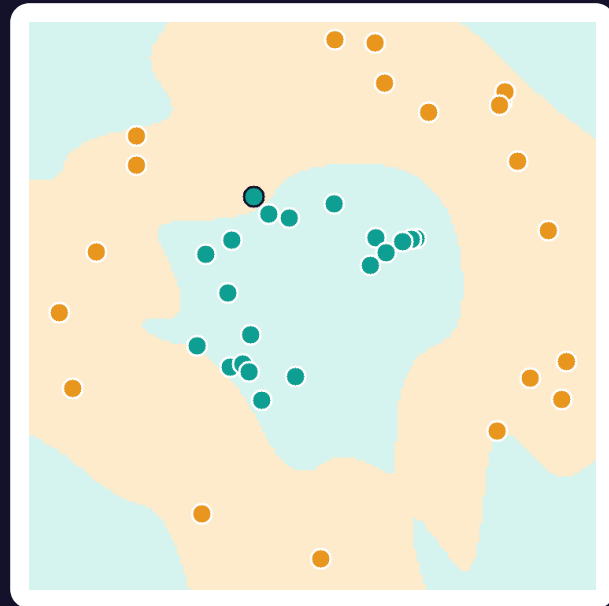


# Classifying data with a quantum kernel

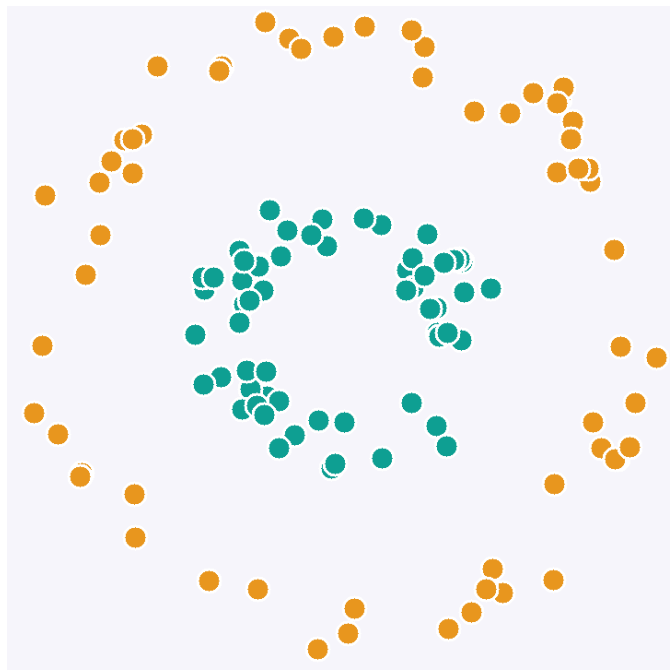
A basic use case of quantum computing for machine learning

V. Anbuchelvan · Quantum Machine Learning · SRM IST



What the classifier learns: two regions, one per ring

# Two rings that no straight line can split



● Inner ring    ● Outer ring

1

## 120 points, two classes

An inner ring and an outer ring. 80 points are used for training and 40 are held back for testing.

2

## Two features per point

Both are scaled to the range 0 to  $\pi$ , so they can later be used directly as rotation angles.

3

## A straight line fails

A linear support vector machine scores 60% on the test points, barely better than guessing.

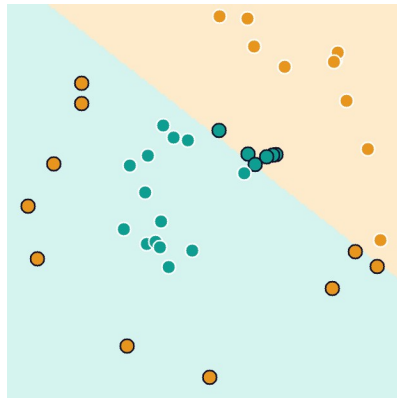
# The kernel idea

A support vector machine only ever asks one question about the data:

**How similar are  $x$  and  $y$ ?**

$k(x, y)$  = one number per pair of points

Swap the similarity measure and the same algorithm draws a different boundary. A richer measure acts like lifting the data into a larger space where a flat cut is enough.



