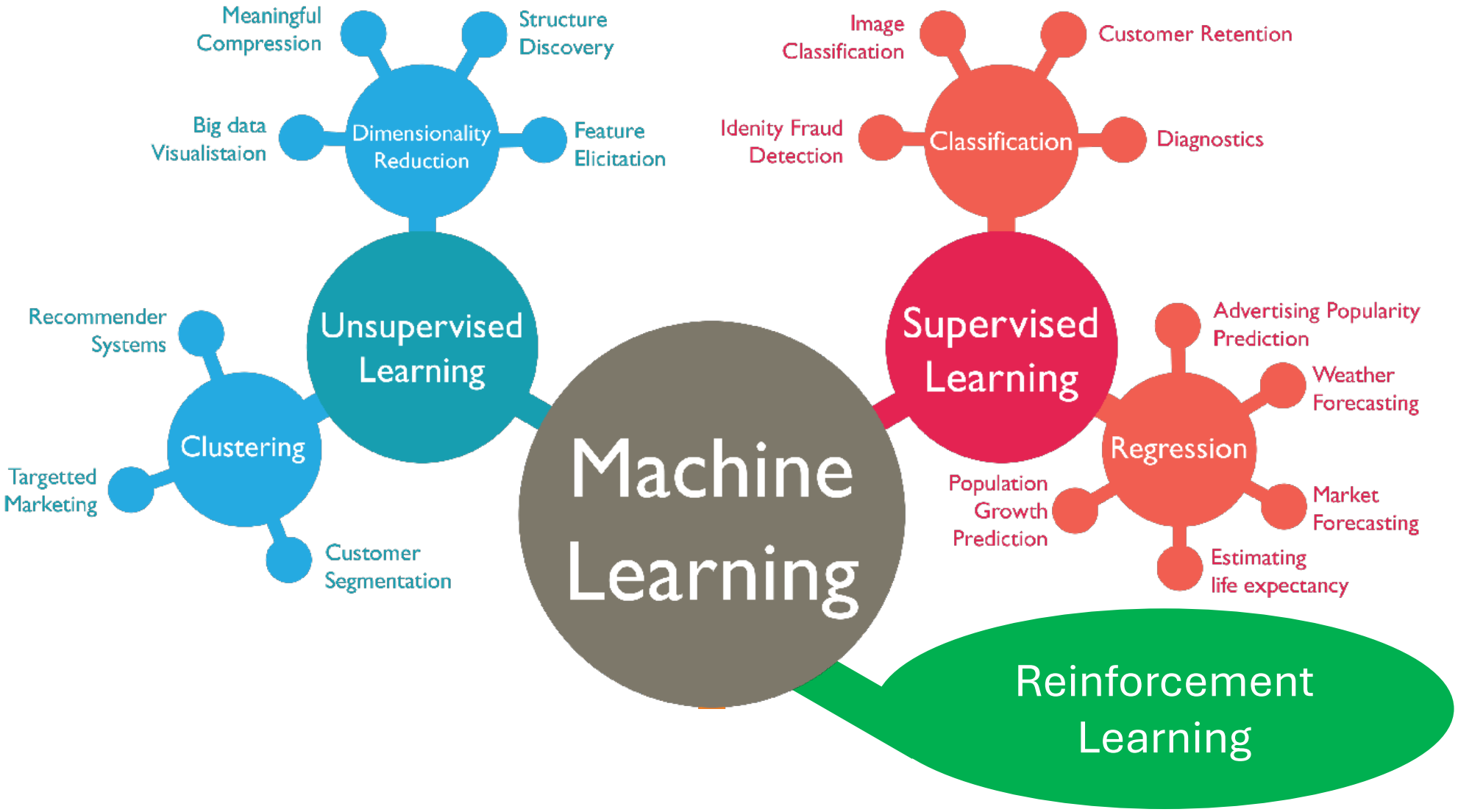
Unsupervised learning

- The **machine determines** the correlations and relationships by analyzing available data.

Reinforcement learning

- A technique that teaches an AI model to find the best result by **trial and error**.

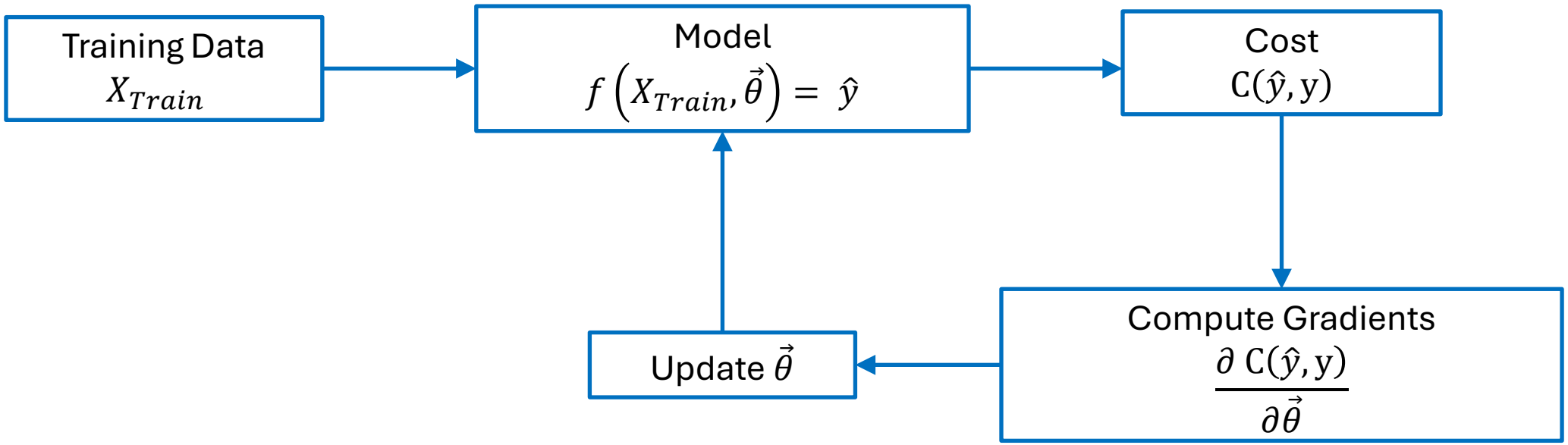
Machine Learning



Machine Learning Types

1. Supervised Learning
   - Classification
   - Regression
2. Unsupervised Learning
   - Dimensionality reduction
   - Clustering
   - Some generative models like GAN, autoencoder, etc.
3. Reinforcement Learning
   - Agent maximizing rewards in an environment

Supervised Learning - Workflow



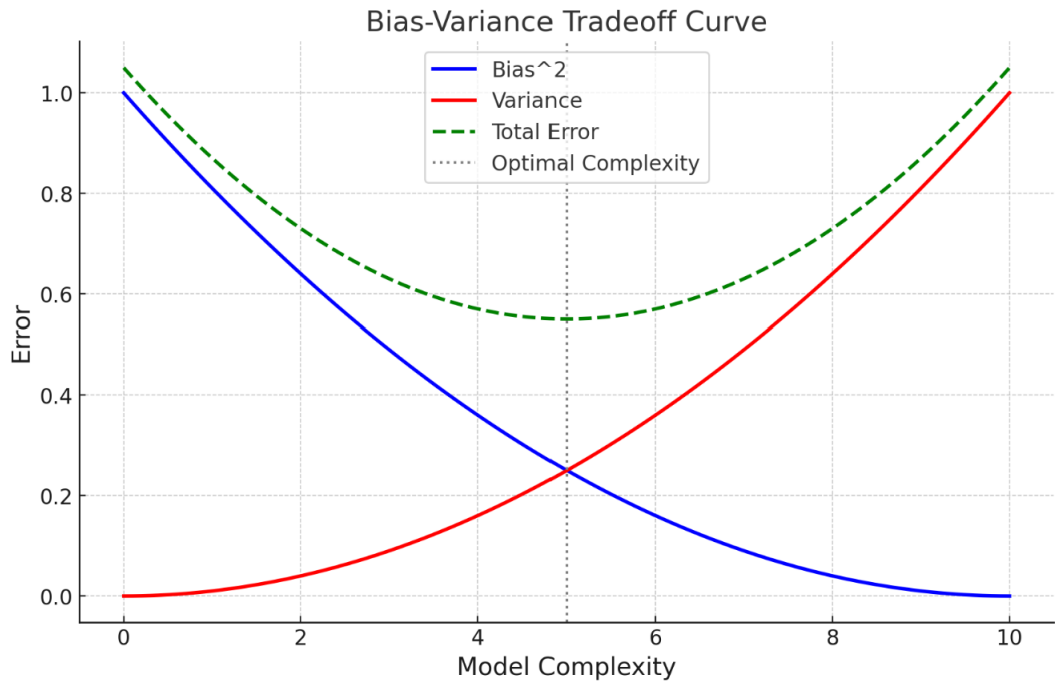
Precision, Recall, F1, Confusion Matrix, Accuracy, Jaccard Index, …

Model Validation

The Model should work well both on **Training and Test Data**

The Model should **not overfit nor underfit** to the Training Data

There is a bias-variance trade-off curve



- **Bias** decreases as the model complexity increases → the error rate decreases
- **Variance** increases with more complex models → the error rate increases
- **Total Error** initially decreases as model complexity increases but eventually rises due to overfitting
- The **optimal complexity** occurs where the total error is minimized, balancing bias and variance

What we do not elaborate now …

Regression

- Linear, Polynomial, Non-Linear

Classification

- K-Nearest Neighbors, Decision Trees, SVMs, Logistic Regression

Clustering

- K-Means Clustering, Hierarchical Clustering, Density Based Clustering

Recommender Systems

- Content based, Collaborative

Neural Networks

- Convolutional Neural Networks

Much More …

Key Performance Indicators

- Precision
- Recall
- F1 Score
- Accuracy
- Confusion Matrix

Multiple Methods in Machine Learning

Linear Models

- Regression, Classification, Clustering, Recommenders

Artificial Neural Networks (NN)

- Perceptron
  Binary classifier
- Feed Forward NN
  Concatenate Linear and Activation Functions (Sigmoid, ReLU, RBF)
- Recurrent NN
  Hopfield NN ("associative memory"), LSTM
- Boltzmann Machine
  Energy Based Model (statistical physics)

Graphical Models

- Bayes Network
  DAG represents set of variables and dependencies
- Hidden Markov Models
  - Modeled system = Markov Process X with unobservable states we learn from process Y

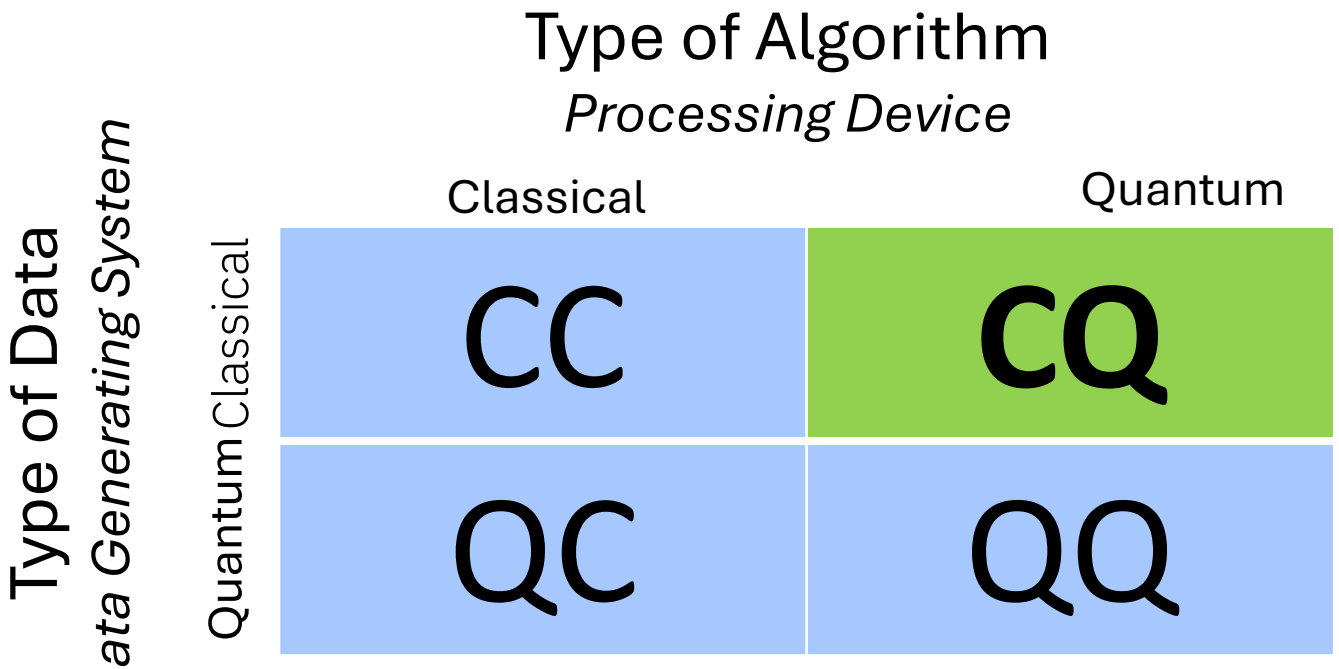
Kernel Methods

- Solve Machine Learning task based on the idea of a similarity measure between data points
- Kernel Function can be written as an **inner product** of data that is mapped into a suitable Feature Space F by a Feature Map $\varphi: X \rightarrow F: K(x, x') = \langle \varphi(x), \varphi(x') \rangle$
- **Kernel Trick** turns one model into another by replacing K by another Kernel K' – data are mapped into a higher dimensional space
- Machine Learning algorithm can solve problems by "simple linear algebra"
- **Sparse Kernel methods** are of particular interest
  - Kernel Density Estimation
  - K-Nearest Neighbor
  - Support Vector Machine
  - Gaussian Process

*Reproducing Kernel Hilbert Space or **RKHS***

Quantum Machine Learning

Positioning Quantum Machine Learning (QML)

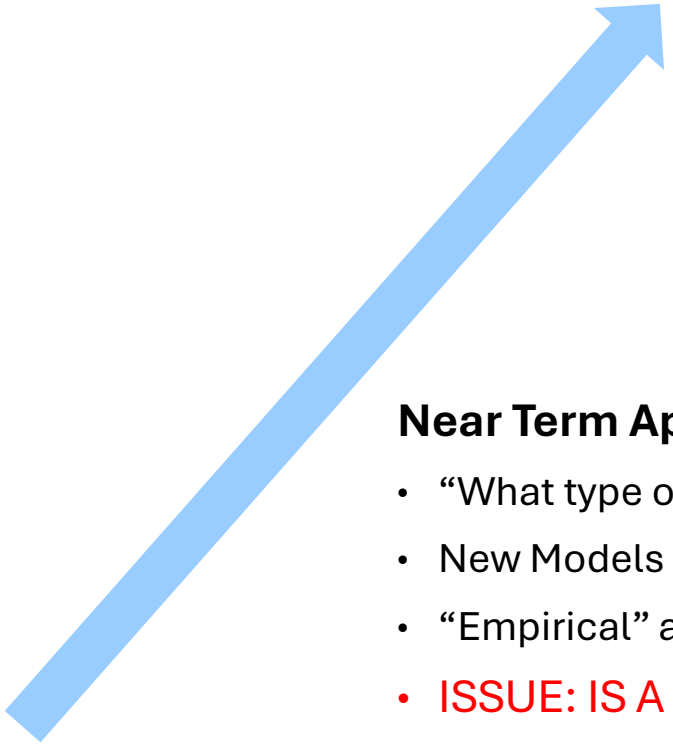


Maria Schuld, Francesco Petruccione
Machine Learning with Quantum Computers", Page 7

<u>Considerations beyond "the straightforward"</u>

- ***CC**: in this context: Machine Learning based on methods borrowed from Quantum Information research, or "Quantum-Inspired"*
- ***QC**: how can Machine Learning help with Quantum Computing?*
- ***CQ**: synonym for QML. Data come from classical systems like text, images, time series, macro-economic variables*
- *Requires Quantum-Classical interface*
- ***QQ**: closely related to CQ. Data can be measured from quantum system or dataset can be Quantum States*

Near-Term and Fault-Tolerant Methods



Emerging Approach - Third Wave

- Based on deeper understanding of ML Potential of Quantum phenomena
- Combine the CML knowledge with Quantum Information Theory
- Train Quantum Models we cannot simulate any more …
- **QUANTUM INSPIRED METHODS**

> 2021

Near Term Approach - Second Wave

- "What type of ML Model fits the physical characteristics of a small-scale Quantum Device?"
- New Models and Algorithms derived
- "Empirical" and "Heuristic" mindset
- ISSUE: IS A CLASSICAL ML MINDSET SUFFICIENT… ?

