UNIVERSITY OF EXETER

```
# Import pyQuil modules
from pyquil.quil import Program
from pyquil.api import QVMConnection
from pyquil.gates import H
from functools import reduce

# Create a connection to the Quantum Virtual Machine (QVM)
qvm = QVMConnection()

# Apply the Hadamard gate to three qubits to generate 8 possible randomized results
dice = Program(H(0), H(1), H(2))

# 8 possible results: [[0,0,0], [0,0,1], [0,1,1], [1,1,1], [1,1,0], [1,0,0], [0,1,0]] [0,0,1]]
# Measure the qubits to get a result, i.e. roll the dice
roll_dice = dice.measure_all()

# Execute the program by running it on the QVM
result = qvm.run(roll_dice)

# Example result: [[0,1,0]]
# Format and print the result as a dice value between 1 and 8
dice_value = reduce(lambda x, y: 2*x + y, result[0], 0) + 1
print("Your quantum dice roll returned:", dice_value)
```

Quantum Scientific Machine Learning:

A path to enhanced SciML

Oleksandr Kyriienko

University of Exeter, UK

https://kyriienko.github.io/

Quantum SciML

1. Quantum neural networks

- data embedding
- kernels
- variational circuits
- Fourier vs Chebyshev
- capacity/expressivity
- trainability

2. DQC: derivative quantum circuits

- circuit differentiation
- ODE solving
- convergent-divergent nozzle
- model discovery
- barren plateaus

3. Quantum kernels for SciML

- kernel models
- support vector regression
- convex DE solver
- fraud detection

4. Generative modelling by quantum quantile mechanics

- QCBM
- GAN
- QGAN
- quantile functions
- QQM
- OU sampling

5. Quantum generative modelling with latent space models

- model transformation
- DQGM
- OU sampling
- copula multivariate models

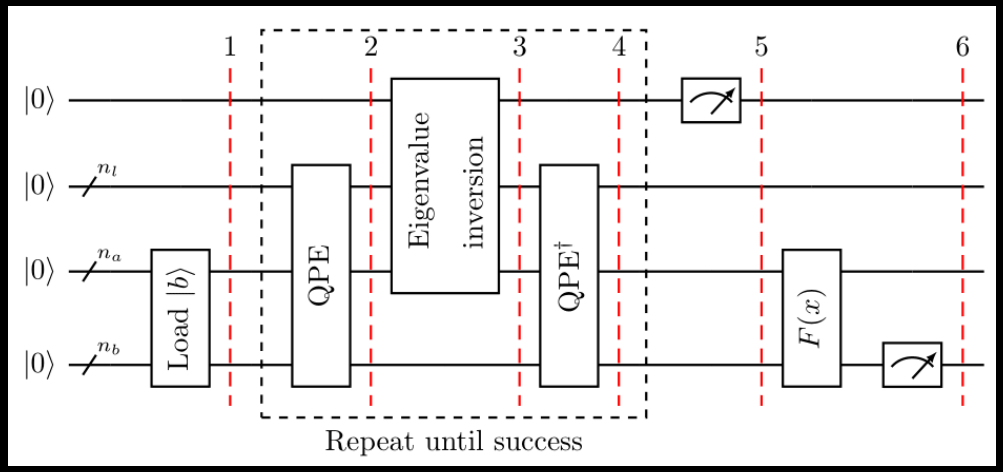
QC for linear systems

We reformulate a linear system problem as inversion of Hermitian operator **A:** $|x\rangle = \hat{A}^{-1}|b\rangle$

Quantum solution runs in O(log(N) s^2 κ^2 /ε) time, and can be further improved to κ log(1/ε). This can be used as a subroutine in solving linear differential equations.

However, the challenges of described scheme include:
1) the input problem;
2) the output problem;
3) deep circuits and ancilla overhead;
4) linearity;
5) dependence on the finite differencing.

Overall we can expect polynomial speed ups, but this dependence on the problem considered (dimensionality, correlations).

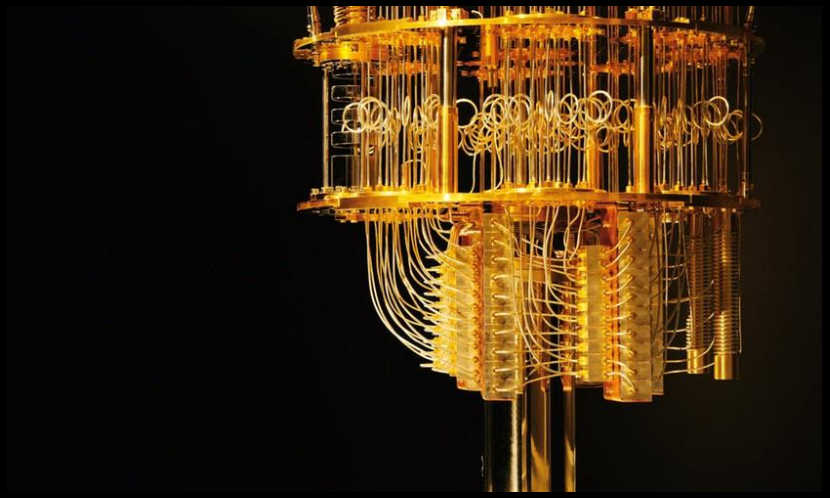


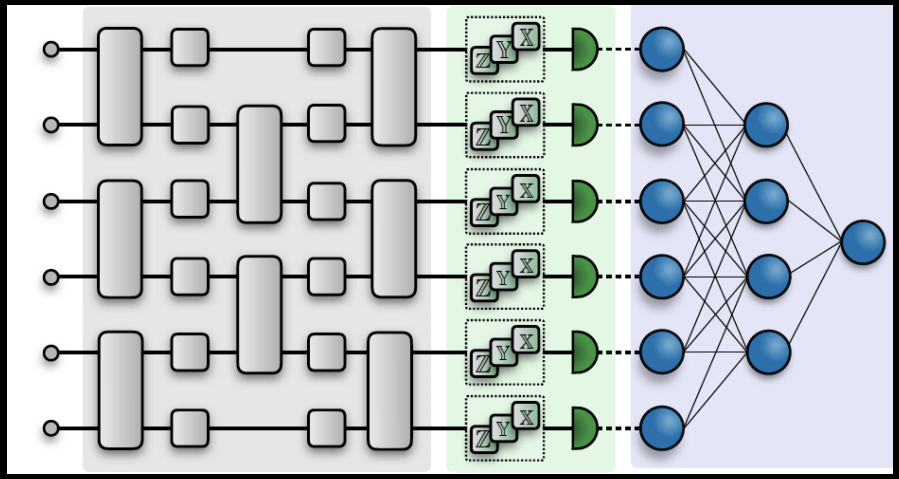
HHL protocol **[Phys. Rev. Lett. 15, 150502 (2009)]**

We may need to rethink the way quantum machine learning is approached

Instead of speeding up linear algebra protocols, let us look advantage in representing function and boost neural network based approaches.

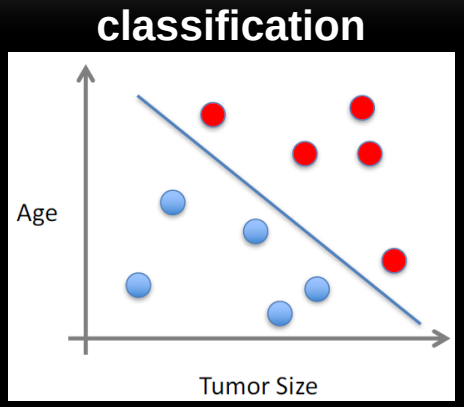
Quantum Machine Learning

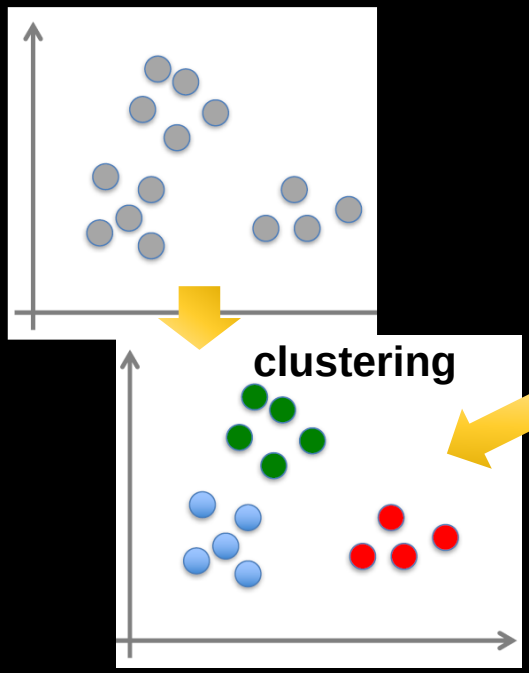




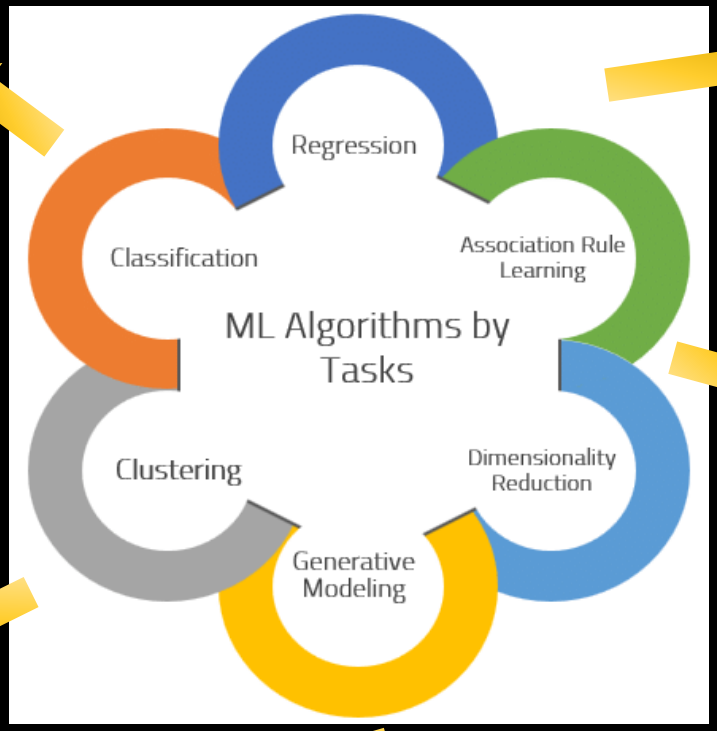
UNIVERSITY OF EXETER

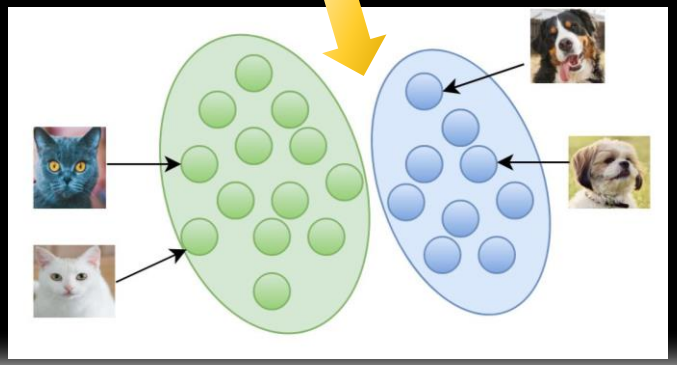
Machine learning

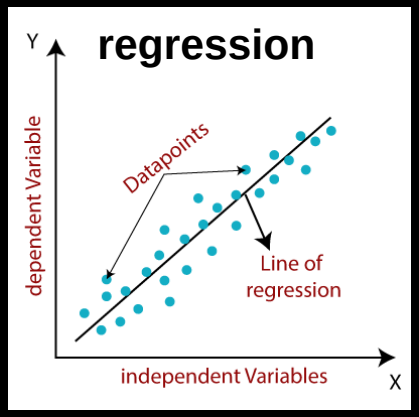


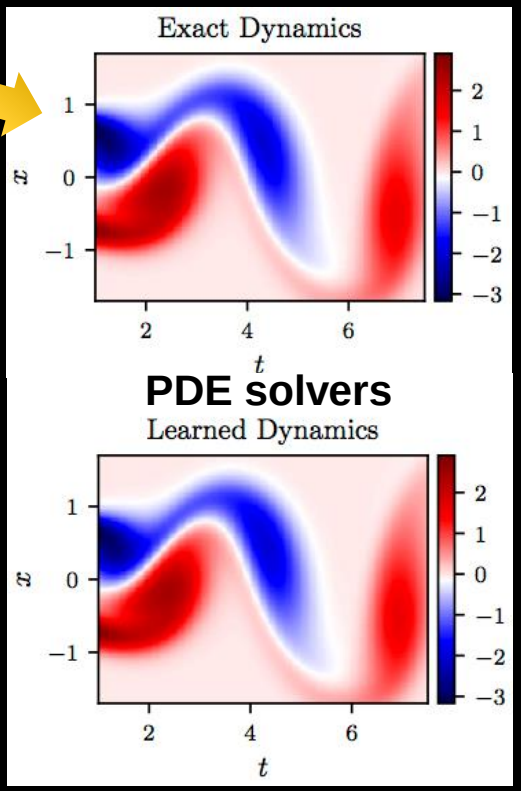


generative modelling





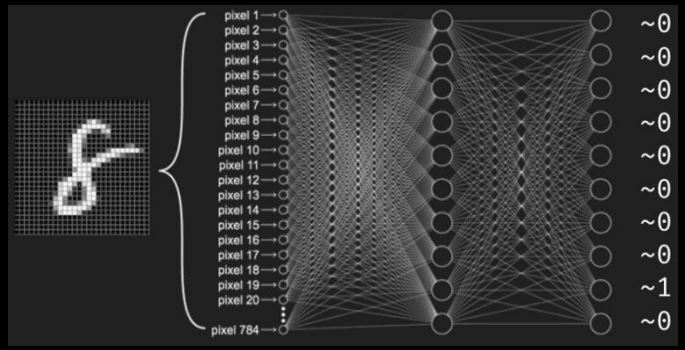




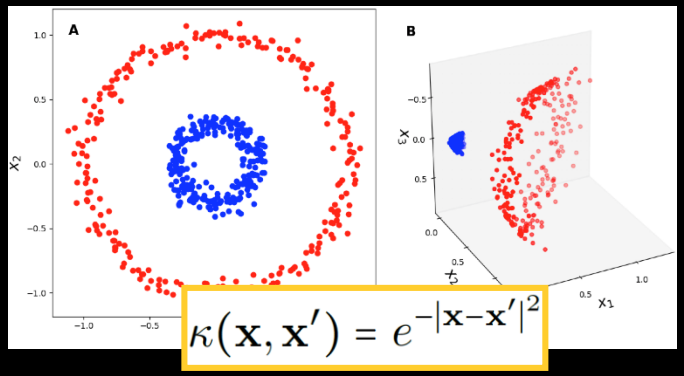
Machine learning algorithms

We can loosely divide machine learning algorithms into two families of algorithms:

- deep neural network-based protocols and
- kernel-based protocols



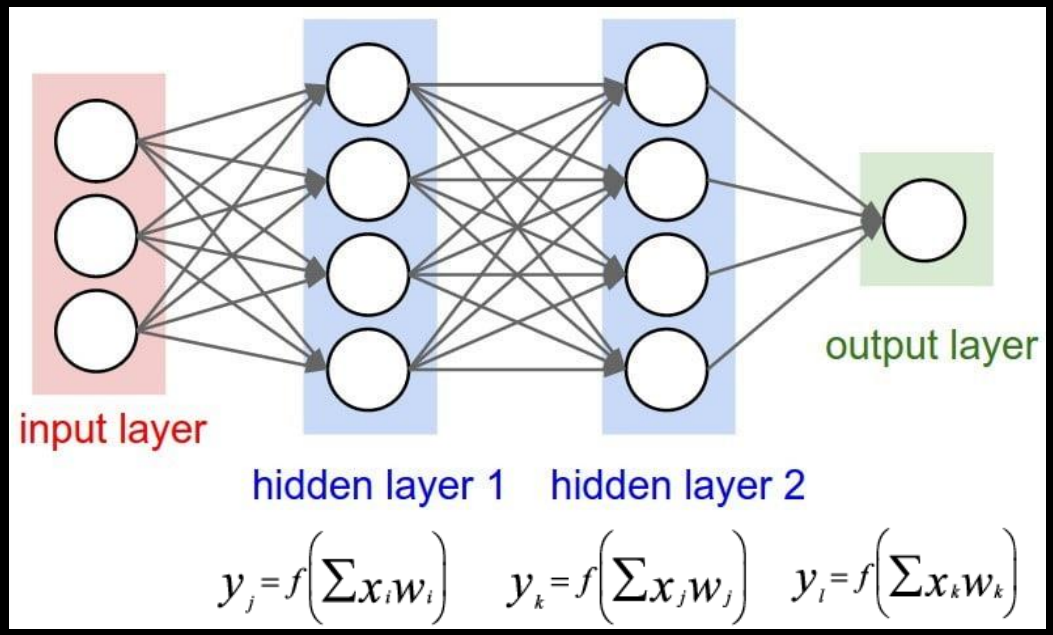
deep neural net



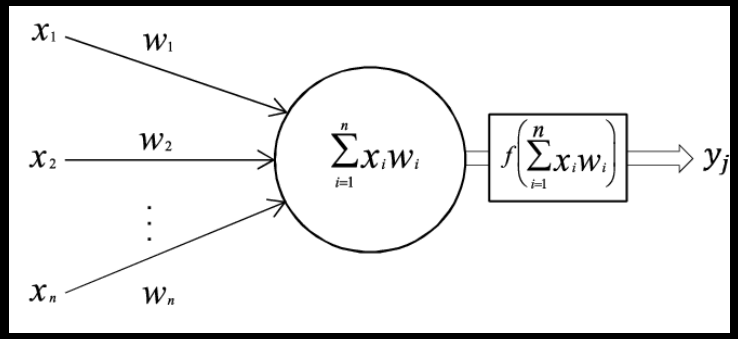
classical kernel

Machine learning

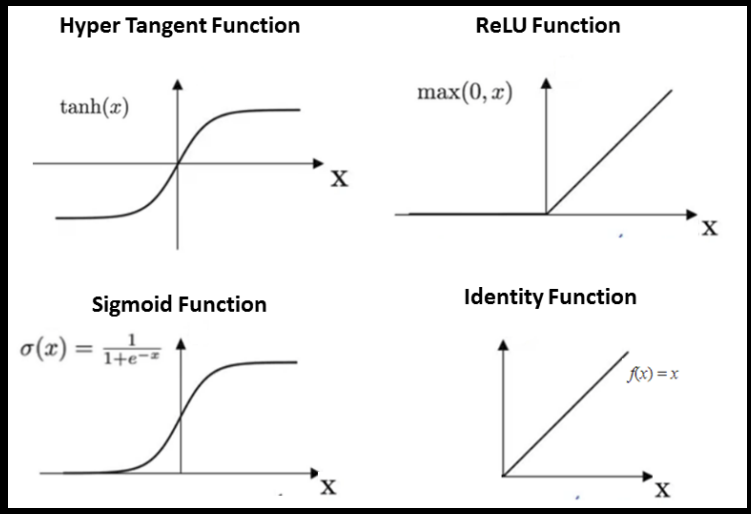
Classical **machine learning** corresponds to a highly efficient way to represent generic parametrised nonlinear function – we use **classical neural networks** as universal approximators.



artificial neural network



action of nonlinear neuron

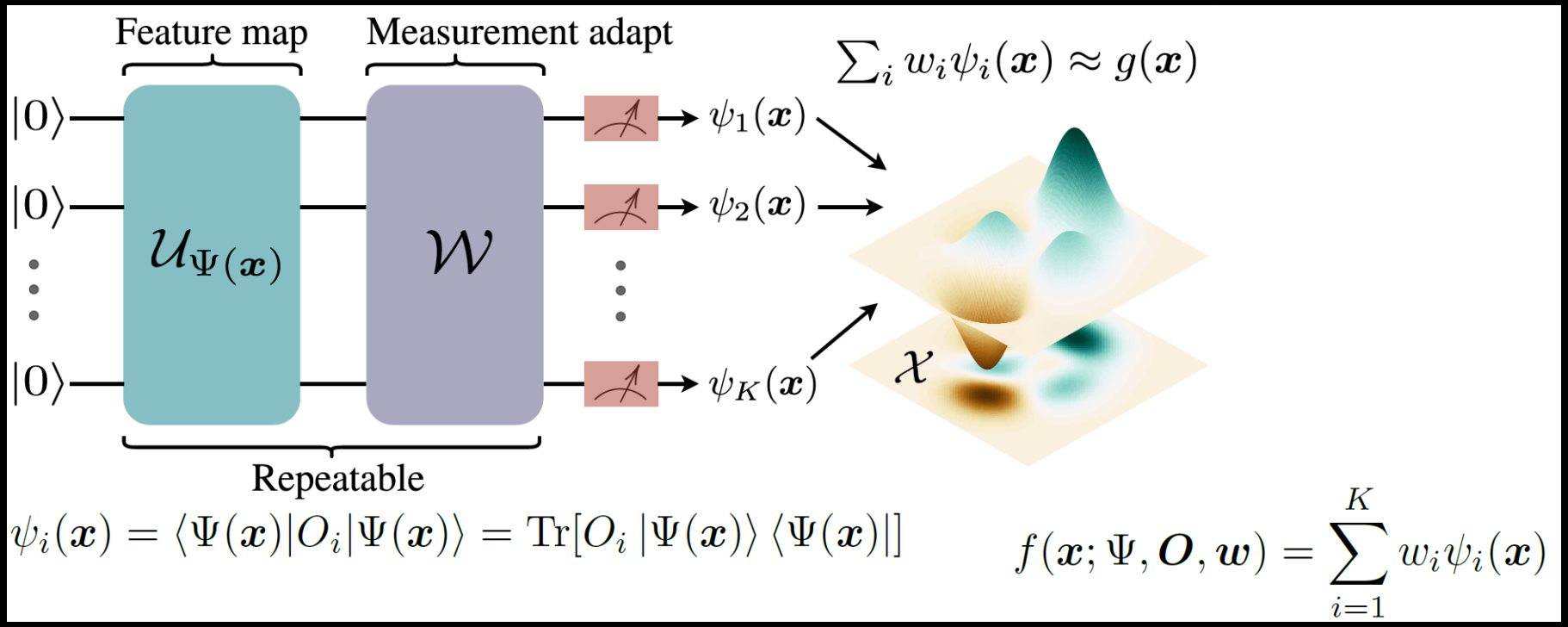


activation functions

Once we have our parametrisable model, we need to **train** it with **gradient descent**, or do any other data analysis in the unsupervised way.

