Variational Quantum Classifiers

Train a variational classifier with a correctly defined loss and gradient estimator.

Undergraduate 21 min lesson Dr. Charith Jayasekara

By the end, you'll be able to

- Derive the toy model's probability and squared loss.
- Apply parameter-shift to an expectation before the chain rule.
- Evaluate a frozen model on unseen data.

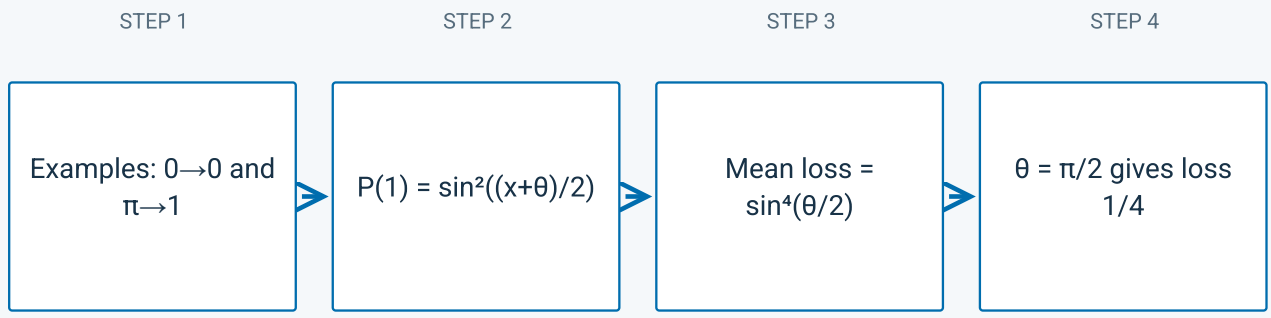
Train a circuit to produce a useful probability

A variational quantum classifier uses trainable circuit parameters to map an example into measured predictions. A classical optimiser adjusts those parameters to reduce a loss on labelled training data. The procedure resembles other differentiable models in its workflow, but its function class, measurement cost and physical constraints come from the chosen quantum circuit.

Prerequisites: rotation gates, measurement and the quantum-learning lesson. A primary proposal studies trainable quantum circuits for classification and explicitly includes small simulated examples rather than a universal learning-speedup theorem. Farhi and Neven.

WORKED DIAGRAM · A31

A trainable threshold



At θ = 0 the toy training loss is 0; at π/2 it is 1/4; at π it is 1. Parameter-shift applies to probabilities before differentiating a nonlinear loss.

The model setup

Define how an input x becomes a quantum state, which trainable gates act afterward, which observable is measured and how that expectation becomes a probability or label. Define the loss, optimiser, number of measurements and

stopping rule. These choices determine an algorithm. The phrase “quantum neural network” alone does not specify any of them.

For a binary classifier, one can estimate the probability of measuring one on a designated wire and compare it with a threshold. Other designs use an expectation between negative and positive one or several observables. Calibration, class imbalance and confidence near the decision boundary need the same care as in ordinary statistical learning.

Worked example: a trainable one-qubit threshold

Encode x with a Y rotation and apply another Y rotation with parameter theta. Rotations about the same axis add, so the one probability is the sine squared of half x plus theta. Use two training examples: x zero with label zero and x pi with label one. With theta zero, both predictions are correct deterministically.

Use mean squared probability error as the loss. On the first example the prediction is sine squared of half theta. On the second it is cosine squared of half theta, whose error relative to label one is the negative of that same sine-squared value. Squaring and averaging gives the loss shown in the probability model below.

At theta zero the loss is zero. At theta pi over two, both predictions are one half and the loss is one quarter. At theta pi, the predictions are reversed and the loss is one. This entire learning landscape is classically calculable; the example makes the training task description visible rather than demonstrating an advantage.

The probability model explains the loss formula and its two training examples.

$$p_\theta(x) = \sin^2\left(\frac{x+\theta}{2}\right), \qquad L(\theta) = \frac{1}{2}\left[p_\theta(0)^2 + (p_\theta(\pi)-1)^2\right] = \sin^4(\theta/2).$$

The examples are $(x, y) = (0, 0)$ and $(\pi, 1)$. The loss averages squared probability errors, so it is a nonlinear function of the measured probabilities.

