

Secrets revealed in this session:

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



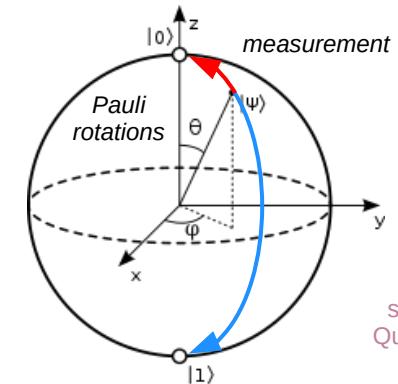
QML workshop
QML team
QML and aims
Parameterised circuits
Data encoding
 Angle encoding
 The good, the bad and the ugly
State measurement
Quantum model training
Parameters optimisation
Model geometry and gradients
QML readings
PennyLane demo
Summary

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 Zajac

Assistant Professor
SGH Warsaw School of
Economics
[LinkedIn](#)



Dr Paweł Gora

Coordinator of QPoland and
CEO of Fundacja Quantum AI
[LinkedIn](#)



Tomek Rybotycki

QML Researcher
SRI PAS, NCAC PAS, CEA AGH, KPLabs
[LinkedIn](#)



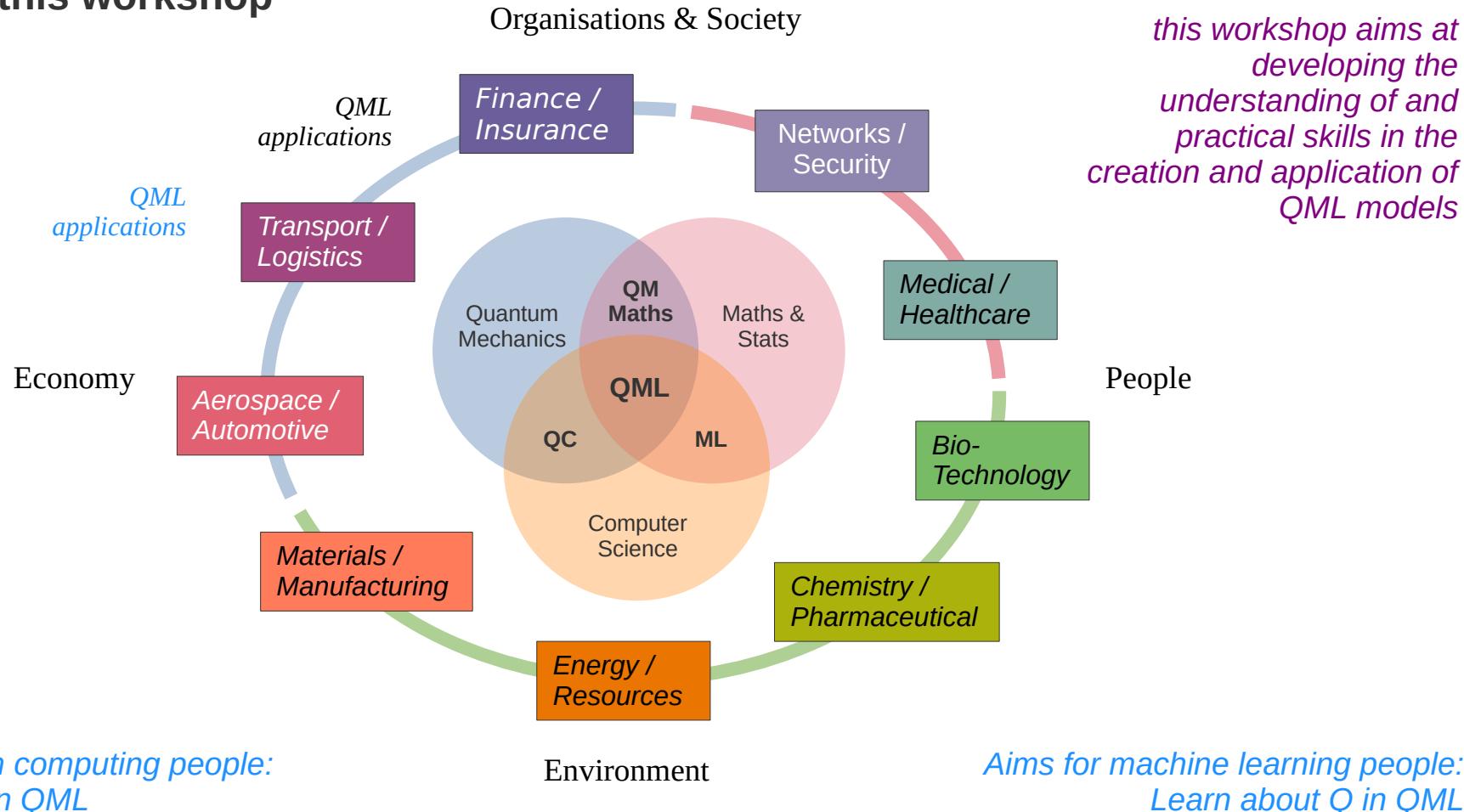
Jacob Cybulski
Founder, Researcher, Consultant
at Enquanted

and
Honorary A/Prof
In Quantum Computing
Deakin University
[LinkedIn](#)

