Secrets revealed in this session:

To explore the principles of quantum machine learning models, their parameterisation and optimisation

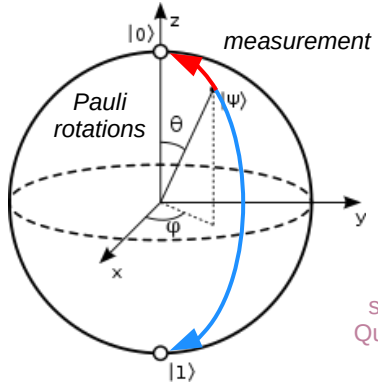
QML workshop
QML team
QML and aims
Parameterised circuits
Data encoding
- *Angle encoding*
- *The good, the bad and the ugly*

State measurement
Quantum model training
Parameters optimisation
Model geometry and gradients
QML readings
PennyLane demo
Summary

Quantum Machine Learning

Introduction

Jacob L. Cybulski
Enquanted, Melbourne, Australia



We will assume
some knowledge of
Quantum Computing
ML and Python

Our QML Team

Sebastian Zając

Assistant Professor
SGH Warsaw School of Economics
LinkedIn

Dr Paweł Gora

Coordinator of QPoland *and*
CEO of Fundacja Quantum AI
LinkedIn

Jacob Cybulski
Founder, Researcher, Consultant
Enquanted

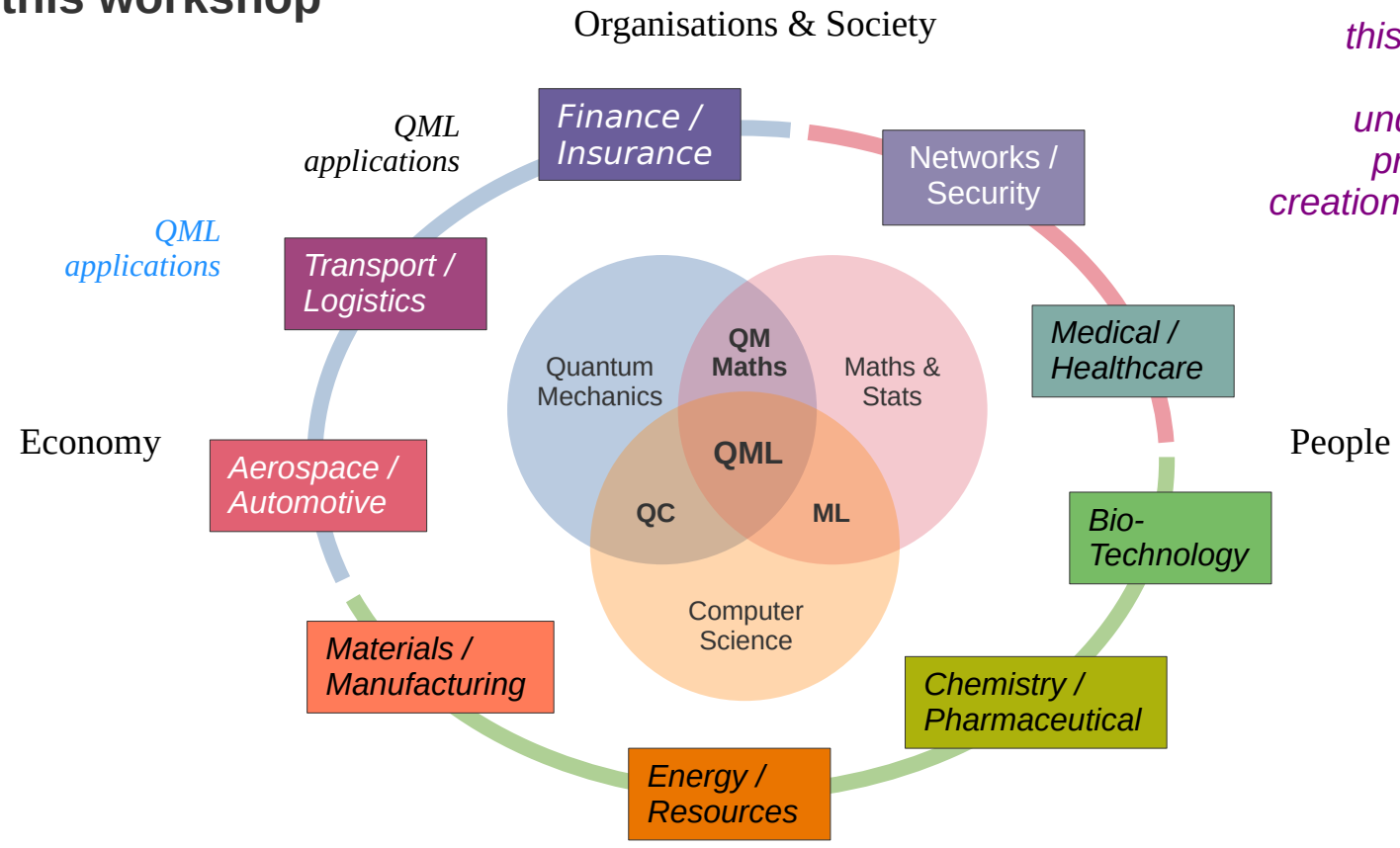
and
Honorary A/Prof
In Quantum Computing
Deakin University
LinkedIn

Tomek Rybotycki

QML Researcher
IBS PAS, NCAC PAS, CEAI AGH, KPLabs
LinkedIn

Jacob L. Cybulski, Quantum Business Series (Deakin, RMIT, ACS, Warsaw School of Economics)
Jacob L. Cybulski, Quantum Computing Intro Series (SheQuantum, Assoc of Polish Profs in Australia)
2021-2025

Quantum ML
aims of this workshop



this workshop aims at developing the understanding of and practical skills in the creation and application of QML models

