Università del Salento
CdLM in Fisica

An Introduction to Quantum Machine Learning

Andrea Zingarofalo

Outline

Basic references


K. P. Murphy.
Probabilistic Machine Learning: An introduction.
MIT Press, 2022.


M. A. Nielsen and I. L. Chuang.
Quantum Computation and Quantum Information.
Cambridge University Press, 2000.

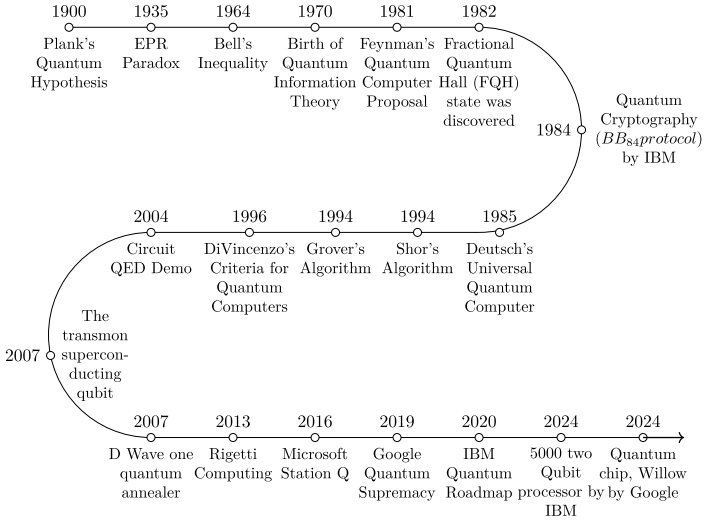

M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini.
Parameterized quantum circuits as machine learning models.
Quantum Science and Technology, 4(4):043001, nov 2019.


M. Schuld and F. Petruccione.
Machine Learning with Quantum Computers.
Springer International Publishing, 2021.


C. Trahan, M. Loveland, and S. Dent.
Quantum Physics-Informed Neural Networks.
Entropy, 26(8):649, August 2024.

The Quantum Computing Era

Timeline of Quantum Computing



What Are Quantum Computers Good For?

- **Quantum computers** exploit **quantum coherence** and **entanglement** to perform certain computations beyond the capabilities of classical machines.
- They promise significant advantages in problems involving:
  - Quantum simulation (e.g., molecules, materials)
  - Combinatorial and global optimization
  - Factoring and cryptography
  - High-dimensional data processing
- **Quantum Machine Learning (QML)** combines quantum computing with machine learning to:
  - Speed up learning algorithms by leveraging quantum parallelism
  - Represent data and models more compactly via quantum states
  - Potentially solve learning tasks more efficiently
- While still in early stages, QML could lead to breakthroughs in areas such as drug discovery, finance, pattern recognition, and AI.

Elements of Machine Learning

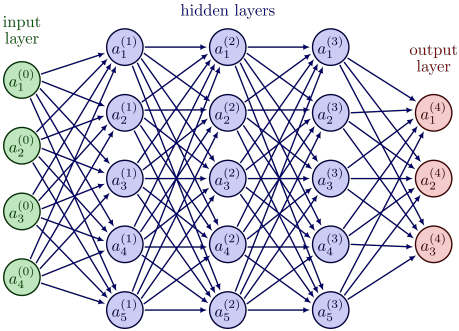
Definition (Tom Mitchell)

A computer program is said to ***learn*** *from experience E with respect to some class of tasks T, and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

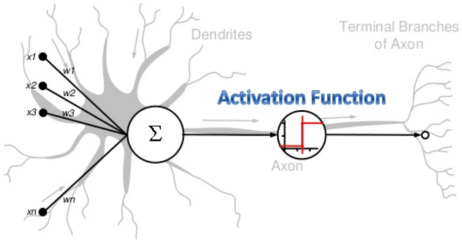
One distinguish many different kinds of learning tasks:

- **Supervised learning**
- **Unsupervised learning**
- **Reinforcement learning**.