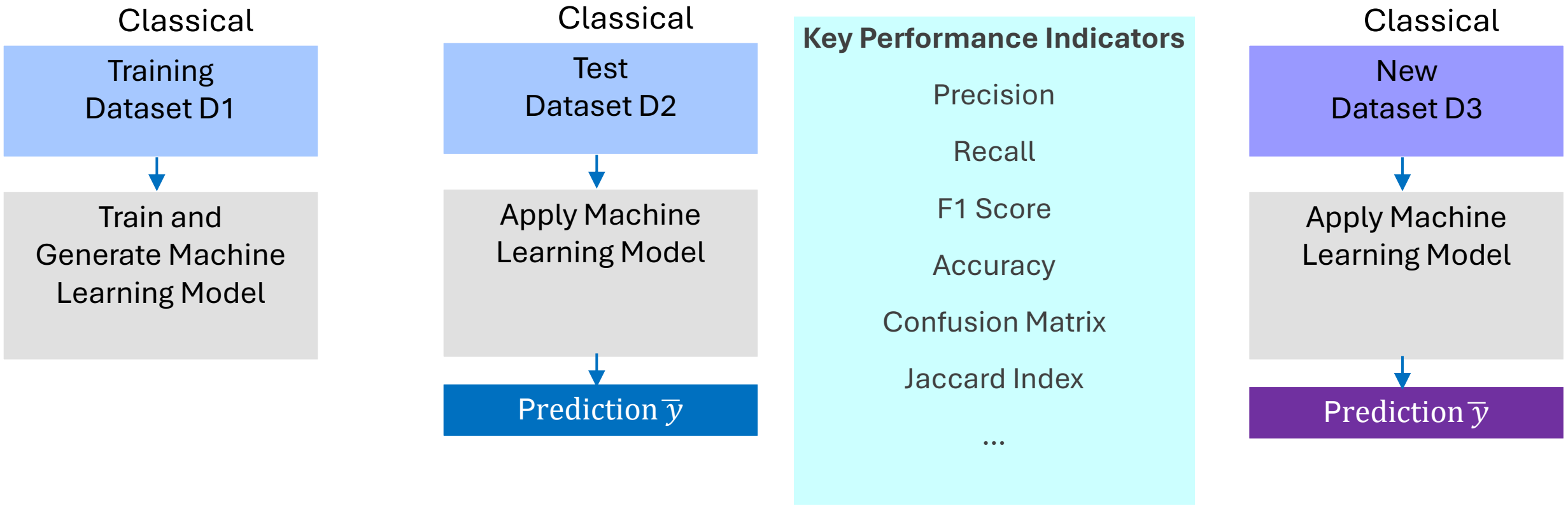
> 2017

Long Term Benefit – First Wave

- Assumes Fault-Tolerant Quantum Computers
- Rather "Academic" and "Mathematical" mindset
- ISSUE: WE ONLY HAVE NISQ TODAY …

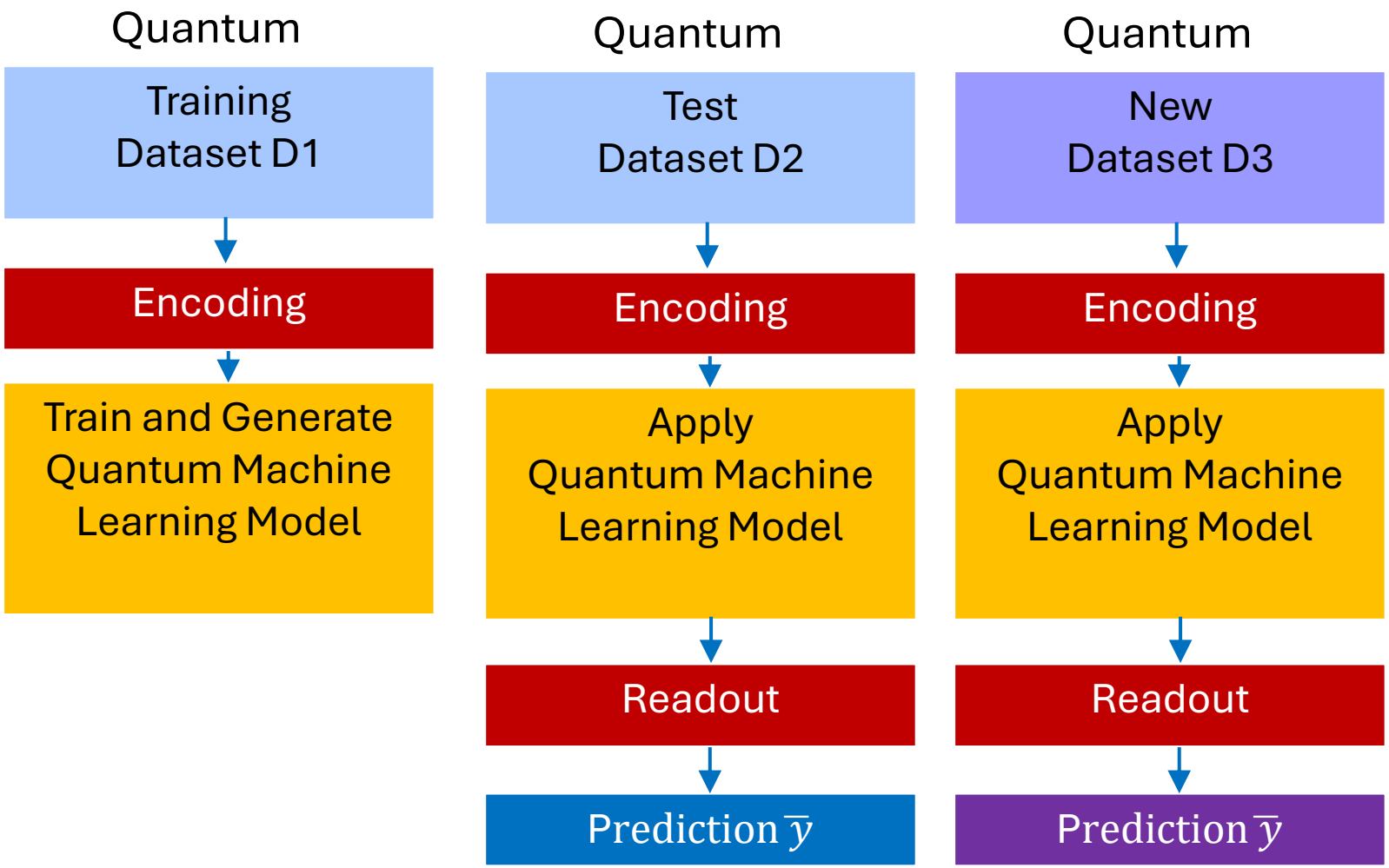
> 2013

The Key Challenge in QML



M. Schuld, F. Petruccione: Supervised Learning with Quantum Computers
M. Schuld, N. Killoran: arxiv 1803.07128

The Key Challenge in QML



Data Encoding

One of the most important parts of a QML approach…

*Frameworks, software and hardware that address the **interface** between classical memory and Quantum Device*

Encoding strategies

- Qubit-Efficient State Preparation
- Resource-Efficient State Preparation

Sample encoding methods

- Basis encoding
- Amplitude encoding
- Angle encoding
- Encode dataset via *Hamiltonian*
- Data Encoding as a *Feature Map*
- … *(ongoing innovation)*

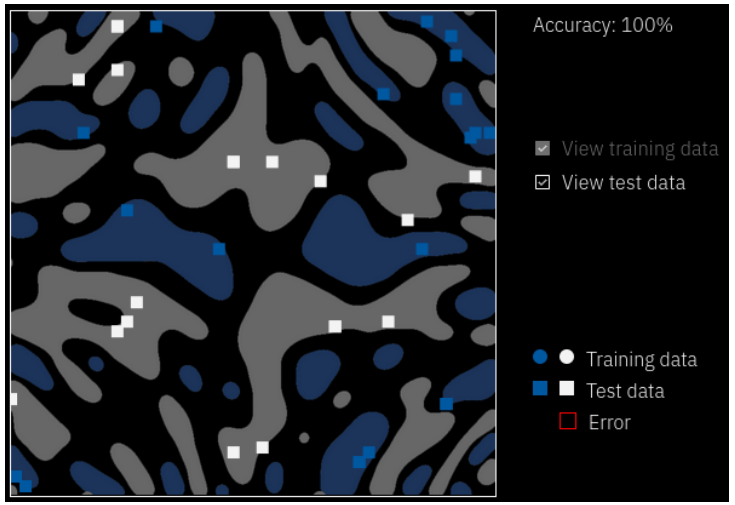
M. Schuld, F. Petruccione: Supervised Learning with Quantum Computers
M. Schuld, N. Killoran: arxiv 1803.07128

Near-term QML Algorithms – Key Flavours

Quantum SVM

- Classification, Regression
- Quantum Kernel Estimation method
- Benefit: Quantum Feature Space

Key paper: Havlíček et al, *Nature* 567, pp 209–212 (2019)

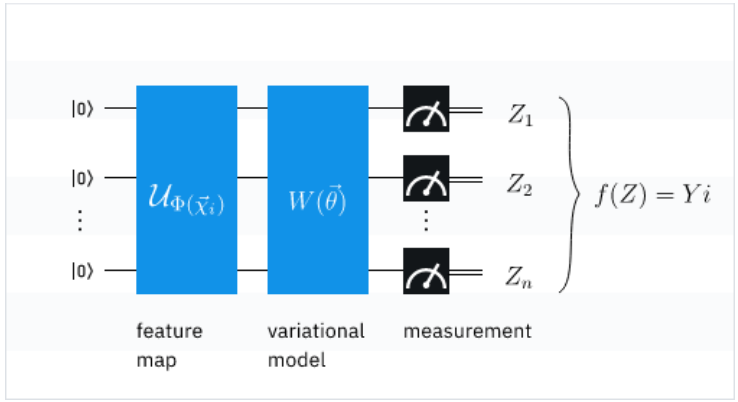


https://www.nature.com/articles/s41586-019-0980-2

Quantum Neural Networks

- Classification, Regression
- Variational Quantum Circuits
- Benefits: model expressiveness, resilience to Barren Plateaus

Key paper: Abbas et al, *Nature Comp. Sci.* 1, pp 403–409 (2021)

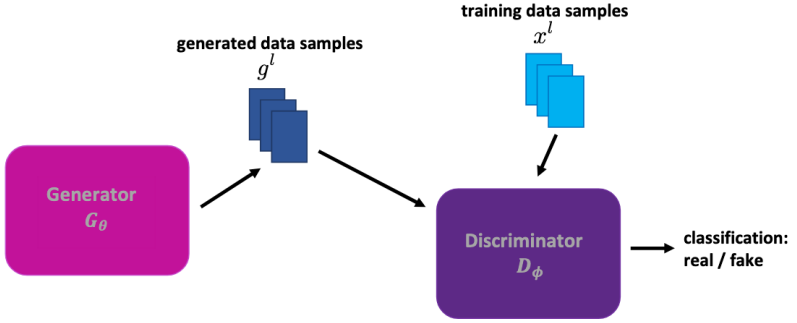


https://arxiv.org/abs/2011.00027

Quantum GANs

- Data Generation
- Quantum and Classical Neural Networks
- Benefits: efficient data sampling

Key paper: Zoufal et al, *npj Quant. Info*. 5, no: 103 (2019)



https://export.arxiv.org/abs/1904.00043

QML on Full Fault-Tolerant Devices

*Are **Quantum Computers** "better" at Machine Learning than Classical Computers?*

- "Traditional Approach" to QML, assuming abundance of perfect Qubits ☺
- NOT novel methods, but novel implementations of the SAME METHODS
- Quality of Algorithm judged thru asymptotic computational complexity

arXiv:1307.0411 (quant-ph) 2013
[Submitted on 1 Jul 2013 (v1), last revised 4 Nov 2013 (this version, v2)]
Quantum algorithms for supervised and unsupervised machine learning
Seth Lloyd, Masoud Mohseni, Patrick Rebentrost

Quantum 'BLAS'*

Amplitude Encoding
Runtime: $O(log(N.M))$ with $N = 2^n$

HHL: Invert Data Matrix:

$$w = (X^T X)^{-1} X^T y$$

- *Linear Regression*

[Quantum Data Fitting - N. Wiebe, D. Braun, S. Lloyd - 2012]

SVMs: Invert Kernel Matrix

- *Classification*

[Quantum SVM for Big Data Classification - P. Rebentrost, M Mohnseni, S. Lloyd - 2013]

Hopfield NN: Invert Adjacency Matrix

- *Associative Memory Recall*

[Quantum Hopfield NN - P. Rebentrost, T Bromley, C. Weedbrook, S. Lloyd - 2014]

*** <u>Caveat: Requires QRAM</u>**

Based on Grover's Search

Amplitude Amplification
Speedup: $O(\sqrt{M})$

Accelerate Amplitude Amplification

[Quantum Associative Memory – Ventura, Martinez - 1998]

Minimize $C(x)$ using Grover

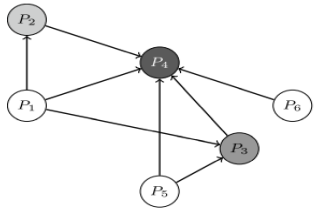
[A Quantum Algorithm for Finding the Minimum - C. Dürr, P. Hoyer - 1999]

Perceptron + Amplitude Amplification

[Quantum Perceptron Models – N. Wiebe, A Kapor, K. Svore - 2016]

Quantum Walks (*E.g.: PageRank*)

[Quantum speed-up of Markov chain- based algorithms]



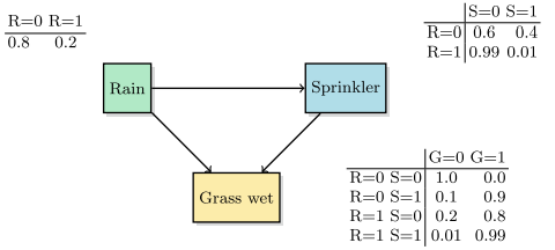
Probabilistic Methods

Measurements sampled from classical probability distributions

Bayesian Net

Elegant State Preparation
Runtime is $O(\sqrt{M})$ better than classical rejection sampling

[Quantum inference on Bayesian Networks, G H Low, T J Yoder, I L Chuang - 2014]



Boltzmann Machines

Qubit Efficient State Preparation
Mean-field approximation

[Quantum Deep Learning- N. Wiebe, A. Kapoor, K.Svore- 2015]

Data Encoding and Mapping

Data Encoding

Basis Encoding: encode each n —bit feature into n Qubits

Basic Example

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 11 \\ 01 \\ 00 \\ 11 \end{bmatrix} = \begin{bmatrix} |11\rangle \\ |01\rangle \\ |00\rangle \\ |11\rangle \end{bmatrix}$$

- We need **8 Qubits** in this example
- The State Vector of $\vec{x}$ has **16 dimensions**

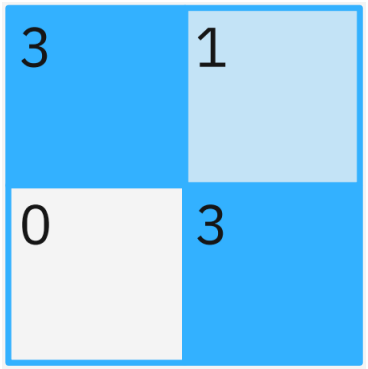
Data Encoding (Cont.)