Aims for quantum computing people: Learn about ML in QML

Aims for machine learning people: Learn about Q in QML

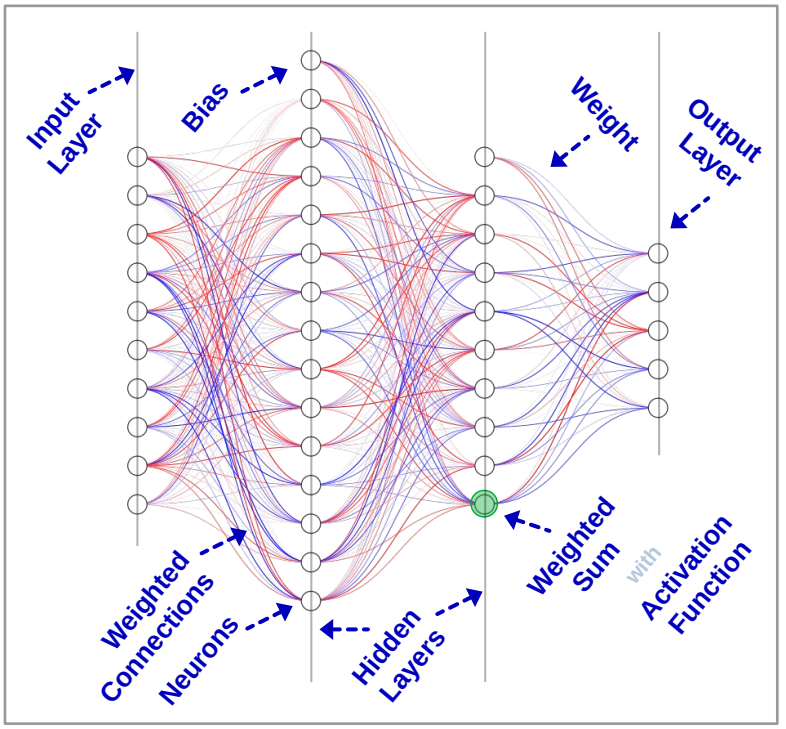
ML/QML objective and categories

Objective	Category	Core Goal	Common Quantum Counterpart (QML)
Classification	Supervised	Assign a discrete label or category	Quantum Support Vector Machines (QSVM)
Estimation	Supervised	Predict a continuous numerical value	Variational Quantum Regressors (VQR)
Clustering	Unsupervised	Find hidden groupings in data	Quantum K-Means / Quantum Clustering
Association	Unsupervised	Discover rules and relationships	Quantum Association Rule Mining
Dimension Reduction	Unsupervised	Compress data while retaining essence	Quantum PCA / AE (QPCA / QAE)
Ranking	Hybrid/Supervised	Determine relative order or relevance	Quantum-enhanced Recommendation
Anomaly Detection	Unsupervised	Identify outliers or abnormal data	Quantum Kernel Anomaly Detection
Synthesis / Generation	Generative	Create new, realistic synthetic data	Quantum Circuit Born Machines (QCBM)
Transformation / Mapping	Reconstruction	Map data from one form/state to another	Quantum Autoencoders (QAE / DQAE)
Action Optimization	Reinforcement	Learn a policy to maximize rewards	Quantum Reinforcement Learning (QRL)

In this workshop, we will only discuss quantum classifiers, estimators and autoencoders.

Multilayer Perceptron (MLP)

A class of Neural Network (NN) models



The cost of each NN model is a point in a multi-D space of weights

w1 x **w2** x ... X **MAE**

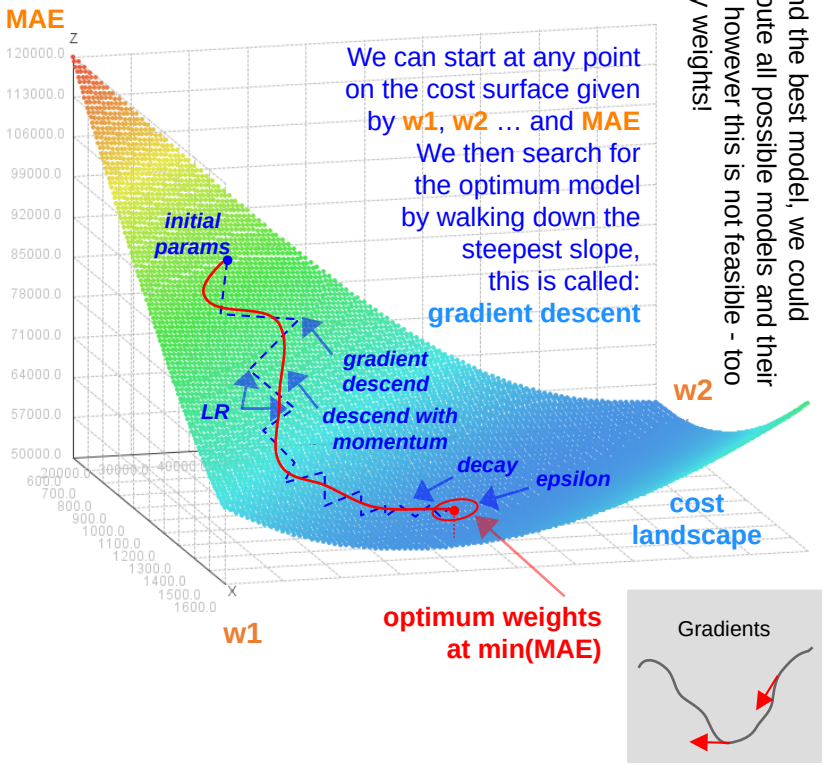
All such points form a “cost” surface.

The shape of such a surface we call the **cost landscape**.

When a model has many weights, the cost surface is multi-dimensional and called a manifold.

Now we will look for the best NN model, i.e. the model which generates the lowest MAE

To find the best model, we could compute all possible models and their cost, however this is not feasible - too many weights!