**Plain dot product**

60% on test points

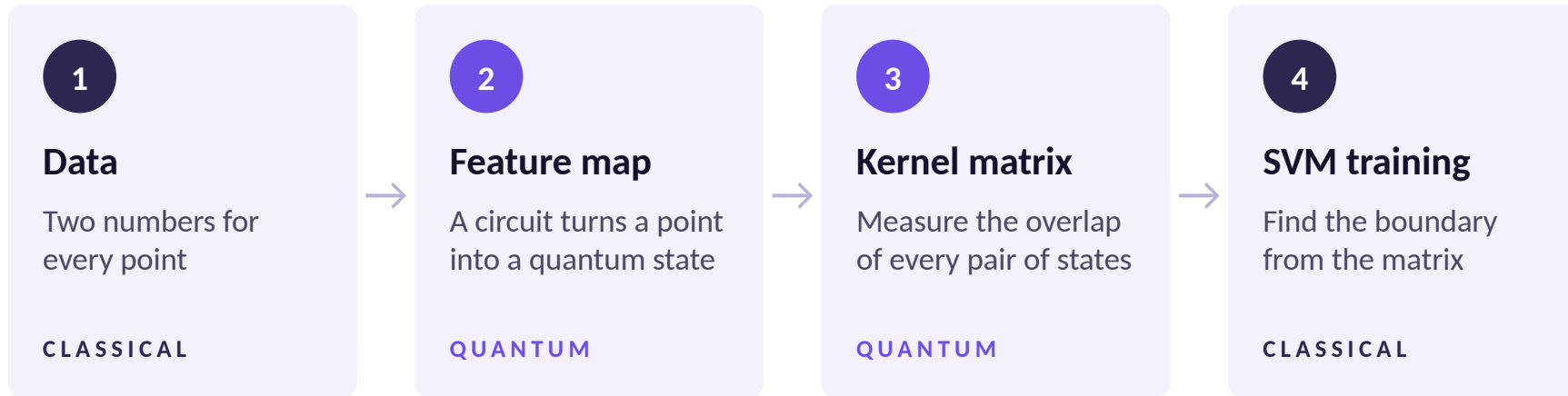


**Richer kernel**

97.5% on test points

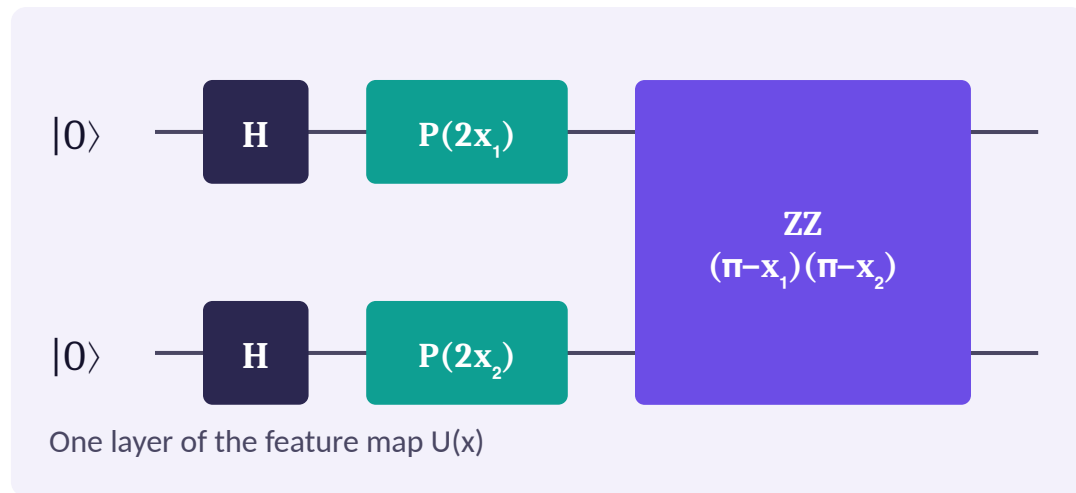
*Dark-ringed points are misclassified.*

# Where the quantum computer fits



Only steps 2 and 3 use qubits. The circuit is shallow, so it can run on the small, noisy machines that exist today.