Amplitude Encoding: Encode into Quantum State Amplitudes

- $|\psi_x\rangle = \sum_{i=1}^{N} x_i |i\rangle$

Basic Example

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \frac{3}{\sqrt{19}} \\ \frac{1}{\sqrt{19}} \\ \frac{0}{\sqrt{19}} \\ \frac{3}{\sqrt{19}} \end{bmatrix} = \frac{3}{\sqrt{19}} |00\rangle + \frac{1}{\sqrt{19}} |01\rangle + \frac{3}{\sqrt{19}} |11\rangle$$



- Amplitudes of **2 Qubits**
- The State Vector of $\vec{x}$ has **4 dimensions**

Data Encoding (Cont.)

Angle Encoding: Encode into Qubit Rotation Angles

- $|x\rangle = \bigotimes_{i=0}^{N-1} \cos(x_i)|0\rangle + sin(x_i)|1\rangle$

Basic Example

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 0 \\ 3 \end{bmatrix} becomes \begin{bmatrix} |0\rangle \\ -\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle \\ |0\rangle \\ |0\rangle \end{bmatrix}$$

$$\theta_0 = 2\pi$$
$$\theta_1 = \frac{2}{3}\pi$$
$$\theta_2 = 0$$
$$\theta_3 = 2\,\pi$$

- Angles of **4 Qubits**
- The State Vector of $\vec{x}$ has **16 dimensions**

In standard **angle encoding**, each feature of the classical data is directly mapped to a separate qubit. Every qubit undergoes a rotation based on the feature value, typically using the RY(θ) or RZ(θ) or RX(θ) gate. This means that if you have a classical data point with n features, would need n qubits to encode that data.

- **Qubits Used:** Linear with respect to the number of features in the classical data.
- **Data Representation:** Each feature gets its own qubit, and the qubit is rotated by an angle corresponding to the feature's value.

- For 3 Features $[x_1, x_2, x_3]$ $|\psi\rangle = RY(\theta_1) \otimes RY(\theta_2) \otimes RY(\theta_3)|0\rangle^{\otimes 3}$

$$\theta_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \cdot 2\pi$$

Data Encoding (Cont.)

- Dense Angle Encoding

- $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 0 \\ 3 \end{bmatrix}$ *becomes* $\begin{bmatrix} \mathrm{RY}(2\pi)\mathrm{RZ}(\frac{2}{3}\pi) \\ \mathrm{RY}(0)\mathrm{RZ}(2\pi) \end{bmatrix}$

- Qubit 1: Encodes θ_0 and θ_1 on the first Qubit
  $|\psi_1\rangle = \mathrm{RY}(\theta_0)\mathrm{RZ}(\theta_1)|0\rangle = \mathrm{RY}(2\pi)\mathrm{RZ}(\frac{2}{3}\pi)|0\rangle$

- Qubit 2: Encodes θ_2 and θ_3
  $|\psi_2\rangle = \mathrm{RY}(\theta_2)\mathrm{RZ}(\theta_3)|1\rangle = \mathrm{RY}(0)\mathrm{RZ}(2\pi)|1\rangle$

- Quantum State:
  $\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \mathrm{RY}(2\pi)\mathrm{RZ}(\frac{2}{3}\pi)|0\rangle \otimes \mathrm{RY}(0)\mathrm{RZ}(2\pi)|1\rangle$

- You need **2 Qubits**

- The State Vector of $\vec{x}$ has **4 dimensions**

- Dense angle encoding is a more qubit-efficient method
- Multiple classical features are encoded into fewer qubits
- Compresses the classical data into fewer qubits, sometimes by using multiple rotations or higher-dimensional entanglements within the quantum system
- Fewer qubits are needed compared to the number of features
- A single qubit might encode multiple features of a classical data point.

Normalization

$\theta_0 = 2\pi$

$\theta_1 = \frac{2}{3}\pi$

$\theta_2 = 0$

$\theta_3 = 2\,\pi$

Data Encoding (Cont.)

- Higher Order Encoding: Feature Maps or other Unitary Circuits

General representation:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_{N-1} \end{bmatrix} \rightarrow |\psi_x\rangle = U_{\phi(x)}|0\rangle$$

Encoding	# Qubits	State prep runtime
Basis	nN	$O(N)$
Amplitude	$\log(N)$	$\frac{O(N)}{O(\log(N))}$
Angle	N	$O(N)$
Arbitrary	n	$O(N)$

Encoding Data - Summary

- Basis Encoding
  Encode each n-bit feature into n qubits
  $x = (b_{n-1}, \dots, b_1, b_0) \rightarrow |x\rangle = |b_{n-1}, \dots, b, b_0\rangle$
  Combine M data points in superposition

- Amplitude Encoding
  Encode into quantum state amplitudes
  $$x = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix} \rightarrow |\psi_x\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$$
  Combine M data points in superposition

- Angle Encoding
  Encode values into qubit rotation angles
  $$|x\rangle = \bigotimes^{N} \cos(x_i)\,|0\rangle + \sin(x_i)\,|1\rangle$$

- Arbitrary Encoding (Feature Map)
  Encode N features in constant-depth circuit with n qubits
  $$x = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix} \rightarrow |\psi_x\rangle = \mathcal{U}_{\Phi(x)}|0\rangle$$

Encoding	# Qubits	State prep runtime
Basis	nN	$O(N)$
Amplitude	$\log(N)$	$\frac{O(N)}{O(\log(N))}$
Angle	N	$O(N)$
Arbitrary	n	$O(N)$

N features each

The impact of Entanglement in the Quantum Feature map *(Basic)*

Process

- Apply *H* gates on all qubits
- Apply parameterized Pauli rotations gates for each feature x_i
- Repeat *k* times

Pro: Simple to implement

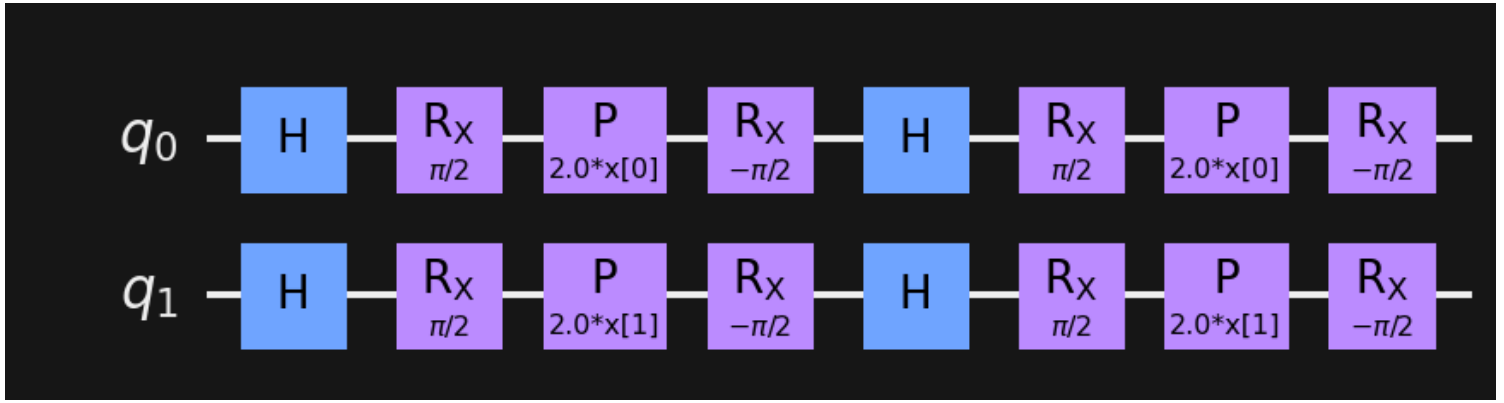
Con: Does not exploit high dimensionality

Encoded state: $|\Phi(x)\rangle = \mathcal{U}_{\Phi(x)}|0\rangle$

$$\mathcal{U}_{\Phi(x)} = (U_{\Phi(x)} H^{\otimes n})^d$$

$$U_{\Phi(x)} = \exp\left(i \sum_i \underbrace{\phi_i(x) P_i}_{\text{Pauli rotation operator}} + \cdots\right)$$

```
feature_map = PauliFeatureMap(feature_dimension=2, reps=2,
                              entanglement='linear', alpha=2.0,
                              paulis=['Y'], data_map_func=None)
```



https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.ZZFeatureMap

The impact of Entanglement in the Quantum Feature Map *(Intermediate)*

Process

- Apply *H* gates on all qubits
- Apply entangling gates and Pauli rotations gates for each feature x_i
- Repeat *k* times

Pro: Exploits Entanglement

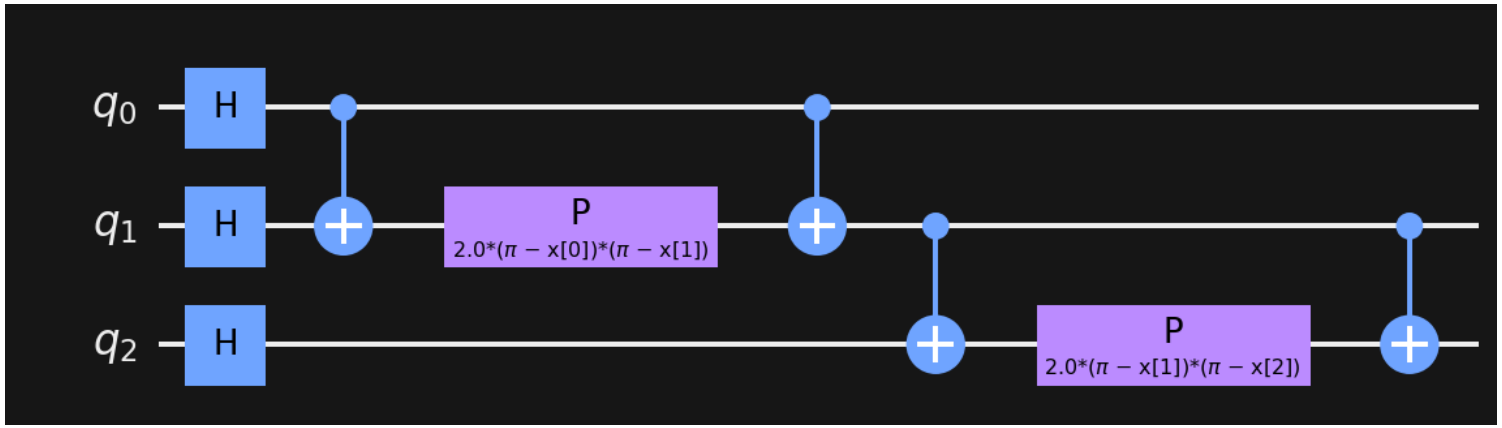
Con: Adds more Gates

$$|\Phi(x)\rangle = \mathcal{U}_{\Phi(x)}|0\rangle$$

$$\mathcal{U}_{\Phi(x)} = (U_{\Phi(x)}H^{\otimes n})^d$$

$$U_{\Phi(x)} = \exp\left(i\sum_i \phi_i(x)P_i + i\sum_{i,j} \underbrace{\phi_{i,j}(x)P_{ij}}_{\text{2-qubit Pauli rotation operator}} + \cdots\right)$$

```
feature_map = PauliFeatureMap(feature_dimension=3, reps=1,
                              entanglement='linear', alpha=2.0,
                              paulis=['ZZ'])
```



The impact of Entanglement in the Quantum Feature Map *(Complex)*

Process

- Apply *H* gates on all qubits
- Apply entangling gates and Pauli rotations gates for each feature x_i
- Repeat *k* times

$$|\Phi(x)\rangle = \mathcal{U}_{\Phi(x)}|0\rangle$$

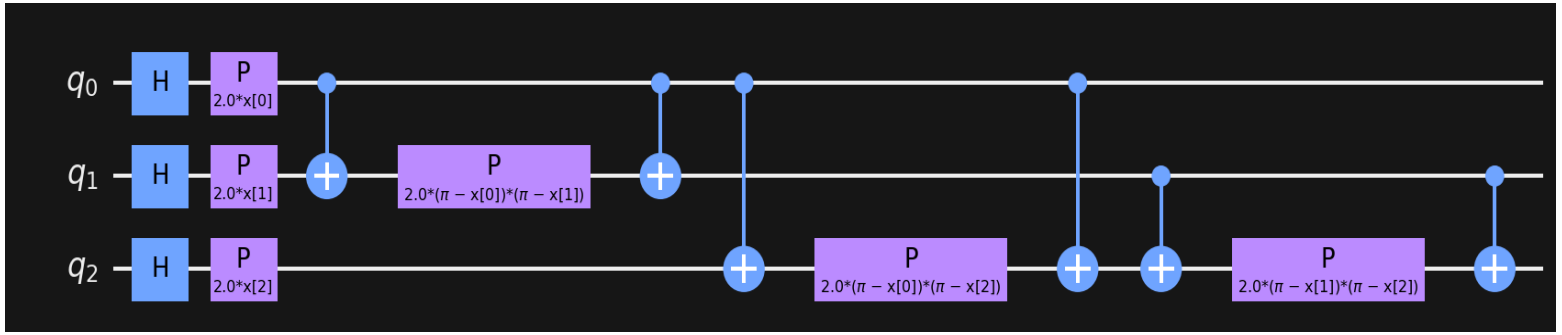
$$\mathcal{U}_{\Phi(x)} = (U_{\Phi(x)}H^{\otimes n})^d$$

$$U_{\Phi(x)} = \exp\left(i\sum_i \phi_i(x)P_i + i\sum_{i,j}\phi_{i,j}(x)P_{ij} + \cdots\right)$$