- MLP is capable of learning any “smooth” function by associating inputs with outputs
- Other NNs: CNN, AE, GAN, LSTM + QNNs

The optimizer controls this process via hyper-parameters, i.e. parameters of the gradient descent itself:

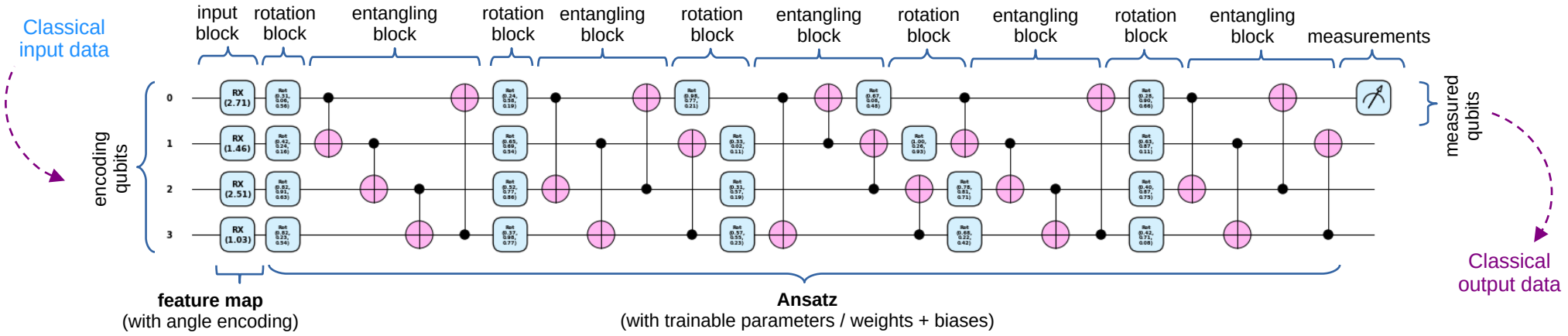
learning rate
momentum
decay
epsilon

By using gradient descent, the optimum cost (and thus the model), was found at:

A=1060 (Lot_Frontage)
B=90000 (Intercept)
MAE=53473.097 (Error)

Variational Quantum Models
= *Parameterised Quantum Circuits*

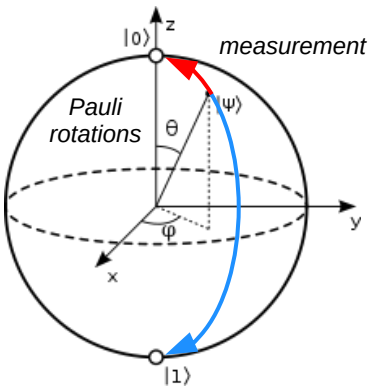
Variational quantum circuits are not executable! They must first be instantiated, i.e. all of their input and weight parameters must be assigned values! Ansatz parameters are trainable.



We can create a “variational” model = a circuit template with parameterised gates, e.g. P(a), Ry(a) or Rz(a), each allowing rotation of a qubit state in x, y or z axis (as per Bloch sphere).

Typically, but now always, the circuit consists of three blocks:

- a feature map (input)
- an ansatz (processing)
- measurements (output)



Classical input data is encoded into the feature map’s parameters, setting the model’s initial quantum state.

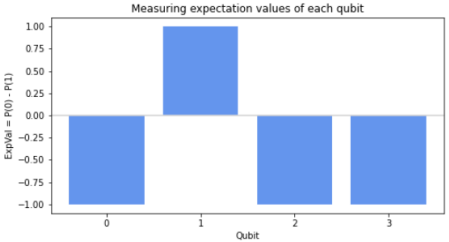
The quantum state is then altered by an ansatz, which consists of parameterised gates (operations), which alter the circuit state. Ansatz parameters are trainable. Qubits and parameters increase the model dimensionality.

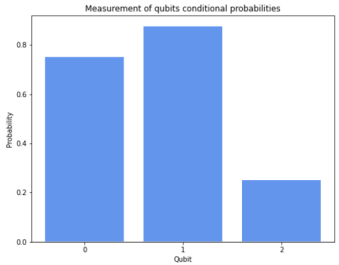
The quantum state of the circuit is then measured and interpreted as the model’s output in classical data form, e.g. as binary values, integer or real value, a single event’s probability or the probability distribution.

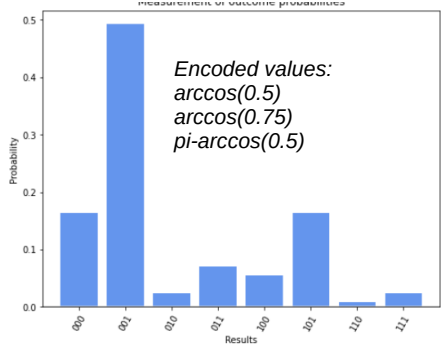
Data Encoding

Maria Schuld and Francesco Petruccione
Machine Learning with Quantum Computers. 2nd ed. Springer, 2021.

Many encoding methods, e.g. basis, angle, amplitude, QRAM, ...



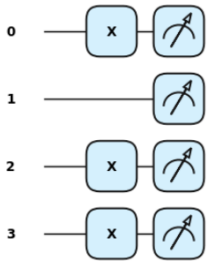




Encoded values:
arccos(0.5)
arccos(0.75)
pi-arccos(0.5)

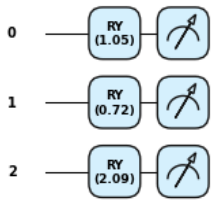
Encoding can be repeated across the circuit, which is called data reuploading

Basis encoding



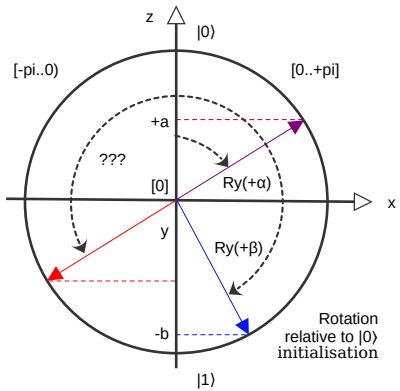
The simplest data encoding and very popular, however, little data can be encoded at a time, you can encode only binary values per each qubit.

