

Lecture 8

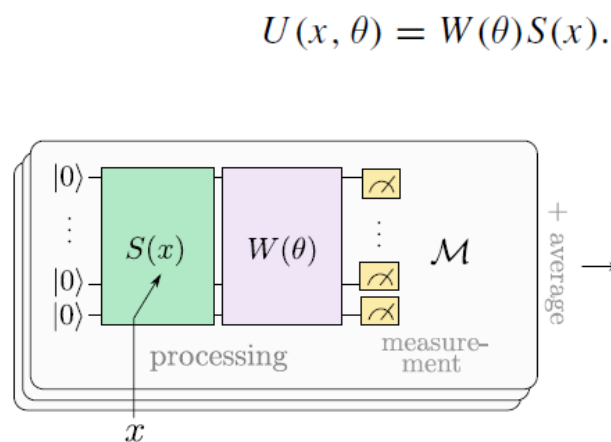
Variational Circuits as ML models

chapter 5

5	Variational Circuits as Machine Learning Models	177
5.1	How to Interpret a Quantum Circuit as a Model	179
5.1.1	Deterministic Quantum Models	179
5.1.2	Probabilistic Quantum Models	181
5.1.3	An Example: Variational Quantum Classifier	183
5.1.4	An Example: Variational Generator	185
5.2	Which Functions Do Variational Quantum Models Express?	186
5.2.1	Quantum Models as Linear Combinations of Periodic Functions	187
5.2.2	An Example: The Pauli-Rotation Encoding	190
5.3	Training Variational Quantum Models	191
5.3.1	Gradients of Quantum Computations	192
5.3.2	Parameter-Shift Rules	194
5.3.3	Barren Plateaus	197
5.3.4	Generative Training	201
5.4	Quantum Circuits and Neural Networks	203
5.4.1	Emulating Nonlinear Activations	204
5.4.2	Variational Circuits as Deep Linear Neural Networks	209
5.4.3	Time-Evolution Encoding as an Exponential Activation	212
	References	213

- Short Review on VQC's structure
- How to build cost functions
- How to draw pictures on the Bloch sphere
- Classical Optimization Methods
- Parametric Shift-Rules for the Gradients
- Expressivity
- Capacity
- Examples from bibliography

Short Review



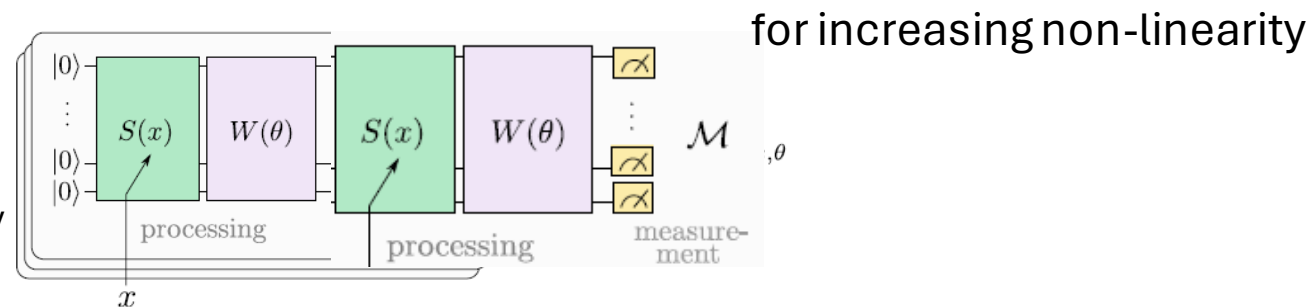
increase the number of weights in the problem (capacity/expressibility) by

$$\rightarrow f(x) = \langle \mathcal{M} \rangle_{x, \theta}$$

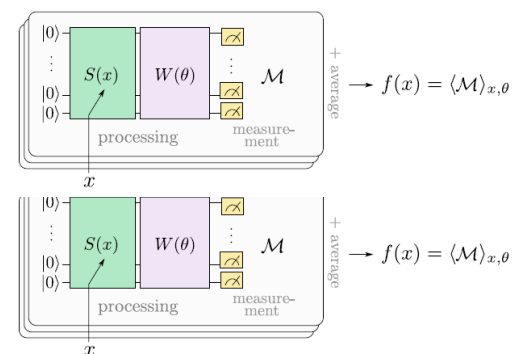


Multiple layers in series (re-uploading)

$$U(x, \theta) = W(\theta)S(x).$$

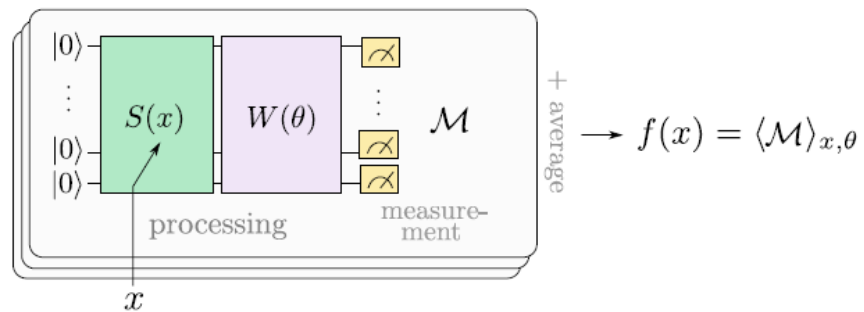


or in parallel

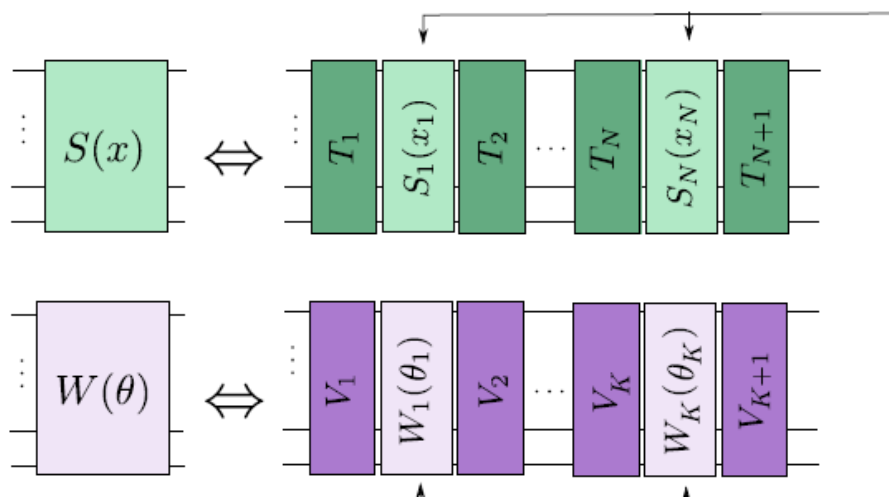


for creating polynomial feature maps

$$U(x, \theta) = W(\theta)S(x).$$



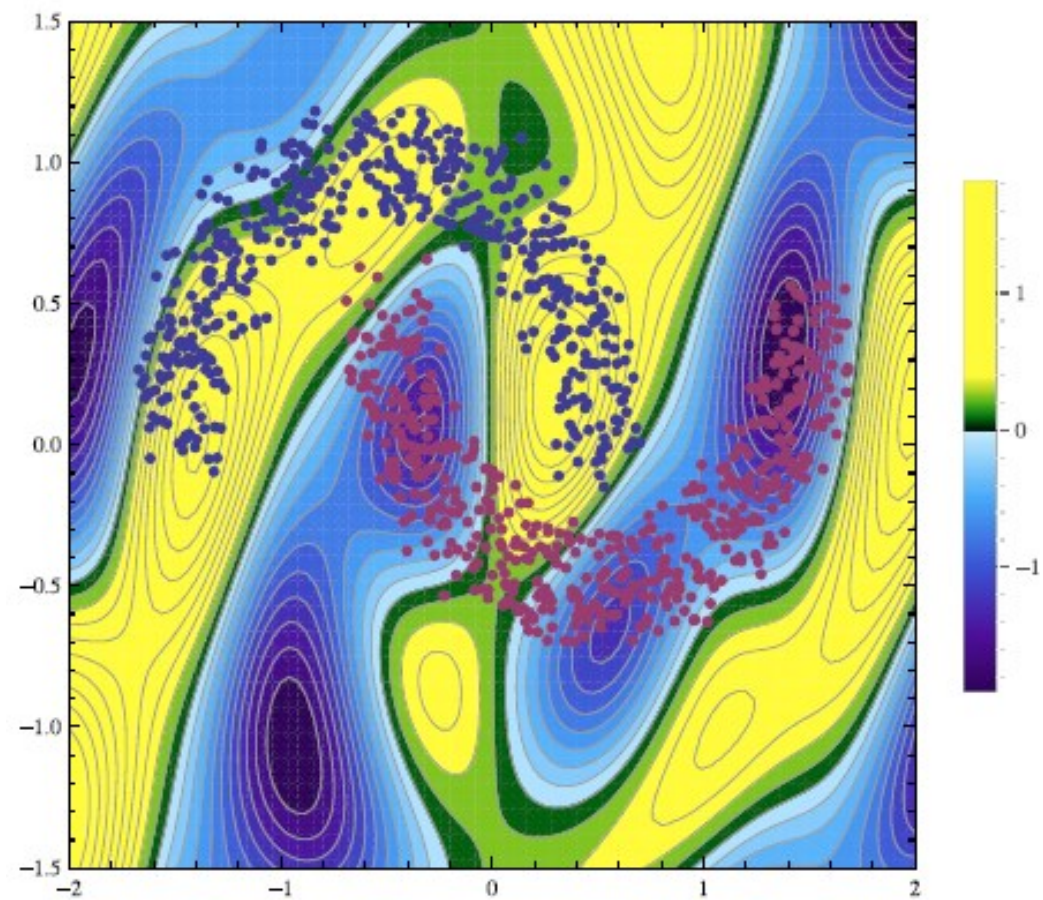
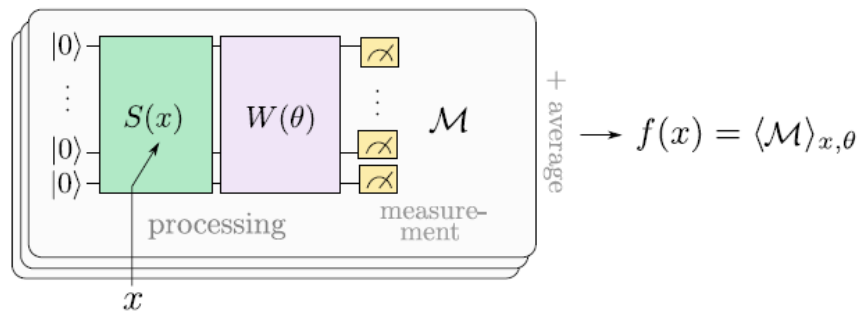
Inside:



$$S(x) = T_{N+1} \prod_{i=1}^N S_i(x_i) T_i,$$

$$W(\theta) = V_{K+1} \prod_{k=1}^K W_k(\theta_k) V_k.$$

$$U(x, \theta) = W(\theta)S(x).$$



2.2.3 Loss

In order to select the best model from a model family, we need a measure of quality of a model. Let us first look at supervised learning, where the figure of merit is formalised as a *loss* which measures how close the model predictions are to the target labels (see Fig. 2.9). There are many different loss functions, and probably the most straight forward one for classification tasks is based on the *accuracy* of a model, or the share of examples that have been classified correctly,

$$\text{accuracy} = \frac{\text{number of correctly classified examples}}{\text{total number of examples}}. \quad (2.11)$$

The corresponding loss is the *error*,

$$\text{error} = 1 - \text{accuracy}. \quad (2.12)$$

However, most training algorithms rely on a loss that is continuous-valued—which is crucial to compute gradients. One of the most popular continuous loss functions (see also Fig. 2.10), is the squared Euclidean distance between prediction and target, which is known as the *mean-squared-error loss*,

$$L(f_{\theta}(x), y) = (f_{\theta}(x) - y)^2. \quad (2.13)$$

Using the absolute value, we get the ℓ_1 instead of ℓ_2 version of the loss

$$L(f_{\theta}(x), y) = |f_{\theta}(x) - y|. \quad (2.14)$$

Another way to quantify the distance between outputs and targets is via the expression $yf(x)$, which is positive when the two numbers have the same sign and negative if they are different. One can use this expression to compute the *hinge loss*,

$$L(f_\theta(x), y) = \max(0, 1 - yf(x)), \quad (2.15)$$

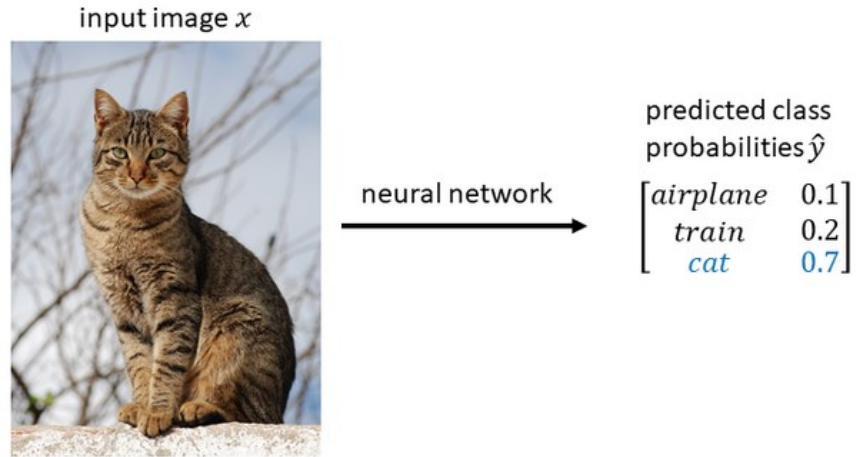
or the *logistic loss*

$$L(f_\theta(x), y) = \log(1 + e^{-yf(x)}). \quad (2.16)$$

For multi-label classification with probabilistic one-hot encoding, the *cross entropy loss* is often used to compare the multi-dimensional target $y = (y_1, \dots, y_D)$, $y \in \{0, 1\}$ with the output probability distribution over predictions $f(x) = (p_1, \dots, p_D)$ via

$$L(f_\theta(x), y) = - \sum_{d=1}^D y_d \log p_d. \quad (2.17)$$

Neural Networks (<https://glassboxmedicine.com/2019/12/07/connections-log-likelihood-cross-entropy-kl-divergence-logistic-regression-and-neural-networks/>)

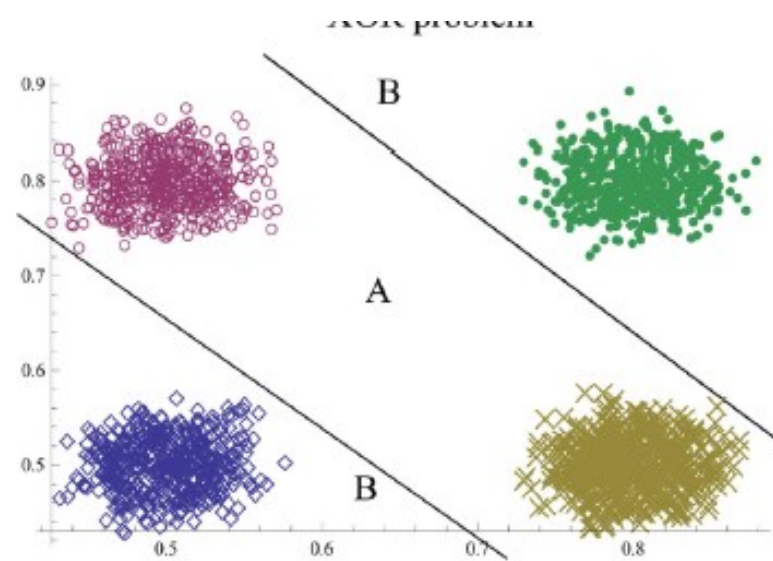


- We want the neural network's predictions to be as close as possible to the ground truth y

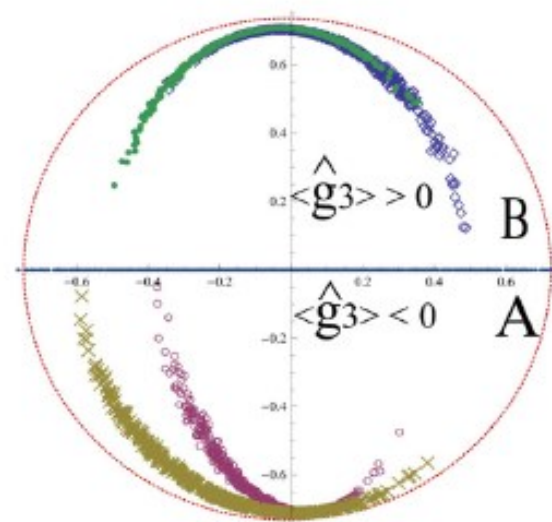
<i>airplane</i>	0
<i>train</i>	0
<i>cat</i>	1
- The likelihood of the observation is $\prod_{j=1}^M \hat{y}_j^{y_j}$
- For example,

predicted class probabilities \hat{y}	ground truth y												
<table border="1"><tr><td><i>airplane</i></td><td>0.1</td></tr><tr><td><i>train</i></td><td>0.2</td></tr><tr><td><i>cat</i></td><td>0.7</td></tr></table>	<i>airplane</i>	0.1	<i>train</i>	0.2	<i>cat</i>	0.7	<table border="1"><tr><td><i>airplane</i></td><td>0</td></tr><tr><td><i>train</i></td><td>0</td></tr><tr><td><i>cat</i></td><td>1</td></tr></table>	<i>airplane</i>	0	<i>train</i>	0	<i>cat</i>	1
<i>airplane</i>	0.1												
<i>train</i>	0.2												
<i>cat</i>	0.7												
<i>airplane</i>	0												
<i>train</i>	0												
<i>cat</i>	1												

$$\prod_{j=1}^M \hat{y}_j^{y_j} = 0.1^0 \times 0.2^0 \times 0.7^1 = 1 \times 1 \times 0.7 = 0.7$$