Quantum ML

aims of this workshop

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

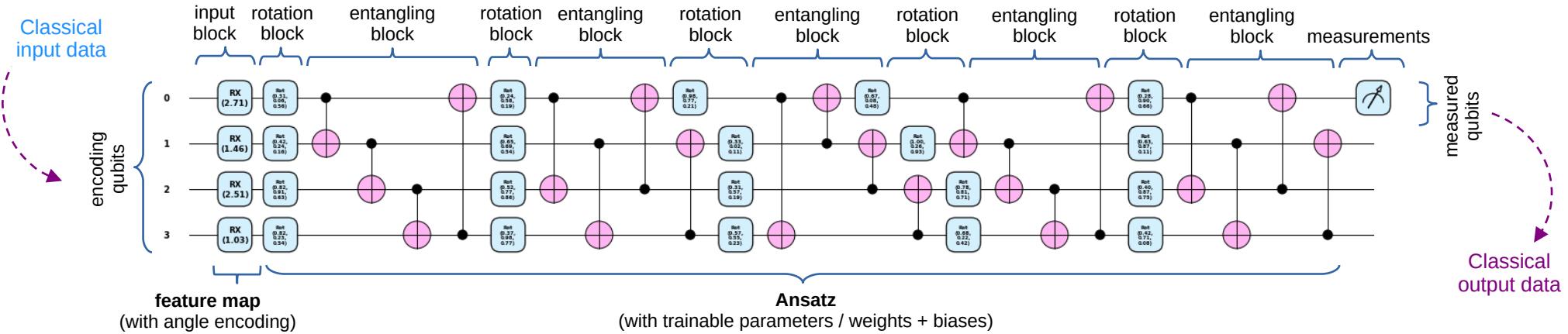


Variational Quantum Models

= Parameterised Quantum Circuits

Variational quantum circuits are not executable!

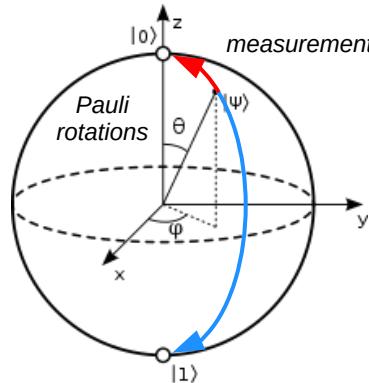
They must first be instantiated, i.e. all of their input and weight parameters must be assigned values!
Ansatz parameters are trainable.



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 initial quantum state.

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 strategies

Data encoding

There are many methods of data embedding, such as:
the *basis*, *angle*, *amplitude*, *QRAM*, ... encoding,

In this workshop we will rely on *angle encoding* realised as qubit state rotation by the angle defined by the data.

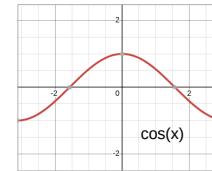
The rotation operators are always available in a quantum platform API (e.g. *Rx*, *Ry*, *Rz* or *Rxyz*).

Typically, the encoding rotation is performed around x or y axis, or both (allowing two values per qubit).

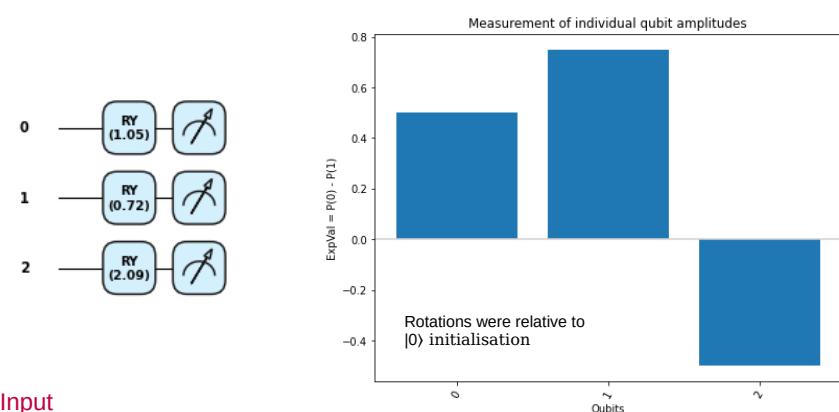
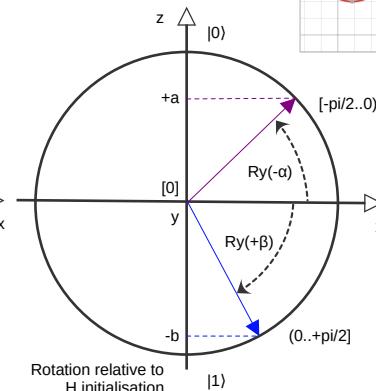
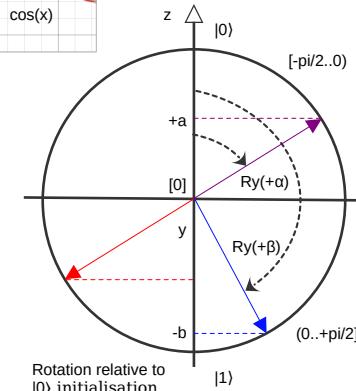
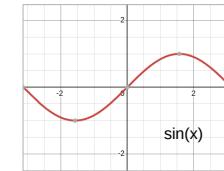
Rotations are *relative to a specific qubit state*, commonly starting at $|0\rangle$ state, or $(|0\rangle+|1\rangle)/\sqrt{2}$, which require qubits to be initialised in these states.