Quantum neural networks

Our goal is to represent functions in the form of **parametrised quantum circuits**, which can be seen as analogues of **quantum neural networks**.



[M. Benedetti et al., Quantum Sci. Technol. 4, 043001 (2019)]

In the simplest case we use a cost function read as measurement of Hermitian operators

$$|f_{\varphi,\theta}(x)\rangle = \hat{\mathcal{U}}_\theta \hat{\mathcal{U}}_\varphi(x)|\emptyset\rangle$$

QNN-based quantum state

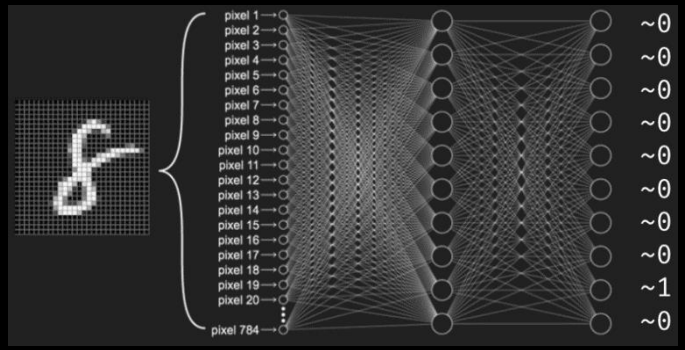
$$f(x) = \langle f_{\varphi,\theta}(x)|\hat{C}|f_{\varphi,\theta}(x)\rangle$$

latent space function

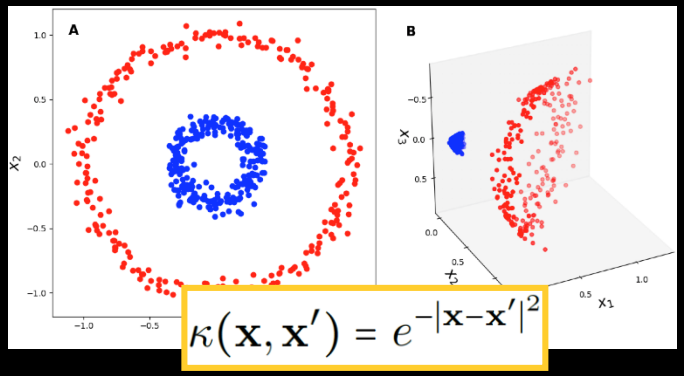
Machine learning algorithms

We can loosely divide machine learning algorithms into two families of algorithms:

- deep neural network-based protocols and
- kernel-based protocols

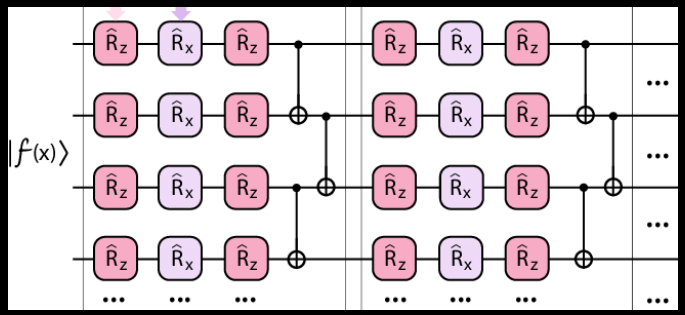


deep neural net

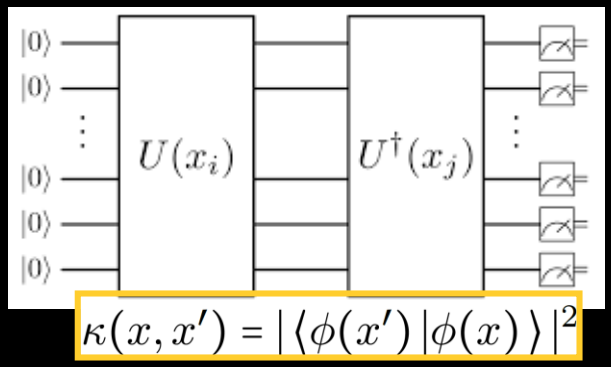


classical kernel

We can design **quantum machine learning** methods based on similar principles:



quantum neural net

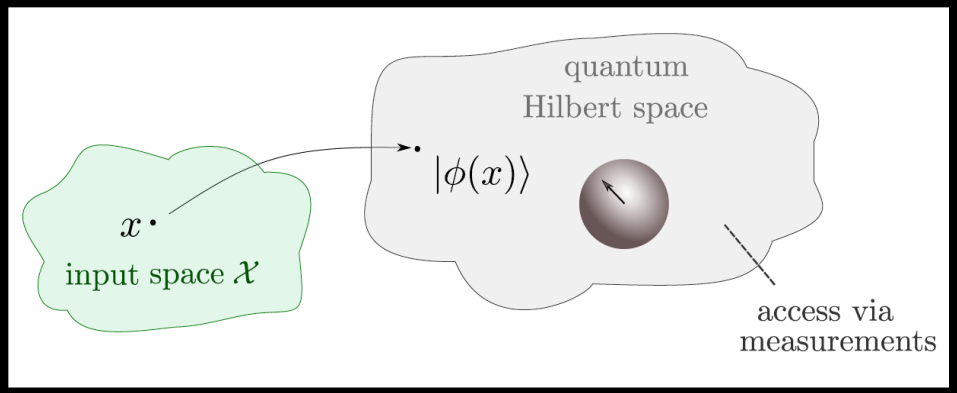


quantum kernel

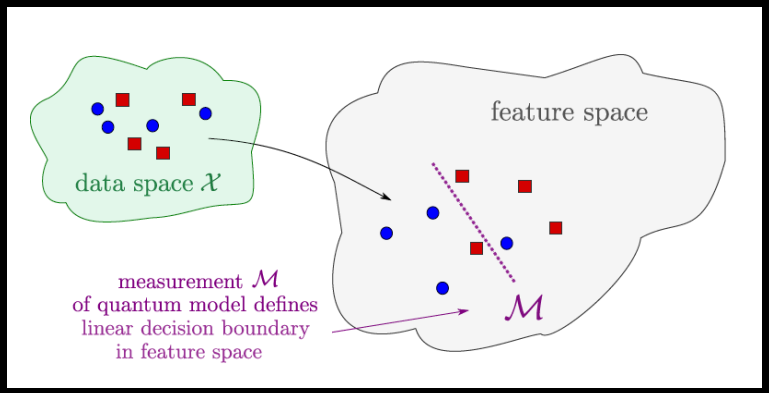
Each has pros/cons, and we will use them depending on the context.

Quantum neural networks

The **power** of **quantum machine learning** comes from representing **data** as quantum states in **high-dimensional Hilbert space**.



quantum embedding (feature map)

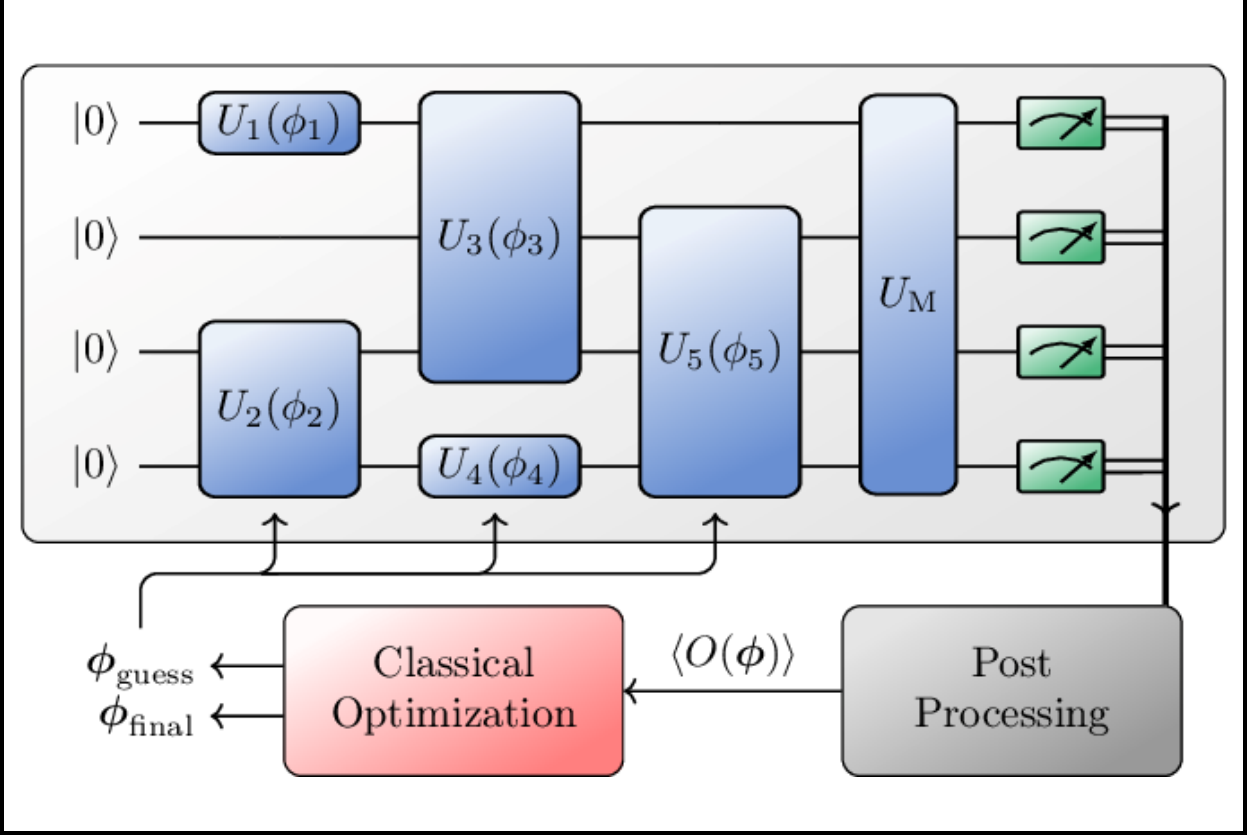


drawing decision boundary in a feature space

[M. Schuld, arXiv:2101.11020 (2021)]

- **Expressivity** of our model (Rademacher complexity of a function class) is defined by the structure of the quantum feature map
- To solve specific ML tasks we can train our **parametrized model** by adjusting angles to minimize a **loss function**
- Alternatively, we can use quantum embedding followed by **kernel** measurements and convex optimisation

Variational quantum algorithms



workflow for hybrid quantum-classical algorithms (VQE)

Key ingredients:

- **variational quantum circuit** (similar to choosing an architecture of neural network)
- **loss function** (energy for VQE, but can be fidelity and overlaps for generic protocols)
- **optimization schedule** (derivative-based vs derivative-free, frugal or not, influence of noise)
- **measurement schedule** (we need to average Hamiltonian, equivalent to measuring all commuting sets of Pauli strings)

Quantum circuit learning

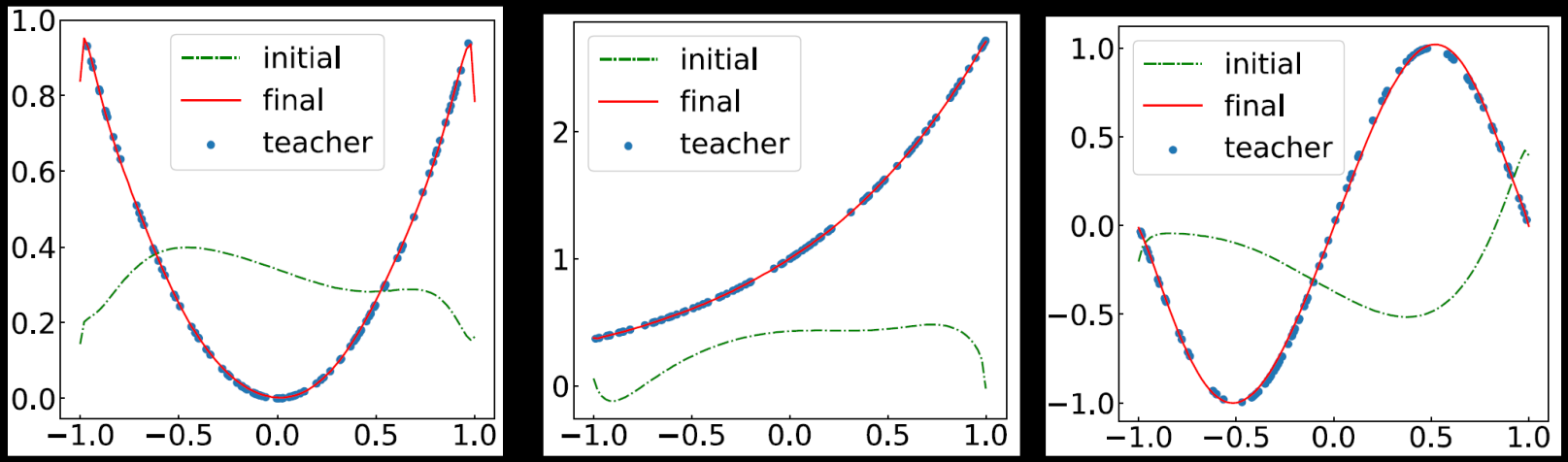
One of the first problems we can solve corresponds to **regression**.

$\{x_i\}$ $\{f(x_i)\}$ $y_i = y(x_i, \theta)$

input data for training **variationally prepared function**

MSE loss: $L = \sum_i \| f(x_i) - y_i \|^2$

To match the training data we need to assign the **loss function**. The simplest and most intuitive form is **mean squared error (MSE)** also known as **L_2 loss**.



nonlinear regression by QNNs [K. Mitarai, PRA 98, 032309 (2018)]

Getting **advantage** requires working with **multidimensional functions** and feature maps.

Quantum circuit learning

One of the first problems we can solve corresponds to **regression**.

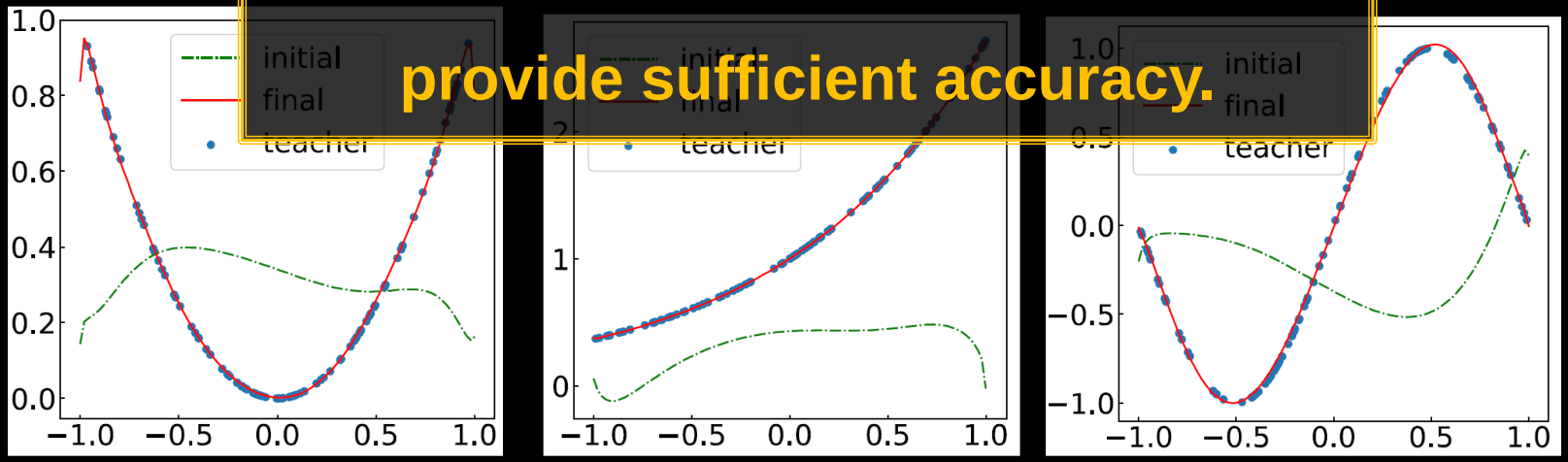
$\{x_i\}$ $\{f(x_i)\}$ $y_i = y(x_i, \theta)$

input data for training **variationally prepared function**

MSE loss: $L = \sum_i \| f(x_i) - y_i \|^2$

The challenge is finding problems where classical solutions do not provide sufficient accuracy.

To match the training data we need to assign the **loss function**. The simplest and most intuitive form is mean-squared-error (MSE) loss:



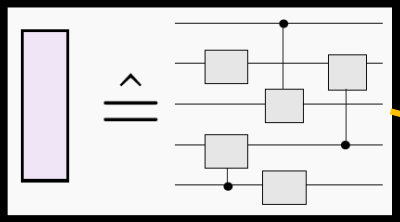
nonlinear regression by QNNs [K. Mitarai, PRA 98, 032309 (2018)]

Getting **advantage** requires working with **multidimensional functions** and feature maps.

Quantum feature maps

We can consider a generic **feature map** with interleaved $\boldsymbol{x}$-dependent unitaries $\boldsymbol{S(x)}$ (data embedding) and parameterized circuits $\boldsymbol{W}$ ("weights"). This is called a **data reuploading** technique.

Circuits $\boldsymbol{W}$ may have some arbitrary structure.