Pro: Exploits more Entanglement

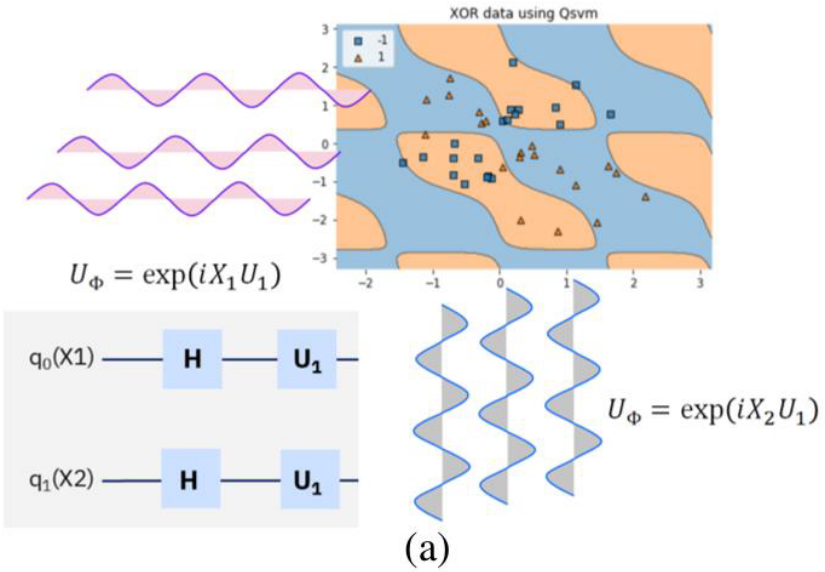
Con: Adds more Gates

```
feature_map = PauliFeatureMap(feature_dimension=3, reps=1,
                              entanglement='full', alpha=2.0,
                              paulis=['Z','ZZ'])
```

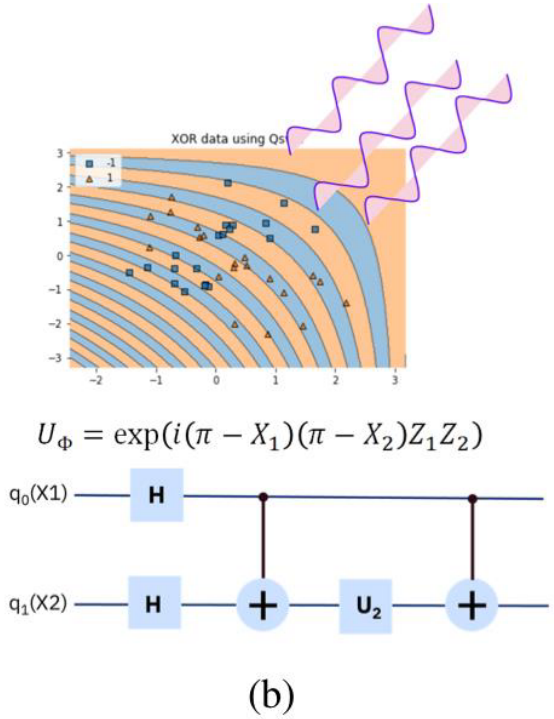


The impact of Entanglement in the Quantum Feature Map *(Outcome)*

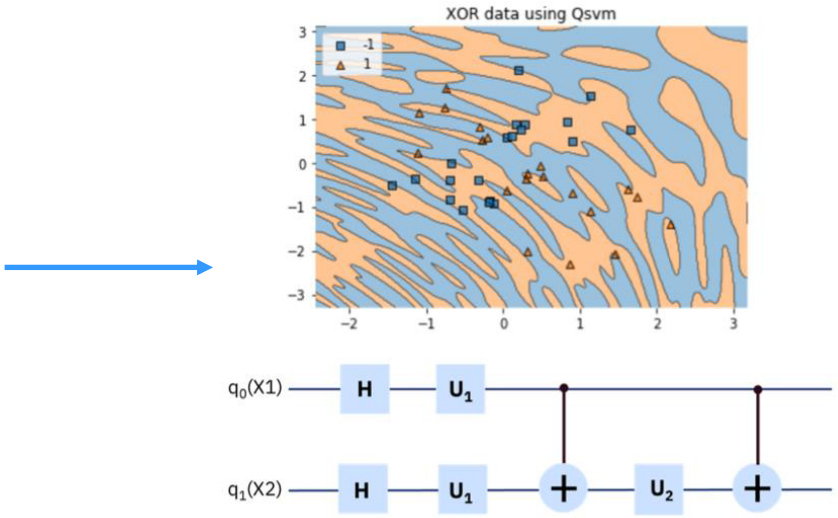
Simple rotations



Entangled unitary rotations



Simple and Entangled unitary rotations



Park et al, arXiv: 2012.07725v1

https://arxiv.org/abs/2012.07725

QML Dependencies

Exponential Speedup

- QFT
- Phase Estimation
- Shor
- QSDP
- QREC
- QSVE
- HHL
- QPCA
- QClustering

Quadratic Speedup

- Amplitude Amplification
- Amplitude Estimation
- Grover
- Quantum Rejection Sampling
- Quantum Monte Carlo Methods
- QRL
- VQE
- VQ Factoring
- QAOA
- QRBM
- QNN
- Quantum Feature Maps
- QSVM

Quantum Support Vector Machines

Classification with Support Vector Machines (SVM)

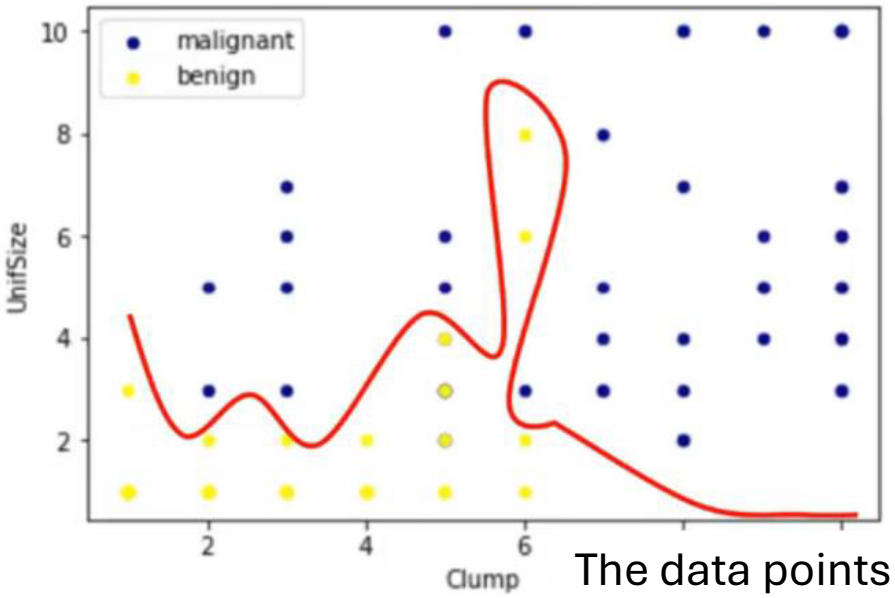
Patients at risk of developing cancer

What is an SVM?

An SVM is a supervised algorithms that classifies cases by finding a separator

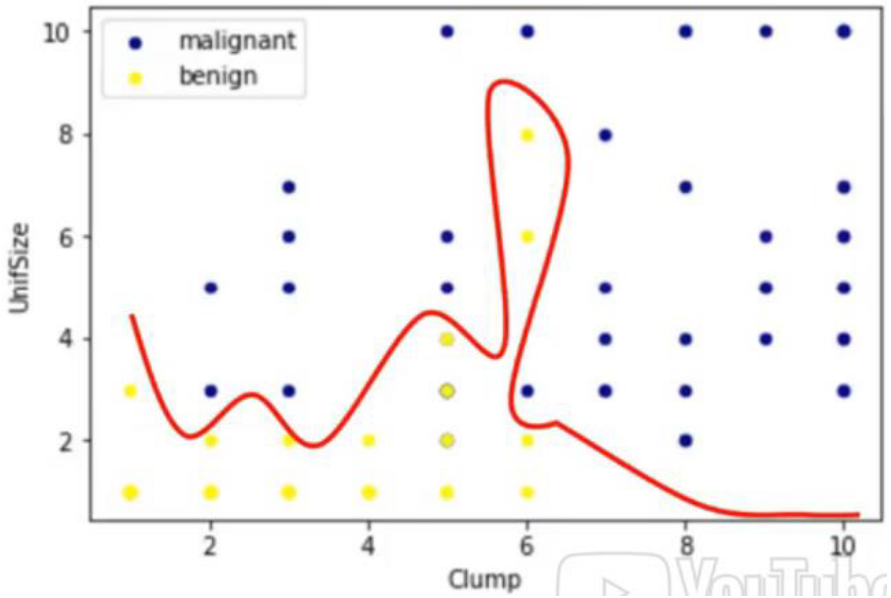
1. Map data to a higher dimensional feature space if necessary
2. Find a separator

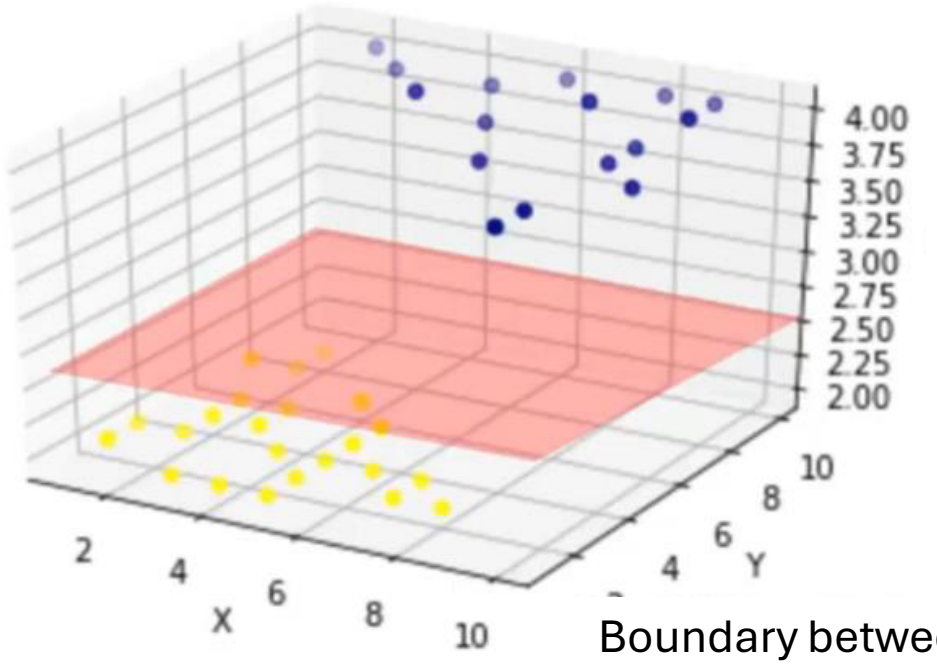
Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl	Mit	Class
5	1	1	1	2	1	3	1	1	benign
5	4	4	5	7	10	3	2	1	benign
3	1	1	1	2	2	3	1	1	malignant
6	8	8	1	3	4	3	7	1	benign
4	1	1	3	2	1	3	1	1	benign
8	10	10	8	7	10		7	1	malignant
1	1	1	1	2	10	3	1	1	benign
2	1	2	H	2	1	3	1	1	benign
2	1	1	1	2	1	1	1	5	benign
4	2	1	1	2	1	2	1	1	benign



The data points are not linearly separable

SVM Approach





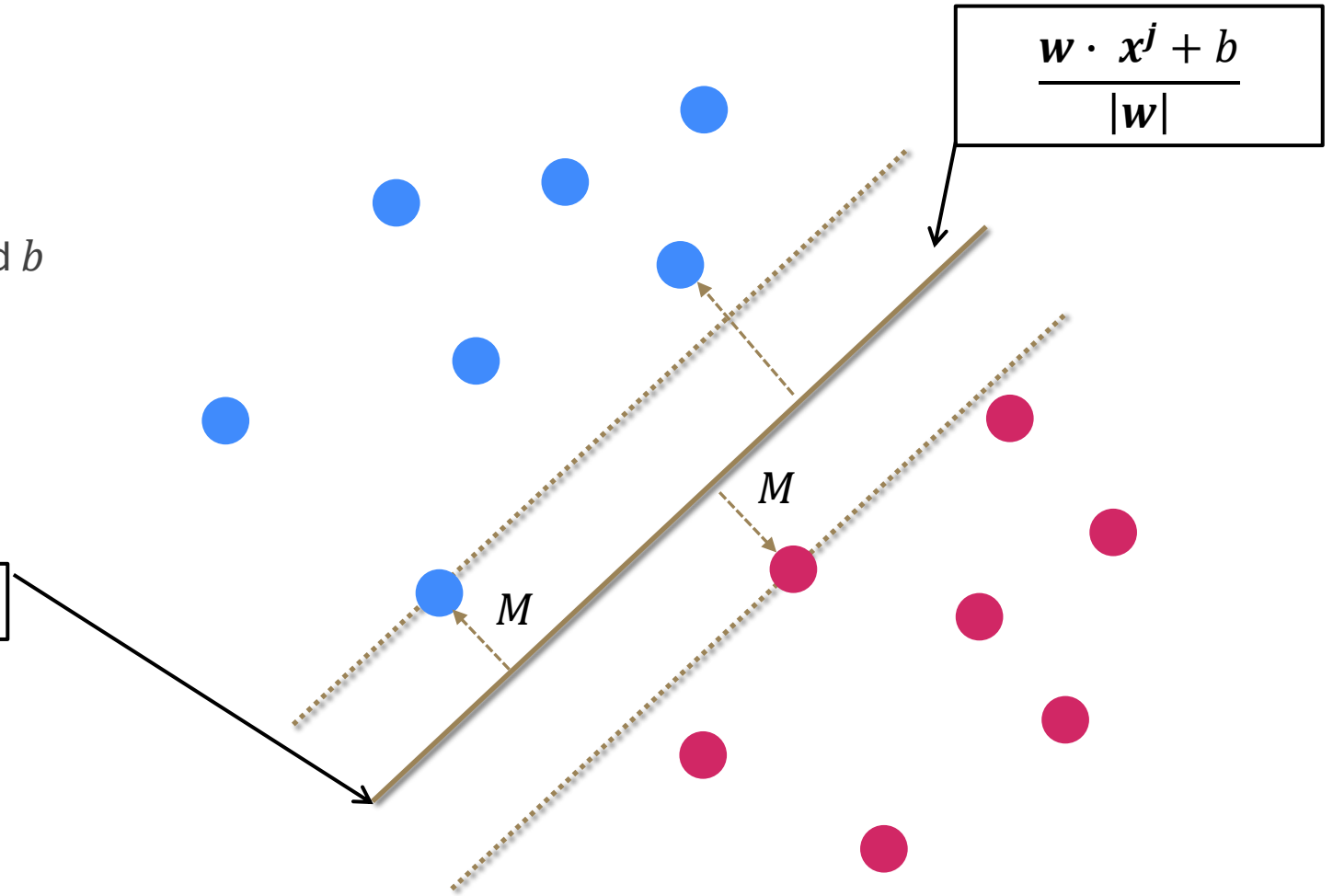
Boundary between categories with a Hyperplane allows to classify unknown cases

- Q1: How do we transform the data in such a way that a separator can be drawn as a hyperplane?
- Q2: How can we find the best optimized hyperplane?