The encoded value could be represented either by the *angular rotation*, or the *amplitude* of the qubit projective measurement (Z).

In some cases, input data is repeatedly encoded and interspersed with ansatz layers, called *data reuploading*, which improves the model performance.



Note that training will place qubit states in areas $x < 0$ and arbitrarily around the z axis. Measurements of such states cannot distinguish them from "pure" $x > 0$ and $z = 0$.



Input

Values entered:
Ry angles used:

[np.arccos(0.5), np.arccos(0.75), np.pi-np.arccos(0.5)]
[1.047, 0.723, 2.094]

Measurements

Probabilities:
Amplitudes:

[[0.25, 0.75], [0.562, 0.438], [0.25, 0.75]]
[0.5, 0.75, -0.5]

Angle encoding

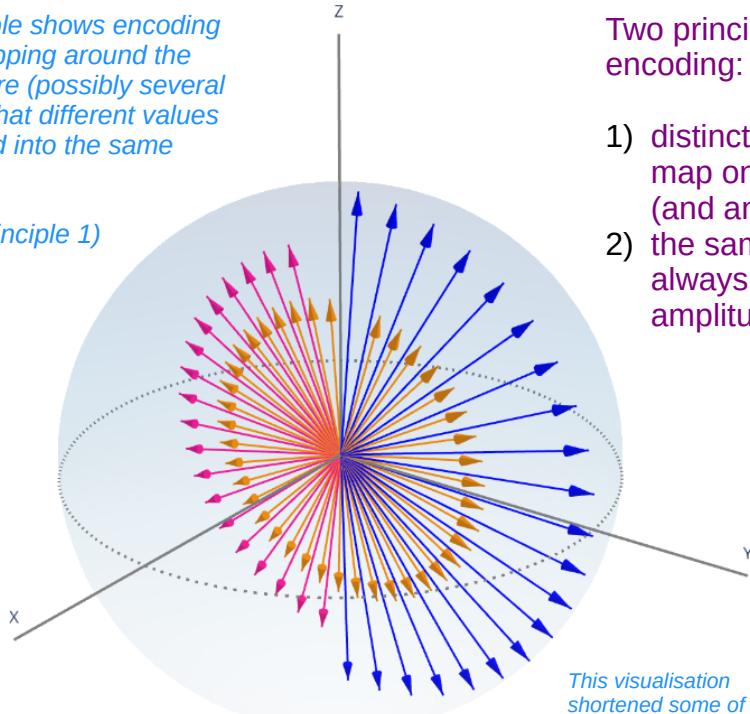
The Good, the Bad and the Ugly

Special care must be taken in those cases where the values form some kind of symmetry or repetition. For example in a case where we were to encode angular values in the range of $-2\pi..2\pi$. In such a case values x and $2\pi+x$ effectively represent the same angular position and their encoding should also be identical.

Angle Range: -6 to 6
 $0..\pi = \text{"blue"}$ (long) | $>\pi = \text{"deeppink"}$ (medium) | $-\pi..0 = \text{"darkorange"}$ (short)

This example shows encoding values wrapping around the Bloch sphere (possibly several times), so that different values are mapped into the same amplitude.

(violates principle 1)



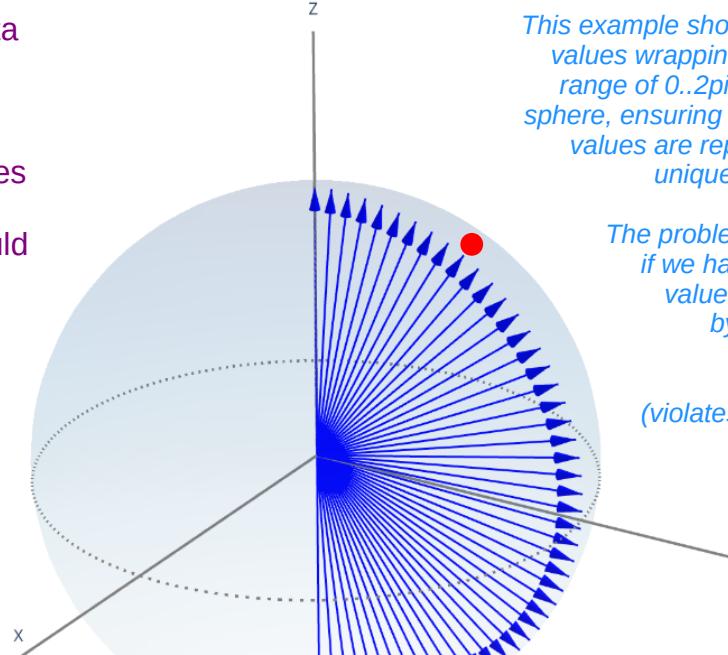
Two principles of quantum data encoding:

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

Angle Range: 0 to 3.141592653589793
 $0..\pi = \text{"blue"}$ (long) | $>\pi = \text{"deeppink"}$ (medium) | $-\pi..0 = \text{"darkorange"}$ (short)

This example shows encoding values wrapping around the range of $0..2\pi$ of the Bloch sphere, ensuring that different values are represented by unique amplitudes.