Estimate a gradient correctly

For a suitable single-parameter Pauli rotation, an expectation derivative can be obtained from two shifted circuit evaluations. In the toy probability, the derivative is one half sine of x plus theta, equal to half the difference between probabilities evaluated at theta plus pi over two and theta minus pi over two.

If the loss is a nonlinear function of that probability, apply the chain rule. The same simple two-shift rule should not be applied blindly to the entire squared loss as if it were one expectation. At theta pi over two in the worked example, the analytic loss derivative is one half. Checking this value is a useful way to catch a gradient implementation that differentiates the wrong quantity.

Each derivative estimate consumes additional circuits and shots. For many parameters, examples and optimiser iterations, training cost can substantially exceed one forward prediction. Finite-shot noise can obscure a small gradient, and a step size that works with exact expectations may behave poorly with noisy estimates.

Apply the parameter-shift rule to the probability, then the chain rule to the nonlinear loss.

$$\partial_\theta p_\theta(x) = \frac{p_{\theta+\pi/2}(x) - p_{\theta-\pi/2}(x)}{2}, \qquad L'(\theta) = 2\sin^3(\theta/2)\cos(\theta/2), \qquad L'(\pi/2) = \frac{1}{2}.$$

The two-shift identity assumes the stated single Pauli rotation dependence. Shifting the full squared loss as though it were a single expectation generally differentiates a different function.

A complete training procedure

Split data before tuning. Fix preprocessing using the training portion. Choose an architecture and initialisation, evaluate losses and gradients under a declared measurement budget, and update parameters. Use validation data for architecture and hyperparameter choices. Freeze the selected model and evaluate a separate test set, including repeated initialisations when training is stochastic.

Record total circuit evaluations, shots, parameter settings, optimiser updates and classical processing. If mitigation or repeated training restarts are used, include them. An impressive best training curve can hide many failed restarts or a large selection budget. Report those choices so the result can be reproduced and compared fairly.

Expressivity is not trainability

Adding gates can enlarge the family of representable functions, but it can also make gradients hard to estimate or introduce redundant parameters. A circuit that can represent an ideal classifier in principle may be difficult to train from practical initialisations. Conversely, a small problem-inspired circuit may be easier to train while representing a narrower family.

Barren-plateau results identify specified settings in which gradients concentrate near zero as systems grow. They motivate careful architecture and cost choices, not a claim that every variational model must fail. McClean and colleagues. Noise, data encoding and observable locality can create further trainability questions distinct from ordinary overfitting.

Generalisation and baselines

Perfect training labels do not establish useful predictions on new data. Compare test loss and calibration with classical models using a comparable tuning budget. For the toy example, an ordinary scalar threshold solves the task immediately, so implementing a circuit adds complexity without a demonstrated benefit. That is a valid educational outcome: understanding a model includes recognising when it is unnecessary.

For more structured tasks, state why the encoding and circuit might provide a useful inductive bias. If the hypothesis is that the circuit is hard to simulate, also show that its measured features are relevant to the labels. Computational difficulty and predictive usefulness are separate requirements.

Exercises and worked answers

Exercise 1. At x zero and theta pi over two, calculate the probability derivative using the two-shift rule.

> Check the worked answer
>
> The shifted angles are pi and zero, giving probabilities one and zero. Half their difference is one half, agreeing with the analytic derivative. The derivative concerns the probability; a loss derivative also includes the loss's derivative with respect to that probability.

Exercise 2. Why should the final model be evaluated on data that did not select theta, the architecture or the optimiser settings?

Check the worked answer

Repeated selection can fit accidental features of the data used for selection. A separate test estimates generalisation after those choices are frozen. Quantum measurements introduce additional sampling variation but do not remove the need to control ordinary data leakage.

As a further check, compare the model with the exact scalar formula before running shots. A disagreement in an exact simulator is an implementation error; variation around the formula in sampled predictions is a statistical issue. Keeping those layers separate makes a learning experiment easier to diagnose.

Check your understanding

PAUSE & THINK

What is the toy mean squared loss at $\theta = \pi/2$?

- [ ] 0
- [ ] 1/4
- [ ] 1

Answer: 1/4. Both predicted probabilities are 1/2, so each squared error is 1/4 and their mean is 1/4.

Sources

- Farhi and Neven
- McClean and colleagues