(a)



(b)

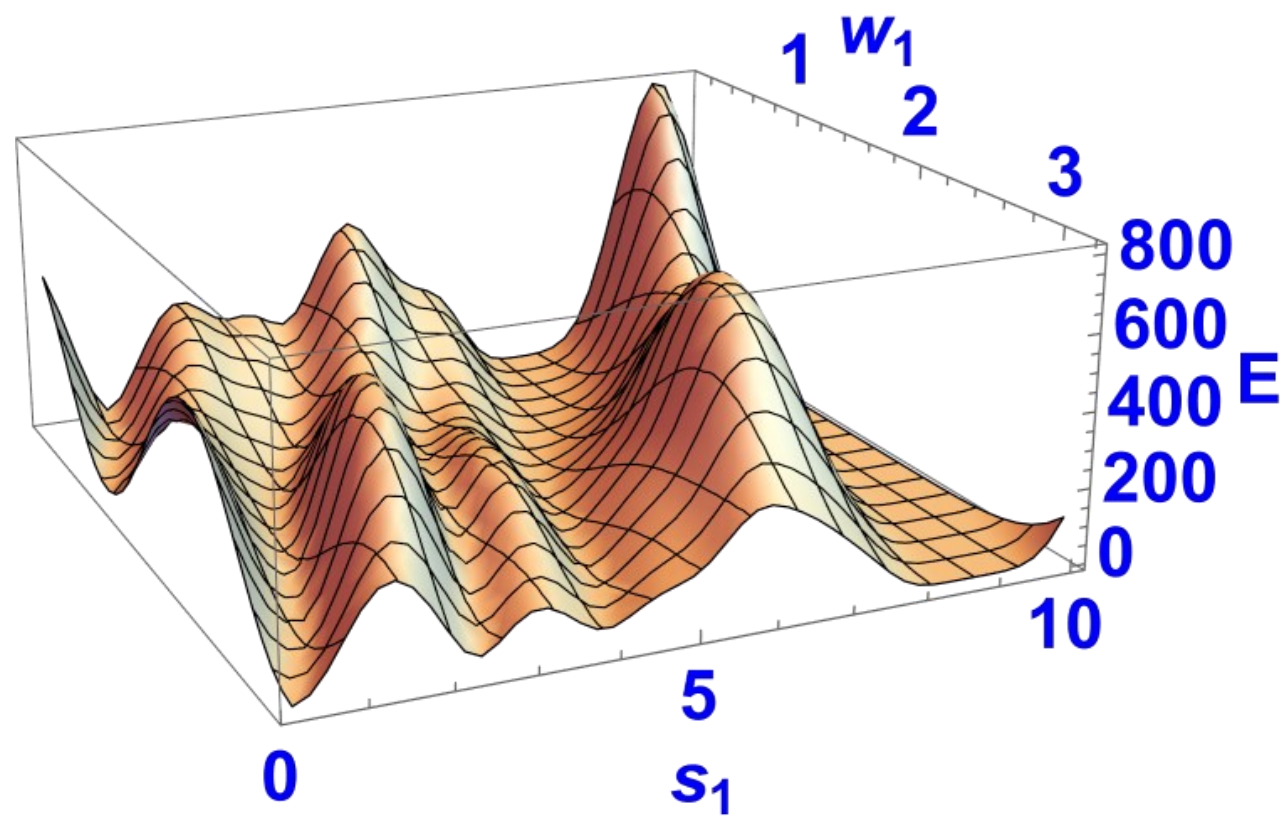
$$\rho_{AB} = \begin{pmatrix} \rho_{11} & \rho_{12} & \rho_{13} & \rho_{14} \\ \rho_{21} & \rho_{22} & \rho_{23} & \rho_{24} \\ \rho_{31} & \rho_{32} & \rho_{33} & \rho_{34} \\ \rho_{41} & \rho_{42} & \rho_{43} & \rho_{44} \end{pmatrix}, \quad (3.13)$$

and the partial trace over the first subsystem leads to the mixed state

$$\mathrm{tr}_A\{\rho_{AB}\} = \begin{pmatrix} \rho_{11} + \rho_{33} & \rho_{12} + \rho_{34} \\ \rho_{21} + \rho_{43} & \rho_{22} + \rho_{44} \end{pmatrix}, \quad (3.14)$$

while the state of the second subsystem is given by the mixed state

$$\mathrm{tr}_B\{\rho_{AB}\} = \begin{pmatrix} \rho_{11} + \rho_{22} & \rho_{13} + \rho_{24} \\ \rho_{31} + \rho_{42} & \rho_{33} + \rho_{44} \end{pmatrix}. \quad (3.15)$$



Random Initialization
+Gradient Descent

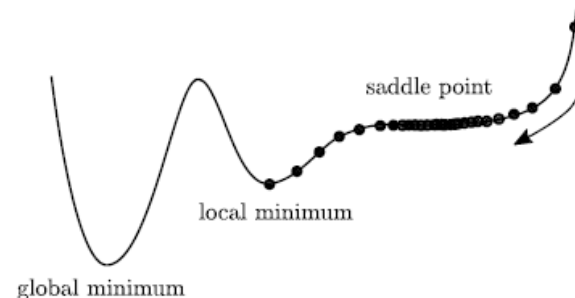


Fig. 2.14 Steps of the training updates in a one-dimensional parameter space. While being often the only option in high-dimensional non-convex optimisation, gradient descent can get stuck in local minima and convergence becomes slow at saddle points

(for more details see [24]). We will see below that the least-squares loss together with a model function that is linear in the parameters forms a rather simple convex quadratic optimisation problem that has a closed-form solution.

For general non-convex optimisation problems much less is known, and many machine learning problems fall into this category. Popular methods are therefore searches which iteratively compute better candidates for the parameters of a model. When there are many parameters and we optimise in high-dimensional spaces, information about the cost landscape is crucial, and the most common information used is gradients. A gradient is the vector of partial derivatives of a function's output with respect to its inputs, and the gradient gives us the direction of the steepest ascent in the cost landscape.

Gradient descent is a well-established optimisation technique that updates the parameters θ of a cost function $C(\theta)$ successively by the rule

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla C(\theta^{(t)}), \quad (2.25)$$

where η is a hyperparameter called the *learning rate* and t an integer keeping track of the current iteration. Since the gradient $\nabla C(\theta^{(t)})$ points towards the ascending direction in the landscape of C , following its negative means to descend into valleys. Note that the cost function, and hence the gradient at each step, depends on the training data.

Stochastic gradient descent uses only a subset of the training data for each step to calculate the gradient of the cost function. While the original definition of stochastic gradient descent, in fact, only considered *one* randomly sampled training input per

5.3.2 *Parameter-Shift Rules*

Parameter-shift rules refer to a family of rules that express the partial derivative of a quantum expectation with respect to a gate parameter as a linear combination of the same expectation, but with the parameter “shifted” (Fig. 5.10):

Definition 5.4 (*Parameter-shift rule*) Let $f_\mu = \langle \mathcal{M} \rangle_\mu$ be a quantum expectation value that depends on a classical parameter μ . A parameter-shift rule is an identity of the form

$$\partial_\mu f_\mu = \sum_i a_i f_{\mu+s_i}, \quad (5.47)$$

where $\{a_i\}$ and $\{s_i\}$ are real scalar values.

$$\partial_\mu f_\mu = \frac{1}{2 \sin(s)} \left(\langle \psi | \mathcal{G}^\dagger(\mu + s) B \mathcal{G}(\mu + s) | \psi \rangle - \langle \psi | \mathcal{G}^\dagger(\mu - s) B \mathcal{G}(\mu - s) | \psi \rangle \right) \quad (5.53)$$

$$= \frac{1}{2 \sin(s)} \left(f_{\mu+s} - f_{\mu-s} \right), \quad (5.54)$$

$$\begin{aligned} & \frac{\partial}{\partial \theta} f(\theta) \\ &= \langle \psi | [+iaG] U_G^\dagger(\theta) A U_G(\theta) | \psi \rangle \end{aligned} \quad (8a)$$

$$\begin{aligned} & + \langle \psi | U_G^\dagger(\theta) A [-iaG] U_G(\theta) | \psi \rangle \\ &= \frac{r}{2} \langle \psi | U_G^\dagger(\theta) (I + i\frac{a}{r}G) A (I - i\frac{a}{r}G) U_G(\theta) | \psi \rangle \end{aligned} \quad (8b)$$

$$\begin{aligned} & - \frac{r}{2} \langle \psi | U_G^\dagger(\theta) (I - i\frac{a}{r}G) A (I + i\frac{a}{r}G) U_G(\theta) | \psi \rangle \\ &= r \langle \psi | U_G^\dagger(\theta + \frac{\pi}{4r}) A U_G(\theta + \frac{\pi}{4r}) | \psi \rangle \end{aligned} \quad (8c)$$

$$\begin{aligned} & - r \langle \psi | U_G^\dagger(\theta - \frac{\pi}{4r}) A U_G(\theta - \frac{\pi}{4r}) | \psi \rangle \\ &= r \left[f(\theta + \frac{\pi}{4r}) - f(\theta - \frac{\pi}{4r}) \right] \end{aligned} \quad (8d)$$

$$r = \frac{a}{2}(e_1 - e_0), \text{ and } s = \frac{1}{2}(e_1 + e_0)$$

5.2 Which Functions Do Variational Quantum Models Express?

Article preview

Abstract

References (26)