The problem may arise if we have the same value represented by two distinct amplitudes



Commonly used measurements and interpretation

Quantum circuits can be measured in many ways, e.g.

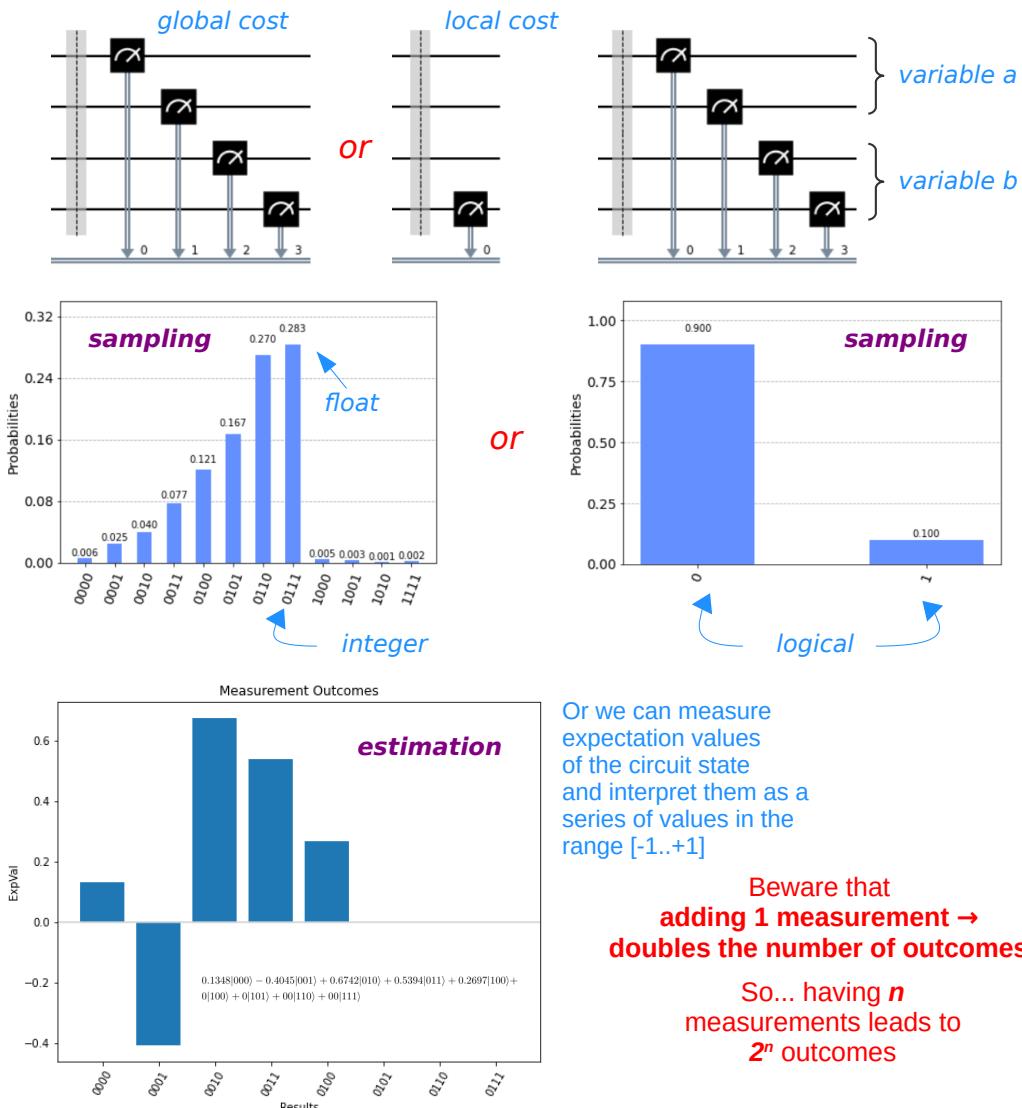
- all qubits (global cost / measurement)
- a few selected qubits (local cost / measurement)
- groups of qubits (each as a variable value)

And received in many different formats, e.g.

- as counts of outcomes (repeated measurements)
- as probabilities of outcomes (e.g. $P(|0111\rangle)$)
- as Pauli expectation values (i.e. of eigenvalues)
- as expectation of interpreted values (e.g. 0 to 15)
- as variance, etc.

Repeated measurement can be interpreted as outcomes of different types, e.g.