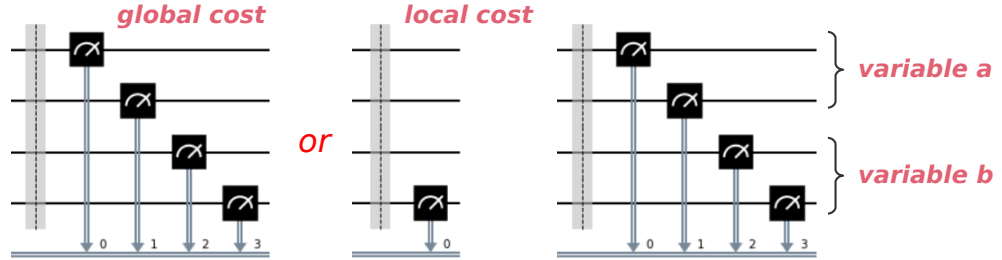
Angle encoding

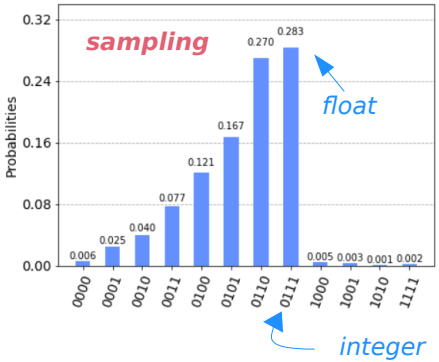


One of the most flexible and very effective, you can encode floating point numbers.

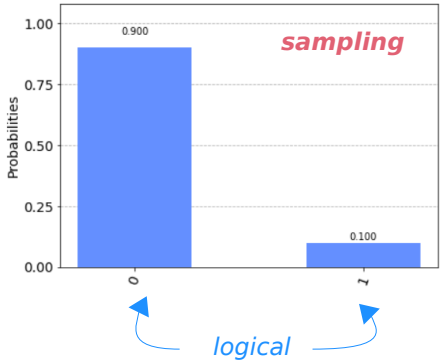
What you encode depends on your intention!

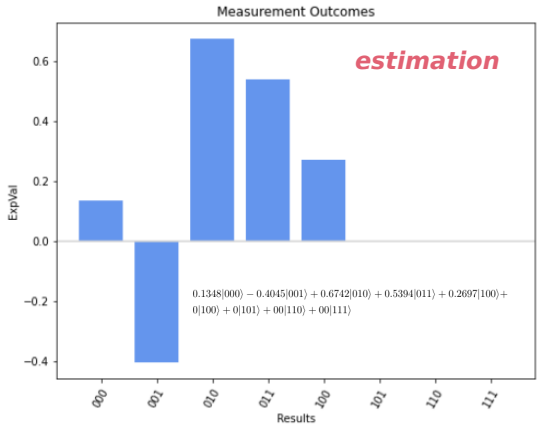






or





Measurements must be repeated, collected and then can be interpreted in many different ways

It is also possible to measure mid-circuit, however, beware as the circuit is no longer unitary and not reversible!

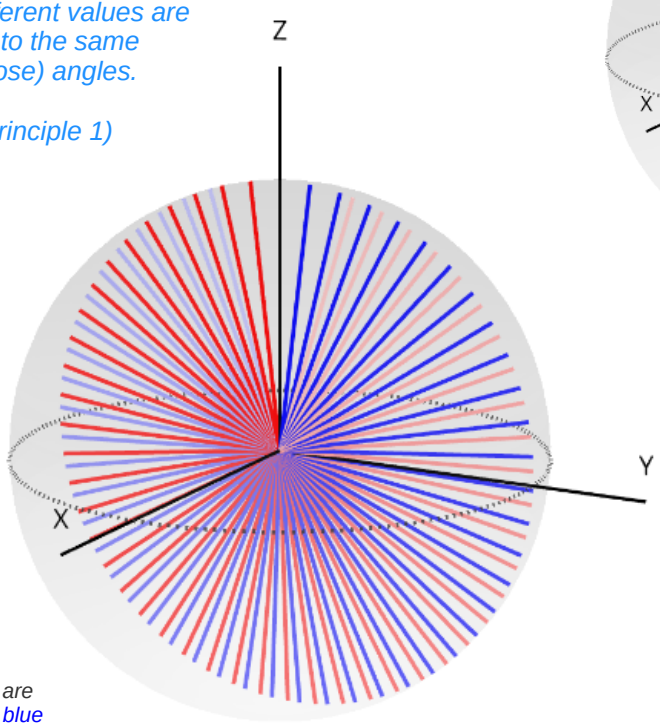
State Measurement

Angle encoding

The Good, the Bad and the Ugly

This example shows encoding values wrapping around the Bloch sphere (e.g. -6 to 6), so that different values are mapped into the same (or very close) angles.

(violates principle 1)

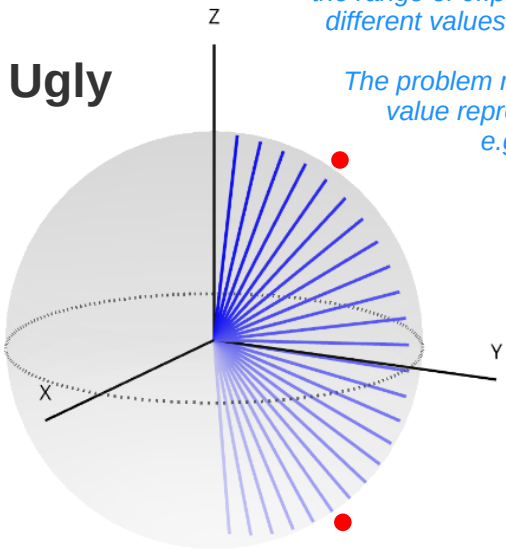


Positive values are represented by blue vectors and negative values by red vectors.