Cited by (18464)



Neural Networks


Volume 2, Issue 5, 1989, Pages 359-366






Original contribution

Multilayer feedforward networks are universal approximators

Kurt Hornik, Maxwell Stinchcombe, Halbert White¹ 

Show more 

 Add to Mendeley  Share  Cite

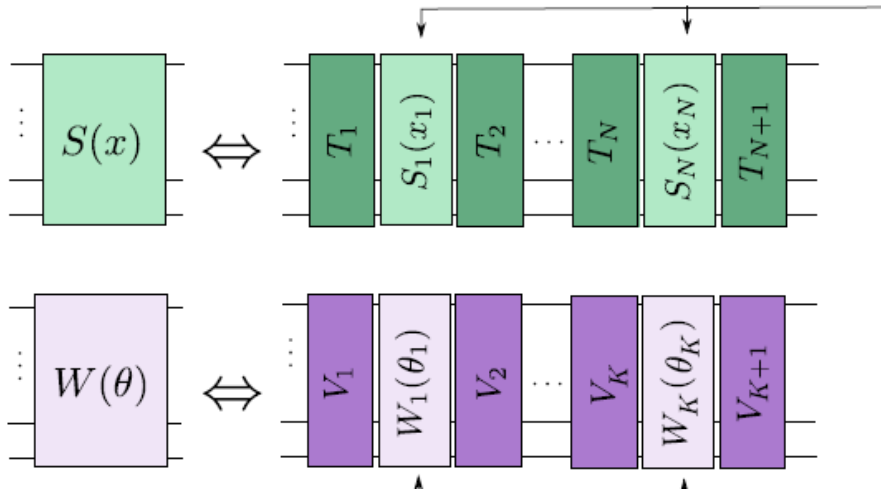
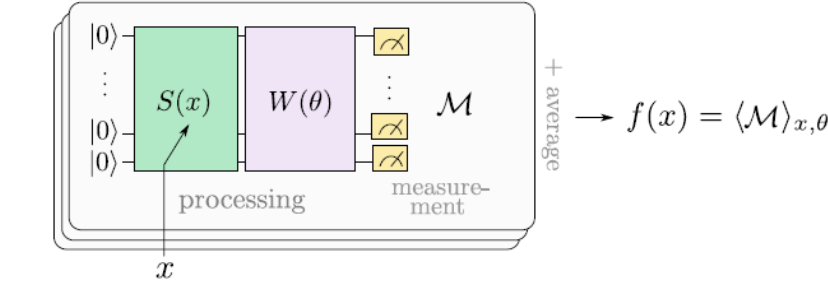
[https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8) 

[Get rights and content](#) 

Abstract

This paper rigorously establishes that standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators.

$$U(x, \theta) = W(\theta)S(x).$$



$$S(x) = T_{N+1} \prod_{i=1}^N S_i(x_i) T_i,$$

$$W(\theta) = V_{K+1} \prod_{k=1}^K W_k(\theta_k) V_k.$$

$$S_i(x_i) = e^{-ix_i G_i}, \quad i = 1, \dots, N, \quad (5.21)$$

where we assume without loss of generality that G_i is a diagonal operator $\text{diag}(\lambda_1^i, \dots, \lambda_d^i)$, where d is the dimension of the Hilbert space. If this is not the

unitaries. Finally, we assume that $\mathbf{x} \in \mathcal{X} = \mathbb{R}^N$ is given by $\mathbf{x} = (x_1, \dots, x_N)^T$.

Theorem 5.1 (Function class of quantum models) *Let $\mathcal{X} = \mathbb{R}^N$, $\mathcal{Y} = \mathbb{R}$ and $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ be a deterministic quantum model as defined in (5.1) with circuit $U(\mathbf{x}, \theta)$ as in Eq. (5.20). Accordingly, the i 'th feature is encoded by gate $e^{-ix_i G_i}$. Then, f_θ can be written as*

$$f_\theta(\mathbf{x}) = \sum_{\omega_1 \in \Omega_1} \dots \sum_{\omega_N \in \Omega_N} c_{\omega_1 \dots \omega_N}(\theta) e^{i\omega_1 x_1} \dots e^{i\omega_N x_N}, \quad (5.22)$$

where the frequency spectrum of the i 'th feature, $i = 1, \dots, N$, is given by

$$\Omega_i = \{\lambda_s^i - \lambda_t^i | s, t \in \{1, \dots, d\}\}. \quad (5.23)$$

This frequency spectrum is the set of all values produced by differences between any two eigenvalues of G_i . We are guaranteed that $0 \in \Omega$, and that for each $\omega \in \Omega$ there is $-\omega \in \Omega$ too with $c_\omega(\theta) = c_{-\omega}^*(\theta)$. This symmetry guarantees that f_θ is real-valued, and that the sum can be rewritten with cosine and sine functions.

Risk Minimisation in Supervised Learning

expected

$$\mathcal{R}_{f_\theta} = \mathbb{E}[L_{f_\theta}] = \int_{\mathcal{X} \times \mathcal{Y}} L(f_\theta(x), y) p(x, y) dx dy. \quad (2.18)$$

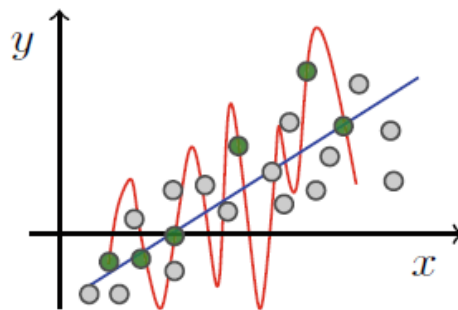
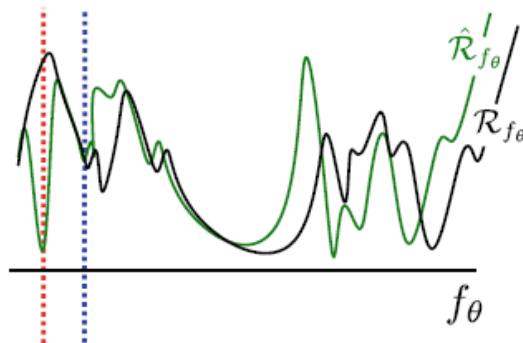
The integral runs over *all possible data pairs*, and the expectation is hence taken over the data distribution. The risk tells us how well the model is performing on the entire data domain, and a lower risk is better.

$$\hat{\mathcal{R}}_{f_\theta} = \hat{\mathbb{E}}[L_{f_\theta}] = \frac{1}{M} \sum_{m=1}^M L(f(x^m), y^m).$$

empirical

Definition 2.6 (*Empirical risk minimisation*) Let $\hat{\mathcal{R}}_{f_\theta}$ be the empirical risk of a supervised learning dataset \mathcal{D} . Empirical risk minimisation is the problem of finding

$$\theta^* = \min_{\theta} \hat{\mathcal{R}}_{f_\theta}. \quad (2.20)$$



$$C(\theta) = \hat{\mathcal{R}}_{f_\theta} + g(f_\theta)$$

$$g_{\ell_1}(\theta) = \sum_i |\theta_i|,$$

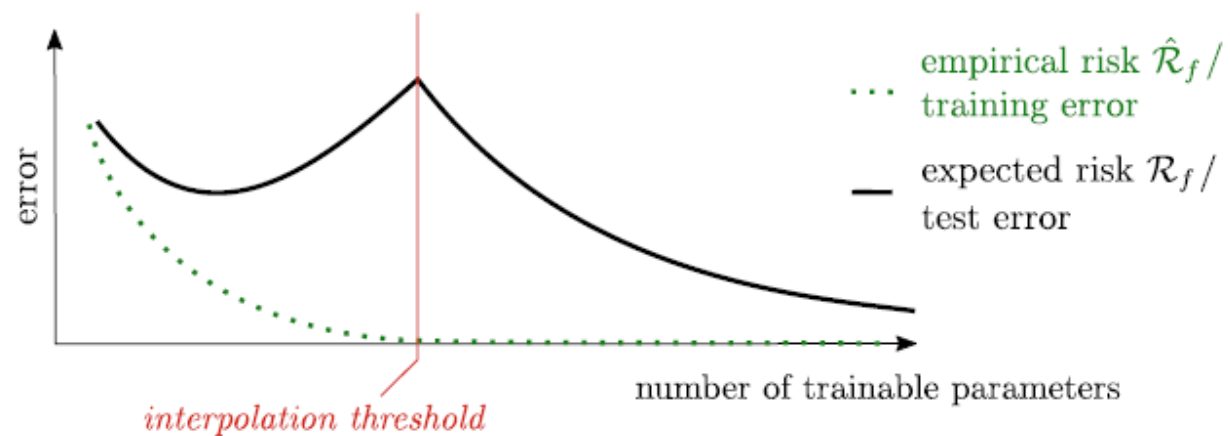
regularization term

$$g_{\ell_2}(\theta) = \sum_i \theta_i^2,$$

generalization (error) bound

$$\mathcal{R}_{f_\theta} \leq \hat{\mathcal{R}}_{f_\theta} + \sqrt{\frac{1}{M} \left(d_{\text{VC}} \left(\log \left(\frac{2M}{d} \right) + 1 \right) + \log \left(\frac{4}{\delta} \right) \right)}.$$

deep learning




ARTICLE

DOI: 10.1038/s41467-018-07090-4

OPEN

Barren plateaus in quantum neural network training landscapes

Jarrod R. McClean¹, Sergio Boixo ¹, Vadim N. Smelyanskiy¹, Ryan Babbush¹ & Hartmut Neven¹