- as a probability distribution (as is)
- as a series of values (via expvals)
- as a binary outcome:
single qubit measurement or parity of kets
- as an integer:
most probable ket in multi-qubit measurement
- as a continuous variable:
probability of the selected ket (e.g. $|0^n\rangle$)

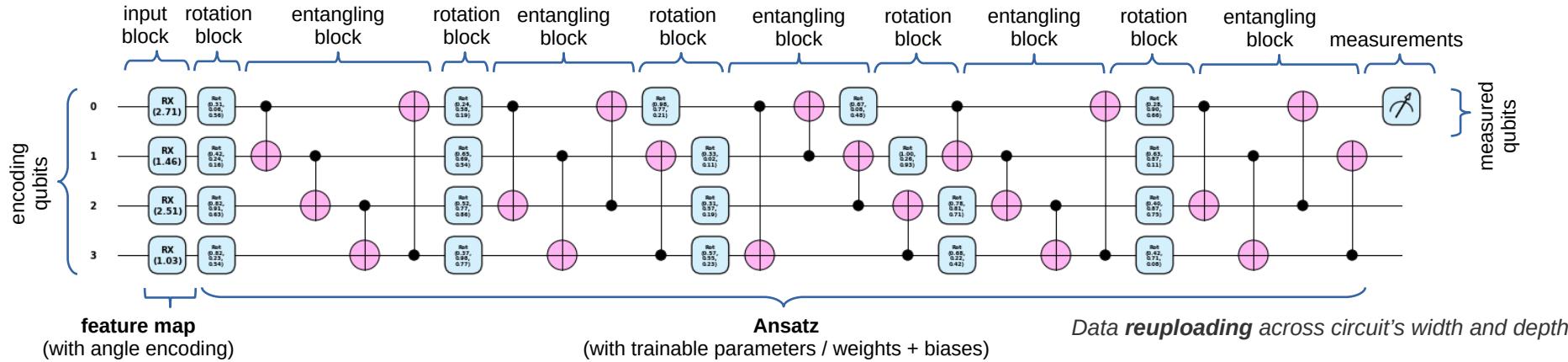


Ansatz design and training

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!



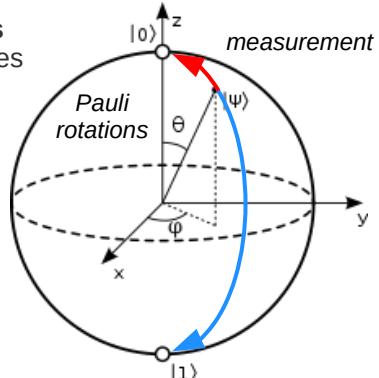
feature maps vary in:
structure and function

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

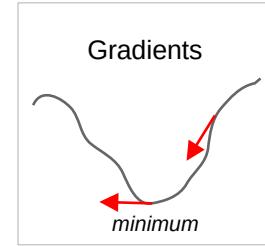
different optimisers:
gradient based (Adam, NAdam and SPSA)
linear approximation methods (COBYLA)
non-linear approximation methods (BFGS)
quantum natural gradient optimiser (QNG)

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

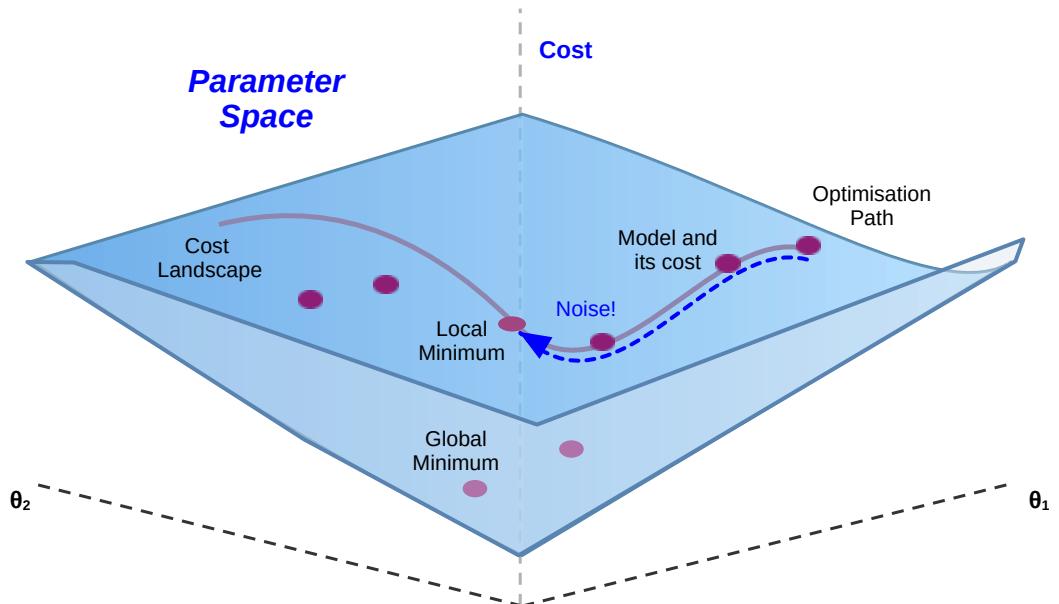
DL models and their optimisation

- A Deep Learning model, such as a neural network, aims to represent some problem.
- The model takes *inputs* and calculates *outputs* via the layers of interconnected nodes.
- Each node has an activation, which is calculated as a *weighted* sum of nodes on its input, a *bias* added, and an *activation function* applied to produce its new value.
- All possible model parameterisations (weights and biases) form a multi-dimensional *parameter space*.
- The model quality is assessed by the *cost function*, where the lower the cost = the better the model.
- The costs of model parameterisations form a manifold over the parameter space - the *cost landscape*.
- The *optimisation process* relies on the shape of the landscape, which in turn is reflected in the *gradient* of points on the cost landscape.
- *Gradient descent* algorithms can assist in the identification of the model with the *minimum cost*.
- *Backpropagation* can also be used to efficiently re-calculate DL model's weights.