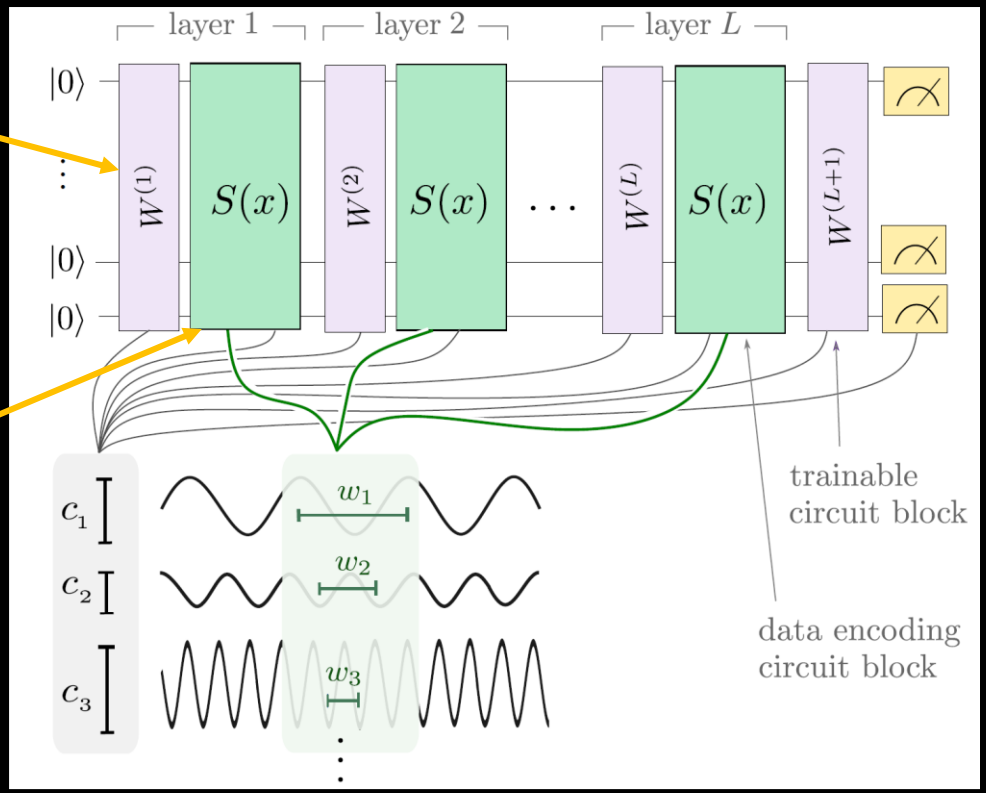
Data embedding $\boldsymbol{S(x)}$ generated by some Hermitian operator and **without pre-processing** of $\boldsymbol{x}$ can be diagonalized as:

$$S(x) = V^\dagger e^{-ix\Sigma} V$$

Our goal is to form a quantum model as an expectation:

$$f_{\boldsymbol{\theta}}(x) = \langle 0 | U^\dagger(x, \boldsymbol{\theta}) M U(x, \boldsymbol{\theta}) | 0 \rangle$$

Next, we can show that the model corresponds to partial Fourier series with controlled coefficients.



developing correspondence between QNN-based models and Fourier series

[M. Schuld et al, PRA 103, 032430 (2021)]

Quantum feature maps

We can consider the action of the feature map in the eigenbasis of embedding (**diagonal generator Σ**). Variable dependence has a complex exponential form that are concatenated, weighted by $\boldsymbol{W}$s, and contracted on the operator $\boldsymbol{M}$.

$$[U(x)\,|0\rangle]_i = \sum_{j_1 \dots j_L = 1}^{d} e^{-i(\lambda_{j_1} + \dots + \lambda_{j_L})x} \times W^{(L+1)}_{i j_L} \dots W^{(2)}_{j_2 j_1} W^{(1)}_{j_1 1}$$

$$a_{\boldsymbol{k},\boldsymbol{j}} = \sum_{i,i'} (W^*)^{(1)}_{1 k_1} (W^*)^{(2)}_{j_1 j_2} \dots (W^*)^{(L+1)}_{j_L i} M_{i,i'} \times W^{(L+1)}_{i' j_L} \dots W^{(2)}_{j_2 j_1} W^{(1)}_{j_1 1}$$

Contracting the resulting state on the operator $\boldsymbol{M}$ we arrive to the quantum model of the type:

QNN-based model as partial Fourier series:

$$f(x) = \sum_{\omega \in \Omega} c_\omega e^{i\omega x}$$

where

$$c_\omega = \sum_{\substack{\boldsymbol{k},\boldsymbol{j} \in [d]^L \\ \Lambda_{\boldsymbol{k}} - \Lambda_{\boldsymbol{j}} = \omega}} a_{\boldsymbol{k},\boldsymbol{j}}$$

The resulting model is characterised by its **spectrum**

$$\Omega = \{\Lambda_{\boldsymbol{k}} - \Lambda_{\boldsymbol{j}},\ \boldsymbol{k},\boldsymbol{j} \in [d]^L\}$$

Major properties of this spectrum are

$$K = (|\Omega| - 1)/2$$ **size** and $$D = \max(\Omega)$$ **degree**

Let's characterise the properties of the models we build.

Quantum feature maps

The number of available frequencies depends on the number of **nontrivial gaps** (difference of eigenvalues) and **number of reuplodings**. This can be **serial and parallel**.

We generally consider **three key properties** of quantum models (informal):

1) capacity (K, D, redundancy etc)

characterizes available fitting functions

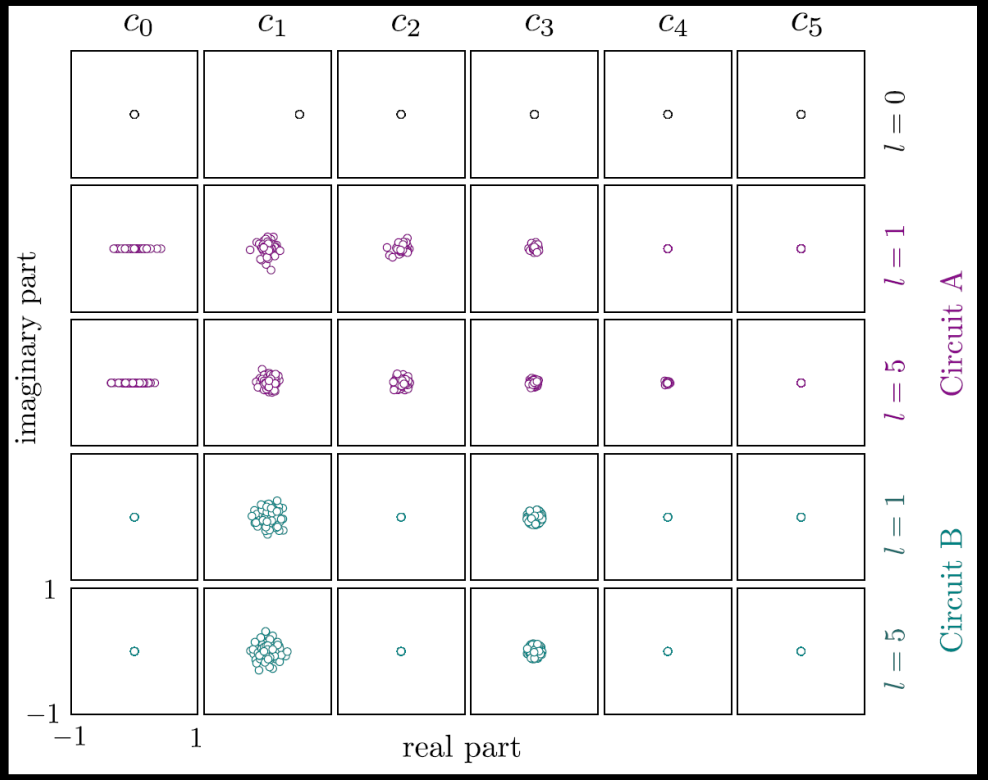
2) expressivity ($c[\theta]$)

accessible models defined by variational parameters

3) trainability (var[grad[θ]])

ability to find suitable (quasi)optimal parameters

Identifying high-capacity quantum models with excellent expressivity and trainability is an ongoing challenge



coefficient distribution for random θ

Quantum feature maps

Next, we can also develop feature maps based on different basis function. For instance, we have designed a **Chebyshev quantum feature map**

[OK, A. Paine, V. Elfving, Phys. Rev. A 103, 052416 (2021)]

Let us use rotations of the form

$$\hat{\mathcal{U}}_\varphi(x) = \bigotimes_{j=1}^{N} \hat{R}_{y,j}(2n[j]\arccos x)$$

parameterised embedding

$$\hat{R}_{y,j}(\varphi[x]) = T_n(x)\mathbb{1}_j + U_n(x)\hat{X}_j\hat{Z}_j$$

Chebyshev basis functions

For TR-symmetric ansatze we can write a corresponding model as

$$f(x) = \sum_{n\in\mathcal{N}} c_{n,\theta} T_n(x)$$

quantum Chebyshev model

Finally, we have also introduced a **phase feature map** with **exponential capacity**

[OK, A. E. Paine, V. Elfving, arXiv:2202.08253 (2022)]

$$\hat{\mathcal{U}}_\varphi(x) = \prod_{j=1}^{N} \left[\hat{R}_j^z\left(\frac{2\pi x}{2^j}\right)\hat{H}_j\right]$$

$$|\widetilde{x}\rangle := \hat{\mathcal{U}}_\varphi(x)|\phi\rangle$$

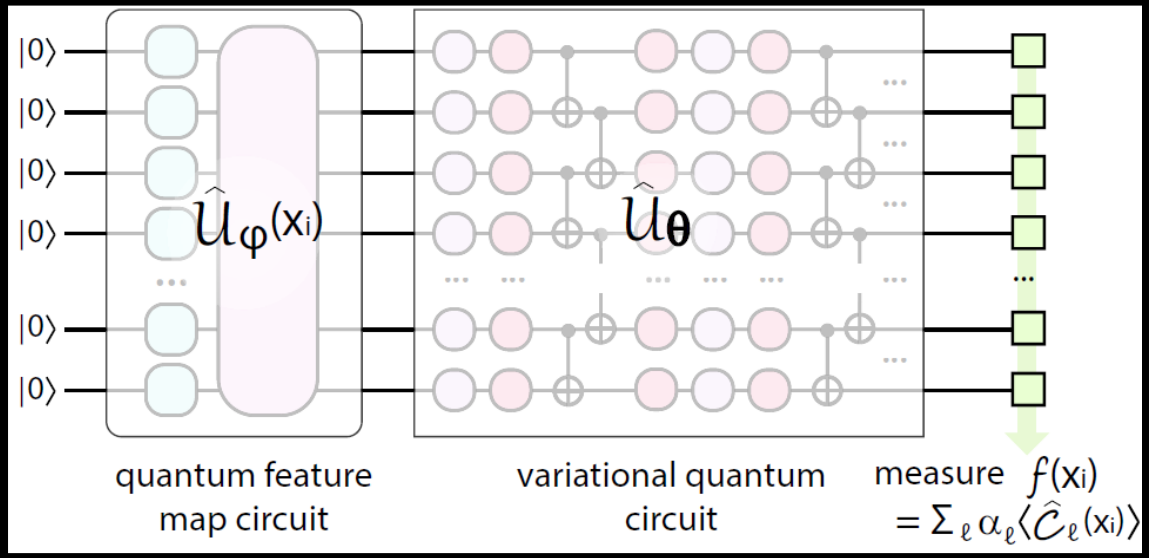
$$|\widetilde{x}\rangle = \frac{e^{-i\Phi/2}}{2^{N/2}} \bigotimes_{j=1}^{N} \left(|0\rangle_j + \exp\left(-i\frac{2\pi x}{\xi_j 2^j}\right)|1\rangle_j\right)$$

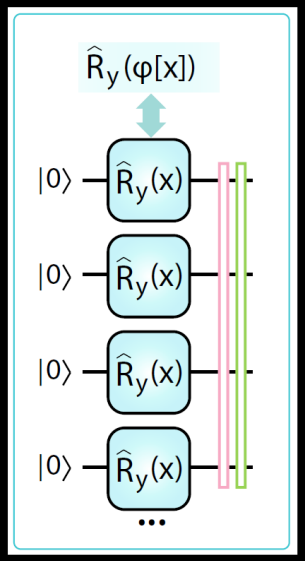
$$f(x) = \sum_{\omega=1}^{2^N-1} c_{\omega,\theta}\exp(-i\omega x)$$

phase map model

DQCs for nonlinear DEs

Our goal is to represent functions in the form of **differentiable quantum circuits** (DQCs) aka quantum neural networks, as used in the quantum circuit learning.





quantum feature map

In the simplest case we use a cost function readout as measurement of Hermitian operators (any chosen pool)

$$|f_{\varphi,\theta}(x)\rangle = \hat{U}_\theta \hat{U}_\varphi(x)|\emptyset\rangle$$

DQC-based quantum state

$$f(x) = \langle f_{\varphi,\theta}(x)|\hat{C}|f_{\varphi,\theta}(x)\rangle$$

latent space function

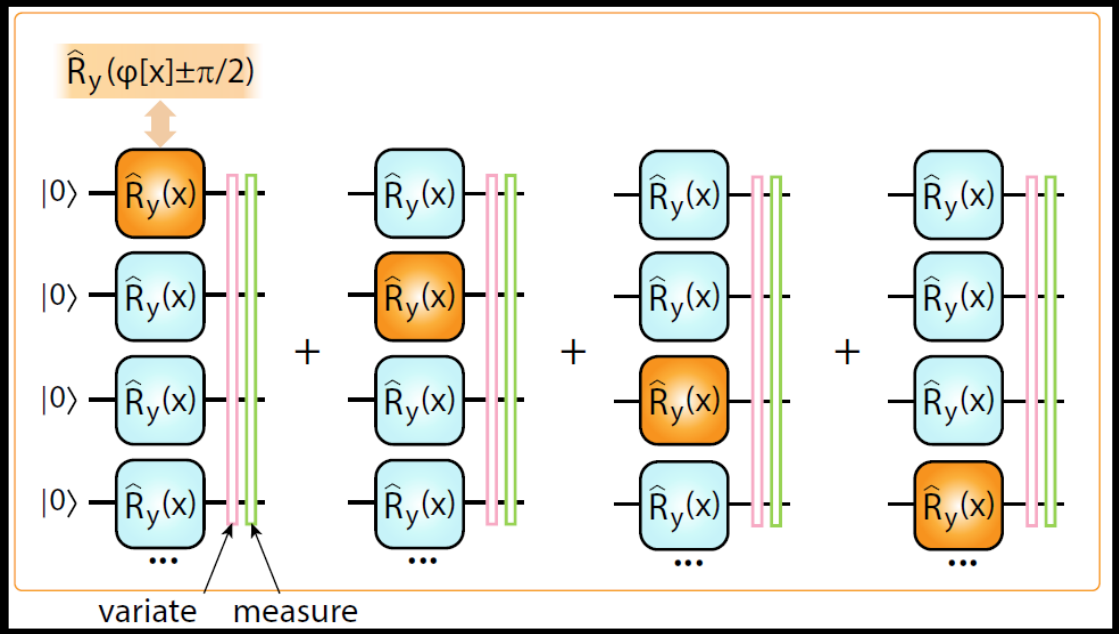
$$\hat{U}_\varphi(x) = \bigotimes_{j=1}^{N'} \hat{R}_{\alpha,j}(\varphi[x])$$

Once we have represented a **function** $f(x)$ using **DQC**, we can also find its **derivative** $\mathrm{d}f(x)/dx$ using **automatic differentiation** (forward path).

[OK, A. E. Paine, V. Elfving, Phys. Rev. A 103, 052416 (2021)]

DQCs for nonlinear DEs

To **differentiate** the **quantum circuit** consisting of gates generated by Pauli string generators we can use the **parameter shift rule**. While this is usually done for **derivative-based optimization**, in our case we also use it for **automatic feature map differentiation**.



differentiated quantum circuits

[K. Mitarai et al., PRA 98, 032309 (2018)]
[M. Schuld et al., PRA 99, 032331 (2019)]

We can also now perform **generalized circuit differentiation** with arbitrary generators for feature maps and variational gates

$$\frac{df(x)}{dx} = -\sum_{s=1}^{S} \Delta_s \left[\sin\left(\frac{x}{2}\Delta_s\right) \mathrm{Re}\{\mathcal{O}_s\} + \cos\left(\frac{x}{2}\Delta_s\right) \mathrm{Im}\{\mathcal{O}_s\} \right]$$

[OK, V. Elfving, PRA 104, 052417 (2021)]

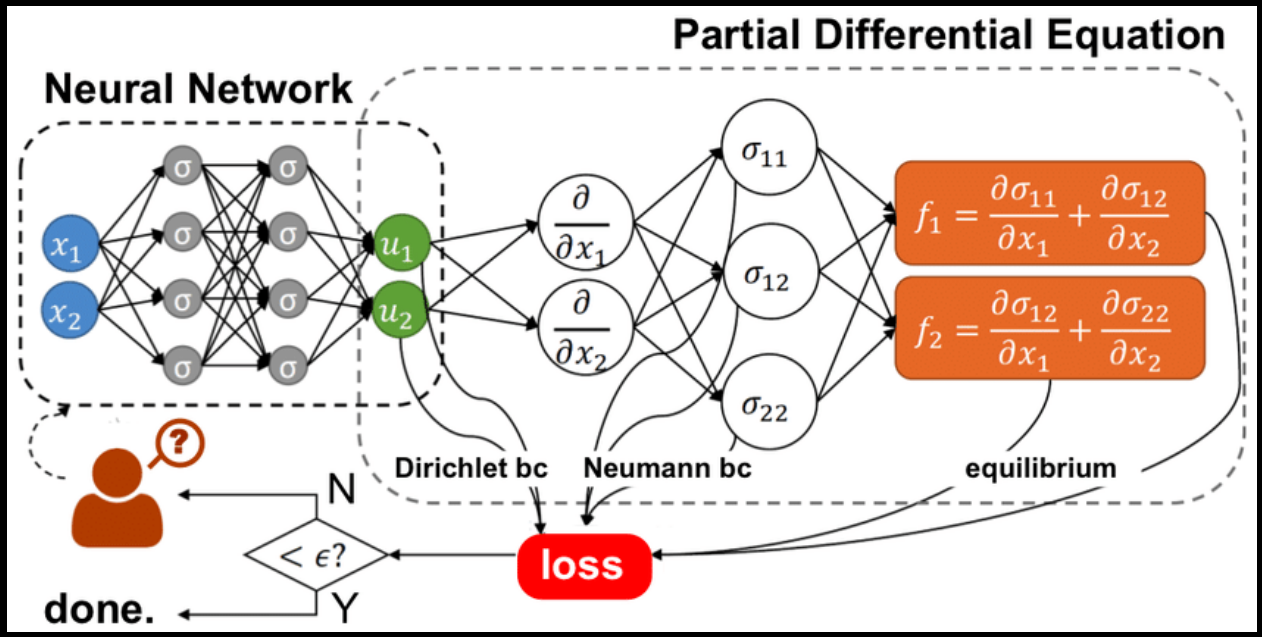
$$df(x)/dx = \frac{1}{2}\sum_j \left(\langle f^+_{d\varphi,j,\boldsymbol{\theta}}(x)|\hat{C}|f^+_{d\varphi,j,\boldsymbol{\theta}}(x)\rangle - \langle f^-_{d\varphi,j,\boldsymbol{\theta}}(x)|\hat{C}|f^-_{d\varphi,j,\boldsymbol{\theta}}(x)\rangle \right)$$

Derivatives can be evaluated at any point of the grid.