Many experimental proposals for noisy intermediate scale quantum devices involve training a parameterized quantum circuit with a classical optimization loop. Such hybrid quantum-classical algorithms are popular for applications in quantum simulation, optimization, and machine learning. Due to its simplicity and hardware efficiency, random circuits are often proposed as initial guesses for exploring the space of quantum states. We show that the exponential dimension of Hilbert space and the gradient estimation complexity make this choice unsuitable for hybrid quantum-classical algorithms run on more than a few qubits. Specifically, we show that for a wide class of reasonable parameterized quantum circuits, the probability that the gradient along any reasonable direction is non-zero to some fixed precision is exponentially small as a function of the number of qubits. We argue that this is related to the 2-design characteristic of random circuits, and that solutions to this problem must be studied.

Gradient concentration in random circuits. We will discuss random parameterized quantum circuits (RPQCs)

$$U(\boldsymbol{\theta}) = U(\theta_1, \dots, \theta_L) = \prod_{l=1}^L U_l(\theta_l) W_l, \quad (1)$$

where $U_l(\theta_l) = \exp(-i\theta_l V_l)$, V_l is a Hermitian operator, and W_l is a generic unitary operator that does not depend on any angle θ_l . Circuits of this form are a natural choice due to a straightforward evaluation of the gradient with respect to most objective functions and have been introduced in a number of contexts already^{26,41}. Consider an objective function $E(\theta)$ expressed as the expectation value over some Hermitian operator H ,

$$E(\boldsymbol{\theta}) = \langle 0 | U(\boldsymbol{\theta})^\dagger H U(\boldsymbol{\theta}) | 0 \rangle. \quad (2)$$

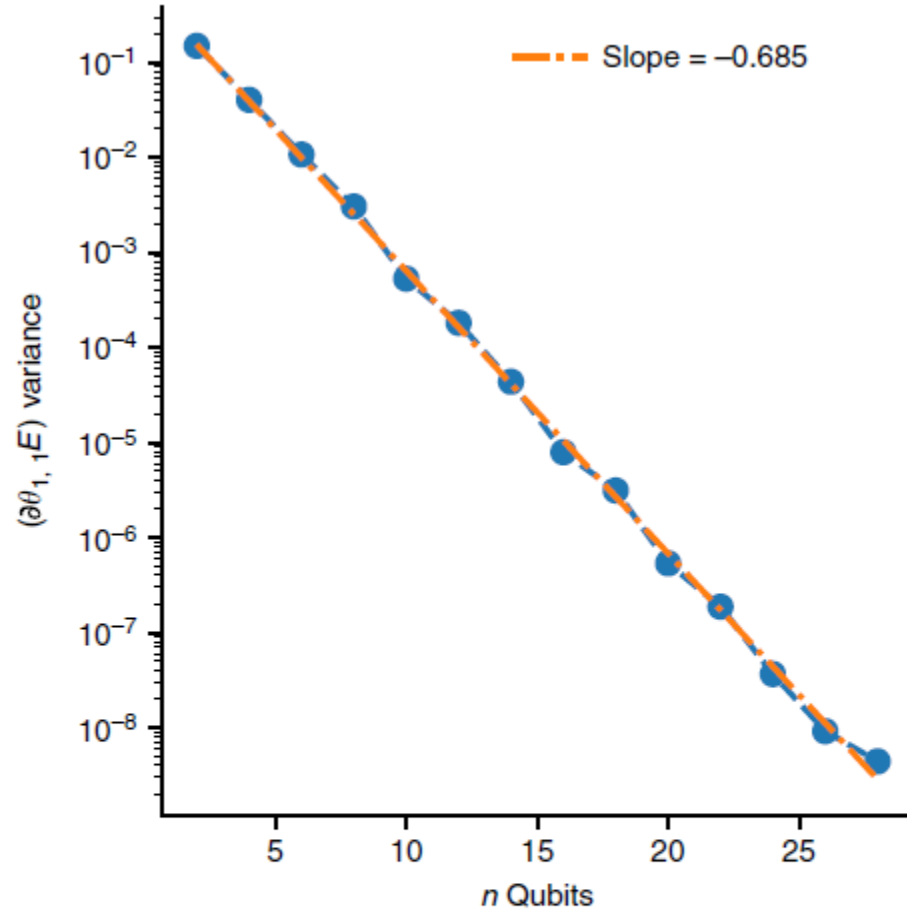


Fig. 3 Exponential decay of variance. The sample variance of the gradient of the energy for the first circuit component of a two-local Pauli term $(\partial_{\theta_{1,1}} E)$ plotted as a function of the number of qubits on a semi-log plot. As predicted, an exponential decay is observed as a function of the number of qubits, n , for both the expected value and its spread. The slope of the fit line is indicative of the rate of exponential decay as determined by the operator

the circuit grows in length from a single layer to a circuit

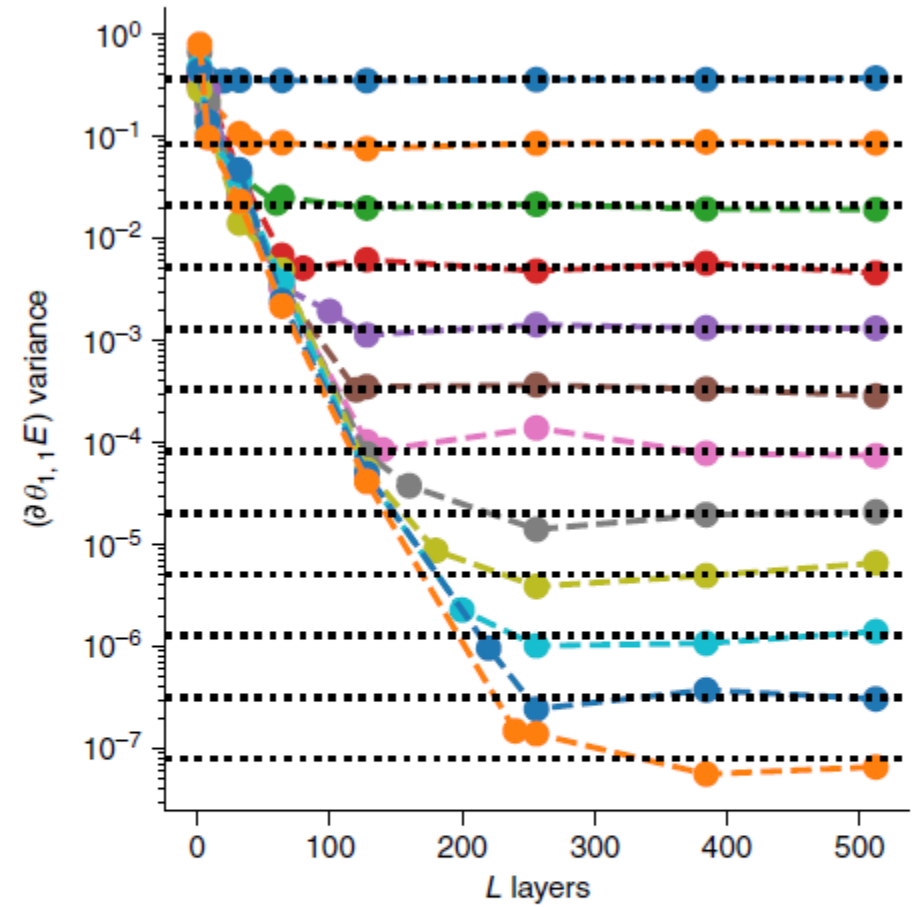


Fig. 4 Convergence to 2-design limit. Here we show the sample variance of the gradient of the energy for the first circuit component of a two-local Pauli term $(\partial_{\theta_{1,1}} E)$ plotted as a function of the number of layers, L , in a 1D quantum circuit. The different lines correspond to all even numbers of qubits between 2 and 24, with 2 qubits being the top line, and the rest being ordered by qubit number. The dotted black lines depict the 2-design asymptotes for this Hamiltonian as determined by our analytic results. This shows the convergence of the second moment as a function of the number of layers to a fixed value determined by the number of qubits

Absence of Barren Plateaus in Quantum Convolutional Neural Networks

Arthur Pesah, M. Cerezo, Samson Wang, Tyler Volkoff, Andrew T. Sornborger, and Patrick J. Coles

Phys. Rev. X **11**, 041011 – Published 15 October 2021

Article

References

Citing Articles (84)

PDF

HTML

Export Citation



ABSTRACT

Quantum neural networks (QNNs) have generated excitement around the possibility of efficiently analyzing quantum data. But this excitement has been tempered by the existence of exponentially vanishing gradients in many QNN architectures.

quantum convolutional neural networks (QCNNs) for analyzing relevant data from quantum devices, implying that the trainability of QCNNs is not limited by random initialization.