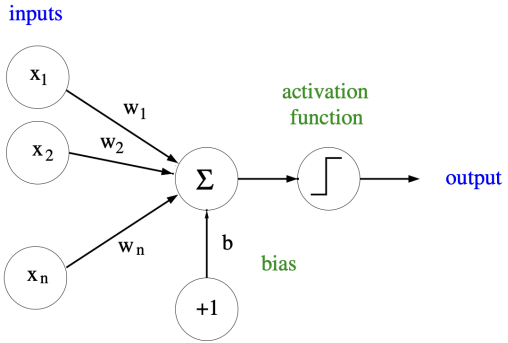
- The task T is to learn a mapping f from inputs $\boldsymbol{x} \in X$ to outputs $\boldsymbol{y} \in Y$.
- The inputs $\boldsymbol{x}$ are also called the **features**, **covariates**, or **predictors**.
- In this case, $X = \mathbb{R}^D$, where D is the number of input features. The output $\boldsymbol{y}$ is also known as the **label**, **target**, or **response**.
- The experience E is given as a set of N input-output pairs $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$, known as the **training set**. (N is called the **sample size**).
- The performance measure P depends on the type of output we are predicting.
- Use cases: classification, regression, object detection, natural language processing, image and video analysis, speech recognition, ...



- Nodes are connected only from one layer to the next and all possible connections are present.
- Size of input and output layers are fixed by the problem.
- The parameters to learn are the weights of the connection.



Neuron



Artificial Neuron

- How do you get the right weights? By **training** your network.
- The goal of training is to minimize a **loss function**, as a function of the weights

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{b}) = \frac{1}{2} \sum_i ||\boldsymbol{y}(\boldsymbol{x}_i) - \hat{\boldsymbol{y}}(\boldsymbol{x}_i)||^2$$

- The loss function measures how well the neural network is performing on a given dataset.

For a MLP:

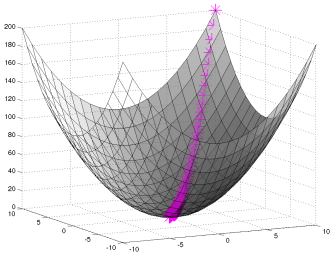
1. Start with random weights
2. Compute the prediction for a given input $\boldsymbol{x}$ and check the difference with target $\boldsymbol{y}$ and the loss.
3. Estimate an update for the weights that reduce the loss
4. Iterate from (2), repeating for all samples of the training dataset.

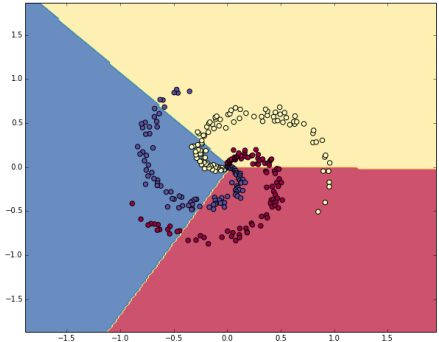
Gradient Descent

- One minimizes the loss function as a function of weights by using the **Gradient Descent method**.
- The gradient of the loss function $\mathcal{L}(\boldsymbol{w}, \boldsymbol{b})$ with respect to the parameters indicates the direction of steepest descent.
- By taking small steps in the opposite direction of the gradient, the algorithm converges towards the optimal set of parameters that minimize the loss function:

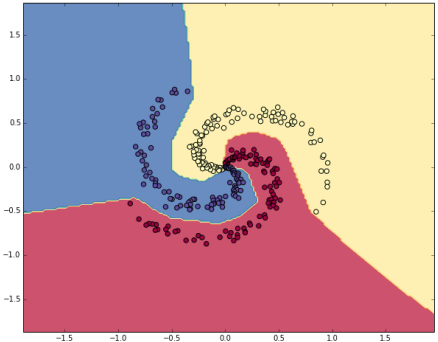
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{b}),$$

  where η is the **learning rate**.
- The gradient is computed by the **backpropagation algorithm**.
- Variants: Stochastic Gradient Descent, Mini-batch Gradient Descent, ...