PINN: Physics-informed neural network

Recent year neural function representation has led to advances in computational fluid dynamics, where DNNs are used as universal approximators and derivatives are analytical

[Raissi et al, J. Comp. Phys. 378, 686 (2019)]



physics-informed neural networks and neural PDEs

- The approach works for various architectures, and thus is applicable to **analog neural networks** (optical, electrical, spintronic). Quantum offers large feature space and accuracy
- Although a heuristic, using neural PDEs outperformed traditional methods already for systems with 6 equations, and is efficient for d > 3 dimensions **[arXiv:2001.04385]**

DQCs for nonlinear DEs

We start with **differential equation** with appropriate boundary conditions written in the form

DEs: $$F[\{d^m f_n/dx^m\}_{m,n}, \{f_n(x)\}_n, x] = 0$$

Then, we design a **loss function** such that RHS and LHS parts of the differential equation are equal, and search for the optimal solution as

$$\theta_{\text{opt}} = \underset{\theta}{\text{argmin}}(\mathcal{L}_\theta[d_x f, f, x])$$

variational search

$$f(x)|_{\theta\to\theta_{\text{opt}}} \approx u(x)$$

DQC solution **true DE solution**

From the technical side, we develop and use several crucial elements:

- Chebyshev feature map of variable capacity
- adjustable cost functions (~classical NNs)
- **boundary handling** procedures
- add regularization

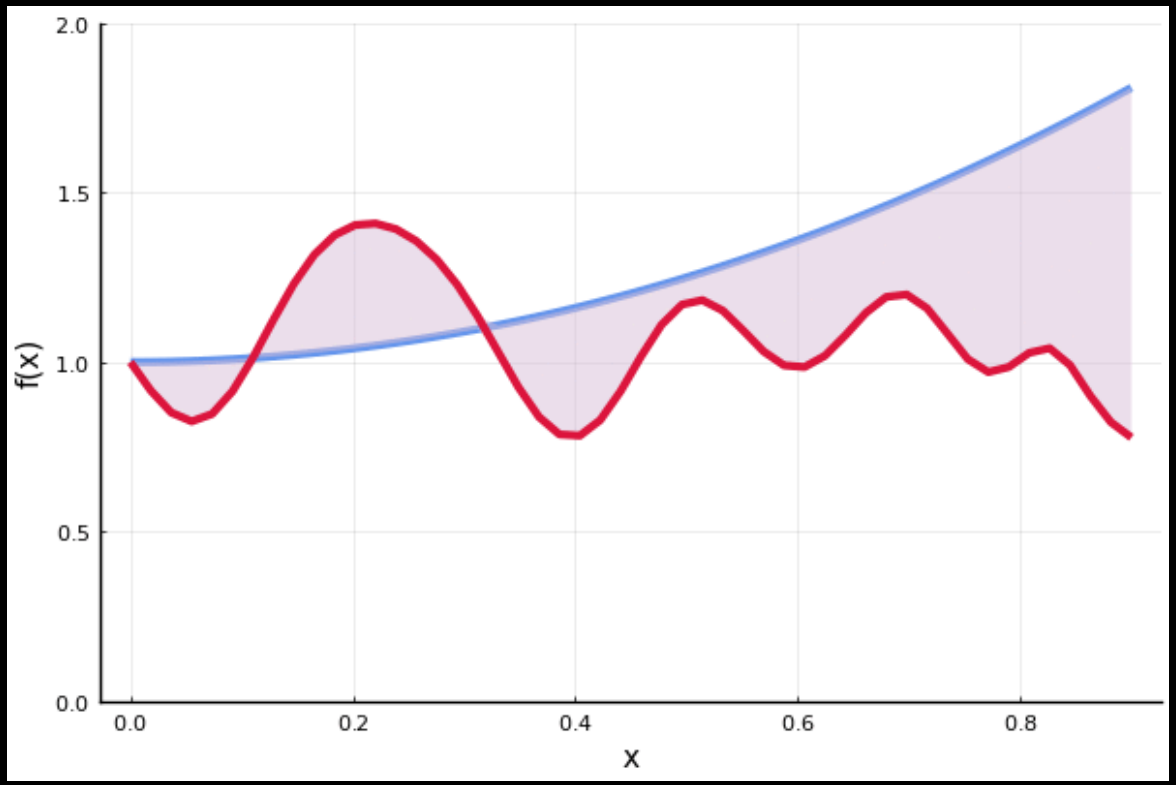
[OK, V. Elfving, PRA 104, 052417 (2021)]

Vincent Elfving
(CTO/sols)

Annie Paine
(PhD student)

DQCs for nonlinear DEs

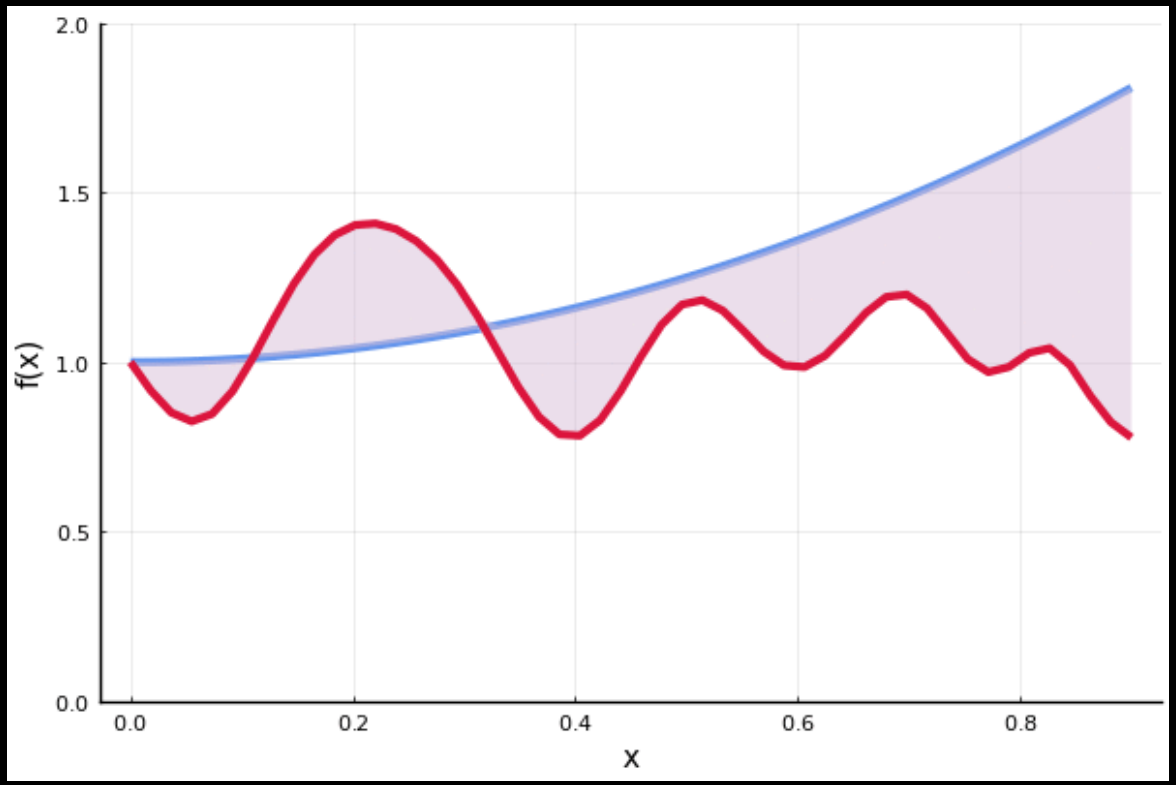
First, let us test it for some simple example with parabolic solution. The function is updated dynamically at each epoch (6q, HEA, depth=6, Adam, Chebyshev feature map)



true solution (blue) and DQC solution (red)

DQCs for nonlinear DEs

First, let us test it for some simple example with parabolic solution. The function is updated dynamically at each epoch (6q, HEA, depth=6, Adam, Chebyshev feature map)

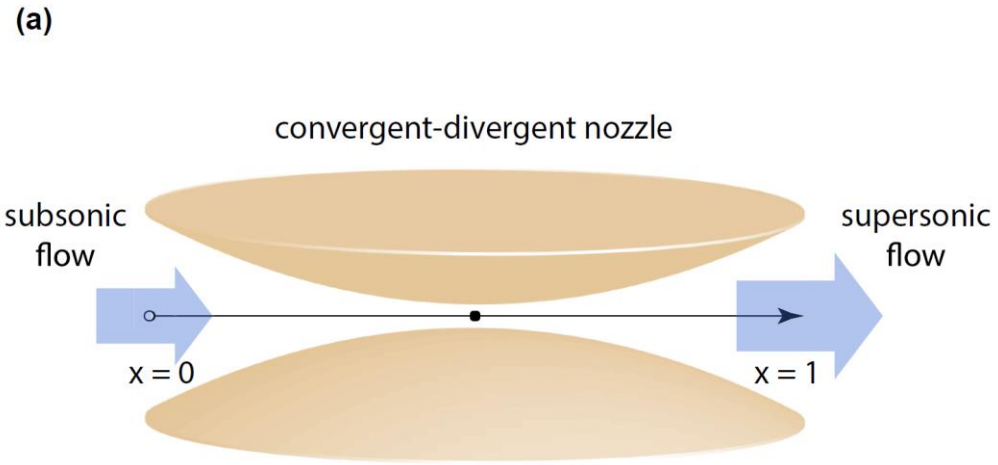


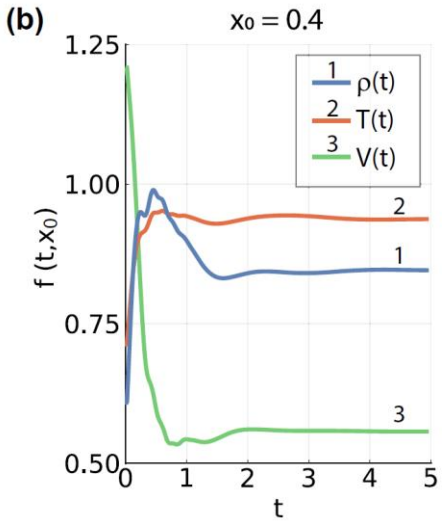
true solution (blue) and DQC solution (red)

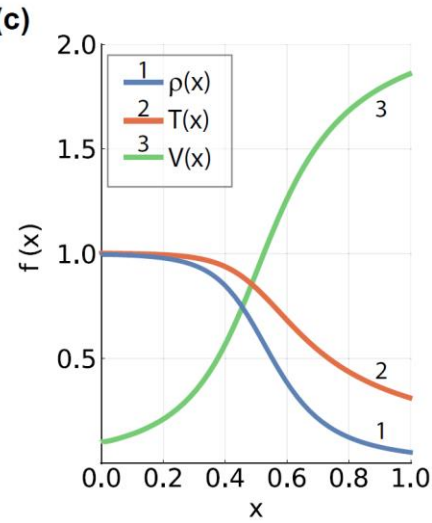
Ok, so it works.

DQCs for nonlinear DEs

Finally, we consider an example from fluid dynamics, being **quasi-1D Navier-Stokes equations** for the **convergent-divergent nozzle**.







$$\frac{\partial \rho}{\partial t} = -\rho \frac{\partial V}{\partial x} - \rho V \frac{\partial (\log A)}{\partial x} - V \frac{\partial \rho}{\partial x},$$
$$\frac{\partial T}{\partial t} = -V \frac{\partial T}{\partial x} - (\gamma - 1) T \left(\frac{\partial V}{\partial x} + V \frac{\partial (\log A)}{\partial x} \right)$$
$$\frac{\partial V}{\partial t} = -V \frac{\partial V}{\partial x} - \frac{1}{\gamma} \left(\frac{\partial T}{\partial x} + \frac{T}{\rho} \frac{\partial \rho}{\partial x} \right),$$

Navier-Stokes equations

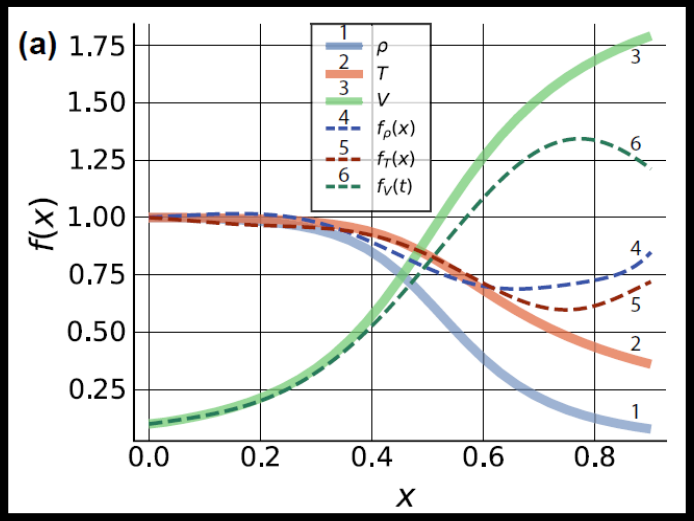
Given the nozzle profile $A(x)$ and **initial conditions**, our goal is to find the steady state profile for **density**, **temperature**, and **velocity**.

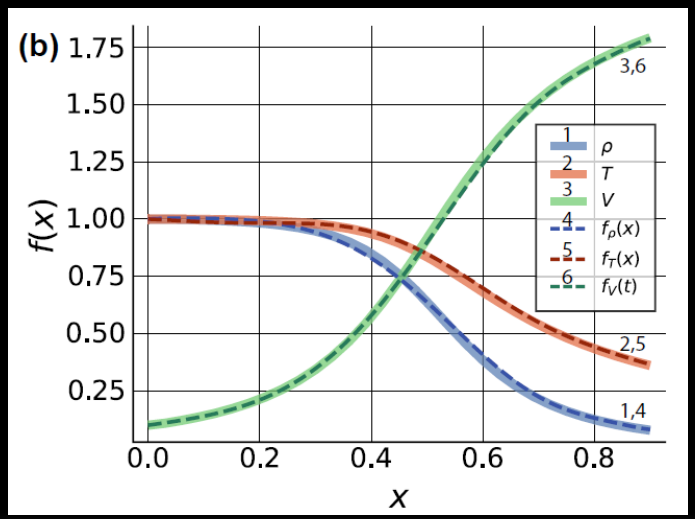
Classically, it is not easy to integrate this system due to divergence at the nozzle throat.

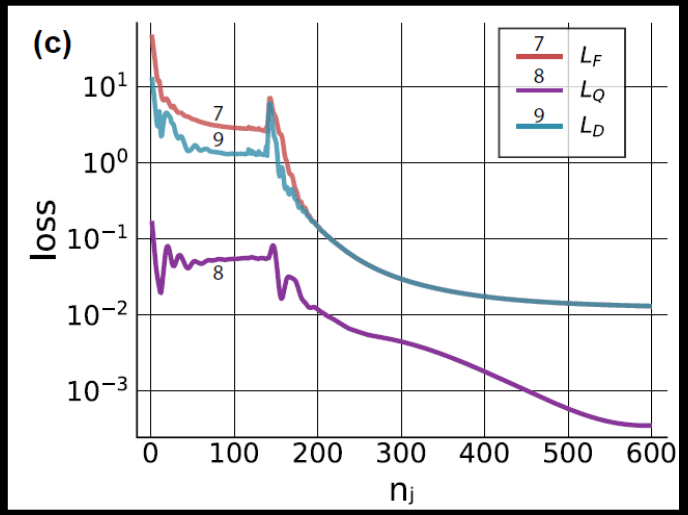
How can we solve these?

DQCs for nonlinear DEs

We train DQC to solve the problem avoiding the divergent region, and show that the solver is able to capture the behaviour.







- train from initial point $x = 0$ and before the throat
- use 20 points and 200 epochs of Adam with 0.01 learning rate

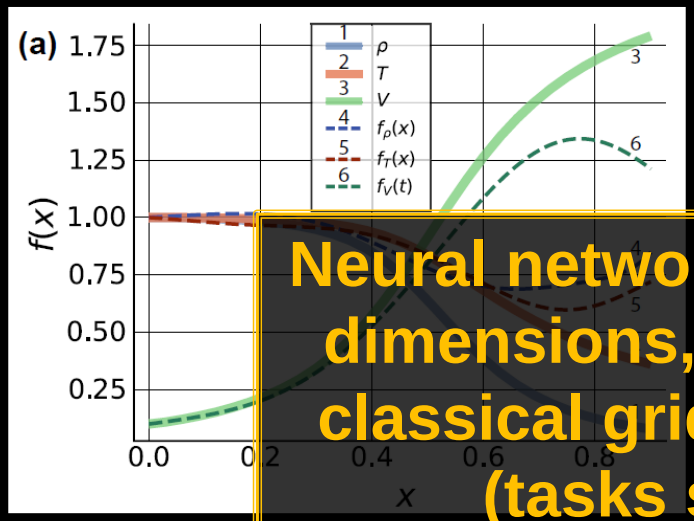
Train the remaining part with 40 points (avoiding the throat) using the first stage for regularization (150 epochs), for 600 epochs in total.

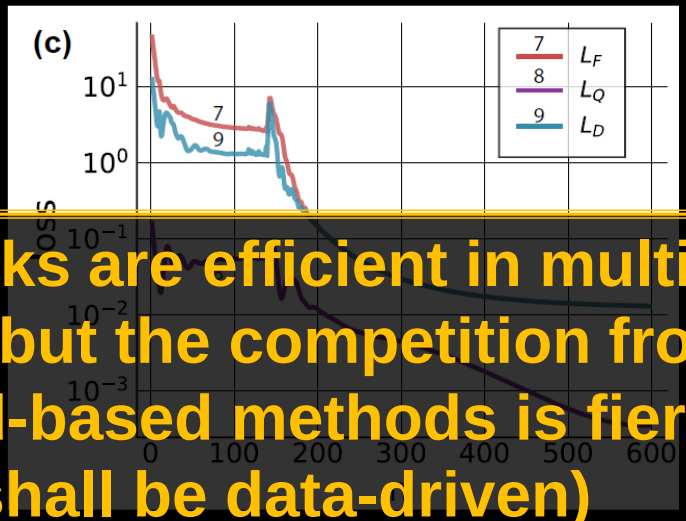
Good quality of solution is obtained even with limited resources.

[OK, A. E. Paine, V. Elfving, Phys. Rev. A 103, 052416 (2021)]

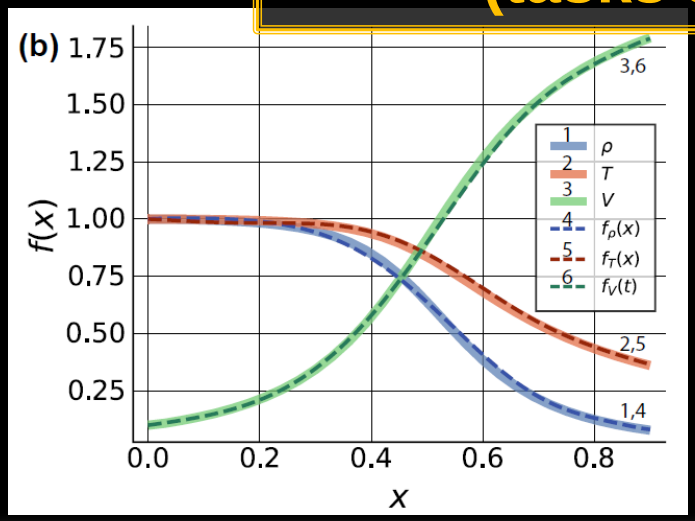
DQCs for nonlinear DEs

We train DQC to solve the problem avoiding the divergent region, and show that the solver is able to capture the behaviour.





Neural networks are efficient in multiple dimensions, but the competition from classical grid-based methods is fierce (tasks shall be data-driven)



- train from initial point $x = 0$ and before the throat
- use 20 points and 200 epochs of Adam with 0.01 learning rate

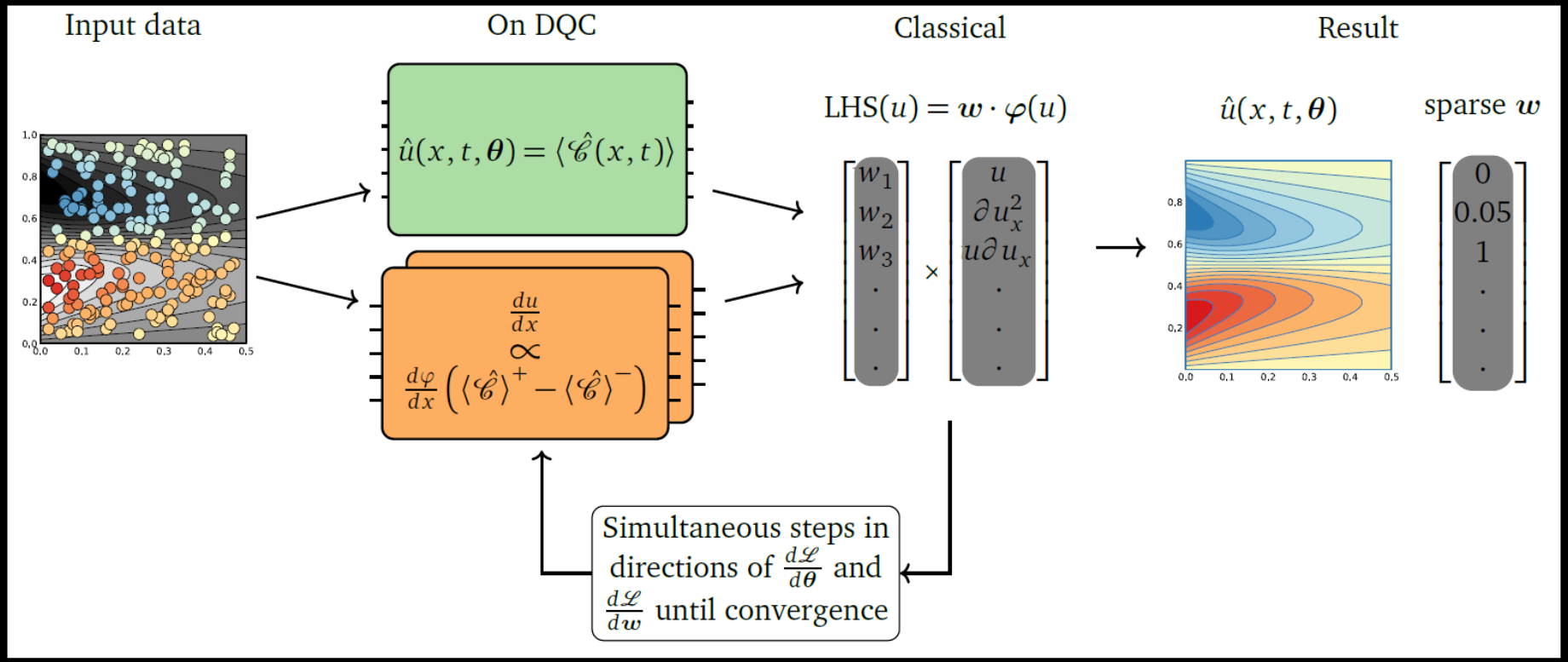
Train the remaining part with 40 points (avoiding the throat) using the first stage for regularization (150 epochs), for 600 epochs in total.

Good quality of solution is obtained even with limited resources.

[OK, A. E. Paine, V. Elfving, Phys. Rev. A 103, 052416 (2021)]

Quantum model discovery

Our goal is to build data-driven physical models. This can be performed in a form of **model discovery**. This corresponds to **QMoD**, in analogy to DNN-based approaches.



flow chart of quantum model discovery

We minimize loss on sparse data where optimal model and DE terms are selected by gradient descent,

[N. Heim, A. Ghosh, OK, V. Elfving, arXiv:2111.06376 (2022)]

Quantum model discovery

Possible test problems include learning **Lotka-Volterra equations**.