Support Vector Machines

- **Task:** Binary classification
- **Data points:** $x_j \ \ (j = 1, 2, \ldots N)$
- **Goal:** Find decision boundary defined by $\boldsymbol{w}$ and b such that the margin M is maximized
- This means that $|\boldsymbol{w}|$ must be minimized

Decision boundary: $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$



- Best Hyperplane: has largest separation or margin between two classes
- **Support Vectors** are closest to the hyperplane (minimal margin)
- Distance Hyperplane –Support Vectors must be maximized

Summary of extended calculations

Handle Optimization Problem by Lagrange Multiplier (easier to handle)

Original Problem

Objective: $\max_{w,b}\left(M:=\frac{1}{|\boldsymbol{w}|}\right)\Leftrightarrow\min_{w,b}\frac{1}{2}|\boldsymbol{w}|^2$

Constraint: $t_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)\geq 1$

Framework in Lagrange Multiplier (transform $\boldsymbol{w}$ into $\boldsymbol{\alpha}$, s.t $\alpha_i\geq 0$)

Objective: $\max_{\alpha_i\geq 0}\left(L(\boldsymbol{\alpha}):=\frac{1}{2}|\boldsymbol{w}|^2-\sum_i\alpha_i[t_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)-1]\right)$

Constraint :

$$\alpha_i[t_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)-1]=0$$

$$\frac{\partial L}{\partial b}=\sum_i\alpha_i t_i=0$$

$$\frac{\partial L}{\partial w_k}=w_k-\alpha_k t_k x_k$$

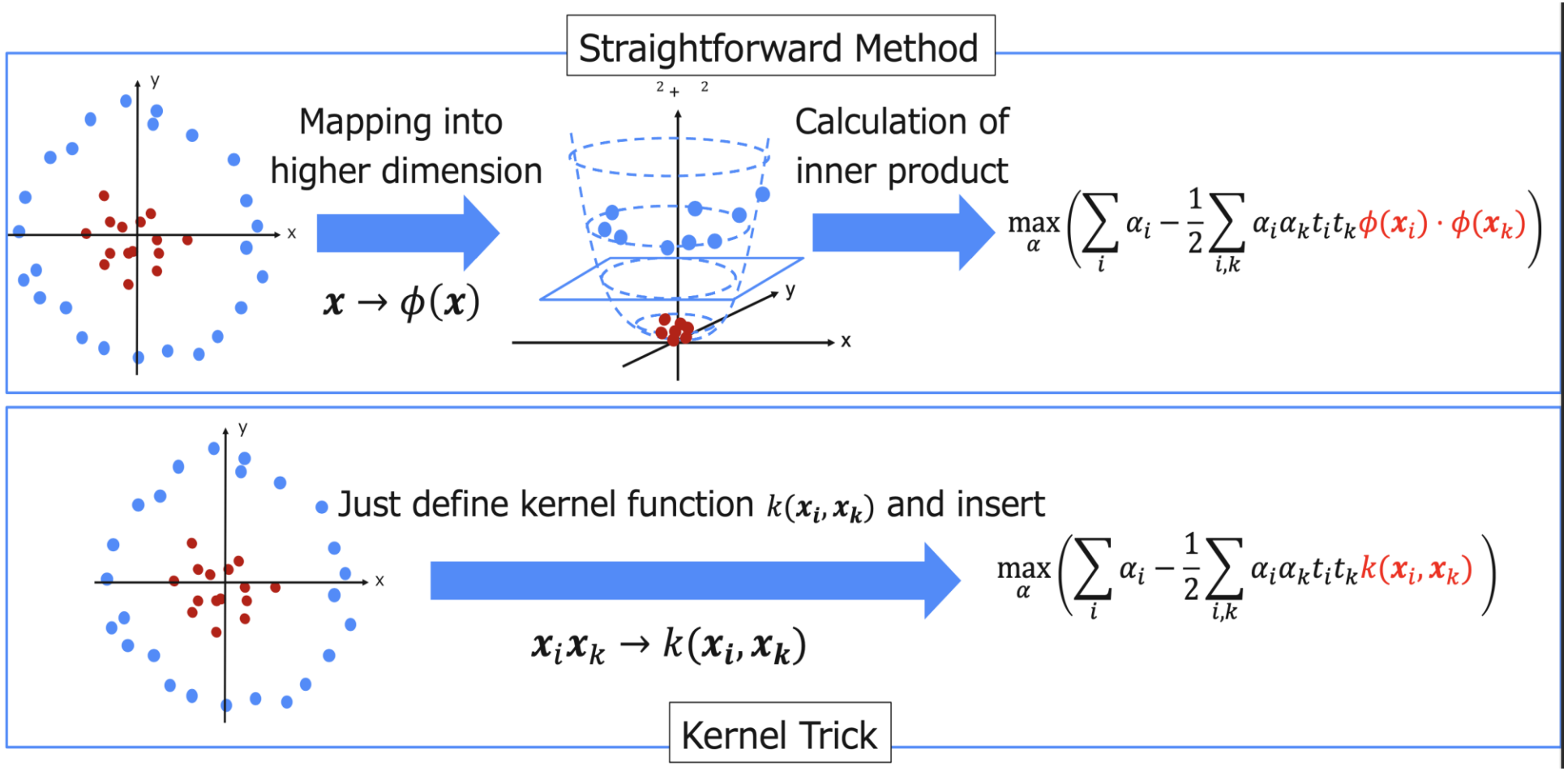
$$=\max_{\alpha_i\geq 0}\left(L(\boldsymbol{\alpha}):=\sum_i\alpha_i-\frac{1}{2}\sum_{i,k}\alpha_i\alpha_k t_i t_k\,\boldsymbol{x}_i\cdot\boldsymbol{x}_k\right)$$

Substitute $w_k=\alpha_k t_k x_k$

The solution gives $\boldsymbol{w}$, which define boundary plane

Accept this ☺

The SVM Kernel Trick



For this Kernel,
Quantum Computing is applied
Another name for Quantum Kernel is "*Quantum Feature Map*"

Mapping data into a higher dimensional space is called "kernelling"

There are different mapping or transformation functions

Most of them are implemented in Data Science programming languages

Breakthrough by IBM-MIT group demonstrating Quantum Advantages in Machine Learning

- Show how to map datasets into quantum feature maps that can be used to classify the datasets.
- The mapping is computed using Variational Quantum Methods
- Show how to compute **inner products** corresponding to **Quantum Feature Maps** efficiently

https://www.nature.com/articles/s41586-019-0980-2

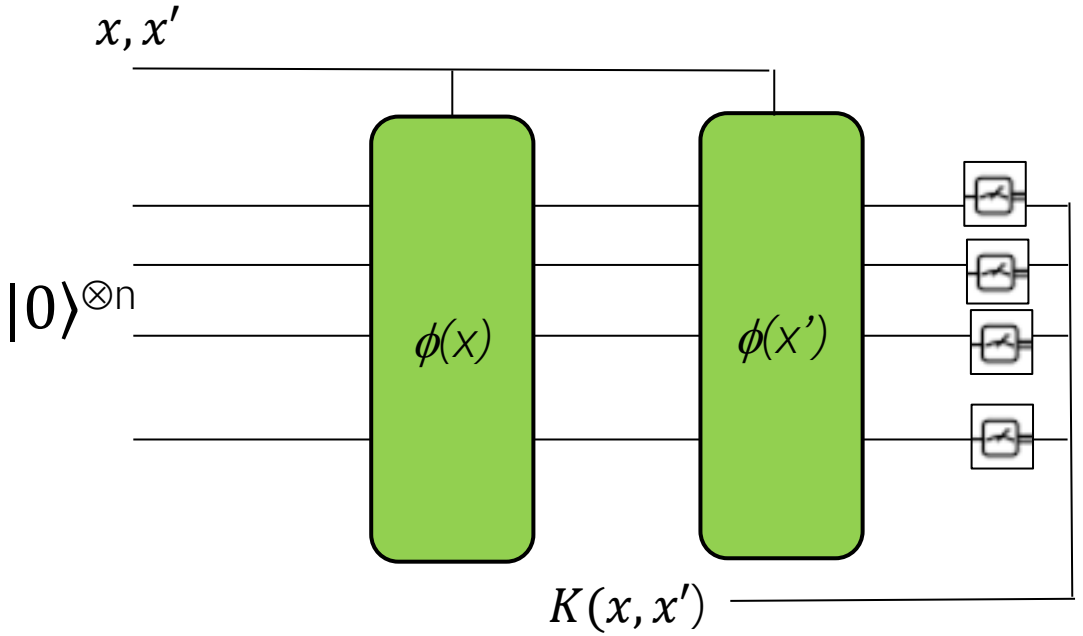
Kernel Methods in General

Main idea: map input data into a higher dimensional space to perform classification tasks

- Data vectors: x and $x' \in X$
- These are mapped to Quantum States $|\varphi(x)\rangle$ and $|\varphi(x')\rangle$
- Here φ is an encoding mapping: $\varphi : X \to H$ (Hilbert Space of the Quantum Register)

The inner product $\langle\varphi(x)|\varphi(x')\rangle$ is evaluated in an exponentially large Hilbert space

- It defines a similarity measure that can be used for classification purposes: 1 is equal, 0 is perpendicular
- Kernel: the matrix $K_{ij} = \langle\varphi(xi)|\varphi(xj)\rangle$ constructed over the training dataset



Can be used along with classical methods such as Support Vector Machines (SVMs) to carry out classification

A particularly promising aspect of such quantum SVMs is the possibility of constructing kernel functions that **are hard to compute classically**, thus potentially leading to **Quantum Advantage** in classification

A rigorous and robust Quantum Speed-Up in Supervised Machine Learning

Quantum Computing in Supervised Machine Learning

Researchers from University of California, Berkeley and IBM Quantum have developed a Quantum Kernel Estimation (QKE) procedure which serves as a quantum classifier created by a support vector machine.

The feature mapping for learning the concept class can be efficiently prepared on a *fault tolerant quantum computer that uses Shor's algorithm as a subroutine* to solve an intractable classification problem, the **discrete logarithm problem (DLP).**

A rigorous and robust quantum speed-up in supervised machine learning

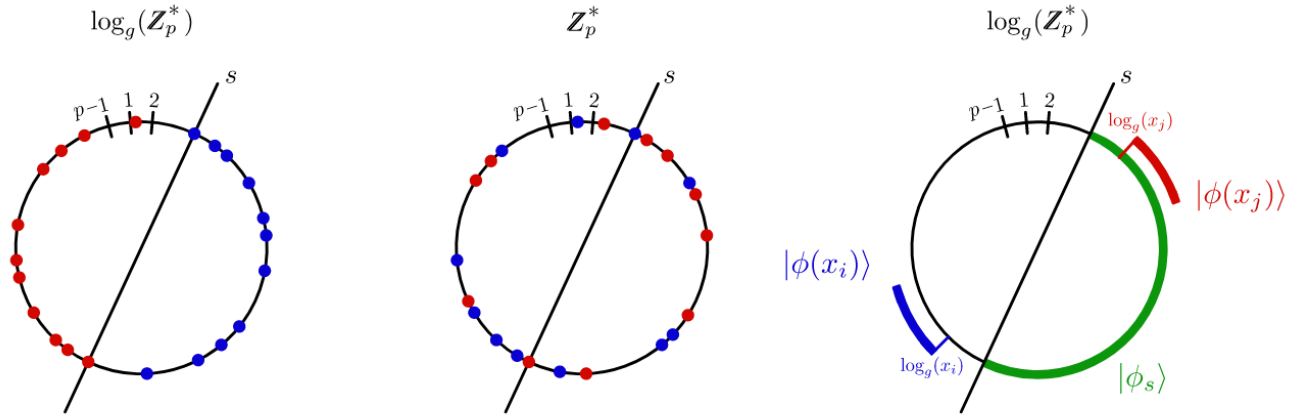
Yunchao Liu,[1,2,*] Srinivasan Arunachalam,[2,†] and Kristan Temme[2,‡]

[1] *Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720*

[2] *IBM Quantum, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598*

(Dated: October 6, 2020)

https://arxiv.org/abs/2010.02174



a

Discrete logarithm of data

b

Original data

c

Data encoded with quantum feature map

Quantum Neural Networks

Classical Artificial Neural Networks (NN)

- Simplest realization: **Feedforward network**
- Mathematically, concatenated application of affine transformations and element wise nonlinearities:

$$\boldsymbol{f_j}(x) = \boldsymbol{\sigma_j}(\boldsymbol{w_j} \cdot x + \boldsymbol{b_j})$$

Where:

σ_j = nonlinear **activation function**

w_j = weight matrix

b_j = **bias vector**

- NN is repeated application of this function

$$\boldsymbol{y} = \boldsymbol{f_L} \circ \boldsymbol{f_{L-1}} \circ \dots \circ \boldsymbol{f_1}(x)$$

Where:

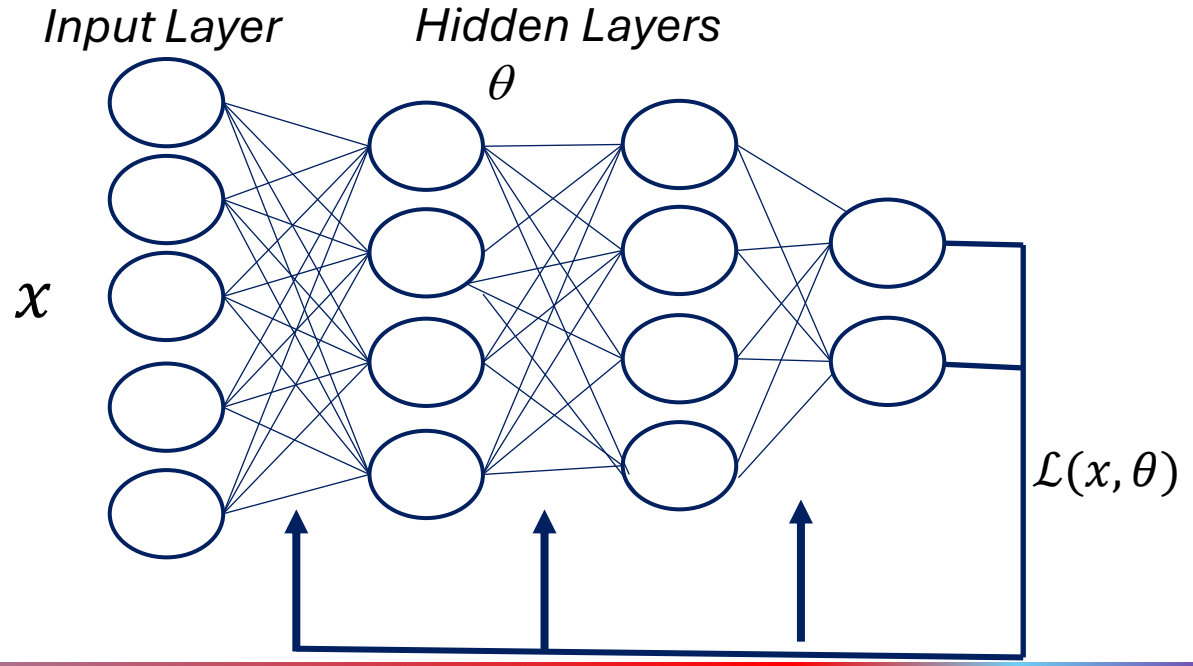
y = output

x = input vector

L = number of layers

Trainable → <u>"Generalization"</u>

- Weight matrices adjusted to learn target task
- Approximates association between inputs x_i and correct outputs y_i
- Predict well on **new** data
- **Discovers hidden patterns/relations in data**