An optimiser uses gradients to recognise the shape of the cost landscape and to navigate it in search of such model parameters that produce the lowest cost



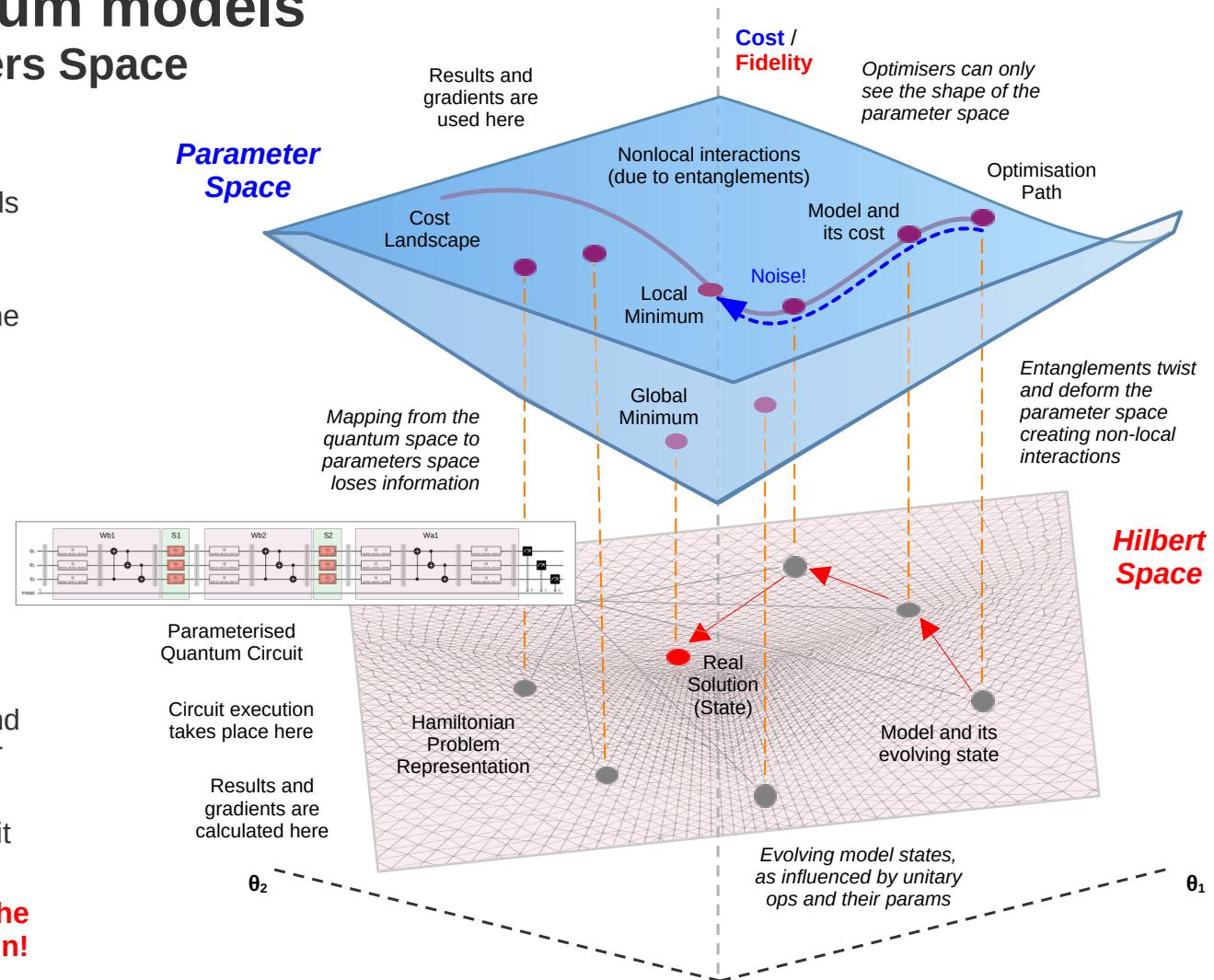
Gradients are local
i.e. during optimisation changes to the model cost influence gradients only in the immediate neighbourhood of the model