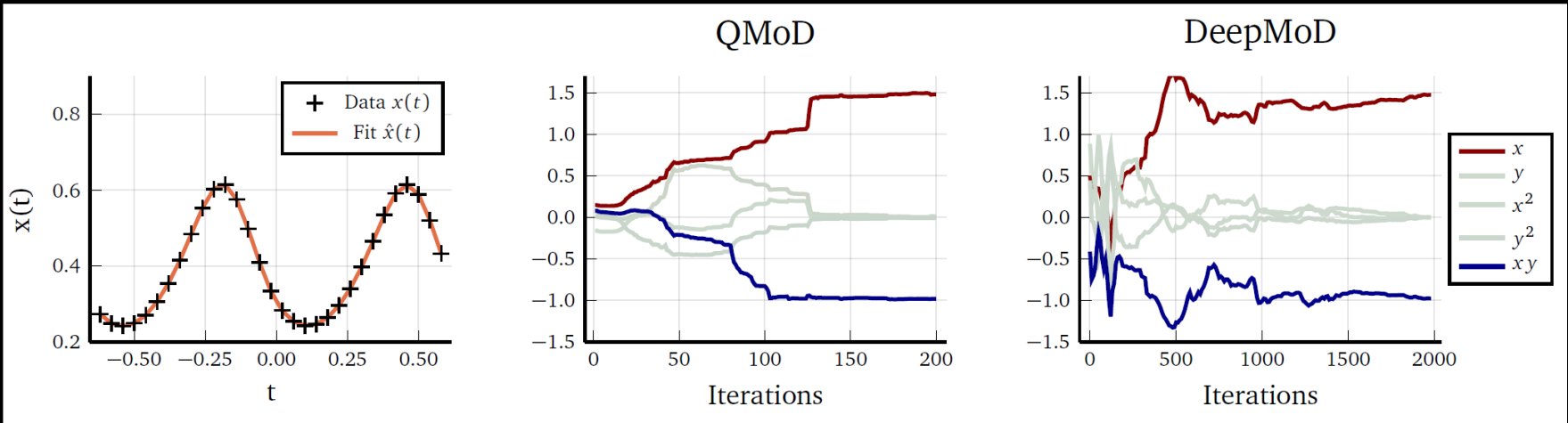
[N. Heim, A. Ghosh, OK, V. Elfving, arXiv:2111.06376 (2022)]

$$\mathcal{L}_F = \frac{1}{M}\sum_{i=1}^{M} \|F_w(\hat{u}, x_i, t_i)\|_2$$

model loss

$$\mathcal{L}_d = \frac{1}{N}\sum_{i=1}^{N} \|\hat{u}(x_i, t_i, \theta) - y_i\|_2$$

data loss

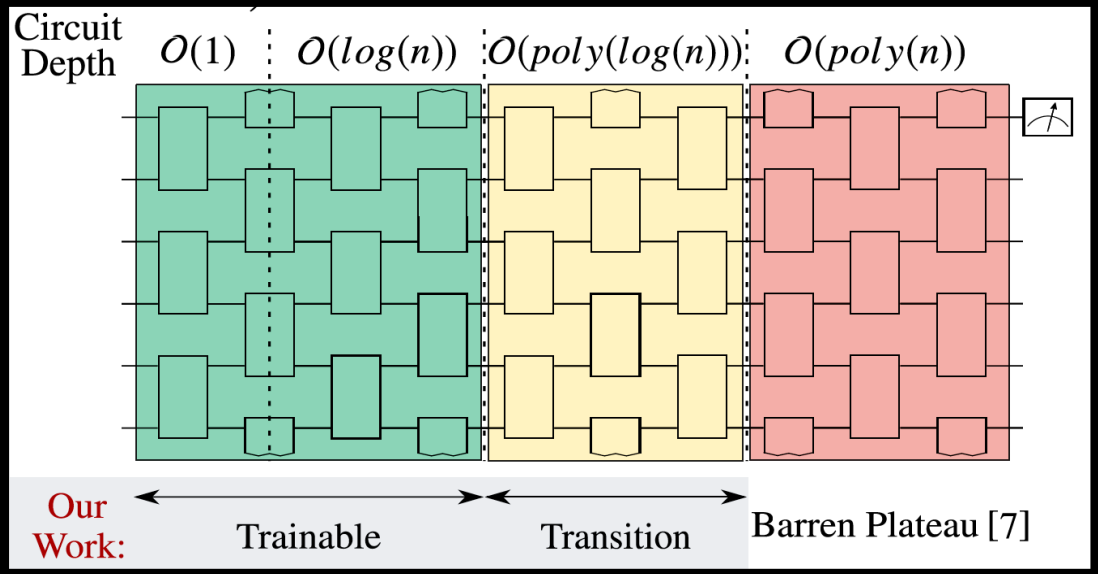


recovered dynamics of LS system

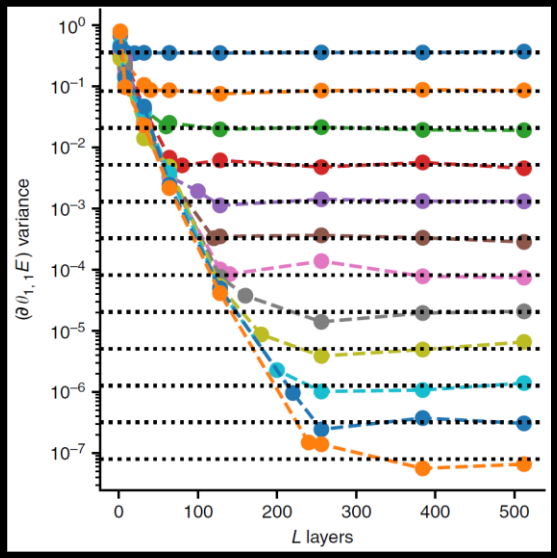
Equation	Model	Basis function x	y	x^2	y^2	xy
$\frac{dx}{dt} = \alpha t_c x - t_c x_c \beta x y$	QMoD	$\alpha = 1.483$	0.001	−0.001	−0.007	$\beta = 0.986$
	DeepMoD	$\alpha = 1.475$	$< 10^{-3}$	$< 10^{-3}$	−0.003	$\beta = 0.979$
	Truth	$\alpha = 1.5$	0	0	0	$\beta = 1$

Barren plateaus

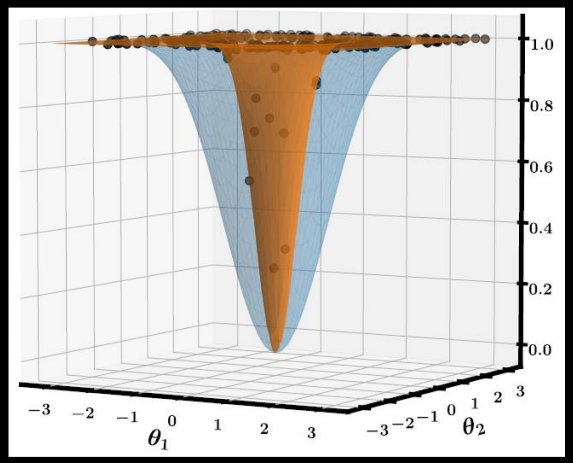
One of the major questions for QNN training corresponds to **barren plateaus** – we try to navigate a **landscape** projected from **exponentially-large Hilbert space**.



emergence of barren plateaus



variance vs depth and # qubits



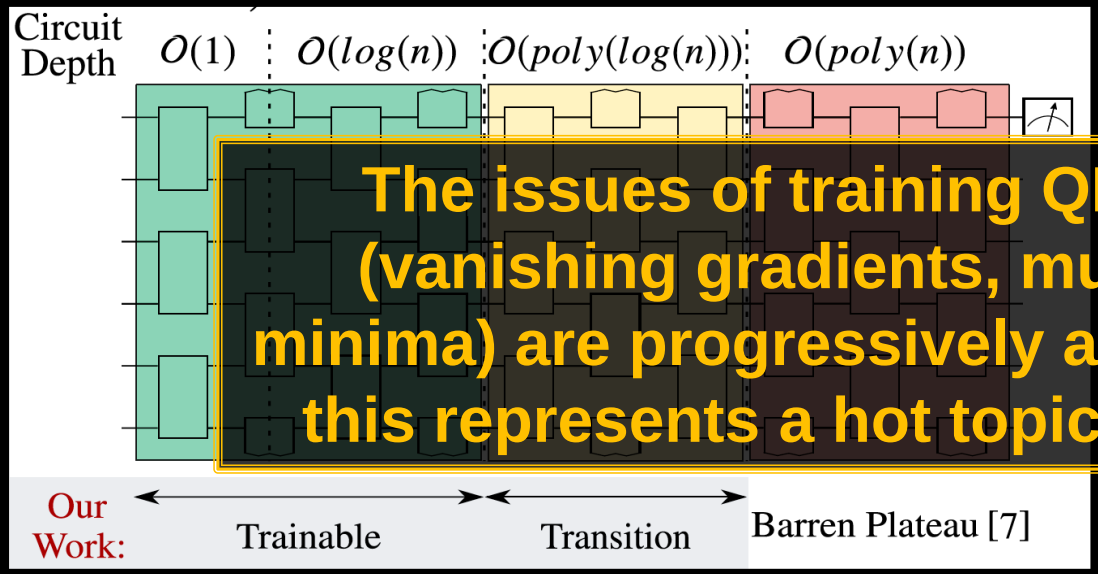
loss landscape

- For <u>some circuits</u> the variance of gradients drops exponentially in the number of qubits – QNN is not trainable.
- Recent results that for convolutional architecture this can be overcome, with polynomial scaling (trainable QNNs).

[J. R. McClean et al. (Google), Nat. Comun. (2018); M. Cerezo et al., Nat. Commun. (2021)]

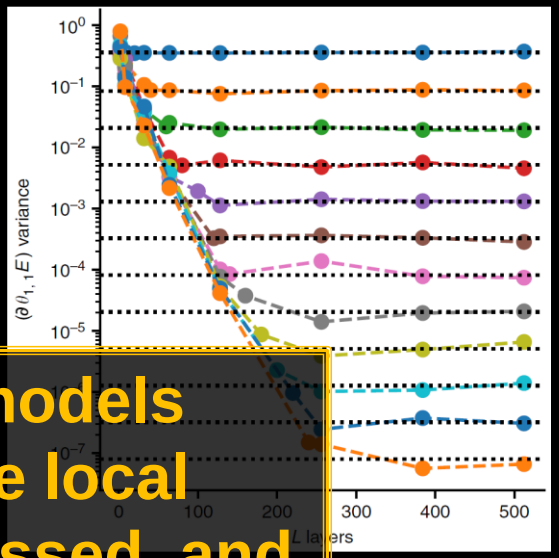
Barren plateaus

One of the major questions for QNN training corresponds to **barren plateaus** – we try to navigate a **landscape** projected from **exponentially-large Hilbert space**.

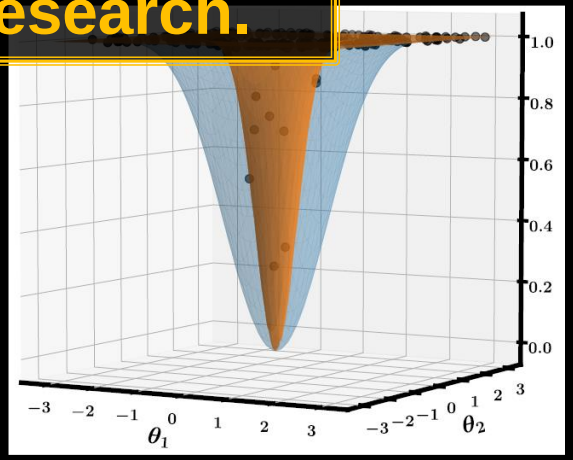


emergence of barren plateaus

The issues of training QML models (vanishing gradients, multiple local minima) are progressively addressed, and this represents a hot topic of research.



and # qubits

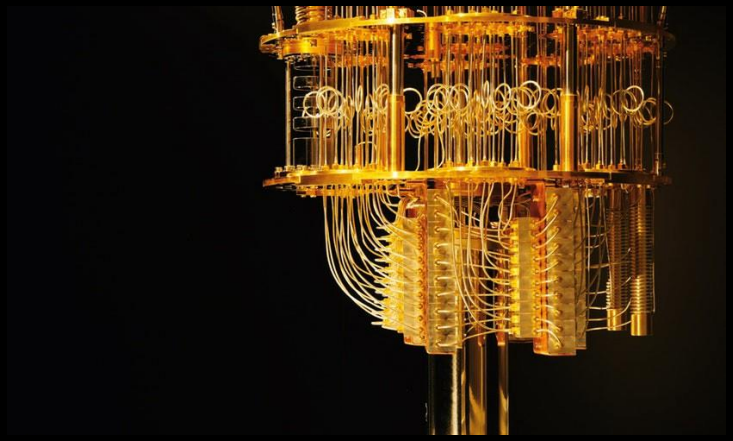


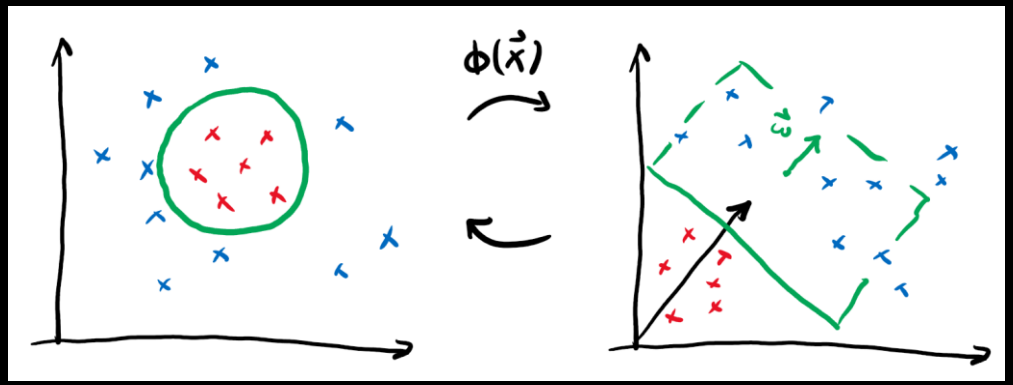
loss landscape

- For some circuits the variance of gradients drops exponentially in the number of qubits – QNN is not trainable.
- Recent results that for convolutional architecture this can be overcome, with polynomial scaling (trainable QNNs).

[J. R. McClean et al. (Google), Nat. Comun. (2018); M. Cerezo et al., Nat. Commun. (2021)]

Quantum Kernel Machine Learning





Quantum differential kernels

We can use **quantum kernels** for solving **nonlinear regression** and **differential equations** with convex optimization.

$$f(x) = \mathbf{w}^\dagger \boldsymbol{\varphi}(x) + b \qquad \Omega_{i,j} = \kappa(x_i, x_j)$$

$$f_\alpha(x) = b + \sum_{i=1}^{|\mathbf{y}|} \alpha_i \kappa(x, y_i)$$

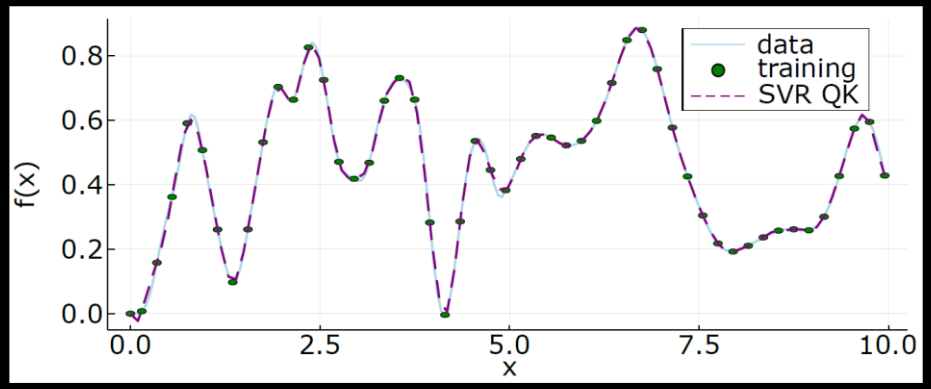
mixed model regression (MMD)

$$\left[\begin{array}{c|c} \Omega + \hat{I}/\gamma & \mathbf{1} \\ \hline \mathbf{1}^T & 0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline b \end{array}\right] = \left[\begin{array}{c} \mathbf{f} \\ \hline 0 \end{array}\right]$$

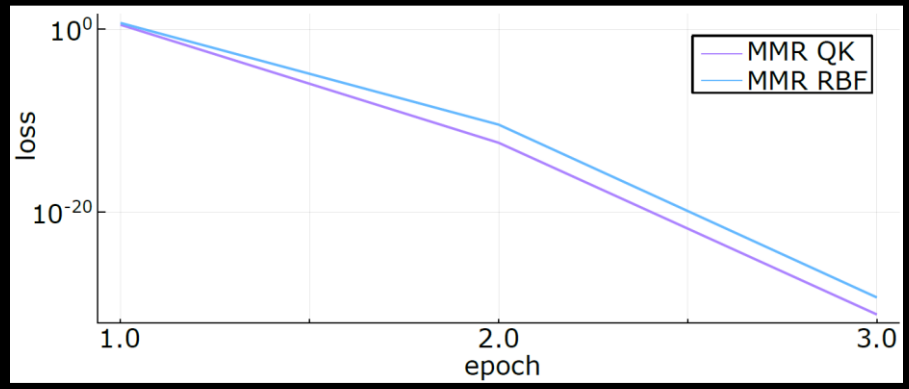
support vector regression (SVR)

For **MMD** we simply form L2 loss, and apply convex optimisation as $\partial^2 \mathcal{L} / \partial \alpha_j^2 \geq 0$

For **SVR** we use dual variables, Lagrangian formulation, and Karush-Kuhn-Tucker (KKT) optimality conditions to find f using the kernel trick.



reproducing quantum correlations



MMR training with Newton's method

[A. E. Paine, V. Elfving, OK; arXiv:2203.08884 (2022)]

Quantum differential kernels

We can exploit the same approach for solving differential equations with **quantum kernels.**

While general solvers are not yet known (active research in classical ML), we suggest solvers for various cases.

$$\mathrm{DE}(x, f, df/dx) = \\ = df/dx - g(x, f) = 0$$

differential constraint

$$[\Omega_n^m]_{i,j} = \nabla_n^m \kappa(x_j, x_i)$$
$$\tilde{\Omega}_n^m = \Omega_n^m + \hat{I}/\gamma,$$
$$[\mathbf{h}_n^m]_i = \nabla_n^m \kappa(x_0, x_i)$$

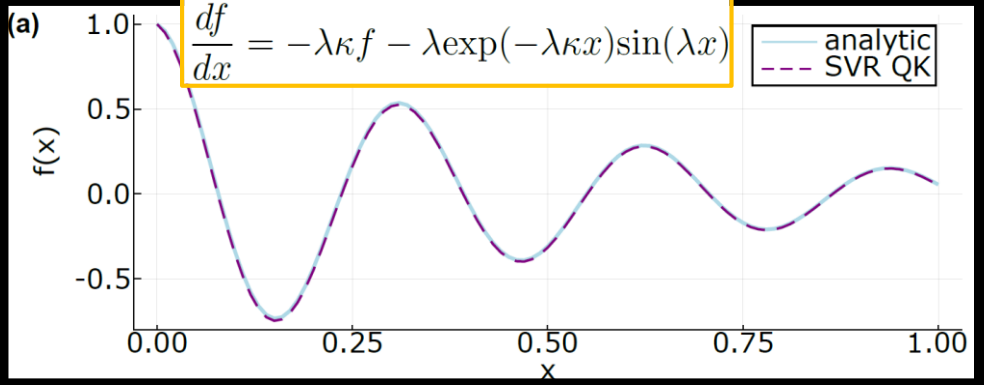
kernel derivatives

We developed quantum kernel differentiation in different forms, and used it to several problems.