General Quantum Neural Networks (QNN)

- QNN is Parameterized Quantum Circuit (PQC)
- Variational Ansatz in multiple repetitions of similar layers:

$$U_{QNN} = \prod_L^1 U_i(\theta_i)W_i$$
$$= U_L(\theta_L)W_L \dots U_1(\theta_1)W_1$$

where:

$U_i(\theta_i)$: variational gate layers

W_i: fixed operations

L: number of layers

- Scientists make the equation **more expressive** to represent more functions of the input data:

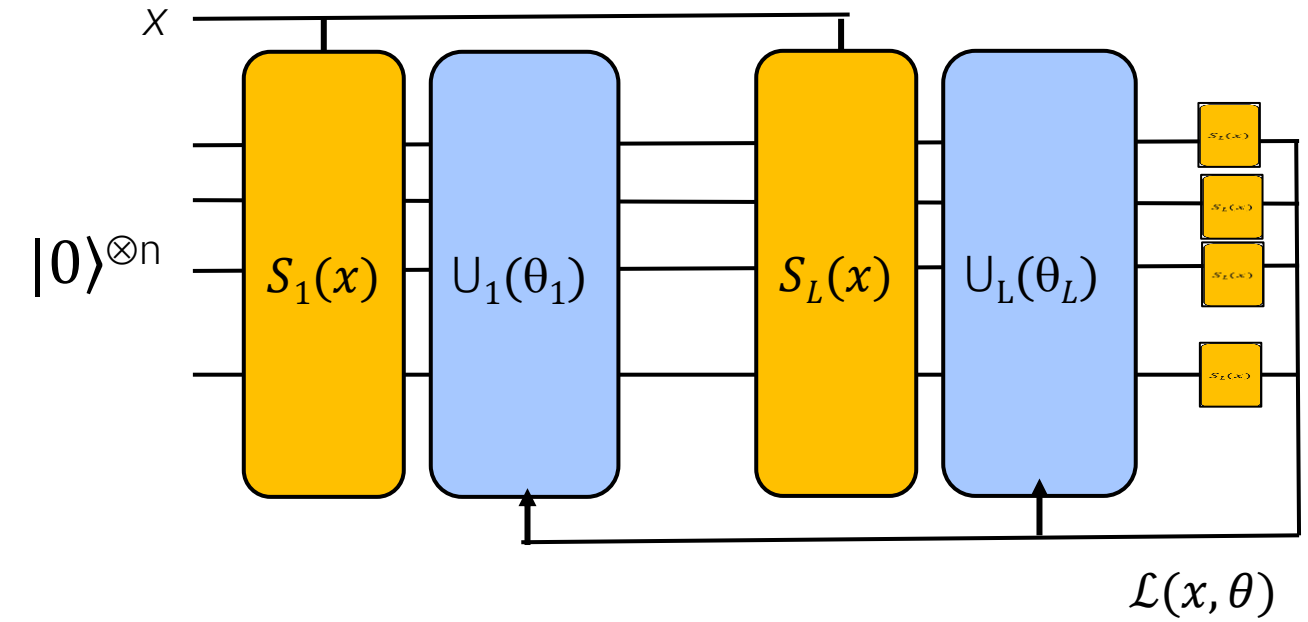
$U_{QNN} = \prod_L^1 \boldsymbol{S_i(x)}U_i(\theta_i)W_i$ OR

$U_{QNN} = \prod_L^1 \boldsymbol{S_i(x)}U_i(\theta_i)$

where $S(x)$ denotes the **data encoding procedure**

Remark: it could be that $S_2 = S_3 = \dots = S_L = I$

Data re-uploading procedure



The more often data are re-uploaded,
the more expressive the QML model becomes

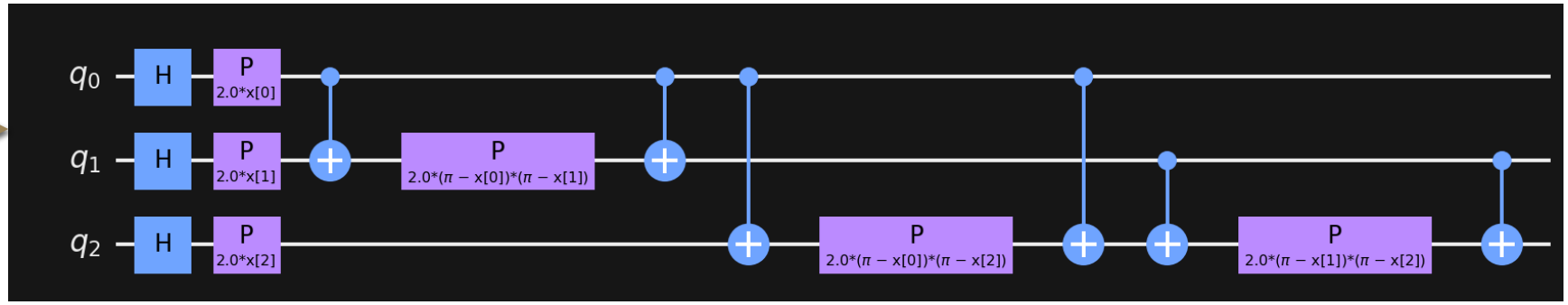
VQC = Variational Quantum Classifier
VQR = Variational Quantum Regressor

Examples of components of a QNN Circuit

Data encoder

$S_1(x)$

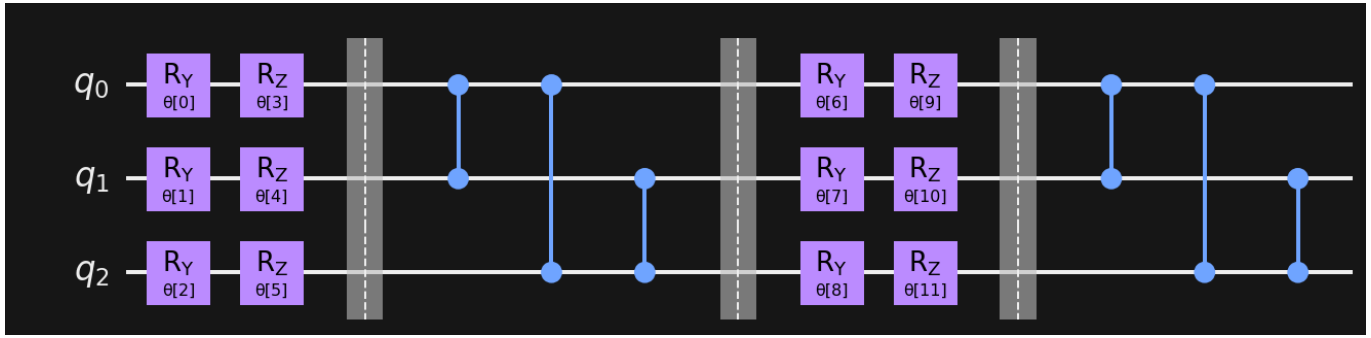
ZZ feature map



Parameterized circuit

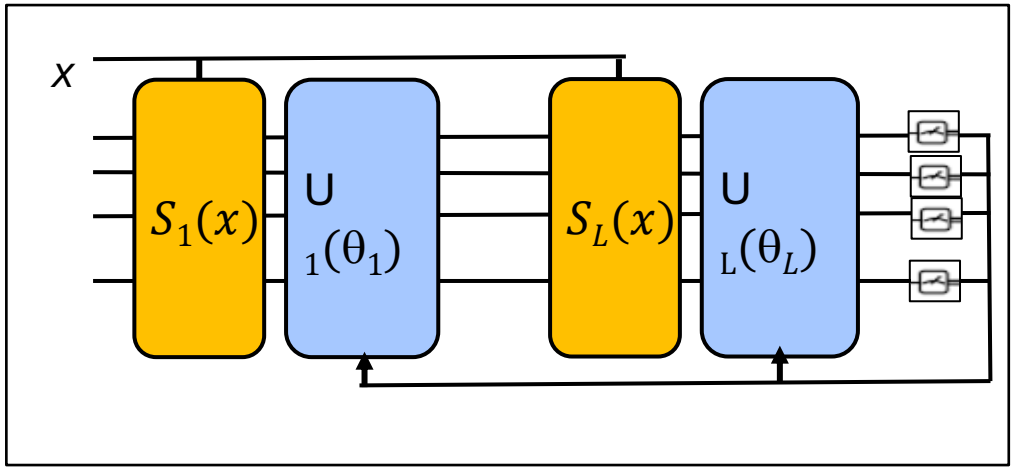
$U_1(\theta_1)$

Two-local (R_Y and R_Z)



Hybrid Training of QNNs

Variational Quantum Circuit



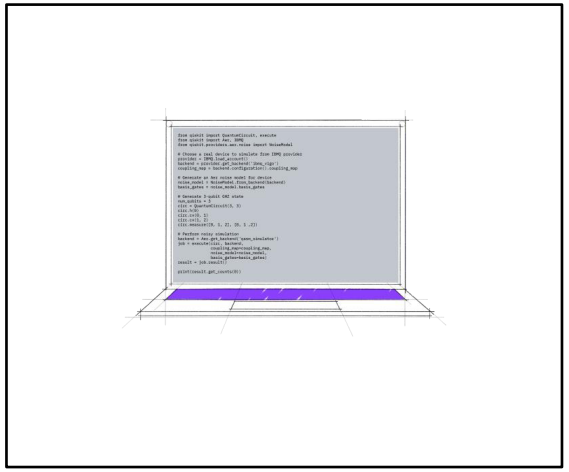
Measurement

$$E(\theta) = \langle\psi(\theta)|H|\psi(\theta)\rangle$$

Parameter update

$$\theta_i \rightarrow \theta_{i+1}$$

Classical Optimizer



Quantum Perceptron Models

- Perceptron = Single Artificial Neuron

- Question: how to get nonlinear activation?

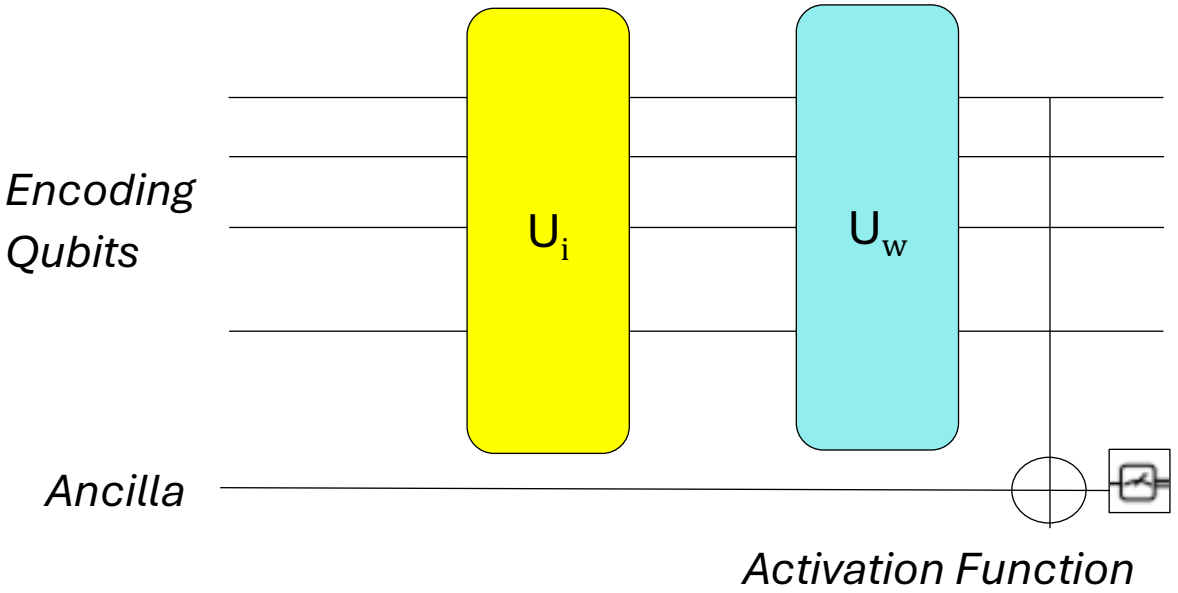
  One approach: measurement

- Simple case: binary classifier

$$f(x) = \begin{cases} 1, & if\ w\ .x + b > 0, \\ 0, & otherwise \end{cases}$$

  where:
  - w: weight vector
  - x: input vector
  - b: bias

- Example task: grayscale/binary image recognition



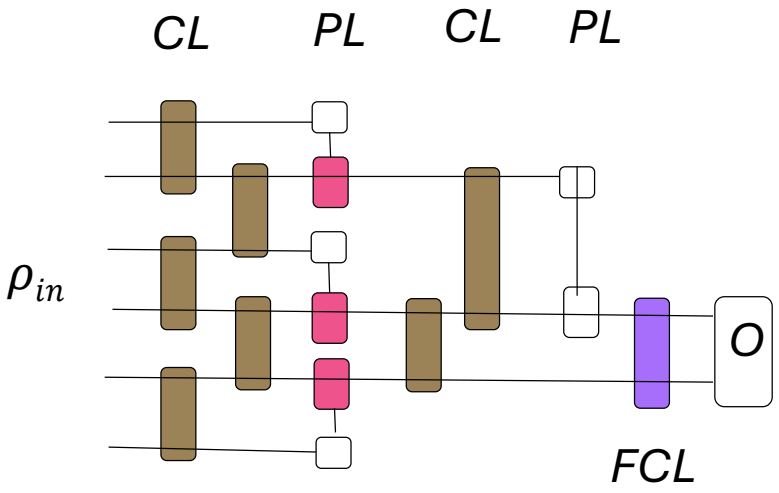
- *U_i: load input data*
- *U_w: multiplication by weights*

[Tacchino, npj Quant. Info., 2019]

https://www.nature.com/articles/s41534-019-0140-4

Quantum Convolutional Neural Networks

- Inspired by Classical CNNs used for image processing
- Architecture
  - Convolution layers (CL): transformation to extract relevant information, translationally invariant
  - Pooling layers (PL): compresses in a lower dimensional representation
  - Fully-connected layers (FCL)