Working with quantum models

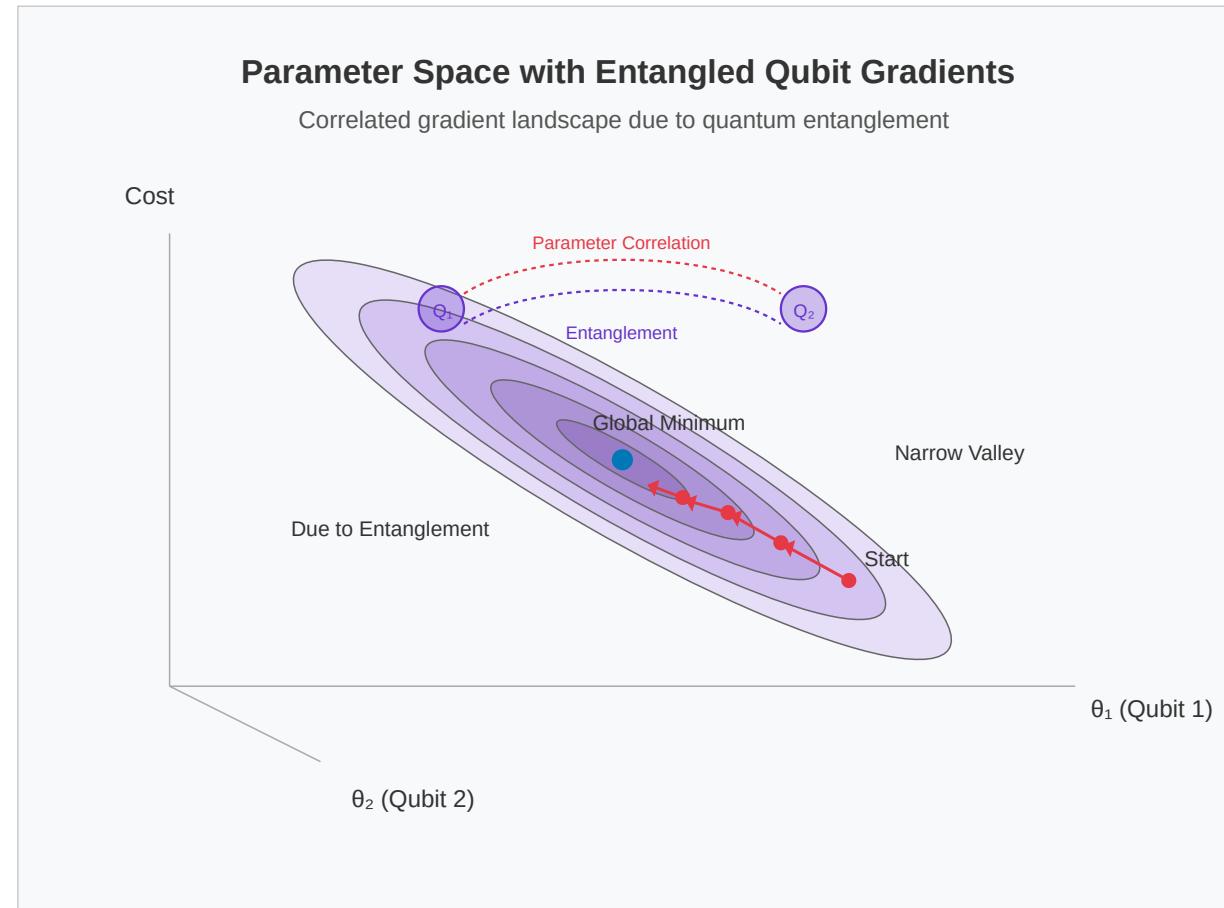
Hilbert Space vs Parameters Space

- **Hilbert state space** (dim = 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
- **Layers of circuit gates** determine the evolution of the quantum model's initial state into its final state during the circuit execution
- **Trainable parameter space** is a classical multi-dimensional space of circuit gate parameters, which the optimiser navigates
- **Entanglements** (defined by CNOTs) create and correlate non-separable qubit states, which alter the parameter space geometry, and also the cost landscape used by the optimiser
- **Measurement** of individual qubits collapses their states, consequently projecting the circuit state onto classical outcomes
- **The mapping from the quantum space to the classical parameter loses some information!**



- Optimisation of quantum model needs unique approaches due to the emergence of *non-local gradients*
- *Entangled qubits* result in *correlated parameters and gradients*, so the changes to one are reflected in the distant others
- The cost landscape of highly entangled circuits commonly features *narrow valleys*
- Also, *backpropagation* cannot be used directly in training quantum circuits, as their state is not directly accessible and the measurement collapses the state
- *Gradient descend* can still be used with *global gradients*, i.e. those derived from the geometry of the cost landscape
- *Stochastic optimisation techniques* are highly effective when the cost landscape is smooth (no quantum noise)
- Other techniques are also available, such as *particle swarm optimisers*, these however are applicable to smaller models