# Encoding a point in two qubits



1

## Superpose

H puts each qubit in an equal mix of 0 and 1.

2

## Write the features

$x_1$  and  $x_2$  become phase angles.

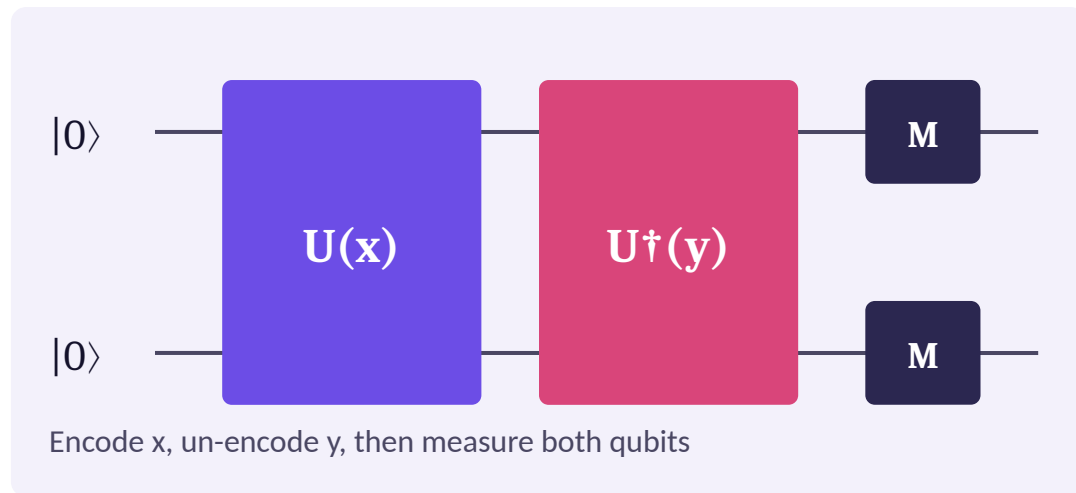
3

## Entangle

$ZZ$  links the qubits using both features.

$|\phi(\mathbf{x})\rangle = U(\mathbf{x})|00\rangle$  The point is now a state in a 4-dimensional space. Repeating the layer makes the map richer.

# Similarity is a measurement



$$P(00) = k(x, y)$$

The share of runs that read 00 is the similarity of the two points.

0.95

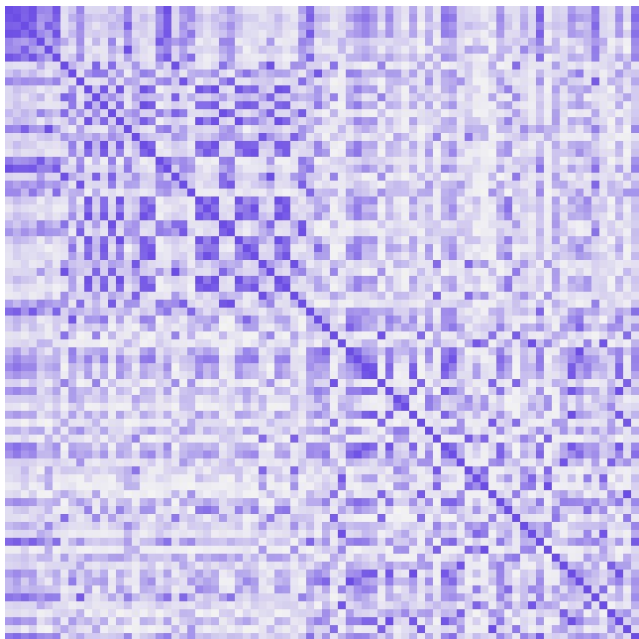
two points on the inner ring

0.22

one inner, one outer point

The same point twice always gives 1. The two values above are from the first training points of each ring.