Function Dependent Barren Plateaus in Shallow Parametrized Quantum Circuits

M. Cerezo,^{1,2,*} Akira Sone,^{1,2} Tyler Volkoff,¹ Lukasz Cincio,¹ and Patrick J. Coles^{1,*}

¹*Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM, USA.*

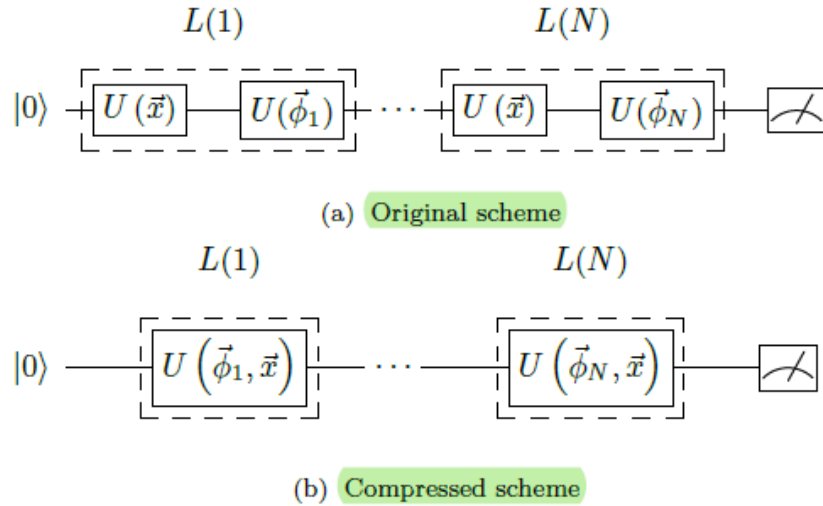
²*Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, NM, USA*

Variational quantum algorithms (VQAs) optimize the parameters θ of a parametrized quantum circuit $V(\theta)$ to minimize a cost function C . While VQAs may enable practical applications of noisy quantum computers, they are nevertheless heuristic methods with unproven scaling. Here, we rigorously prove two results, assuming $V(\theta)$ is an alternating layered ansatz composed of blocks forming local 2-designs. Our first result states that defining C in terms of global observables leads to exponentially vanishing gradients (i.e., barren plateaus) even when $V(\theta)$ is shallow. Hence, several VQAs in the literature must revise their proposed costs. On the other hand, our second result states that defining C with local observables leads to at worst a polynomially vanishing gradient, so long as the depth of $V(\theta)$ is $\mathcal{O}(\log n)$. Our results establish a connection between locality and trainability. We illustrate these ideas with large-scale simulations, up to 100 qubits, of a quantum autoencoder implementation.

some examples of architectures and problems solved

Data re-uploading for a universal quantum classifier

Adrián Pérez-Salinas^{1,2}, Alba Cervera-Lierta^{1,2}, Elies Gil-Fuster³, and José I. Latorre^{1,2,4,5}



$$L(i) = U\left(\vec{\theta}_i + \vec{w}_i \circ \vec{x}\right), \quad (5)$$

where $\vec{w}_i \circ \vec{x} = (w_i^1 x^1, w_i^2 x^2, w_i^3 x^3)$ is the Hadamard product of two vectors. In case the data points have

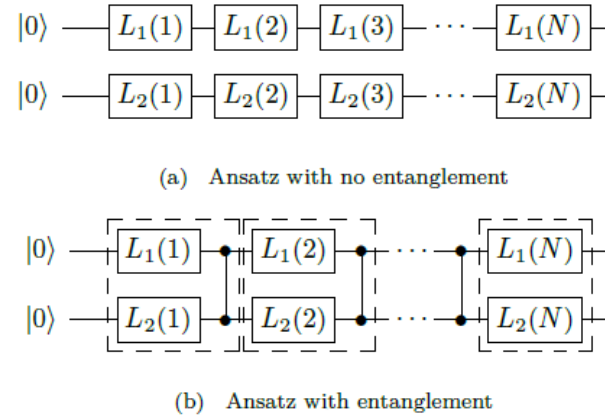
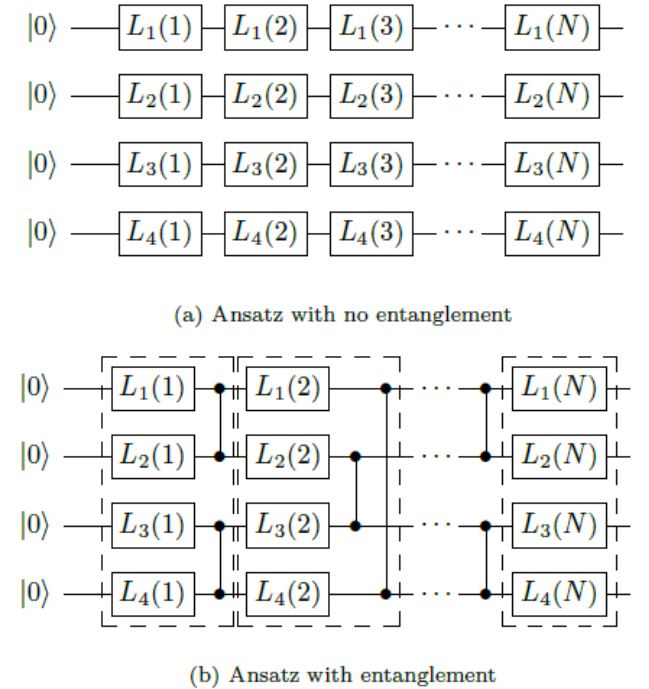


Figure 4: Two-qubit quantum classifier circuit without entanglement (top circuit) and with entanglement (bottom circuit). Here, each layer includes a rotation with data re-uploading in both qubits plus a CZ gate if there is entanglement. The exception is the last layer, which does not have any CZ gate associated to it. For a fixed number of layers, the number of parameters to be optimized doubles the one needed for a single-qubit classifier.



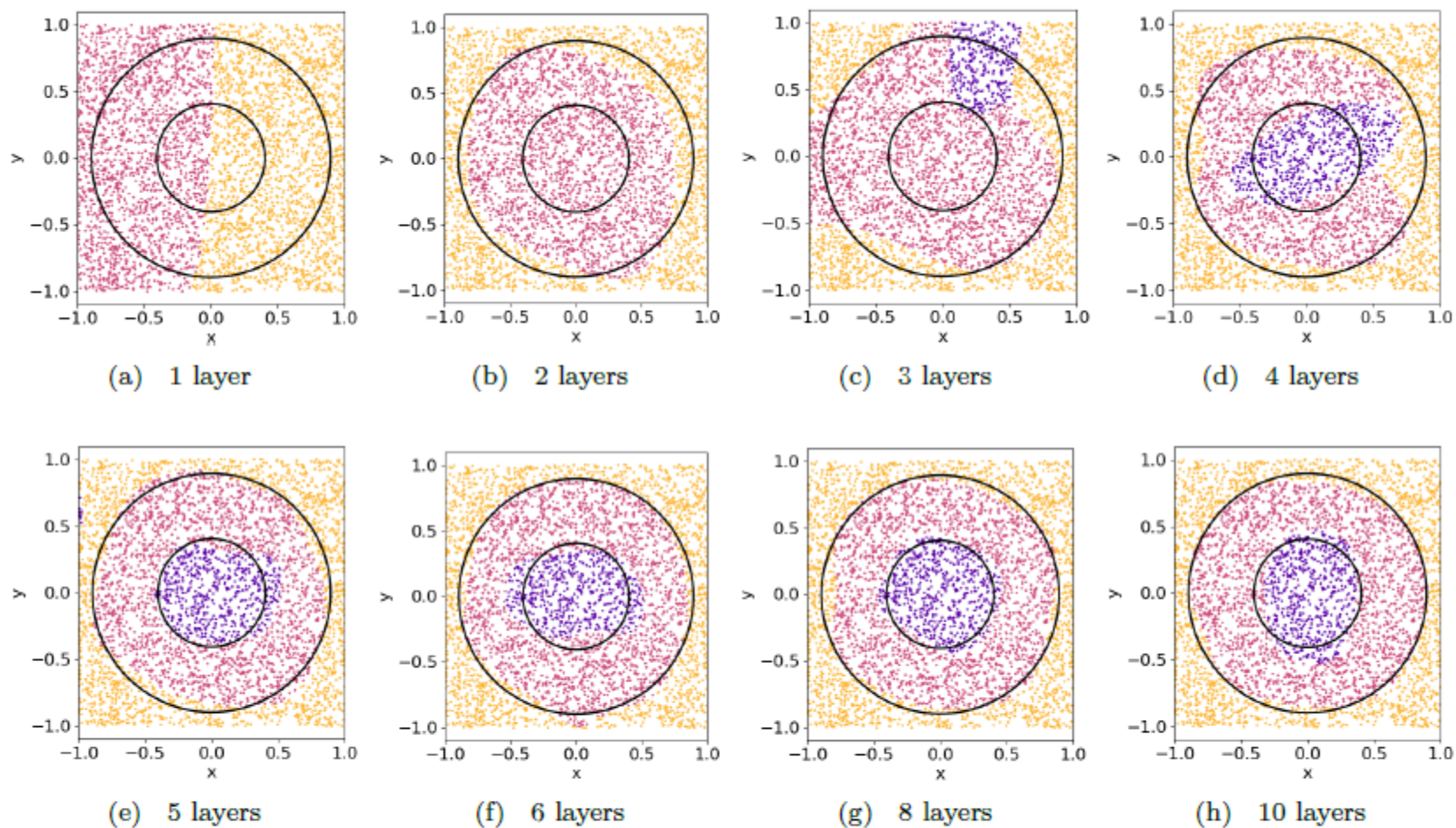


Figure 8: Results obtained with the single-qubit classifier for the annulus problem, using the weighted fidelity cost function during the training. The better results are obtained with a 10 layers classifier (93% of success rate). As we consider more qubits and entanglement, we can increase the success rate up to 96%, as shows Table 4.