Quantum model optimisation



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

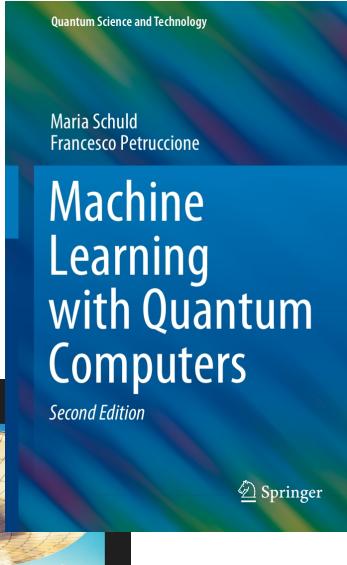
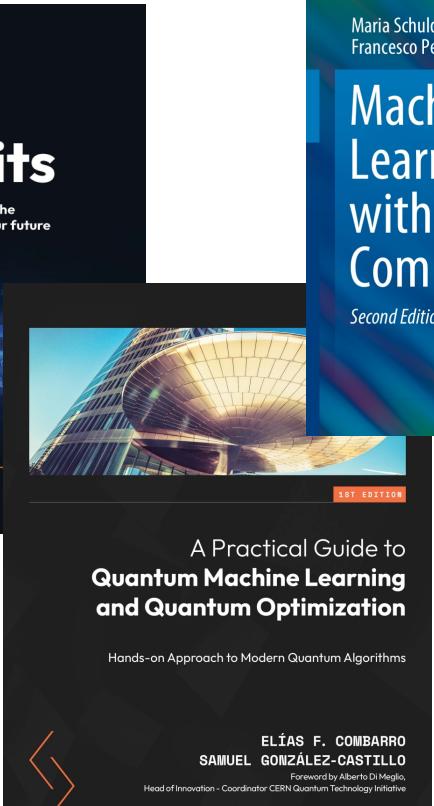
EXPERT INSIGHT

Dancing with Qubits

From qubits to algorithms, embark on the quantum computing journey shaping our future

Second Edition

Robert S. Sutor



PennyLane: Automatic differentiation of hybrid quantum-classical computations

Ville Bergholm,¹ Josh Izaac,¹ Maria Schulz,¹ Christian Gogolin,¹ M. Sohaib Alam,² Shahnaz Ahmad,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 Killoran¹

¹Xanadu, 777 Bay Street, Toronto, Canada

²Rigetti Computing, 2019 Seventh Street, Berkeley, CA 94710

³Wüllenberg Centre for Quantum Technology, Department of Microtechnology and Nanoscience, Chalmers University of Technology, 412 96 Gothenburg, Sweden

⁴Data Cybernetics, Martin-Kolbengser-Str 26, 68899 Landsberg, Germany

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

The framework for optimization and machine learning of quantum and hybrid quantum provides a unified architecture for near-term quantum computing devices, supporting the paradigm. PennyLane's core feature is the ability to compute gradients of variational compatible with classical techniques such as backpropagation. PennyLane thus extends optimizers common in optimization and machine learning to include quantum and hybrid tasks the framework compatible with any gate-based quantum simulator or hardware. We fields, Rigetti Forest, Qiskit, Cirq, and ProjectQ, allowing PennyLane optimizations to be used by devices provided by Rigetti and IBM Q. On the classical front, PennyLane interfaces with libraries such as TensorFlow, PyTorch, and autograd. PennyLane can be used for the eigenvalues, quantum approximate optimization, quantum machine learning models,

Modern applications of machine learning in quantum sciences

Anna Dawid^{1,2*}, Julian Arnold^{3,7}, Borja Requejo^{2,8}, Alexander Gresch⁹, Marcin Płodzień², Kaelan Donatella⁵, Kim A. Nicolaï^{6,7}, Paolo Stornai², Rouven Koch⁸, Miriam Bärtnér⁸, Robert Okula^{10,11}, Gorika Muñoz-Gill¹², Rodrigo A. Vargas-Hernández^{13,14}, Alba Cervera-Lierta¹⁵, Juan Carrasquilla¹⁴, Vedran Dunjko¹⁶, Marylou Gabrié¹⁷, Patrick Huembeli¹⁸, Evert van Nieuwenburg^{16,20}, Filippo Vicentini¹⁸, Lei Wang^{11,22}, Sebastian J. Wetzel²³, Giuseppe Carleo²⁴, Michal Gokh²⁵, Roman Krems²⁶, Florian Marquardt^{26,27}, Michał Tomza¹, Maciej Lewenstein²⁸, and Alexandre Dauphin^{29*}

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 Technologies Institute, University of Regensburg, Germany

⁵ Université de Paris, CNRS, Laboratoire Matière et Phénomènes Quantiques, France

⁶ Machine Learning Group, Technische Universität Berlin, Germany

⁷ BIFOLD, Berlin Institute for Foundations of Learning and Data, 10587 Berlin, Germany

⁸ Department of Applied Physics, Aalto University, Espoo, Finland

⁹ Institute of Physics, Albert-Einstein University of Freiburg, Germany

¹⁰ International Centre for Theoretical Quantum Physics, 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, Toronto, Canada

¹⁵ Barcelona Supercomputing Center, Spain

¹⁶ IJCLab, Leiden University, The Netherlands

¹⁷ CMAP École Polytechnique, France

¹⁸ Institute of Physics, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

¹⁹ Mentor AI, Inc., Palo Alto, California, United States of America

²⁰ 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 Institute for Theoretical Physics, University of Cologne, Germany

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 University 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?



This presentation has been released under the Creative Commons CC BY-NC-ND license, i.e.

BY: credit must be given to the creator.

NC: Only noncommercial uses of the work are permitted.

ND: No derivatives or adaptations of the work are permitted.

Photos from Unsplash

Enquanted is being somewhere in-between Enchanted and Entangled