Linear Classifier



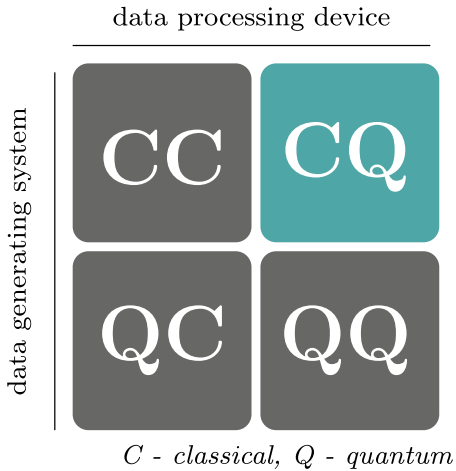
Neural Network Classifier

From `https://cs231n.github.io/neural-networks-case-study/`.

Quantum Machine Learning

Classical vs Quantum Machine Learning

- Classical ML models are able to recognise the statistical patterns in the data and produce the data that possess the same statistical patterns.
- If a small quantum processor can produce some statistical patterns that are difficult for a classical computer to perform computationally, they can perhaps also recognise patterns that are computationally difficult for a classical computer to find.
- This whole observation is the basis for the hope of answering whether an efficient quantum algorithm can be found for ML.
- ML relies heavily on probabilities. This is also what makes quantum systems a good candidate for ML applications.
- Due their strength in handling vast quantities of data and parallel processing, QC can aid in ML tasks.


data processing device
data generating system
CC
CQ
QC
QQ
C - classical, Q - quantum

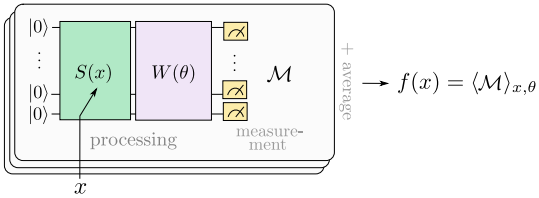
Variational Circuits as Machine Learning Models

- Quantum computing is in the **NISQ era**.
- To perform even the smallest of empirical benchmarks on simulators and early-stage quantum hardware, QML algorithms underlie strict design limitations.
- Large routines are not simulable on classical hardware, and quickly become drowned in noise on real quantum devices.
- Instead of outsourcing the entire ML pipeline to a QC, **hybrid quantum-classical algorithms** consider brief quantum computations as parts of more complex classical ones.
- A second strategy is to investigate generic **quantum models** as alternatives to classical ones.
- A perfect candidate for these requirements are **variational quantum circuits** (VQC), or **quantum neural networks** (QNN).

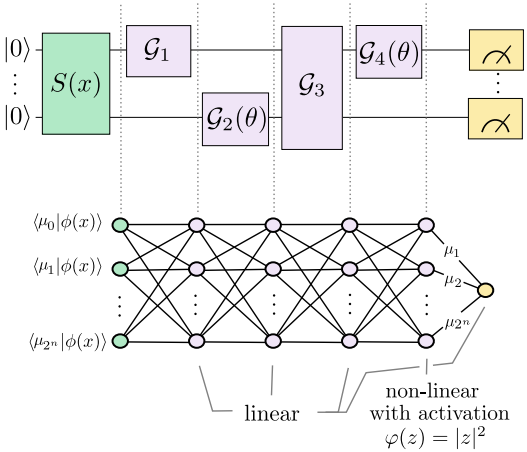
- Let $U(\boldsymbol{x}, \theta)$, with $\boldsymbol{x} \in X$, $\theta \in \mathbb{R}^K$ be a parametrized quantum circuit, and $\mathcal{M}$ a quantum observable.
- Let $|\psi(\boldsymbol{x}, \theta)\rangle \equiv U(\boldsymbol{x}, \theta)\,|0\rangle$. The function