This example shows encoding values wrapping around the range of 0..pi of the Bloch sphere, ensuring that different values are represented by unique angles.

The problem may arise if we may have the same value represented by two distinct amplitudes, e.g. values of sin(x) represented by x

(violates principle 2)

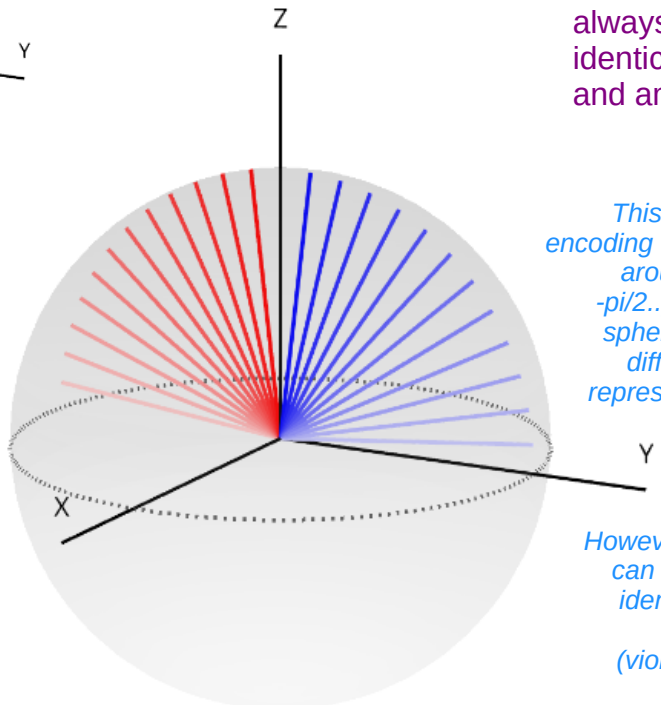


Special care must be taken in those cases where the values form some kind of symmetry or repetition. e.g. in a case values x and 2pi+x effectively represent the same angular position and their encoding should also be identical.

The red dot represents the same value which on two occasions maps onto two different angles and thus two amplitudes.

Two principles of quantum data encoding:

1) distinct data values should map onto distinct angles and amplitudes
2) the same data values should always map into identical angles and amplitudes



This example shows encoding values wrapping around the range of -pi/2..pi/2 of the Bloch sphere, ensuring that different values are represented by unique angles.

However, some angles can be measured as identical amplitudes

(violates principle 1)

Ansatz design & training

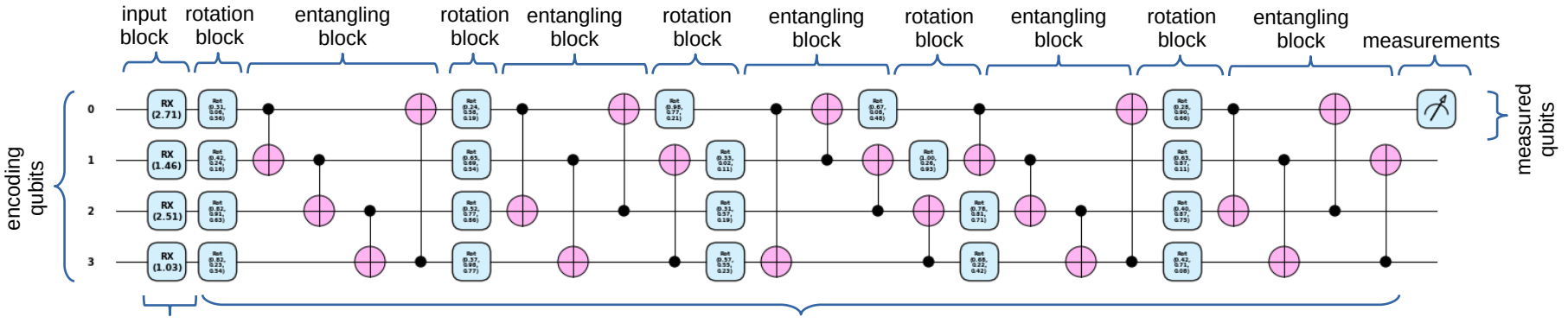
A simple quantum classifier ...

Beware that **adding qubits adds parameters and entanglements!**

The number of states represented by the circuit **grows exponentially** with the number of qubits!

Beware that **adding 1 measurement doubles the number of outcomes!**

So... having ***n*** measurements leads to 2^n outcomes



feature map
(with angle encoding)

Ansatz
(with trainable parameters / weights + biases)

*Data **reuploading** across circuit's width and depth*

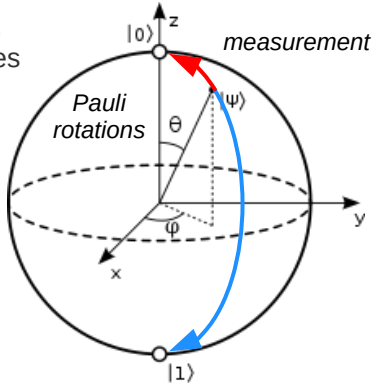
feature maps vary in:
structure and function

ansatze vary in:

- width (qubits #)
- depth (layers #)
- dimensions (param #)
- structure (e.g. funnelling)
- entangling (circular, linear, sca)

ansatz layers consist of:
rotation blocks and entangling blocks of R(x, y, z) and CNOT gates
(rotation) (entanglement)

rotation gates
alter qubit states around x, y, z axes



To execute a circuit we just apply it to input data and the optimum parameters

different cost functions:
R2, MAE, MSE, Huber, Poisson, cross-entropy, hinge-embedding, Kullback-Leibner divergence

different optimisers:
gradient based (Adam, NAdam and SPSA)
linear approximation methods (COBYLA)
non-linear approximation methods (BFGS)
quantum natural gradient optimiser (QNG)