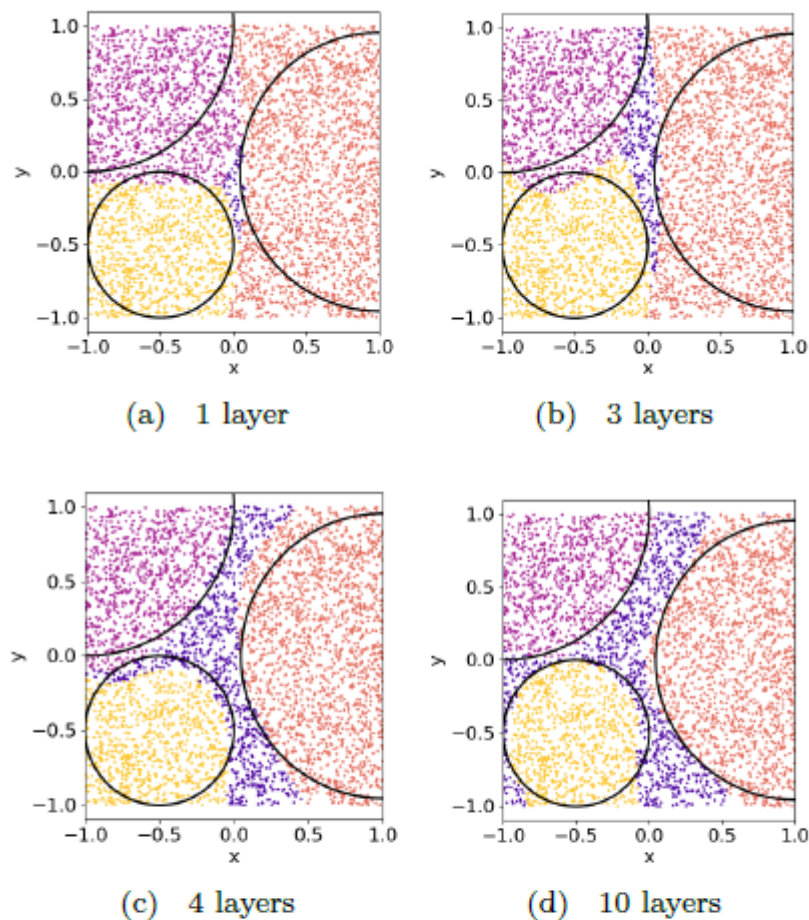


Figure 7: Results for the 3-circles problem using a single-qubit classifier trained with the L-BFGS-B minimizer and the weighted fidelity cost function. With one layer, the classifier intuits the four regions although the central one is difficult to tackle. With more layers, this region is clearer for the classifier and it tries to adjust the circular regions.

Circuit-centric quantum classifiers

Maria Schuld,^{1,2,3} Alex Bocharov,³ Krysta Svore,³ and Nathan Wiebe³

¹*Quantum Research Group, School of Chemistry and Physics,
University of KwaZulu-Natal, Durban 4000, South Africa*

²*National Institute for Theoretical Physics, KwaZulu-Natal, Durban 4000, South Africa*

³*Quantum Architectures and Computation Group,
Station Q, Microsoft Research, Redmond, WA (USA)*

(Dated: April 3, 2018)

The current generation of quantum computing technologies call for quantum algorithms that require a limited number of qubits and quantum gates, and which are robust against errors. A suitable design approach are variational circuits where the parameters of gates are learnt, an approach that is particularly fruitful for applications in machine learning. In this paper, we propose a low-depth variational quantum algorithm for supervised learning. The input feature vectors are encoded into the amplitudes of a quantum system, and a quantum circuit of parametrised single and two-qubit gates together with a single-qubit measurement is used to classify the inputs. This circuit architecture ensures that the number of learnable parameters is poly-logarithmic in the input dimension. We propose a quantum-classical training scheme where the analytical gradients of the model can be estimated by running several slightly adapted versions of the variational circuit. We show with simulations that the circuit-centric quantum classifier performs well on standard classical benchmark datasets while requiring dramatically fewer parameters than other methods. We also evaluate sensitivity of the classification to state preparation and parameter noise, introduce a quantum version of dropout regularisation and provide a graphical representation of quantum gates as highly symmetric linear layers of a neural network.

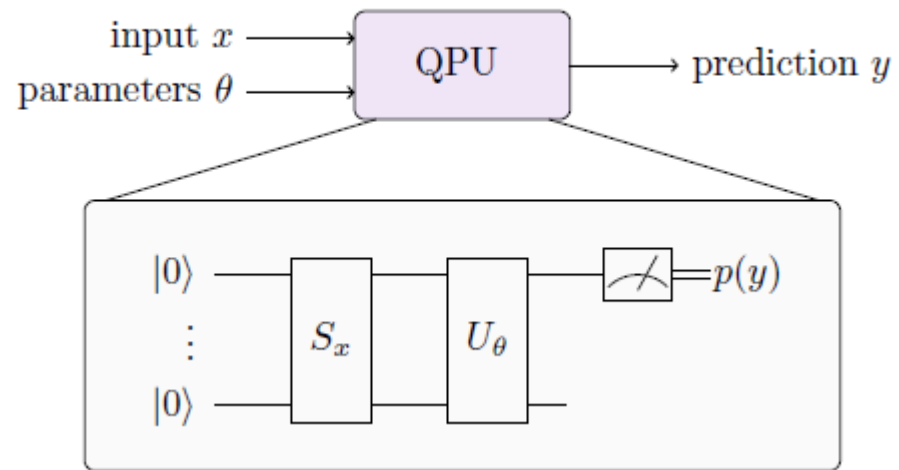


FIG. 1. Idea of the circuit-centric quantum classifier. Inference with the model $f(x, \theta) = y$ is executed by a quantum device (the quantum processing unit or QPU) which consists of a *state preparation circuit* S_x encoding the input x into the amplitudes of a quantum system, a *model circuit* U_θ , and a single qubit measurement. The measurement retrieves the probability of the model predicting 0 or 1, from which in turn the binary prediction can be inferred. The classification circuit parameters θ are learnable and can be trained by a variational scheme.