Quantum Convolutional Neural Networks

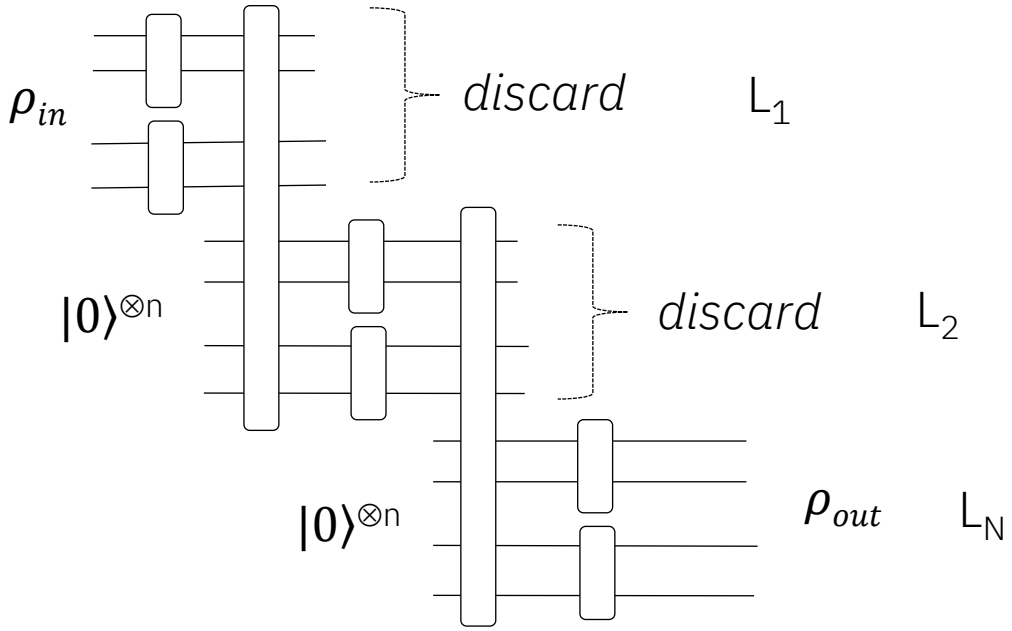
- CL applied on a Quantum State
  - *Parametrized Unitary Operation acting on individual subsets of the Qubits*
- PL measures some of the Qubits
- Only logarithmically many parameters → efficient training

- *CL = Convolutional Layer*
- *PL = Pooling Layer*
- *FCL = Fully-Connected Layer*

Quantum Dissipative Neural Networks

- Each Qubit represents a Node
- Edges connect Qubits in different Layers, representing Unitary Operators

- **“Dissipative”:**
- Progressive discarding of Layers of Qubits after a given Layer has interacted with the following Layer



Usage

- Learn general Quantum transformations to carry out Universal Quantum Computation
- Possible route towards Quantum “analogues” of Deep Neural Networks analyzing Quantum Data

Qubits in L_n are coupled to ancilla Qubits in L_{n+1}, loading the result of the computation of L_n

Previous layers are discarded or dissipated

The Challenge of the Barren Plateaus

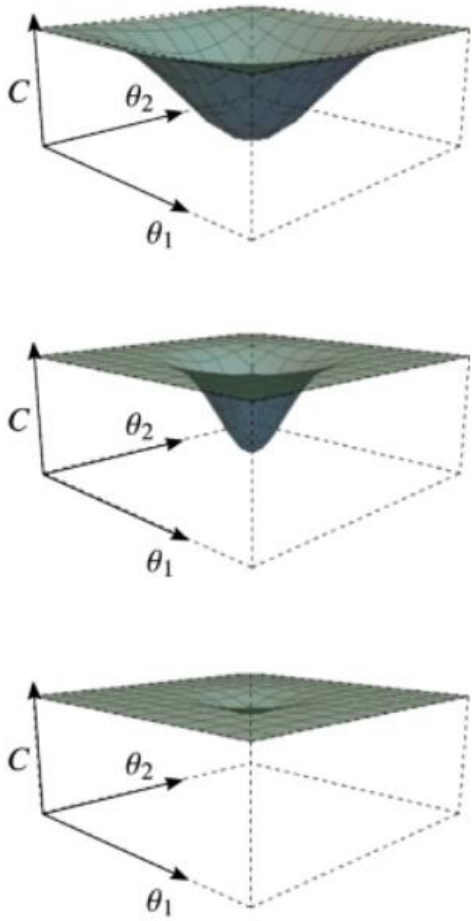
Key "Victims"

- Noisy Near-Term Quantum Computers
- Highly expressive Circuits and large Hilbert Spaces
- Deep Quantum Circuits
- Circuits with global measurements
- High entanglement entropy
- Dissipative Quantum Neural Networks

Convolutional QNNs are **resilient** to Barren Plateaus

[See: Pesah et al., Phys Rev X, 2021]

https://journals.aps.org/prx/pdf/10.1103/PhysRevX.11.041011



Some strategies to avoid or mitigate Barren Plateaus

- Initialize and train parameters in batches
- Reduce depth of circuit by randomly initializing a subset of the total number of parameters
- Introduce correlations between the parameters in multiple layers of the QNN to reduce the overall dimensionality of the parameter space
- Leverage Classical recurrent NNs to find good parameter initializing heuristics such that the network starts close to a minimum
- Choice of local cost functions and specific entanglements between hidden and visible units

The Power of Quantum Neural Networks

Quantum Computing in Machine Learning

IBM Research, University of ZwaZulu-Natal, and ETH Zurich demonstrated that well-designed quantum neural networks (QNNs) offer advantages over comparable classical neural networks. These advantages include better loss function values and faster training. That is, well-designed QNNs take less time to train, and can achieve lower errors, than comparable classical neural networks.

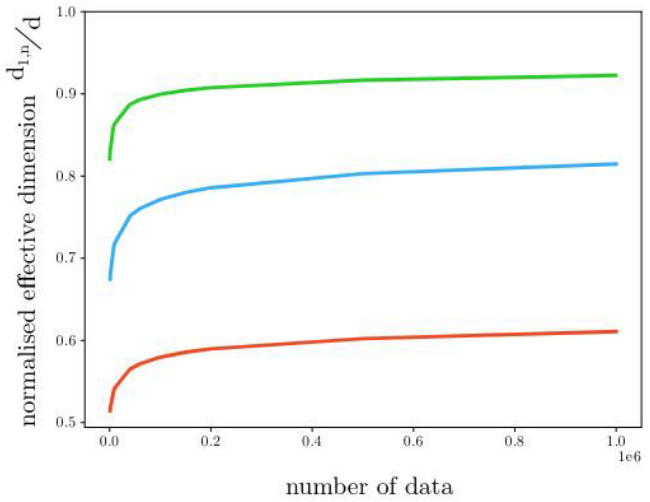
https://arxiv.org/abs/2011.00027

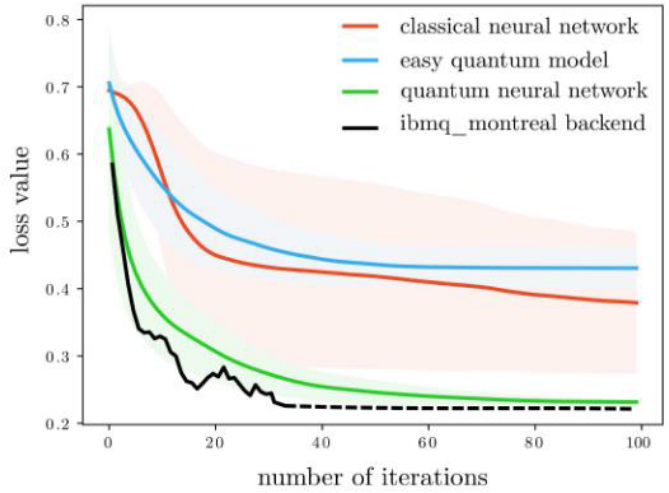
Nature Comp. Sci. 1, 403-409 (2021)

The power of quantum neural networks

Amira Abbas[1,2], David Sutter[1], Christa Zoufal[1,3], Aurelien Lucchi[3], Alessio Figalli[3], and Stefan Woerner[1,*]

[1] *IBM Quantum, IBM Research – Zurich*
[2] *University of KwaZulu-Natal, Durban*
[3] *ETH Zurich*

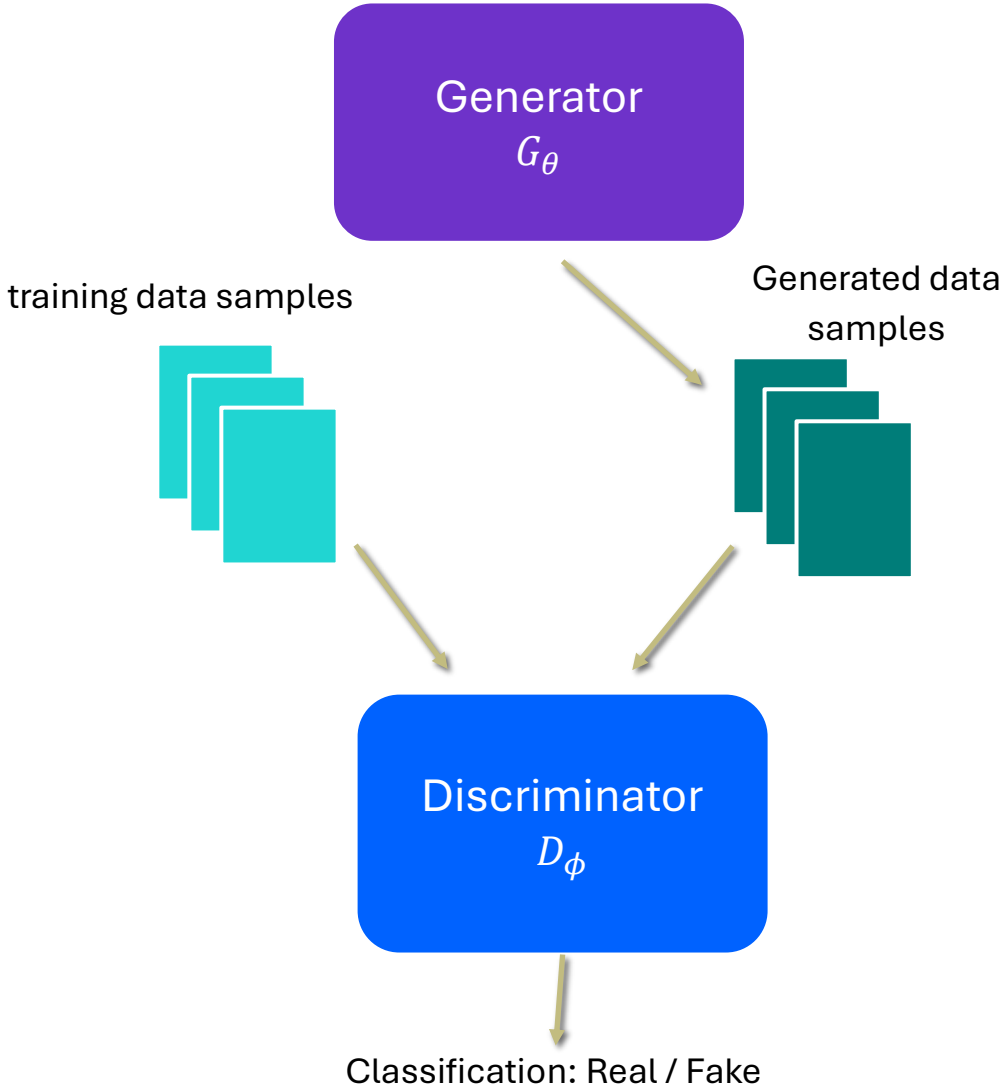




Quantum Generative Adversarial Networks

How do GANs work?

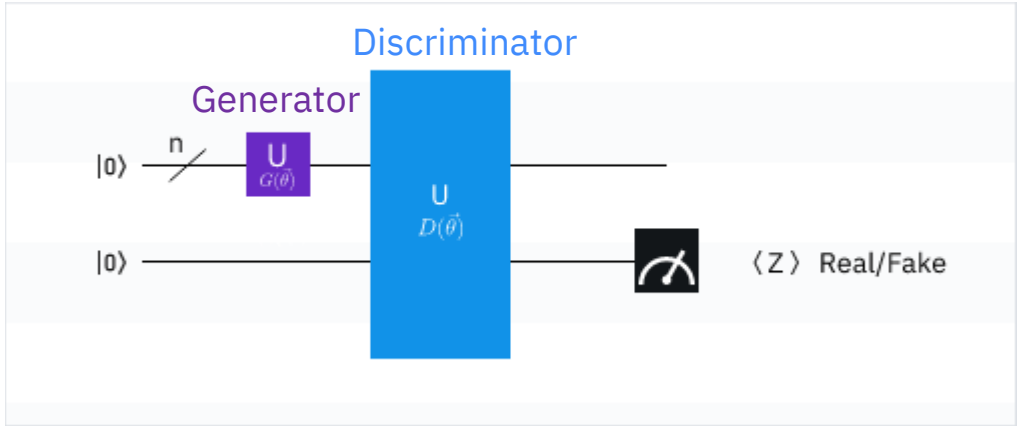
- **Generator:** learns to create data that cannot be distinguished from training data
- **Discriminator**: learns to distinguish training (real) data and generated (fake) data
- **Goal:** Generate realistic-looking data by learning the probability distribution of the training data