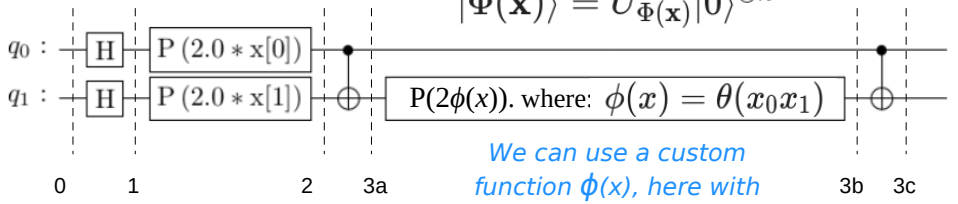
circuit execution on:
simulators (CPUs), accelerators (GPUs) and real quantum machines (QPUs)

What is the Hilbert space?

Example: Consider the following ZZ feature map

The story of the Hilbert space, the circuit states, the state vector paths and manifolds – and this is just the beginning...

$$|\Phi(\mathbf{x})\rangle = U_{\Phi(\mathbf{x})}|0\rangle^{\otimes n}$$



We can use a custom function ϕ(x), here with θ trainable parameter

the final state:

$$|\Phi(\mathbf{x})\rangle = \frac{1}{2}\begin{pmatrix} 1 \\ e^{i(2x_1+2\phi(x))} \\ e^{i(2x_0+2\phi(x))} \\ e^{i(2x_0+2x_1)} \end{pmatrix} \begin{matrix} \leftarrow |00\rangle \\ \leftarrow |01\rangle \\ \leftarrow |10\rangle \\ \leftarrow |11\rangle \end{matrix}$$

Step 0: Initial State

$$|\psi_0\rangle = |0\rangle \otimes |0\rangle = |00\rangle$$

Step 1: Superposition (Hadamard Gates $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$)

$$|\psi_1\rangle = (H \otimes H)|00\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

Step 2: Phase Gates (adds phase when qubit is in state |1>)

$$|\psi_2\rangle = \frac{1}{2}\left(|00\rangle + e^{i2x_1}|01\rangle + e^{i2x_0}|10\rangle + e^{i(2x_0+2x_1)}|11\rangle\right)$$

Step 3: Entanglement (CNOT – Phase – CNOT)

3a. First CNOT (Control q0, Target q1)

$$|\psi_{3a}\rangle = \frac{1}{2}\left(|00\rangle + e^{i2x_1}|01\rangle + e^{i2x_0}|11\rangle + e^{i(2x_0+2x_1)}|10\rangle\right)$$

3b. Middle Phase Gate P(2ϕ) on q1

$$|\psi_{3b}\rangle = \frac{1}{2}\left(|00\rangle + e^{i2x_1}e^{i2\phi}|01\rangle + e^{i2x_0}e^{i2\phi}|11\rangle + e^{i(2x_0+2x_1)}|10\rangle\right)$$

3c. Second CNOT (Control q0, Target q1)

$$|\psi_{3c}\rangle = \frac{1}{2}\left(|00\rangle + e^{i(2x_1+2\phi)}|01\rangle + e^{i(2x_0+2\phi)}|10\rangle + e^{i(2x_0+2x_1)}|11\rangle\right)$$

The Hilbert space is a coordinate system and a space of directions that the state of your model could take during its execution. As it executes it leaves a path, a “trail” its state vector follows. Because of entanglement, the state vector is a “complex” arrow and the trail becomes “knotted”. Because of Heisenberg uncertainty and noise, each time the model executes, it takes a slightly different path. Trainable parameters allow adjusting the direction we take. By varying encoded data and model parameters, all possible paths create a smooth curved surface – a manifold.

The dance of the blind puppeteer

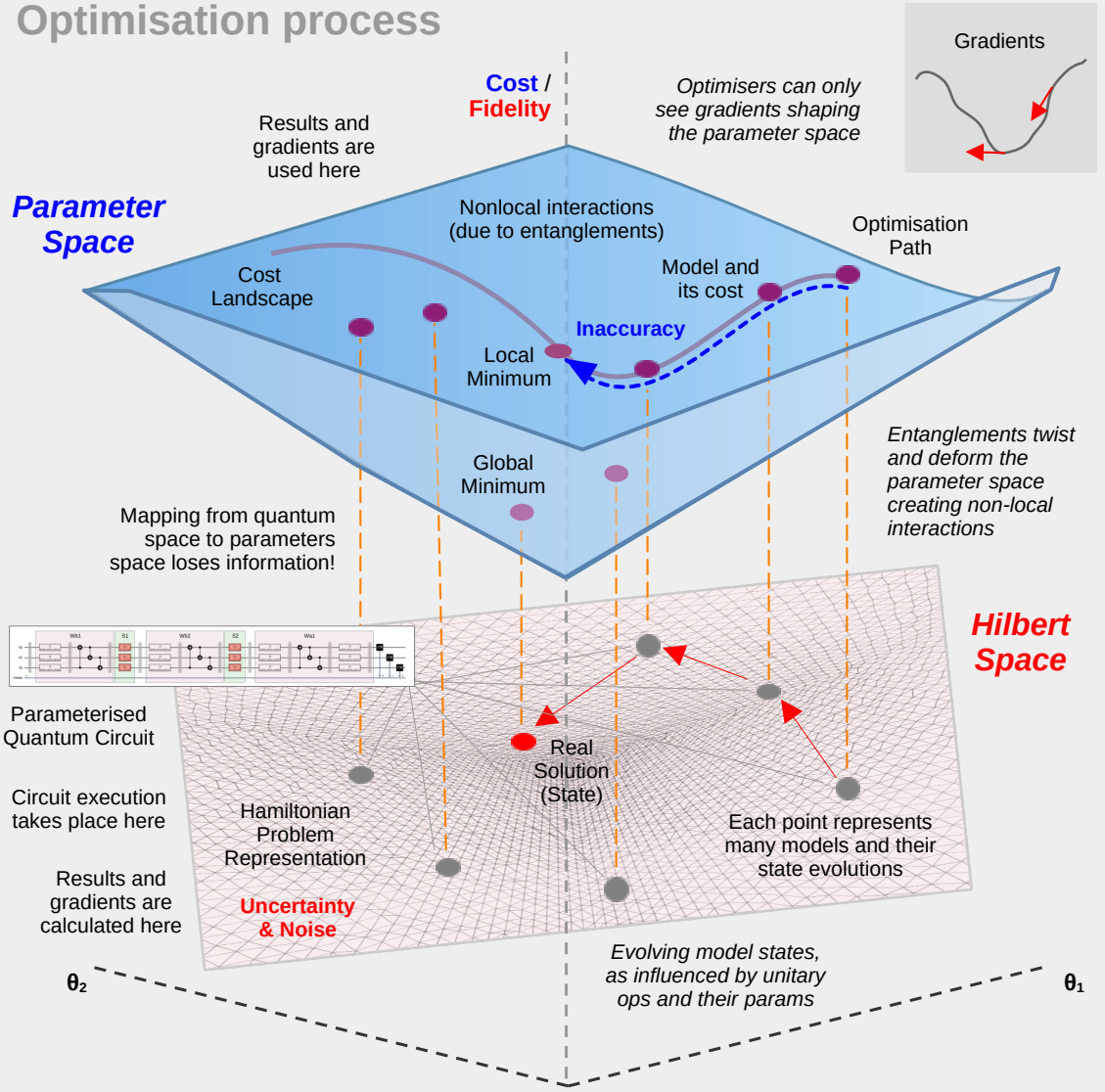
Parameter space
(dim = the number of params) is a classical multi-dim space of trainable gate parameters, which the optimiser navigates