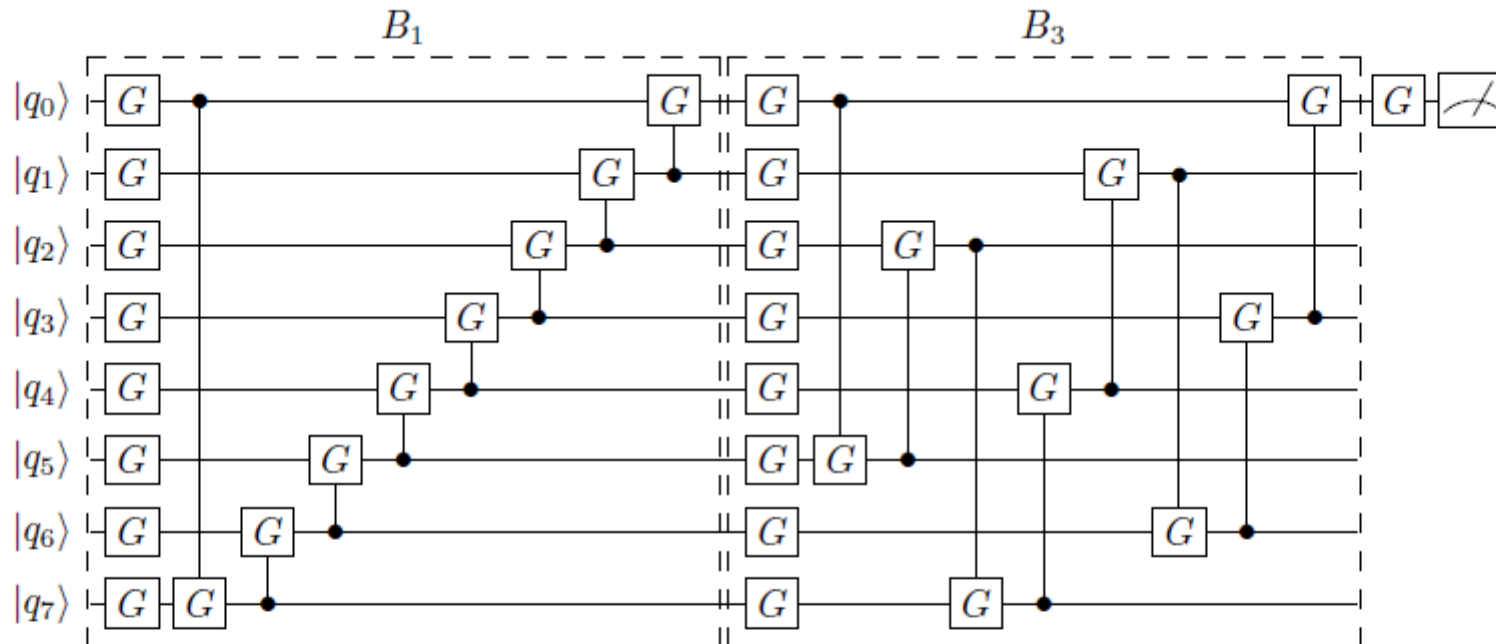


FIG. 4. Generic model circuit architecture for 8 qubits. The circuit consists of two ‘code blocks’ B_1 and B_3 with a range of controls of $r = 1$ and $r = 3$ respectively. The circuit consists of 17 trainable single-qubit gates $G = G(\alpha, \beta, \gamma)$, as well as 16 trainable controlled single qubit gates $C(G)$, which have in turn to be decomposed into the elementary constant gate set used by the quantum computer on which to implement it. If the optimisation methods are used to reduce the controlled gates to a single parameter, we have $3 \cdot 33 + 1 = 100$ parameters to learn in total for this model circuit. These 100 parameters are used to classify inputs of $2^8 = 256$ dimensions, which shows that the circuit-centric classifier is a much more compact model than a conventional feed-forward neural network.

ID	domain	# features	# classes	# samples	preprocessing
CANCER	decision	32	2	569	none
SONAR	decision	60	2	208	padding with noninformative features
WINE	decision	13	3	178	padding with noninformative features
SEMEION	OCR ^a	256	10	1593	padding with noninformative features
MNIST256	OCR	256	10	2766	coarse-graining and deskewing

^a Optical Character Recognition

TABLE II. Benchmark datasets and preprocessing.

ID	model	fixed hyperparameters	variable hyperparameters
QC	Circuit-centric quantum classifier	entangling circuit architecture	dropout rate, number of blocks, range
PERC	perceptron	-	regularisation type
MLPlin	neural network	dim. of hidden layer = N	regularisation strength, optimiser, initial learning rate
MLPshal	neural network	dim. of hidden layer = $\lceil \log_2 N \rceil$	regularisation strength, optimiser, initial learning rate
MLPdeep	neural network	dim. of hidden layers = $\lceil \log_2 N \rceil$	regularisation strength, optimiser, initial learning rate
SVMpoly1	support vector machine	polynomial kernel with $d = 1$, regularisation strength of slack variables = 1, offset $c = 1$	-
SVMpoly2	support vector machine	polynomial kernel with $d = 2$, regularisation strength of slack variables = 1, offset $c = 1$	-

TABLE III. Benchmark models explained in the text and possible choices for hyperparameters.

▼ Circuit-centric quantum classifiers

Abstract

I Introduction

▼ II The circuit-centric quantum classifier

A State preparation

▼ B The model circuit

1 Decomposition into (controlled) single qubit gates

C Read out and postprocessing

▼ III Circuit architectures

A Strongly entangling circuits

B Optimising the architecture

C Graphical representation of gates

▼ IV Training

A Cost function

B Hybrid gradient descent scheme

C Classical linear combinations of unitaries

D Dropout

▼ E Performance analysis

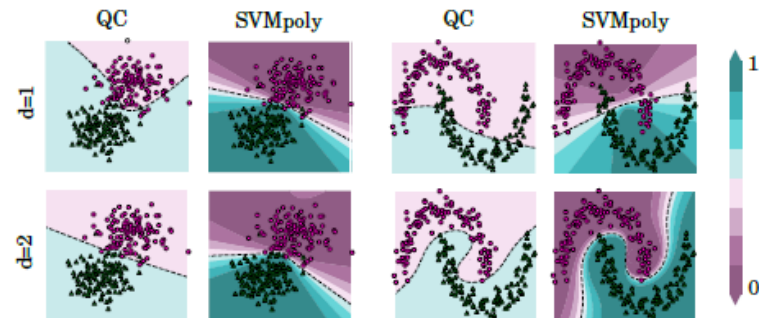


FIG. 9. Comparison of the decision boundary for the circuit-centric quantum classifier (QC) and a support vector machine with polynomial kernel (SVMpoly). The 2-dimensional data gets embedded into a 4-dimensional feature space (we padded with 2 non-informative features), where it is classified by the two models. The colour scale indicates the probability that the model predicts class “green circles”. For the QC model, the parameter d corresponds to the degree of the polynomial feature map in amplitude encoding (see Section II A). For the SVMpoly, d is the degree of the polynomial kernel. One can see that for $d = 1$ the QC model is slightly more flexible than the SVMpoly.

ID	QC	PERC	MLPlin	MLPshal	MLPdeep	SVM
CANCER	79	32	1056	165	190	208
SONAR	60	60	1952	305	330	569
WINE	28	13	272	51	60	178
SEMEION	100	256	65792	2056	2120	1593
MNIST256	124	256	65792	2056	2120	800

TABLE IV. Number of parameters each model has to train for the different benchmark datasets. The circuit-centric quantum classifier QC has a logarithmic growth in the number of parameters with the input dimension N and data set size M , while all other models show a linear or polynomial parameter growth in either N or M .

You should provide:

- a) the design of the circuit you have built including measurements. My advice is not to exceed 4 qubits circuit.
- b) the cost function you have selected build by the expectation value of measurements at the end of the circuit
- c) your selected optimization method
- d) the outcomes of classification (e.g. accuracy) on train and test data
- e) your program

- Compare different classical optimization methods and/or loss functions
- Justify your model by presenting geometric representations using Bloch sphere
- Compare the number of parameters, epochs, structure with a classical NN that solves the same problem
- Involve parameter-shift rule for the evaluation of gradients
- Use your classifier to solve another binary classification problem with 3 inputs and report the outcomes.
- Any other subject that you find interesting

KAN: Kolmogorov–Arnold Networks

Ziming Liu^{1,4*} Yixuan Wang² Sachin Vaidya¹ Fabian Ruehle^{3,4}
James Halverson^{3,4} Marin Soljačić^{1,4} Thomas Y. Hou² Max Tegmark^{1,4}

¹ Massachusetts Institute of Technology

² California Institute of Technology

³ Northeastern University

⁴ The NSF Institute for Artificial Intelligence and Fundamental Interactions

Abstract

Inspired by the Kolmogorov–Arnold representation theorem, we propose Kolmogorov–Arnold Networks (KANs) as promising alternatives to Multi-Layer Perceptrons (MLPs). While MLPs have *fixed* activation functions on *nodes* (“neurons”), KANs have *learnable* activation functions on *edges* (“weights”). KANs have no linear weights at all – every weight parameter is replaced by a univariate function parametrized as a spline. We show that this seemingly simple change makes KANs outperform MLPs in terms of accuracy and interpretability. For accuracy, much smaller KANs can achieve comparable or better accuracy than much larger MLPs in data fitting and PDE solving. Theoretically and empirically, KANs possess faster neural scaling laws than MLPs. For interpretability, KANs can be intuitively visualized and can easily interact with human users. Through two examples in mathematics and physics, KANs are shown to be useful “collaborators” helping scientists (re)discover mathematical and physical laws. In summary, KANs are promising alternatives for MLPs, opening opportunities for further improving today’s deep learning models which rely heavily on MLPs.

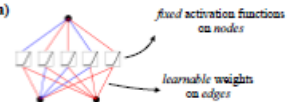
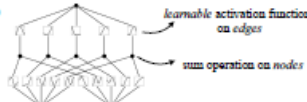
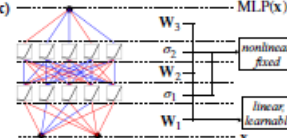
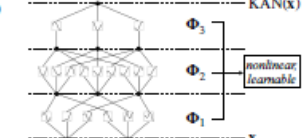
Model	Multi-Layer Perceptron (MLP)	Kolmogorov–Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov–Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \phi_{qp}(x_p) \right)$
Model (Shallow)	(a)  fixed activation functions on nodes learnable weights on edges	(b)  learnable activation functions on edges sum operation on nodes
Formula (Deep)	$\text{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\text{KAN}(\mathbf{x}) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(\mathbf{x})$
Model (Deep)	(c)  nonlinear fixed linear learnable	(d)  nonlinear learnable

Figure 0.1: Multi-Layer Perceptrons (MLPs) vs. Kolmogorov–Arnold Networks (KANs)