# Training the classifier



Points sorted by ring. Darker = more similar.

80

training points

3,160

pairs, one circuit each

42

support vectors kept

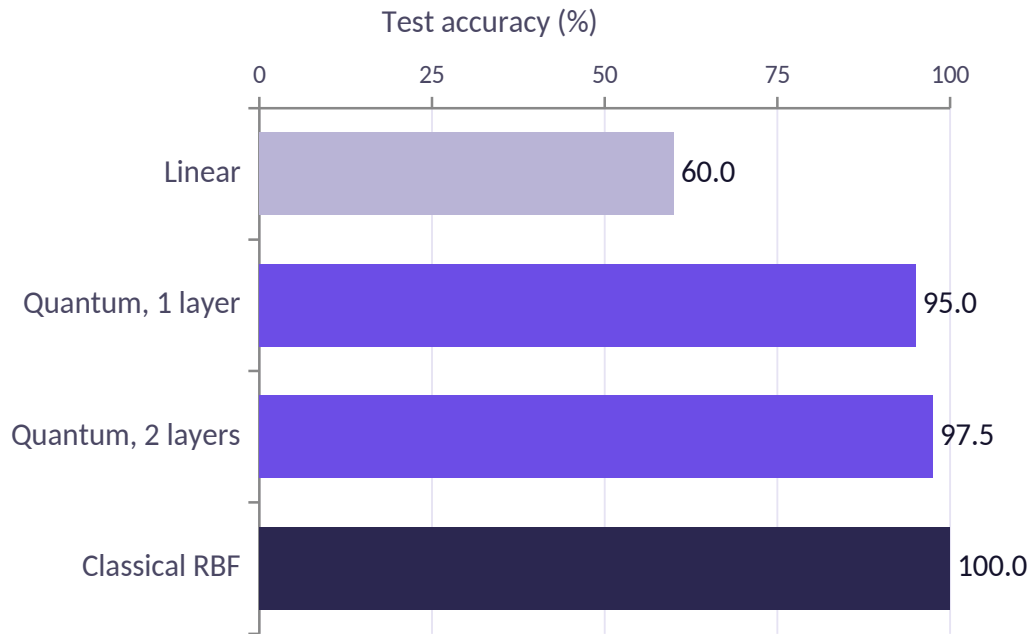
## The quantum circuit fills the matrix.

A classical solver then picks the points that sit near the boundary and weights them. Nothing about the solver is quantum.

## To classify a new point

run the circuit against each support vector and add up the weighted similarities.

# Result on 40 unseen points



Quantum kernel, 2 layers: 39 of 40 test points correct

# What this example does not show

1

## No quantum advantage

A classical RBF kernel scores 100% on the same points. Two qubits are easy to simulate on any laptop.

2

## An ideal simulation

These numbers are exact. Real hardware estimates each kernel value from repeated runs and adds noise.

3

## Cost grows with pairs

$n$  training points need  $n(n-1)/2$  circuits. Large datasets get expensive quickly.

Where a benefit is expected: feature maps that are hard to simulate classically and that match the structure of the data. Finding real datasets like that is an open research question.

# Takeaways

1

## A kernel is a similarity measure

An SVM needs nothing else from the data.

2

## A quantum circuit can compute one

Encode two points, measure the overlap, read off  $k(x, y)$ .

3

## The rest stays classical

A shallow, hybrid method that fits today's small machines.

## Try the live demo

[zanyanbu.github.io/  
quantum-kernel-classifier](https://zanyanbu.github.io/quantum-kernel-classifier)

## References

Havlíček et al., “Supervised learning with quantum-enhanced feature spaces”, Nature 567 (2019).

Schuld and Killoran, “Quantum machine learning in feature Hilbert spaces”, Phys. Rev. Lett. 122 (2019).