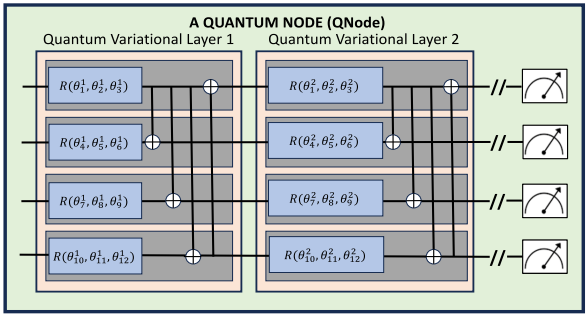
$$f_\theta(\boldsymbol{x}) := \langle\psi(\boldsymbol{x}, \theta)|\,\mathcal{M}\,|\psi(\boldsymbol{x}, \theta)\rangle\,,$$

defines a **deterministic variational quantum model**.



- Both VQCs and NNs can be thought of as layers of connected computational units controlled by adjustable parameters.
- This has led some authors to refer to VQCs as **quantum neural networks** but...
- First, quantum circuit operations are unitary and therefore linear; classical NNs use nonlinear activation functions.
- Second, it is impossible to access the quantum state at intermediate points during computation.
- This implies that executing the VQC cannot be seen as performing the forward pass of a neural network.
- Moreover, it is difficult to conceive a circuit learning algorithm that truly resembles backpropagation.


$|0\rangle$
$|0\rangle$
$S(x)$
$\mathcal{G}_1$
$\mathcal{G}_2(\theta)$
$\mathcal{G}_3$
$\mathcal{G}_4(\theta)$
$\langle\mu_0|\phi(x)\rangle$
$\langle\mu_1|\phi(x)\rangle$
$\langle\mu_{2^n}|\phi(x)\rangle$
μ_1
μ_2
μ_{2^n}
linear
non-linear
with activation
$\varphi(z) = |z|^2$



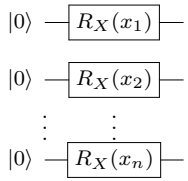
Hardware-efficient circuits alternate layers of native entangling gates and single-qubit rotations.

Angle (Rotation) Encoding

Goal: Encode classical data $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ into a quantum state using single-qubit rotations.

Idea: Use the data values as rotation angles for quantum gates acting on qubits.

Circuit Representation:



- Each classical input x_i controls a rotation gate.
- Common choices: $R_X(x_i)$, $R_Y(x_i)$, or $R_Z(x_i)$.
- Transforms each qubit into:
  $|\psi_i\rangle = \cos(x_i/2)\,|0\rangle + \sin(x_i/2)\,|1\rangle$
- Resulting quantum state: $|\psi(\boldsymbol{x})\rangle = \bigotimes_{i=1}^{n} |\psi_i\rangle$

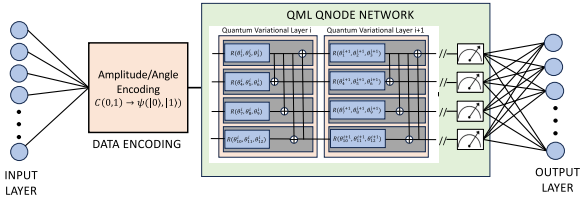
Pros:

- Simple and hardware-efficient
- Maintains normalization

- We want to optimize the parameters of a VQC in order to minimize a classical loss function.
- Quantum circuits do not allow direct computation of derivatives as in classical software.
- The **parameter-shift rule** provides an analytical method to estimate gradients using finite shifts.
- For a single parameter θ_i, the rule is:

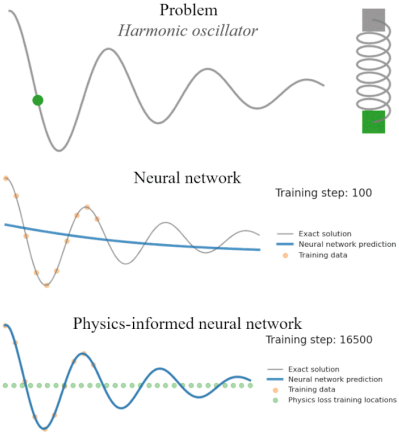
$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_i} = \frac{1}{2}\left[\mathcal{L}(\boldsymbol{\theta}_i^+) - \mathcal{L}(\boldsymbol{\theta}_i^-)\right], \qquad \boldsymbol{\theta}_i^{\pm} = \boldsymbol{\theta} \pm \frac{\pi}{2}\boldsymbol{e}_i.$$

- This means we must evaluate the quantum circuit twice per parameter, with positive and negative shifts.
- In **hybrid quantum-classical models** automatic differentiation can still be used.
- This allows training QNNs similarly to classical NNs.


QML QNODE NETWORK
Quantum Variational Layer i
Quantum Variational Layer i+1
Amplitude/Angle Encoding
$C(0,1) \rightarrow \psi(|0\rangle, |1\rangle)$
DATA ENCODING
INPUT LAYER
OUTPUT LAYER
Amplitude/Angle Encoding
$C(0,1) \rightarrow \psi(|0\rangle, |1\rangle)$
DATA ENCODING 1
QNN 1
Amplitude/Angle Encoding
$C(0,1) \rightarrow \psi(|0\rangle, |1\rangle)$
DATA ENCODING 2
QNN 2
INPUT LAYER
OUTPUT LAYER

Example: Quantum Physics-Informed Neural Networks I

- **Quantum Physics-Informed Neural Networks** (QPINNs) are a class of NNs that leverage QC principles to solve PDEs.
- While NN function approximation/regression has shown to be a valuable tool for data-rich scenarios, data-only models are not constrained by physics and can perform poorly in sparse- or no-data regions.
- To build reliable physical models in these regions, **Physics-Informed NNs** (PINNs) can be used.



Example: Quantum Physics-Informed Neural Networks II

- PINNs supplement the data-driven loss with **PDE residuals** representing physical conservation principles.
- Consider a time-dependent PDE of the form

$$u_t + \mathcal{N}[u] = 0, \qquad t \in [0, T], \quad x \in \Omega$$

subject to initial and boundary conditions

$$u(0, x) = g(x), \quad x \in \Omega, \qquad \mathcal{B}[u] = u_0, \quad t \in [0, T], \quad x \in \partial\Omega$$

where $\mathcal{B}[\cdot]$ is a boundary operator corresponding to the equation's boundary conditions (Dirichlet, Neumann, etc.).

Example: Quantum Physics-Informed Neural Networks III

- If the NN solution is given by $u_\theta(t, x)$, where θ denotes the tunable parameters of the network, then the parameterized solution of the PDE in residual form is given by

$$\mathcal{R}_\theta(t, x) \coloneqq \frac{\partial u_\theta}{\partial t} + \mathcal{N}[u_\theta],$$

and the PINN is trained on the composite loss function

$$\mathcal{L}_\theta \coloneqq \mathcal{L}_{\text{IC}}(\theta) + \mathcal{L}_{\text{BC}}(\theta) + \mathcal{L}_r(\theta),$$

where $\mathcal{L}_{\text{IC}}$ and $\mathcal{L}_{\text{BC}}$ are the losses associated with the ICs and BCs, respectively, and $\mathcal{L}_r$ is the residual loss given by

$$\mathcal{L}_r(\theta) \coloneqq \frac{1}{N_r} \sum_{i=1}^{N_r} ||\mathcal{R}_\theta(t_i, x_i)||^2,$$

- Recently, QPINNs have been shown capable of increasing model accuracy when compared to purely classical NNs in computational fluid dynamics.