[A. E. Paine, V. Elfving, OK; arXiv:2203.08884 (2022)]

$$\left[\begin{array}{c|c|c|c|c} \tilde{\Omega}_1^1 & \Omega_0^1 & \mathbf{h}_0^1 & \mathbf{0} & \hat{0} \\ \hline \Omega_1^0 & \tilde{\Omega}_0^0 & \mathbf{h}_0^0 & \mathbf{1} & -I \\ \hline \mathbf{h}^{T0}_1 & \mathbf{h}^{T0}_0 & \tilde{h} & 1 & \mathbf{0}^T \\ \hline \mathbf{0}^T & \mathbf{1}^T & 1 & 0 & \mathbf{0}^T \\ \hline \hat{D} & I & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \boldsymbol{\alpha} \\ \hline \boldsymbol{\eta} \\ \hline \beta \\ \hline b \\ \hline \mathbf{y} \end{array}\right] = \left[\begin{array}{c} \tilde{\mathbf{g}} \\ \hline \mathbf{0} \\ \hline f_0 \\ \hline 0 \\ \hline \hat{0} \end{array}\right]$$

system of equation for SVR-based ODE solver



SVR based solution

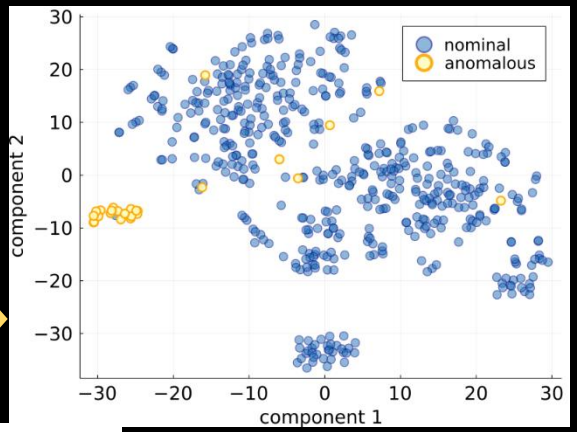
Quantum fraud detection

Another area where **quantum kernels** may be beneficial is fraud detection, where **unsupervised** machine learning is performed using **one-class support vector machine (OC-SVM)**.

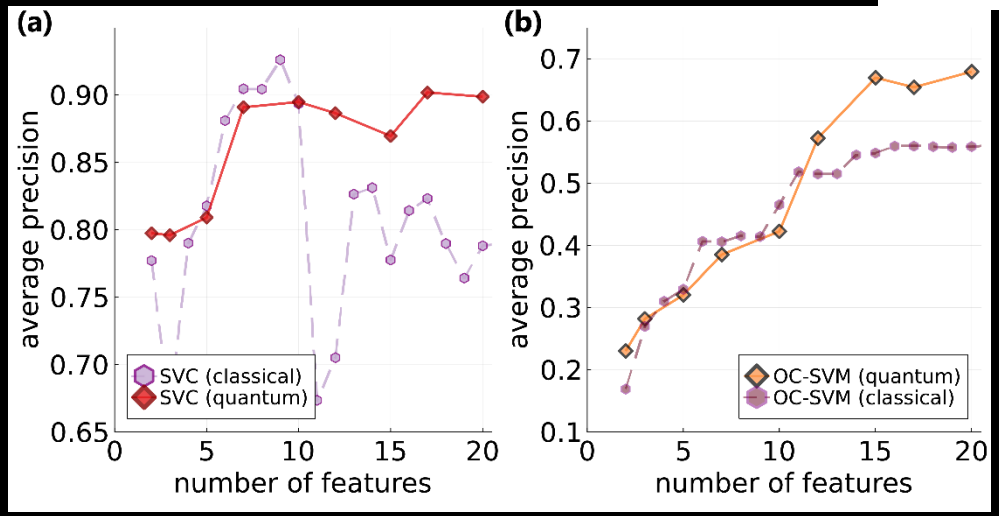
Used Kaggle dataset with PCA pre-processing, average precision (AP) metric, and various quantum embeddings (IQP, QAOA, HEA).

 # features = # qubits

t-SNE visualisation



Einar Bui Magnusson (HSBC)



quantum vs classical benchmarks

- advantage is dataset dependent (correlations)
- training is consuming due to $O(L^2)$ Gram matrix estimation, but can be reduced
- absolute clock rates of quantum computers and sampling rates are important

[OK, EBM; arXiv:2208.01203 (2022)]

Quantum fraud detection

Another area where **quantum kernels** may be beneficial is fraud detection, where **unsupervised** machine learning is performed using **one-class support vector machine (OC-SVM)**.

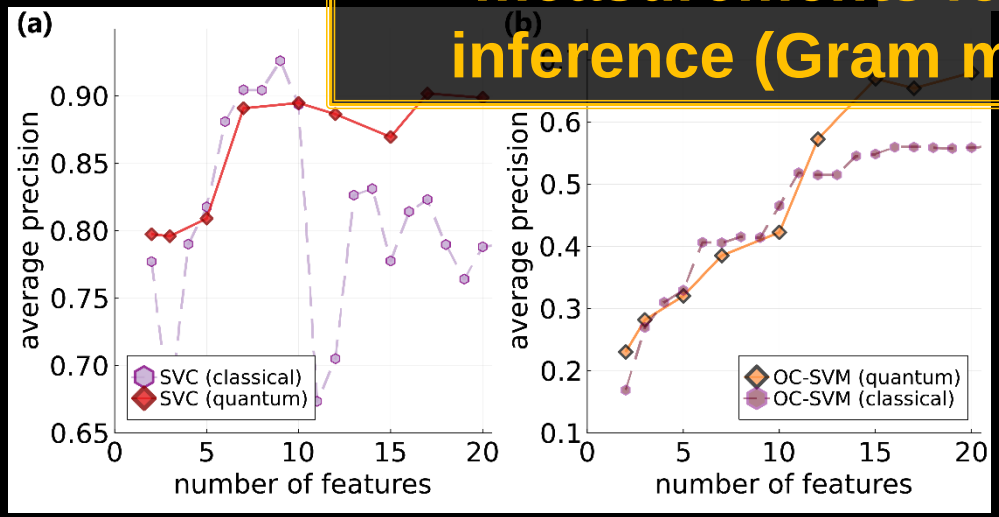
Used Kaggle dataset with PCA pre-processing, average precision (AP) metric, and various quantum embeddings (IQP, QAOA, HEA).

 # features = # qubits

visualisation

Quantum kernel methods use convex methods, but require more measurements for training and inference (Gram matrix updates)

Einar Bui Magnusson (HSBC)

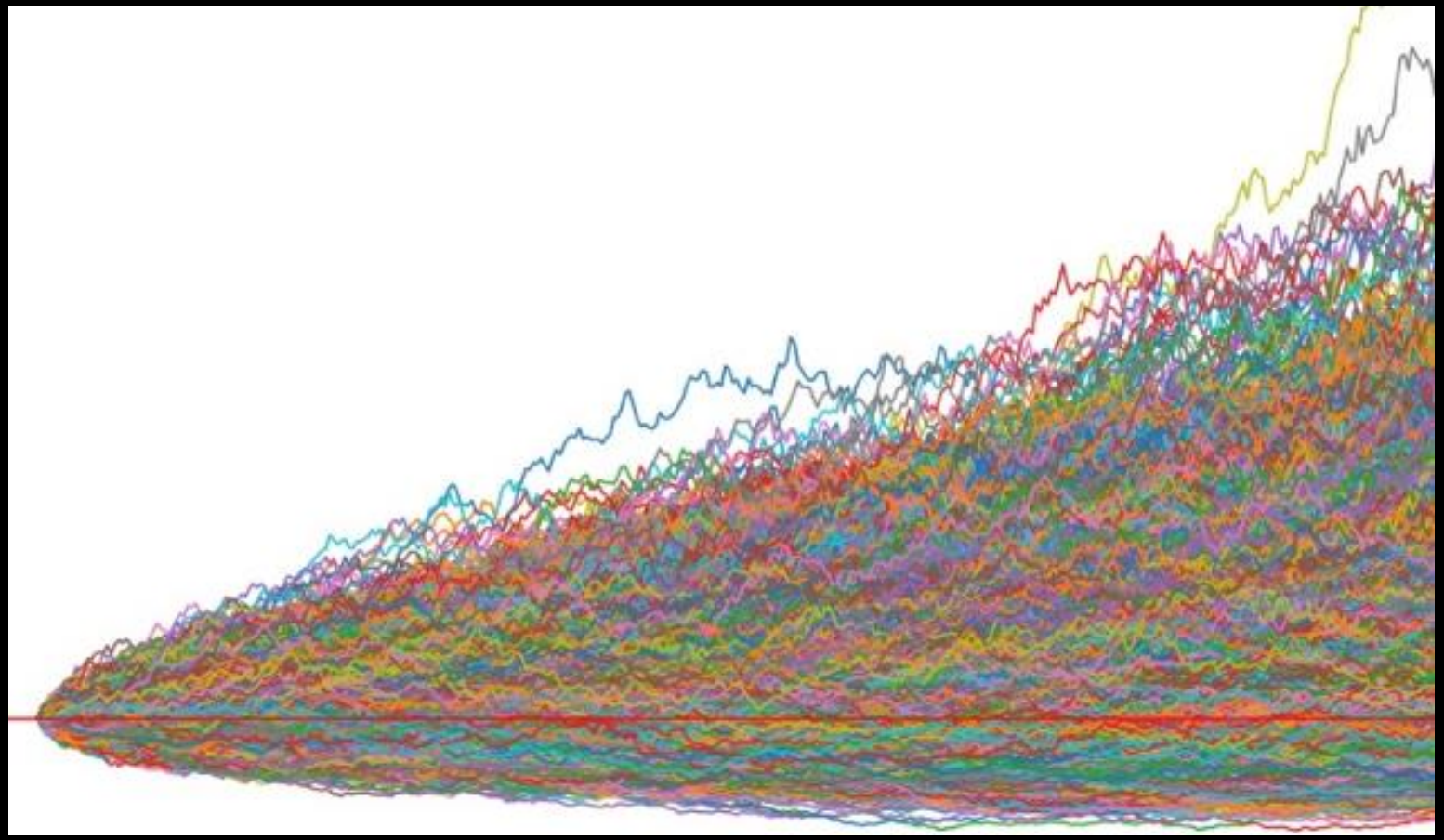


quantum vs classical benchmarks

- advantage is dataset dependent (correlations)
- training is consuming due to $O(L^2)$ Gram matrix estimation, but can be reduced
- absolute clock rates of quantum computers and sampling rates are important

[OK, EBM; arXiv:2208.01203 (2022)]

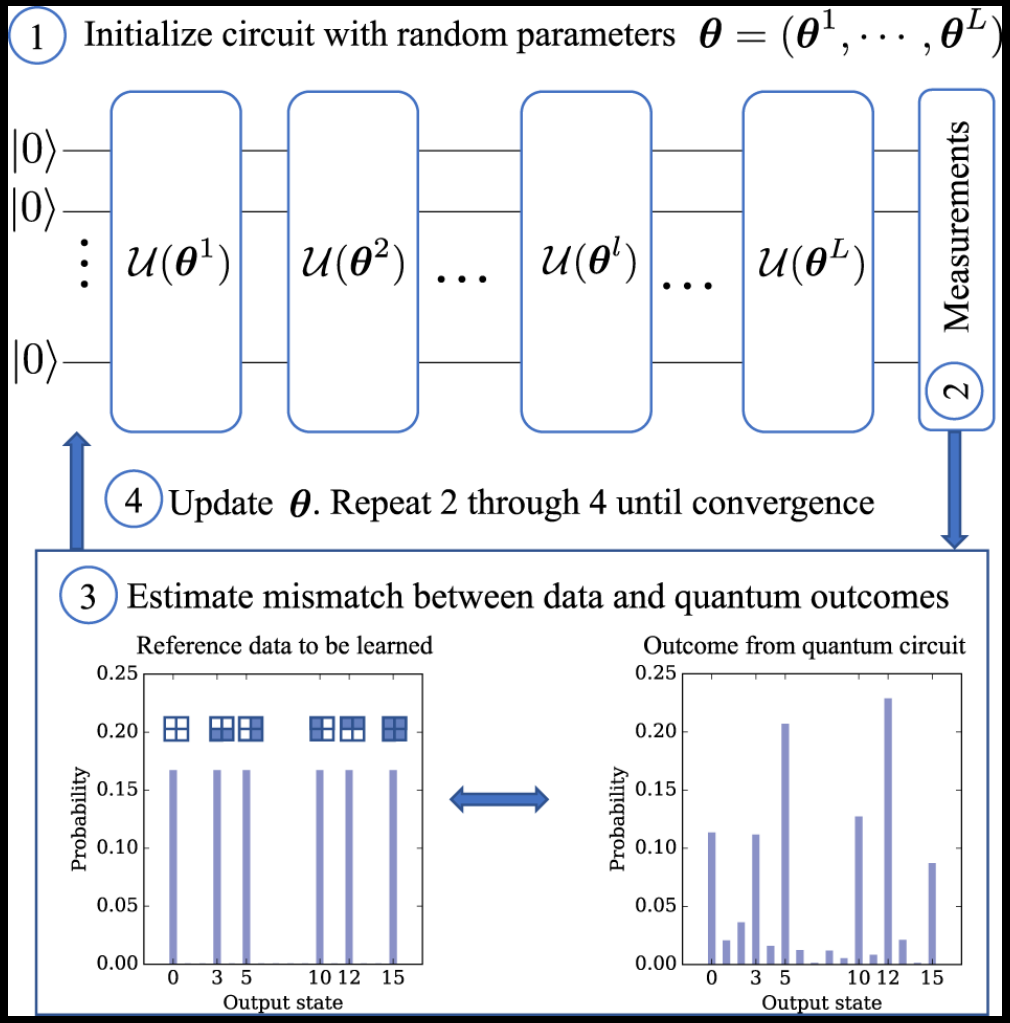
Generative modelling



ours vision of generative modelling…

Quantum generative modelling

One important QML application corresponds to **generative modelling**.



[Benedetti et al., npj Quantum Information 5, 45 (2019)]

Thanks to properties of quantum mechanics, quantum states can represent quasi-probability distribution, with each **measurement** giving a sample according to the **Born rule**

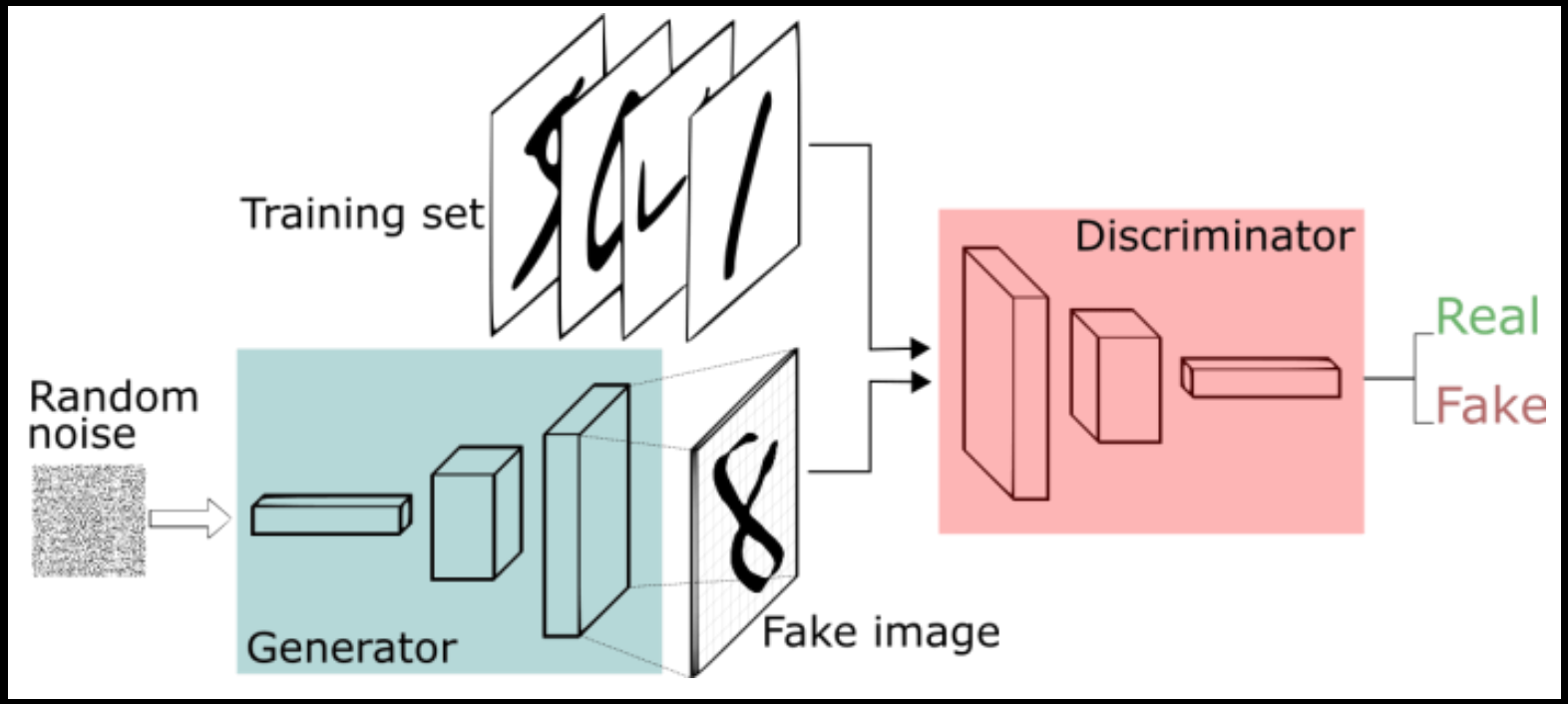
$$p_\theta(x) = |\langle x|\psi_\theta\rangle|^2$$

QCBM (quantum circuit Born machine)

- Kullback–Leibler divergence as a loss function is expensive
- Other options MMD loss, Sinkhorn divergence etc.
  [B. Coyle et al., npj Quant. Inf. 6 , 60 (2020)]
- Finally generator can be trained in the adversarial fashion (**QGAN**)

Quantum generative modelling

The goal of generative adversarial network (**GAN**) as a **variationally trained neural net** is to provide a **synthetic sample** that looks like a **real sample**, as judged by a **discriminator network**.



generative adversarial training schedule

The training is based on **game theory** approach with an objective to find **Nash equilibrium** between the two networks, **Generator** and **Discriminator**.

[Ian Goodfellow et al., Generative Adversarial Networks, arXiv:1406.2661]

Quantum generative modelling

Now let's dissect the classical GAN approach. First we assign a **latent space random variable**

$z \in \mathrm{Z} \sim p_z(z)$

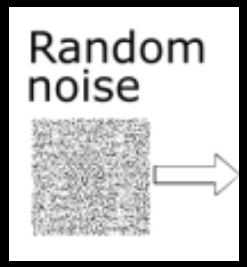
random latent variable from prior distribution

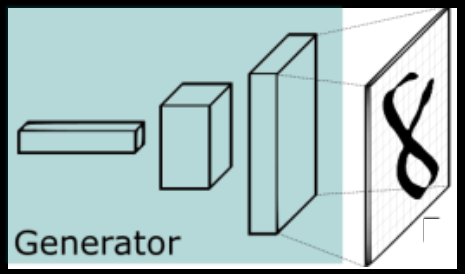
$G(\boldsymbol{z};\theta_g) \sim p_g(z)$

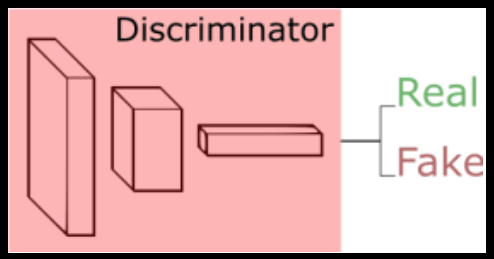
parametrized generator function

$D(\boldsymbol{x};\theta_d)$

parametrized discriminator function with single *scalar* output







(probability that a sample came from data)

Then we **train** $\boldsymbol{D}(\boldsymbol{x})$ to **maximize** the probability of assigning correct labels (since we know if it was real of fake), and simultaneously train $\boldsymbol{G}(\boldsymbol{z})$ by **minimizing** the difference

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

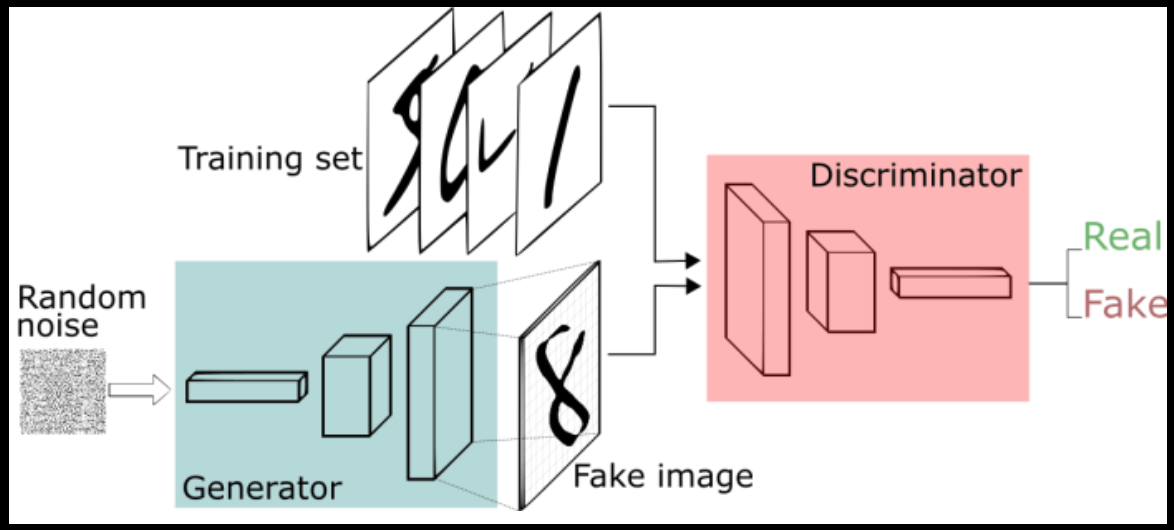
minimax game

$\max \mathrm{E}[\log(\mathrm{D}(\mathrm{G}(\mathrm{z}))]$

Quantum generative modelling

For quantum GANs we can use quantum chips for performing sampling.