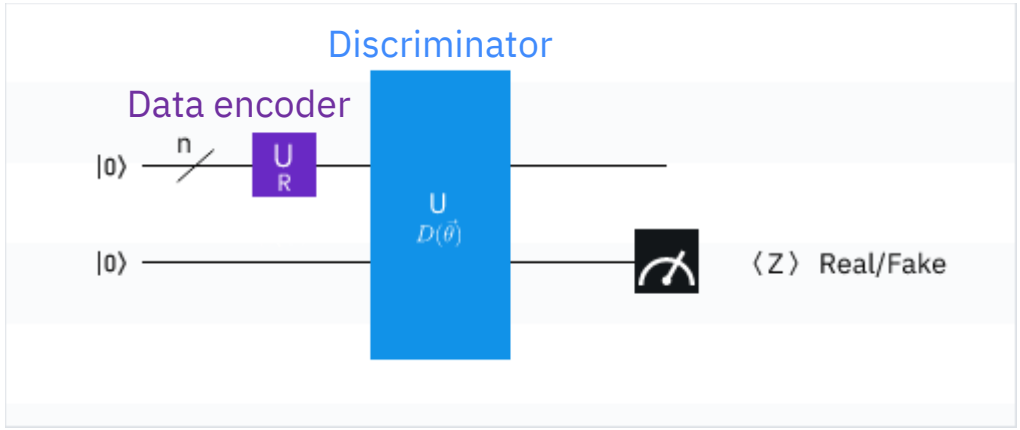
Quantum GAN Architectures

Quantum-Quantum QGAN

Input generated data

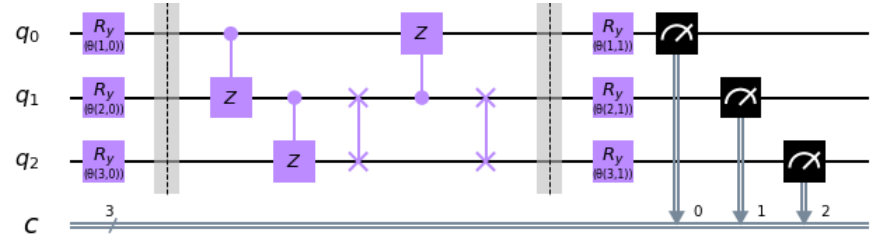


Input training data

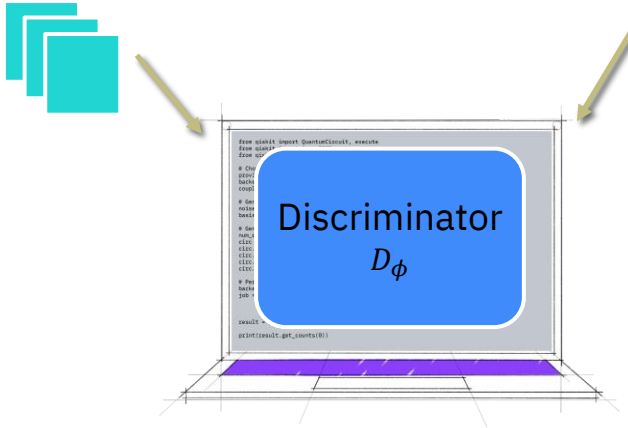


Quantum-Classical QGAN

Generator network circuit



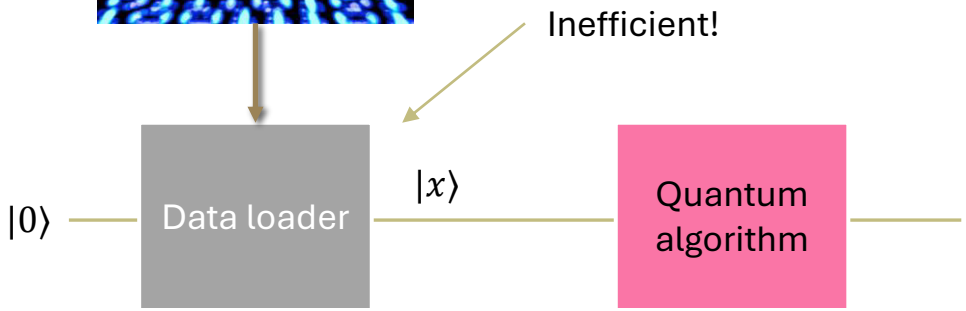
training data



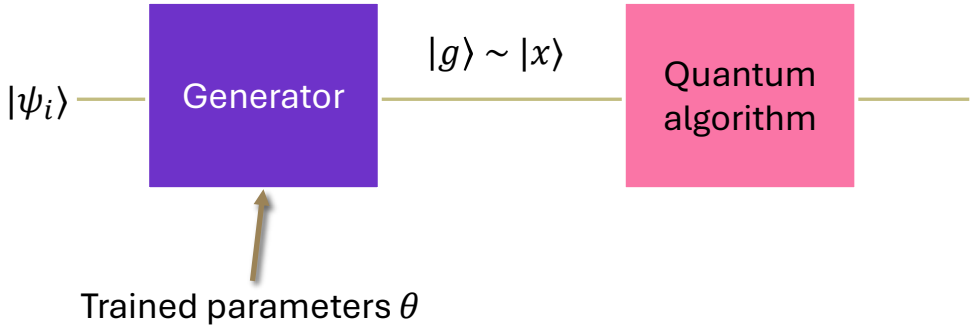
https://arxiv.org/abs/1901.00848

Application: Approximate Data Loading

- **Scenario:** need to load data (e.g. a probability distribution) into Quantum Computer
- **Problem:** generic data loading runtime O (2^n)
- **Impacts speedup of downstream Quantum Algorithms!**



- **Alternative:** train a quantum generator from a QGAN to approximate the data
- **Benefit:** Avoids exponential overhead



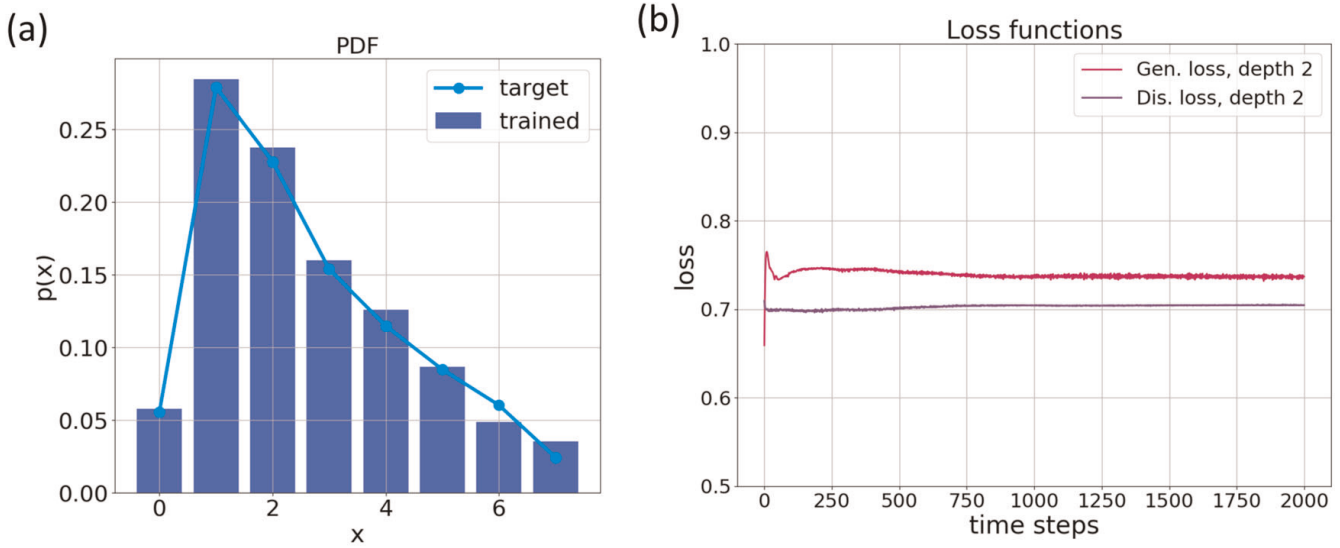
Quantum Generative Adversarial Networks for Learning and Loading Random Distributions

Quantum Computing Optimization of Simulations

IBM Quantum researchers have developed an efficient way to prepare the initial state of a quantum system for subsequent quantum algorithms that rely on the distribution, such as Monte-Carlo simulation, using a quantum version of a generative adversarial network (GAN).

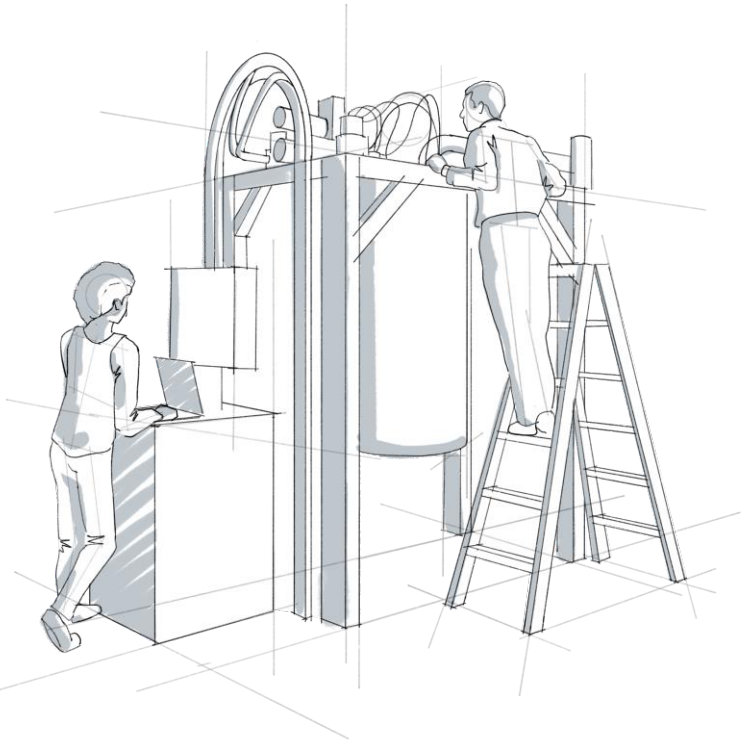
Quantum Generative Adversarial Networks for Learning and Loading Random Distributions

Christa Zoufal,[1,2,*] Aurélien Lucchi,[2] and Stefan Woerner[1]
[1] *IBM Research – Zurich, Rueschlikon 8803, Switzerland*
[2] *ETH Zurich, Zurich 8092, Switzerland*
(Dated: October 1, 2019)



Quantum Machine Learning with Qiskit

Qiskit – Ecosystem



https://www.ibm.com/quantum/ecosystem

Extend the power of Qiskit

The Qiskit ecosystem is a collection of open-source capabilities and tools created by researchers and developers who use Qiskit every day.

https://qiskit-community.github.io/qiskit-machine-learning/

Qiskit | Ecosystem

Qiskit Machine Learning 0.7.2

Summary

Caveats

- **Data loading:** may incur significant runtime that's longer than the QML algorithm!
- **Quantum RAM:** many fault-tolerant algorithms require QRAM, but not invented yet!
- **Oracles:** some algorithms assume existence of an oracle circuit for the problem
- **Runtime of near-term algorithms:** no guarantees since based on classical heuristic optimization

Potential Quantum Advantages

Commercial expectations clash with complex reality of scientific research ☺

Dissection of Quantum Advantage – 4 criteria carry importance

1. *Quality: the Algorithm generalizes from data samples to unseen data*
2. *Speedup: the Algorithm runs and/or learns faster than classical competitors, usable in practice*
3. *Relevance: the Algorithm solves a problem that is relevant to Machine Learning*
4. *Availability: the technology to implement the Algorithm is (highly probably to be) available*

Today, no QML Algorithm fulfills all 4 of these requirements correctly …

Without proofs nor competitive benchmarks, an Algorithm can still be relevant

Remember: Neural Networks showed poor performance and are now forefront of the 4th Industrial Revolution, without any performance guarantees on paper ☺

Key take-aways on QML

- Potential dimensions of QML Quantum Advantage: Quality (e.g. accuracy), speed and efficiency
- Pay attention to Data Encoding of classical data into a quantum computer
- Near-term QML algorithms like QSVM, QNN and QGANs may provide benefits in terms of quantum enhanced feature space, model expressibility, resilience to barren plateaus and efficient data sampling
- Exponential and quadratic speedups are expected on Fault-Tolerant Quantum Computers (FTQC)

Learn more

- Qiskit Global Summer School 2021 on Quantum Machine Learning

Quantum Computing and GenAI

GenAI as a tool for the Quantum Developer

E.g.: Qiskit Code Assistant

Train LLM s in support of Quantum Computing Research

E.g. Ansatz in VQC for Chemistry

GenAI solution generating input for a Quantum Algorithm

E.g.: GenAI transforms speech into Boolean Expression ans input for a Boolean SAT Problem, solved with Grover Algorithm

Quantum Computing for improving LLM practices

- Faster training
- Real Time Learning
- Save resources
- More complex models

Continuous Research

Qiskit | Ecosystem

Qiskit Machine Learning 0.7.2

[Submitted on 9 Sep 2021]

Quantum Machine Learning for Finance

Marco Pistoia, Syed Farhan Ahmad, Akshay Ajagekar, Alexander Buts, Shouvanik Chakrabarti, Dylan Herman, Shaohan Hu, Andrew Jena, Pierre Minssen, Pradeep Niroula, Arthur Rattew, Yue Sun, Romina Yalovetzky

[Submitted on 28 Nov 2016 (v1), last revised 10 May 2018 (this version, v2)]

Quantum Machine Learning

Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, Seth Lloyd

Challenges and Opportunities in Quantum Machine Learning

M. Cerezo, Guillaume Verdon, Hsin-Yuan Huang, Lukasz Cincio, Patrick J. Coles

(Submitted on 16 Mar 2023)

[Submitted on 10 Sep 2014]

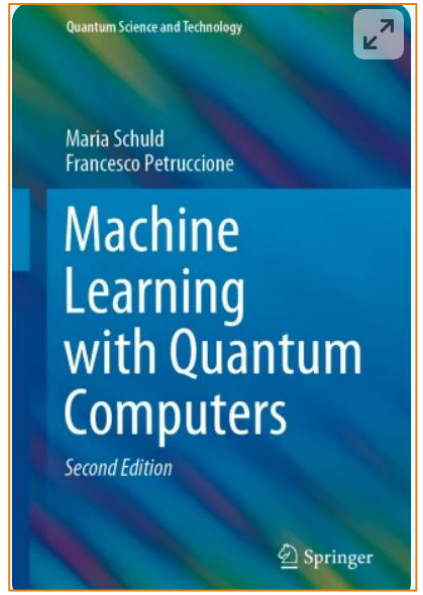
An introduction to quantum machine learning

M. Schuld, I. Sinayskiy, F. Petruccione

[Submitted on 4 Jan 2023 (v1), last revised 6 Mar 2023 (this version, v2)]

Quantum Machine Learning: from physics to software engineering

Alexey Melnikov, Mohammad Kordzanganeh, Alexander Alodjants, Ray-Kuang Lee



[Submitted on 21 Jan 2024 (v1), last revised 31 Mar 2024 (this version, v2)]

A comprehensive review of Quantum Machine Learning: from NISQ to Fault Tolerance

Yunfei Wang, Junyu Liu

Applications of Quantum Machine Learning for Quantitative Finance

Piotr Mironowicz, Akshata Shenoy H., Antonio Mandarino, A. Ege Yilmaz, Thomas Ankenbrand

Top References

- M. Schuld, F. Petruccione: Machine Learning with Quantum Computers
- Stefan Woener et al.: The power of quantum neural networks: https://arxiv.org/abs/2011.00027v1
- Krista Teme et al.: A rigorous and robust quantum speed-up in supervised machine learning: https://arxiv.org/abs/2010.02174
- Stefani Mangini et al.: Quantum computing models for artificial neural networks: https://arxiv.org/abs/2102.03879
- M. Schuld, N. Killoran: Quantum machine learning in feature Hilbert spaces: https://arxiv.org/abs/1803.07128
- Park et all.: Practical application improvement to Quantum SVM: theory to practice: https://arxiv.org/abs/2012.07725
- S. Lloyd et al.: Quantum algorithms for supervised and unsupervised machine learning: https://arxiv.org/abs/1307.0411
- V. Havlicek et al.: Supervised learning with quantum-enhanced feature spaces: Nature 567, 209 (2019)
- IBM Research YT, Almaden: Covariant Quantum Kernels for Data with group Structure: https://arxiv.org/abs/2105.03406
- Qiskit Machine Learning Ecosystem: https://qiskit-community.github.io/qiskit-machine-learning/migration/01_migration_guide_0.5.html

Quantum Machine Learning

End of Chapter 8