Classical parameters
The optimiser can only manipulate the gate parameters but is not aware of their action on qubits states or their correlations via qubit entanglements.

Measurement
of individual qubits collapses their states, consequently projecting the circuit state onto classical outcomes, the only information reaching the optimiser. In the process we lose much of quantum information (e.g. phase).

Taking the blindfold from the puppeteer's eyes

Optimisation process



Hilbert state space
(dim ≈ $2^{\text{the number of qubits}}$) is the quantum realm where the models and their states evolve in response to unitary operations as defined by the circuit gates

Data encoding
brings in classical data into the Hilbert space as unique and correlated quantum states during the model execution

Entanglements
(CNOTs) create non-separable qubit states, which cause non-local interactions, and which alter the state space geometry, and as a side-effect, also the cost landscape

Circuit layers
determine the evolution of the quantum model's initial state into its final state during the circuit execution

PennyLane Demo

Everything is a function!

PennyLane (PL) …

- Supports *differentiable programming paradigm*
- Integrates seamlessly with the *Python*
- Has a range of operations for *state preparation*, *gates* and *measurements*
- Supports creation of flexible *quantum algorithms*
- Executes on *simulators* and *quantum hardware*
- Supports *error mitigation*
- Extends its *quantum gradients* with those from JAX, PyTorch, Keras, TensorFlow, or NumPy
- Supports *hybrid quantum-classical models*
- Allows training with *hardware-compatible gradients* and *higher-order derivatives*
- Provides numerous quantum models, such as: *QNNs*, *quantum kernels* and *Fourier models*
- Can be extended with models and optimisers from other SDKs, e.g. *PyTorch* and *TensorFlow*

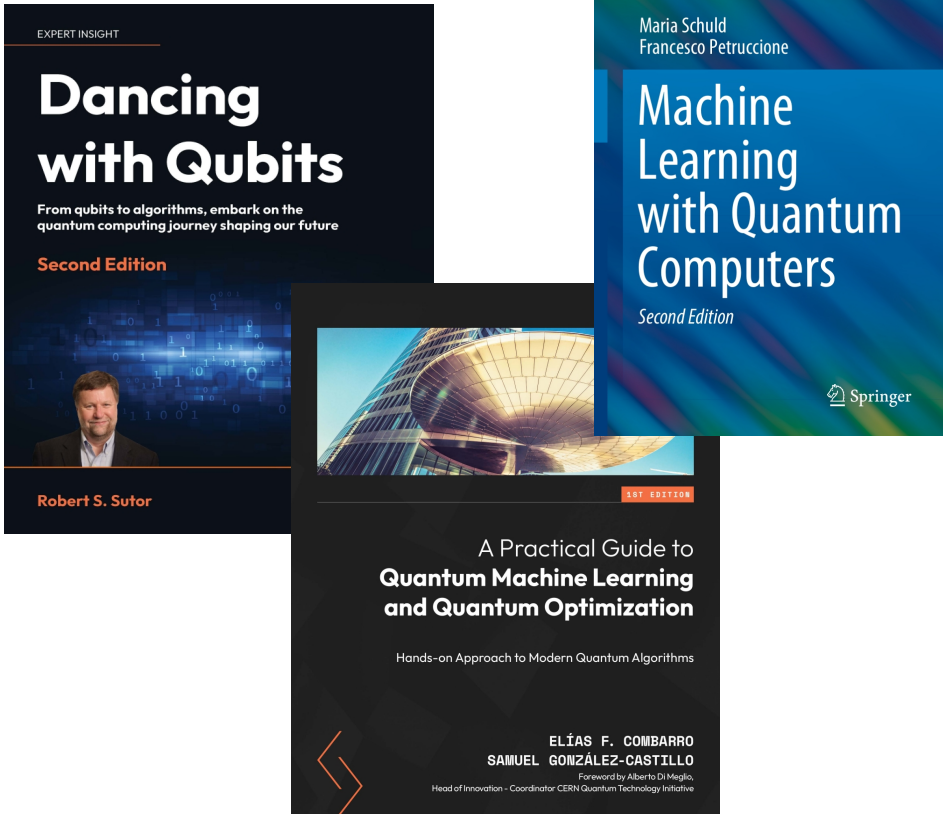
PennyLane Demo:

- Create a simple PL model to fit a simple function
- Learn to initialise model weights
- Explore the impact of ansatz structure on performance
- Create minimalistic quantum models
- Learn the interaction of data encoding and ansatz
- Investigate different types of entangling
- Apply the best solution to more complex data
- Learn about stamina and wisdom in QML development

Key takeaways:

- Plan model development, tests and experiments
- Bad data encoding spoils the bunch!
- Strong entanglement improves the data fit
- More width and depth = the curse of dimensionality
- Carefully consider your quantum model initialisation
- Surprise - a single qubit model still works! (and well)
- More training does not solve the problems
- Data reuploading makes a huge difference!

Recommended reading
on QML



PennyLane: Automatic differentiation of hybrid quantum-classical computations

Ville Bergholm,[1] Josh Izaac,[1] Maria Schuld,[1] Christian Gogolin,[1] M. Sohaib Alam,[2] Shahnawaz Ahmed,[3] Juan Miguel Arrazola,[1] Carsten Blank,[4] Alain Delgado,[1] Soran Jahangiri,[1] Keri McKiernan,[2] Johannes Jakob Meyer,[5] Zeyue Niu,[1] Antal Száva,[1] and Nathan Killoran[1]

[1]*Xanadu, 777 Bay Street, Toronto, Canada*

[2]*Rigetti Computing, 2919 Seventh Street, Berkeley, CA 94710*

[3]*Wallenberg Centre for Quantum Technology, Department of Microtechnology and Nanoscience, Chalmers University of Technology, 412 96 Gothenburg, Sweden*

[4]*data cybernetics, Martin-Kolmsperger-Str 26, 86899 Landsberg, Germany*

[5]*Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany*

14 Feb 2020

Modern applications of machine learning in quantum sciences

Anna Dawid[1,2*], Julian Arnold[3†], Borja Requena[2†], Alexander Gresch[4†], Marcin Płodzień[2], Kaelan Donatella[5], Kim A. Nicoli[6,7], Paolo Stornati[2], Rouven Koch[8], Miriam Büttner[9], Robert Okuła[10,11], Gorka Muñoz–Gil[12], Rodrigo A. Vargas–Hernández[13,14], Alba Cervera-Lierta[15], Juan Carrasquilla[14], Vedran Dunjko[16], Marylou Gabrié[17], Patrick Huembeli[18,19], Evert van Nieuwenburg[16,20], Filippo Vicentini[18], Lei Wang[21,22], Sebastian J. Wetzel[23], Giuseppe Carleo[18], Eliška Greplová[24], Roman Krems[25], Florian Marquardt[26,27], Michał Tomza[1], Maciej Lewenstein[2,28] and Alexandre Dauphin[2*]

1 Faculty of Physics, University of Warsaw, Poland
2 ICFO - Institut de Ciències Fotòniques, The Barcelona Institute of Science and Technology, 08860 Castelldefels (Barcelona), Spain
3 Department of Physics, University of Basel, Switzerland
4 Quantum Technology Research Group, Heinrich-Heine-Universität Düsseldorf, Germany
5 Université de Paris, CNRS, Laboratoire Matériaux et Phénomènes Quantiques, France
6 Machine Learning Group, Technische Universität Berlin, Germany
7 BIFOLD, Berlin Institute for the Foundations of Learning and Data, 10587 Berlin, Germany
8 Department of Applied Physics, Aalto University, Espoo, Finland
9 Institute of Physics, Albert-Ludwig University of Freiburg, Germany
10 International Centre for Theory of Quantum Technologies, University of Gdańsk, Poland
11 Department of Algorithms and System Modeling, Faculty of Electronics, Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Poland
12 Institute for Theoretical Physics, University of Innsbruck, Austria
13 Department of Chemistry, University of Toronto, Canada
14 Vector Institute for Artificial Intelligence, MaRS Centre, Toronto, Canada
15 Barcelona Supercomputing Center, Spain
16 LIACS, Leiden University, The Netherlands
17 CMAP, École Polytechnique, France
18 Institute of Physics, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
19 Menten AI, Inc., Palo Alto, California, United States of America
20 Niels Bohr Institute, Copenhagen, Denmark
21 Beijing National Lab for Condensed Matter Physics and Institute of Physics, Chinese Academy of Sciences, Beijing, China
22 Songshan Lake Materials Laboratory, Dongguan, China
23 Perimeter Institute for Theoretical Physics, Waterloo, Canada
24 Kavli Institute of Nanoscience, Delft University of Technology, NL-2600 GA Delft, The Netherlands
25 Department of Chemistry, University of British Columbia, Vancouver, Canada
26 Max Planck Institute for the Science of Light, Erlangen, Germany
27 Department of Physics, Friedrich-Alexander Universität Erlangen-Nürnberg, Germany
28 ICREA, Pg. Lluís Companys 23, 08010 Barcelona, Spain
† These authors contributed equally.
* Anna.Dawid@fuw.edu.pl, Alexandre.Dauphin@icfo.eu

June 23, 2022

Abstract

In these Lecture Notes, we provide a comprehensive introduction to the most recent advances in the application of machine learning methods in quantum sciences. We cover the use of deep learning and kernel methods in supervised, unsupervised, and reinforcement learning algorithms for phase classification, representation of many-body quantum states, quantum feedback control, and quantum circuits optimization. Moreover, we introduce and discuss more specialized topics such as differentiable programming, generative models, statistical approach to machine learning, and quantum machine learning.

Thank you!

Any questions?

Photos from Unsplash

Enquanted is being somewhere in-between Enchanted and Entangled