[C. Zoufal et al., npj Quantum Inf. 5, 103 (2019)]



- can use **QCBM** as a **generator** with **fast sampling**
- this limits us to **discrete distributions**

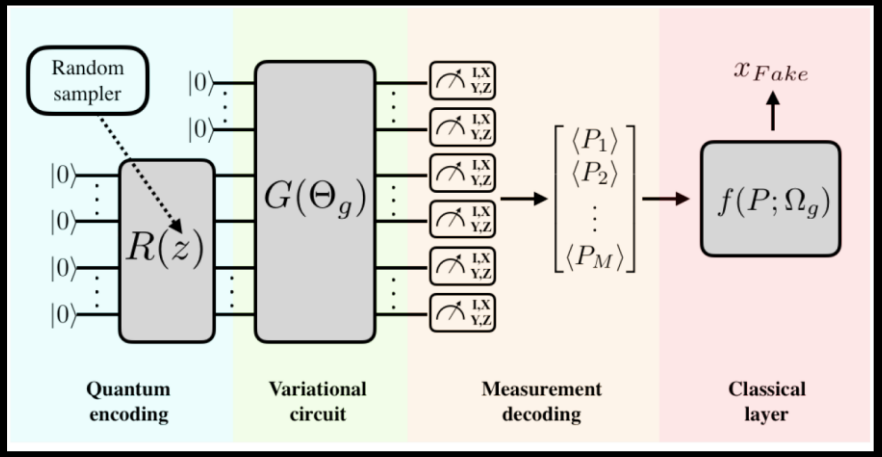
- **discriminator** can be **quantum (QNN)** or **classical (DNN)**

Note that as compared to classical GANs with QCBM we learn (quasi)probability distributions.

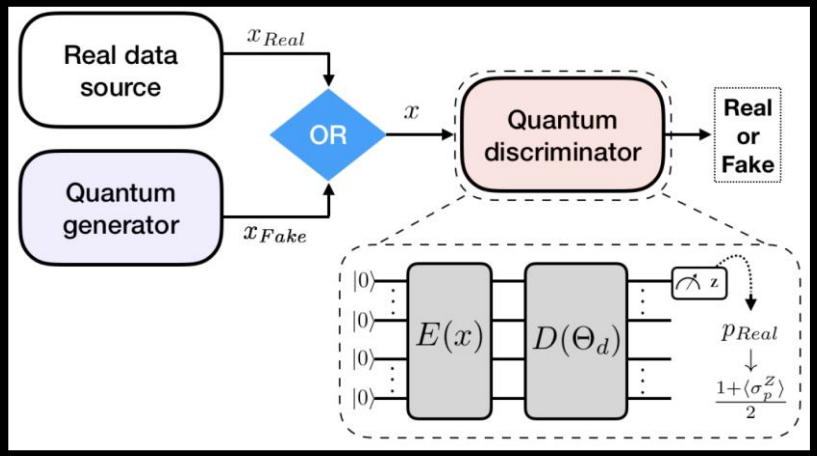
- **potential caveats** come from **trainability of QCBMs** as they are typically based on the global cost operators
- may require large number of samples to learn
- once loaded, we cannot use distributions when considering differential equations

Quantum generative modelling

A different possibility is offered when the **generator** is substituted by a **QNN**. In this case can be used to encode $G(z)$ using a **feature map** for a **latent variable** z and enable continuous **quantum generative adversarial networks.**



QGAN generator (continuous)



QGAN discriminator

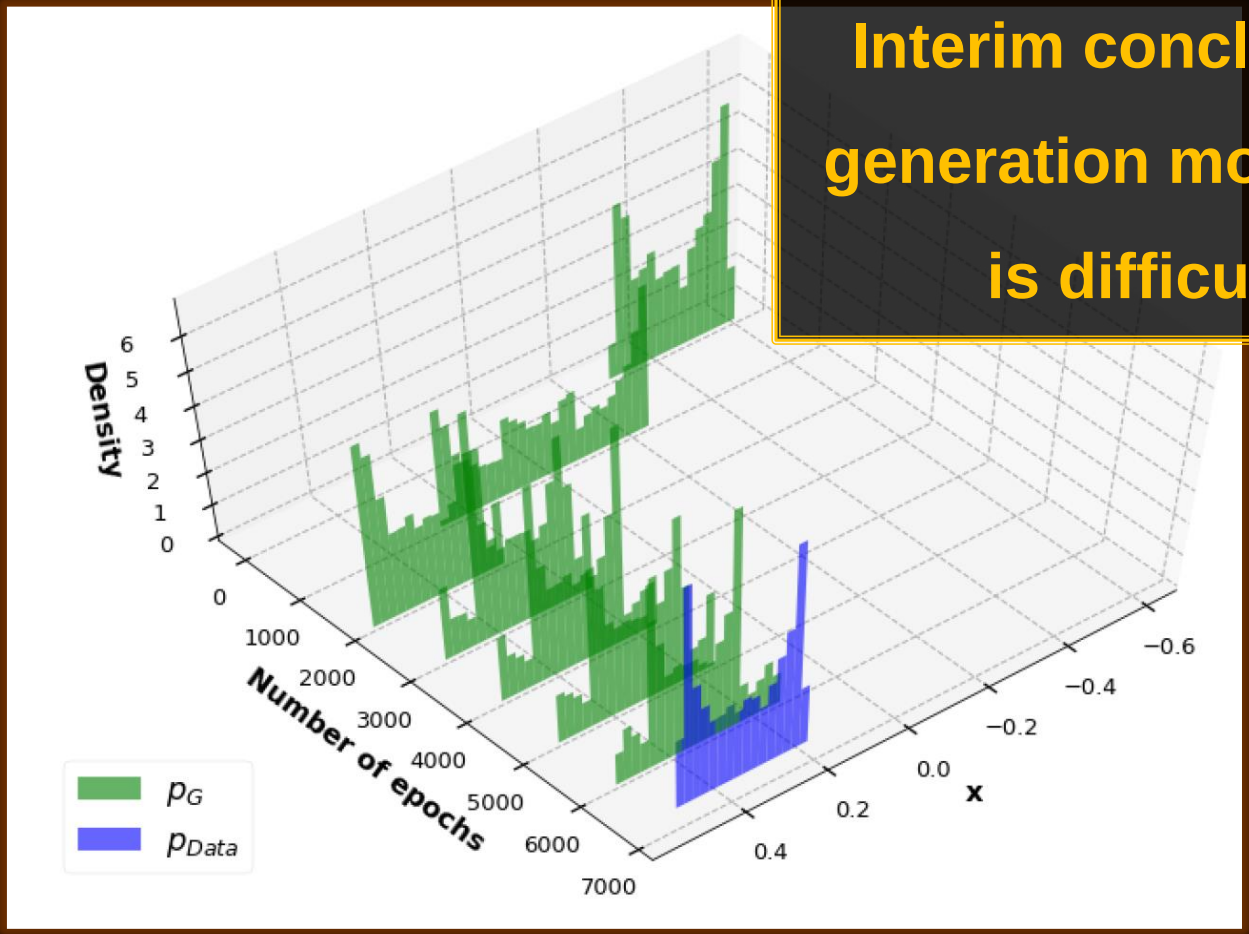
[J. Romero, A. Aspuru-Guzik, Adv. Quantum Technol. 2000003 (2020)]

In this case QGAN resembles the classical analogue, and allows using full QNN machinery. There are positive and negative points though.

- can work with continuous distributions
- training becomes more stable since we do not work with wavefunction-based samples (yet requires equilibrium)
  - to get a sample from trained probability distribution we need to measure expectations of a cost operator (increased measurement budget)

Quantum generative modelling

The example considers a **2-qubit generator**, with the synthetic data coming from pre-defined circuit configuration.



Interim conclusion: generation modelling is difficult.

QGAN training

Quantum generative modelling

We have decided to **approach the task *differently***. For this, let's go through the sampling procedure carefully and understand what we are trying to achieve.

The process we want to perform corresponds to **drawing samples X_t** (random variable) from a **probability distribution** (potentially time-dependent):

$$x \sim p(x,t)$$

Classically, the relevant steps include:

- find a **cumulative distribution function** (CDF)

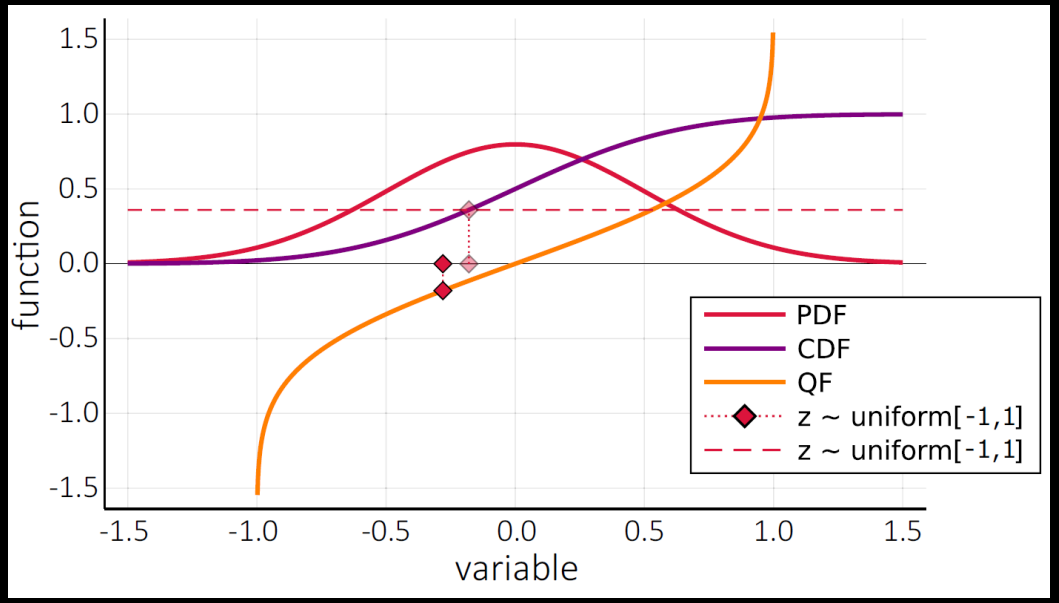
$$F_X(x) = \int_{-\infty}^{x} p(x')dx'$$

- draw a random uniform latent variable z and solve

$$z = F_X(x)$$

- requires CDF inversion
- the inverse of CDF is called a **quantile function**, that gives use samples

$$F_X^{-1}(z) \equiv Q(z)$$



sampling procedure

Quantum quantile mechanics

To summarise, for producing samples we need to construct a **quantile function** $Q(z)$ as a **monotonically increasing** function of a latent variable (almost* like GAN).

Which probability distributions shall we consider? The motivation comes from **stochastic differential equations (SDEs)**:

$$dX_t = f(X_t, t)dt + g(X_t, t)dW_t$$

stochastic differential equation

one opportunity is to solve a corresponding **Fokker-Planck equation** for $p(x, t)$ if we want to consider expectations

More general: [H. Alghassi et al., arXiv:2108.10846 (2021)]

Once we want to perform generative modelling, we should use **quantum quantile mechanics** (aka quantilized FP)

[Steinbrecher & Shaw, Quantile mechanics, Eur. J. Appl. Math. 19, 87 (2008)]

$$\frac{\partial Q(z,t)}{\partial t} = f(Q,t) - \frac{1}{2}\frac{\partial g^2(Q,t)}{\partial Q} + \frac{g^2(Q,t)}{2}\left(\frac{\partial Q}{\partial z}\right)^{-2}\frac{\partial^2 Q}{\partial z^2}$$

quantile mechanics PDE

After solving QQM equations as a function of z and t, representing quantile function in a neural form, we can perform time series generation and dataset augmentation, while learning from data and allowing for trainable SDE parameters.

We will use **differentiable quantum circuits** to solve QQM equations based on feature map differentiation and variational procedure.

Quantum quantile mechanics

Returning back to the **QQM problem**, we combine our loss function to **learn from data** and **satisfy differential equations**

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \mathcal{L}_{\text{SDE}}$$

loss function for training

$$\mathcal{L}_{\text{SDE}} = \frac{1}{M} \sum_{z,t \in \mathcal{Z},\mathcal{T}} \mathfrak{D}\left[\frac{\partial G_{\boldsymbol{\theta}}}{\partial t}, F(z, t, f, g, \frac{\partial G_{\boldsymbol{\theta}}}{\partial z}, \frac{\partial^2 G_{\boldsymbol{\theta}}}{\partial z^2})\right]$$

differential loss function

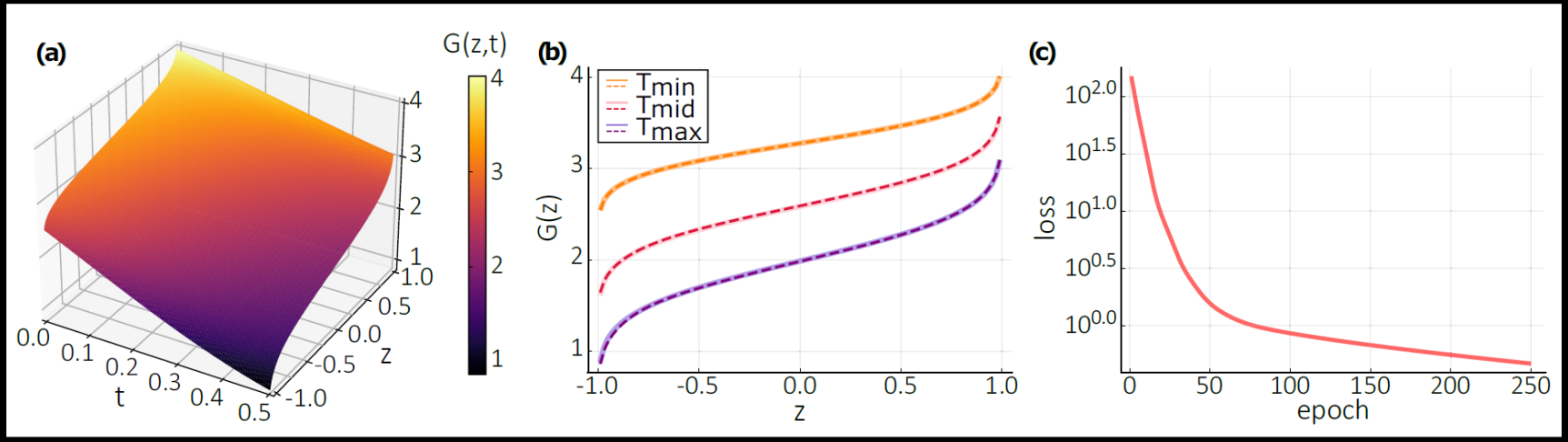
[A. E. Paine, V. Elfving, OK, arXiv:2108.03190 (2021)]

Now let's test the approach by considering **Ornstein-Uhlenbeck** process as an example

$$dX_t = \nu(\mu - X_t)dt + \sigma dW_t$$

Ornstein-Uhlenbeck SDE

The solution for OU is known, where **PDF** is **normal** and **QF** is **inverse error function**.

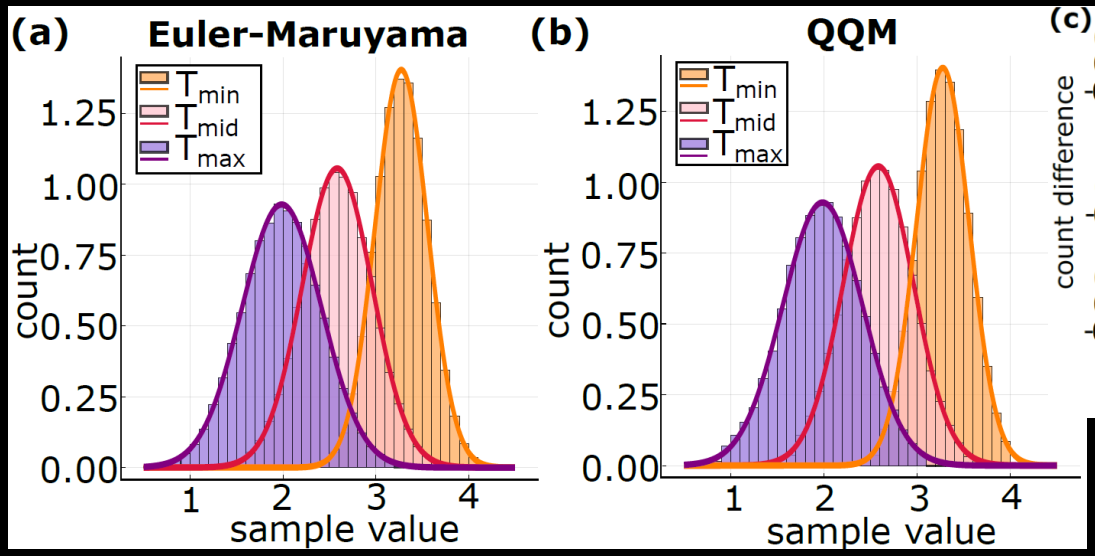


QQM training

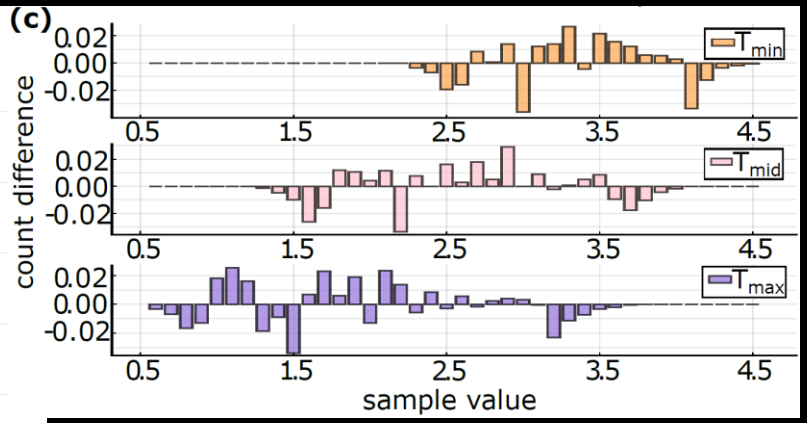
Quantum quantile mechanics

We can now perform sampling from **time-dependent quantum QF**, and compare to **classical SDE solvers**.

[A. E. Paine, V. Elfving, OK, arXiv:2108.03190 (2021)]



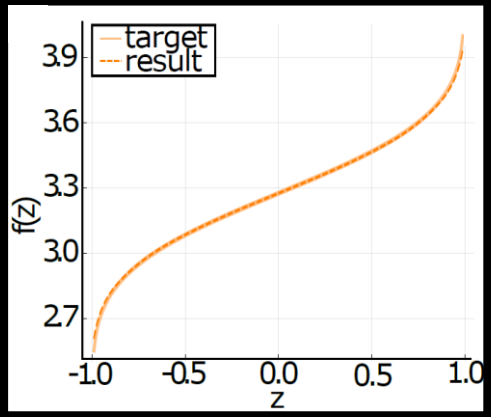
sampled SDE solutions



quantum-classical difference

While results above assume a known **initial quantile** function, we can also learn it from data by **dataset ordering**, and assigning latent variable to each bin.

- excellent* results can be obtained with relatively low number of samples and stable procedure
  - we need to mind the “tails” for high-quality sampling



data-inferred QQF

Differentiable quantum generative models

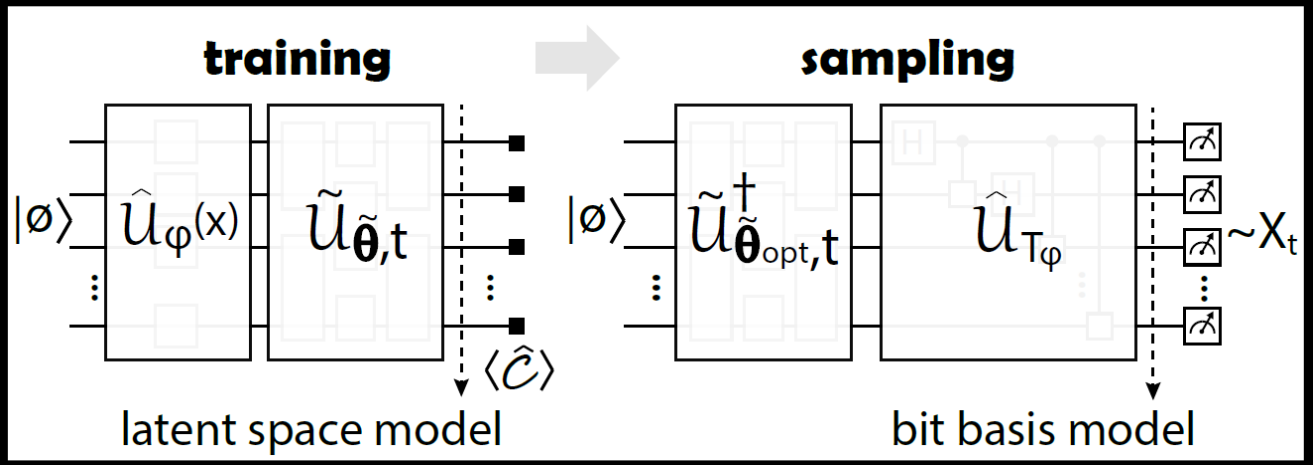
To improve sampling we need to utilize feature map encoding for PDF, yet being able to sample then in the bit basis. We combine both by constructing **differentiable quantum generative models (DQGM)**.

[OK, A. E. Paine, V. Elfving, arXiv:2202.08253 (2022)]

$$p_\theta(x) = |\langle x|\psi_\theta\rangle|^2$$

QCBM (quantum circuit Born machine)

generalize QCBM to make differentiable and



We divide training and sampling stages, where learning is in the phase (latent) space.

$$\widetilde{p}_{\tilde{\boldsymbol{\theta}},t}(x) = \mathrm{tr}\{\hat{\rho}_{\tilde{x}}\widetilde{U}^{\dagger}_{\tilde{\boldsymbol{\theta}},t}\hat{\rho}_0\widetilde{U}_{\tilde{\boldsymbol{\theta}},t}\}$$

latent space model

$$\widetilde{p}_{\tilde{\boldsymbol{\theta}},t}(x) = \mathrm{tr}\{|x\rangle\langle x|\hat{\mathcal{W}}^{\dagger}_{\tilde{\boldsymbol{\theta}},t}\hat{\rho}_0\hat{\mathcal{W}}_{\tilde{\boldsymbol{\theta}},t}\}$$

sampling

For this, we use the **phase feature map:**

$$|\widetilde{x}\rangle := \hat{\mathcal{U}}_{\varphi}(x)|\phi\rangle$$

$$\hat{\mathcal{U}}_{\varphi}(x) = \prod_{j=1}^{N}\left[\hat{R}^z_j\left(\frac{2\pi x}{2^j}\right)\hat{H}_j\right]$$

$$|\widetilde{x}\rangle = \frac{e^{-i\Phi/2}}{2^{N/2}}\bigotimes_{j=1}^{N}\left(|0\rangle_j + \exp\left(-i\frac{2\pi x}{\xi_j 2^j}\right)|1\rangle_j\right)$$

Differentiable quantum generative models

The mapping between phase (latent) space and bit-basis for sampling corresponds to **quantum Fourier transform** (efficient subroutine).

Once we have represented a **function** $\boldsymbol{p(x)}$ using **DQGM**, we can also find its **derivative** $\mathrm{d}\boldsymbol{p(x)/dx}$ using **automatic differentiation** (measurement modification), and introduce differential constraints – solve SDEs.

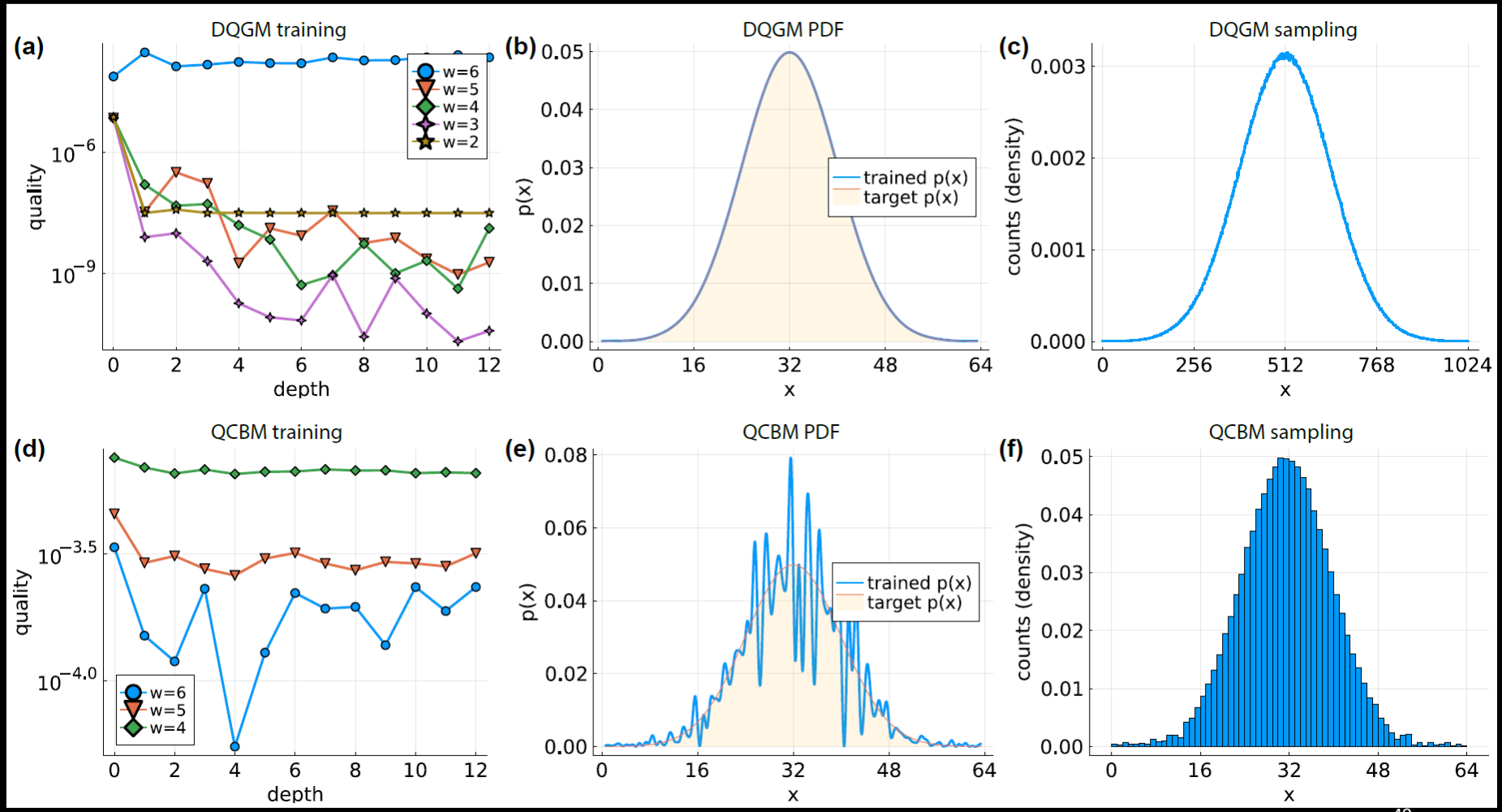
[OK, A. E. Paine, V. Elfving, arXiv:2202.08253 (2022)]

- analysis of the **phase feature map**
- DQGM and generalized QCBM
- frequency taming
- feature map sparsification
- AD via modified measurements
- Fokker-Planck constrained
- model evolution for dynamics
- multivarite distributions and building of **copula models**

Our goal is to join the quantum model expressivity (capacity) advantage and sampling advantage as utilized in "supremacy"-type advantage

Differentiable quantum generative models

The mapping between phase (latent) space and bit-basis for sampling corresponds to **quantum Fourier transform** (efficient subroutine).



Differentiable quantum generative models

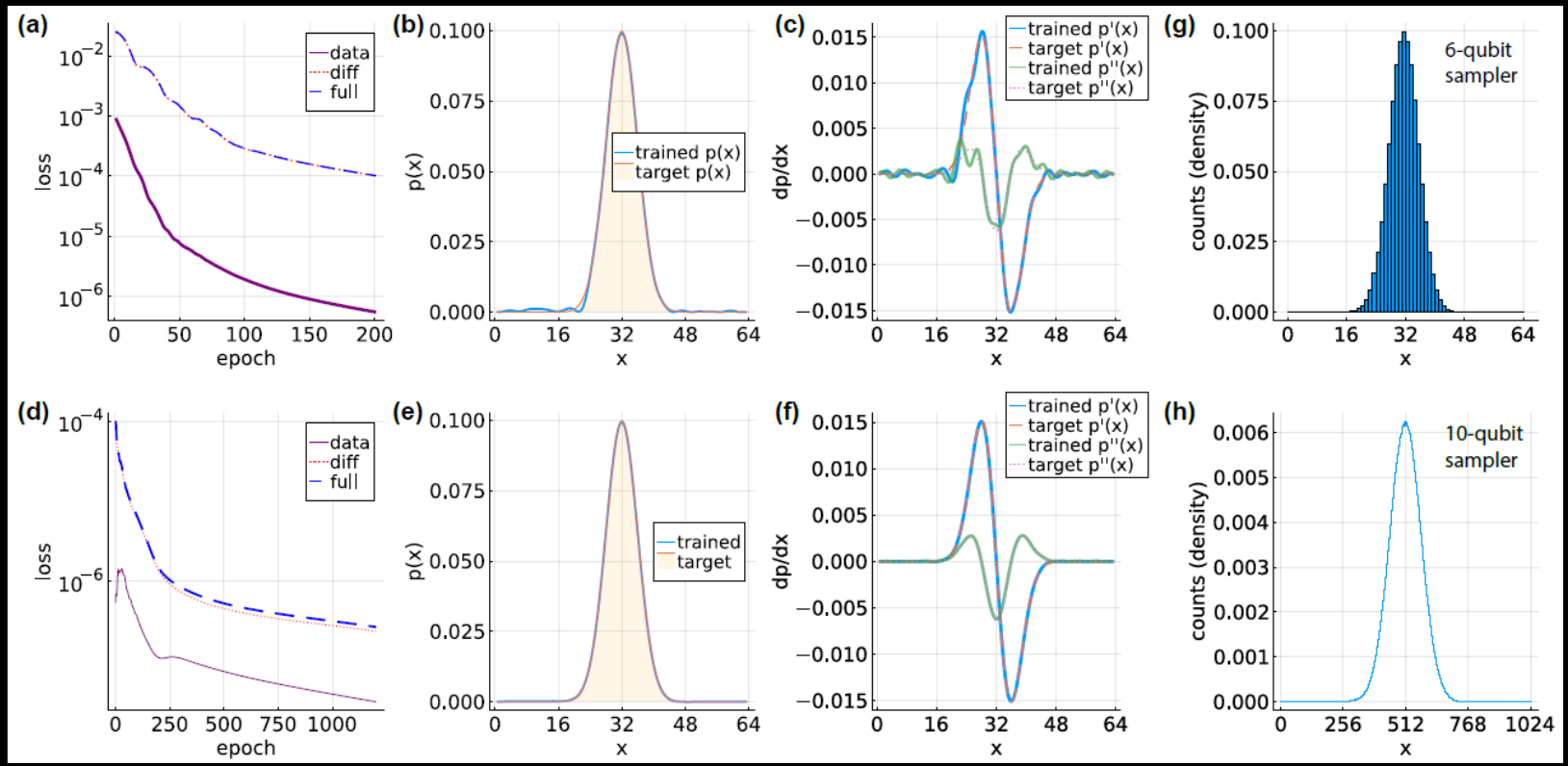
We can then continue to solve SDE for the Ornstein-Uhlenbeck process

$$dX_t = -\nu(X_t - \mu)dt + \sigma dW_t$$

SDE for OU process

$$\nu p(x, t_s) + \nu(x - \mu)\frac{d}{dx}p(x, t_s) + \frac{\sigma^2}{2}\frac{d^2}{dx^2}p(x, t_s) = 0$$

Fokker-Planck equation at stationarity



Model differentiation is important if we want to supplement **learning from data** with **learning from differential constraints** (making it physics-informed/finance-informed etc).

Differentiable quantum generative models

We can use **copula approach** to encode correlations explicitly using entangling operations, and performing training/sampling separately

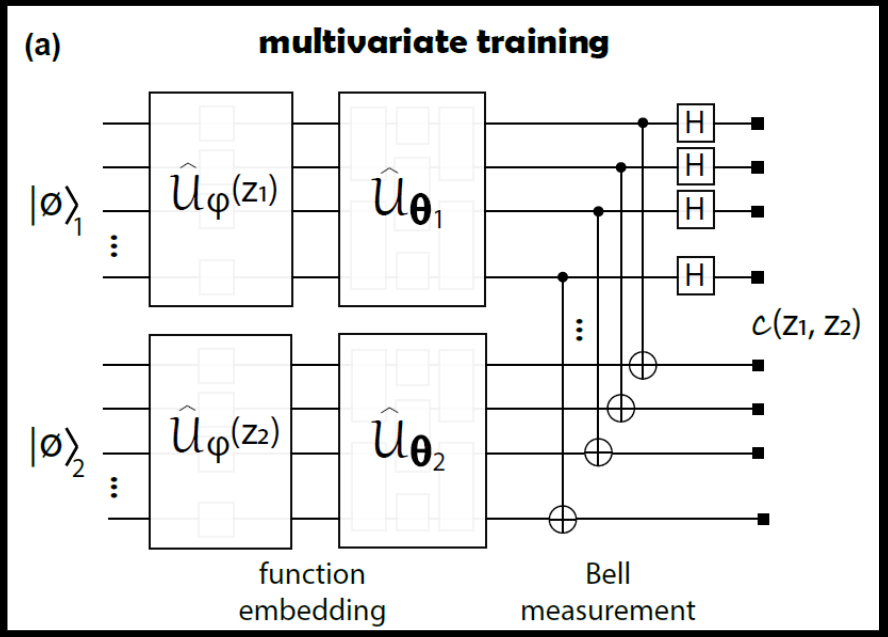
$$c[\boldsymbol{x}] = c[F_1(x_1), \ldots, F_D(x_D)]p_1(x_1) \cdot \ldots p_D(x_D)$$

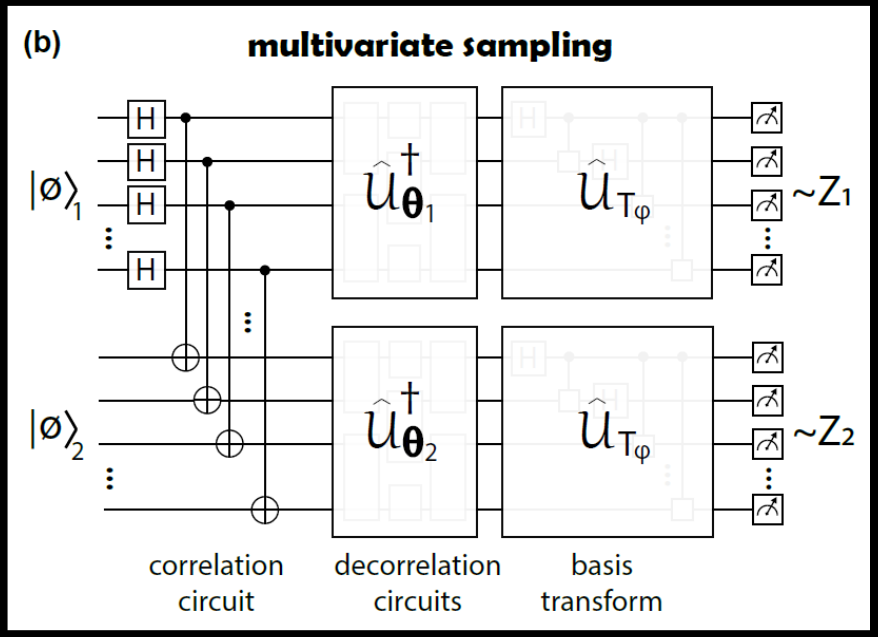
copula PDF

$$(Z_1, Z_2, \ldots, Z_D) \sim c$$

$$\boldsymbol{X} = (Q_1(Z_1), Q_2(Z_2), \ldots, Q_D(Z_D))$$

multivariate sampling

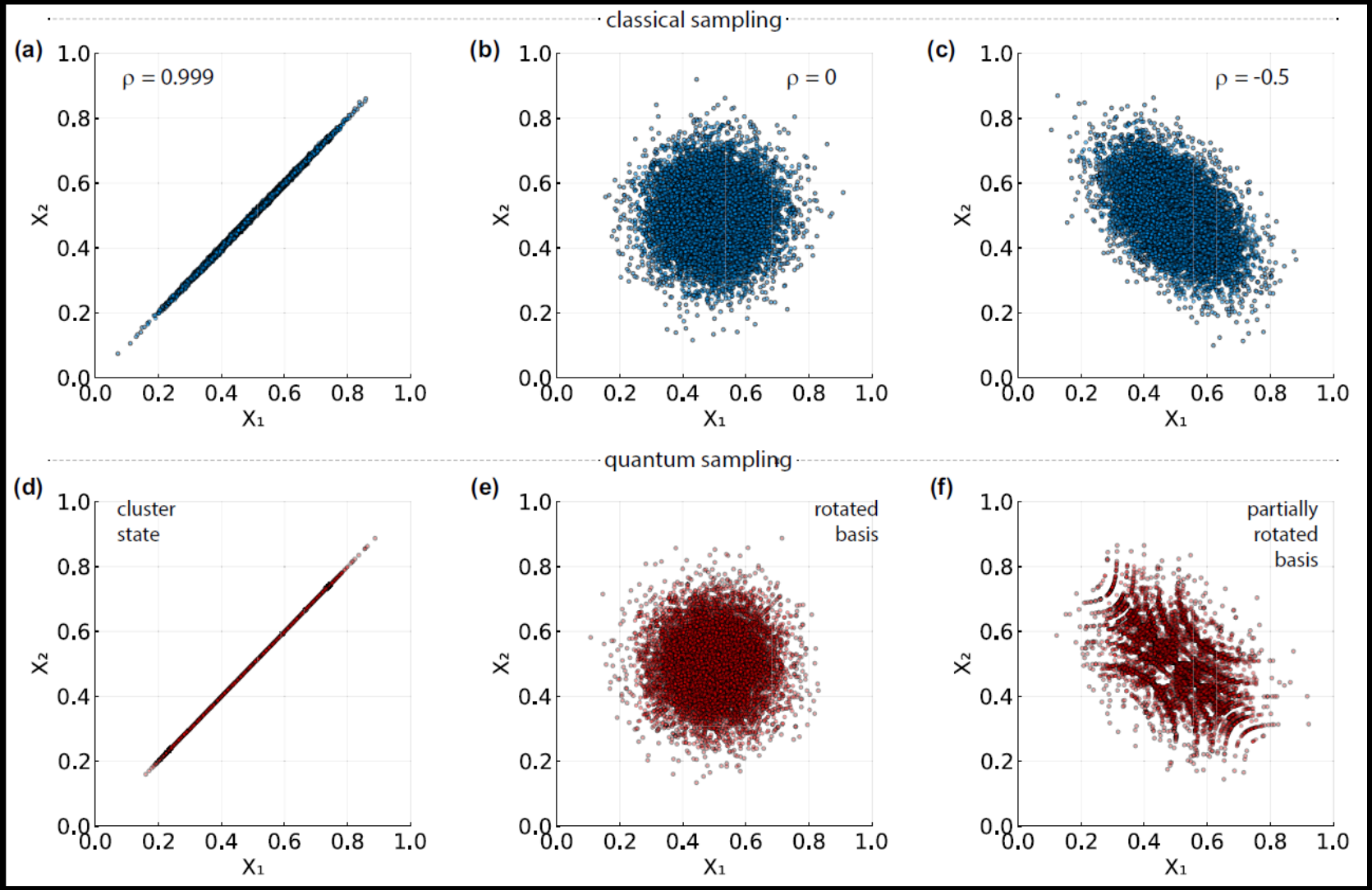




multivariate quantum copula sampling

[OK, A. E. Paine, V. Elfving, arXiv:2202.08253 (2022)]

Differentiable quantum generative models



multivariate quantum copula sampling

Conclusions and questions

- ✓ Quantum machine learning offers a powerful and innovative paradigm for performing data-driven tasks
- ✓ Quantum advantage may arise from building models based on high-dimensional quantum states
- ✓ Quantum models can be differentiated, and used for solving differentia equations
- ✓ We can also use quantum kernel-based approaches to avoid non-convex optimisation
- ✓ Generative modelling represents a task wjere quantum computers may excel
- ✓ We can exploit SDE structure to get access to samples via quantile functions, and propagate them in time (QQM)
- ✓ We can also build latent models for sampling, and get advantage in multidimensional setting