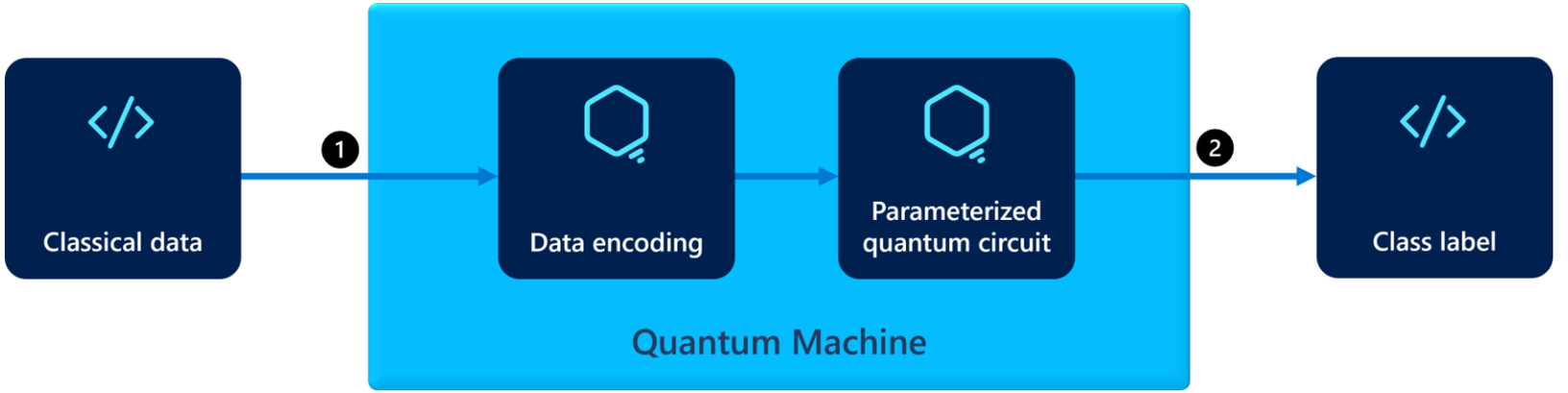
Microsoft

Quantum Machine Learning



Mariia Mykhailova
Principal Software Engineer
Microsoft Quantum Systems

Lecture outline

What is quantum machine learning?

Example: circuit-centric quantum classifiers

Challenges of QML

Microsoft

What is quantum machine learning?

What is quantum machine learning?

Classical machine learning is study of algorithms that act based on logic derived from past experiences that are provided as training data set

This translates to algorithms being able to learn from past data and generalize to future data

1. QML is using a quantum computer to solve a classical machine learning task better than a classical algorithm can

“Better” can mean fewer samples required in training data, faster processing time, higher accuracy, or finding correlations in data that are hard to describe classically

2. QML is using a quantum computer to extract features from quantum states

Finding exponential speedups over classical analogues in the first formulation is hard, but it might be possible to apply QML to problems that don’t have a classical analogue – for example, when data itself is generated by a quantum process

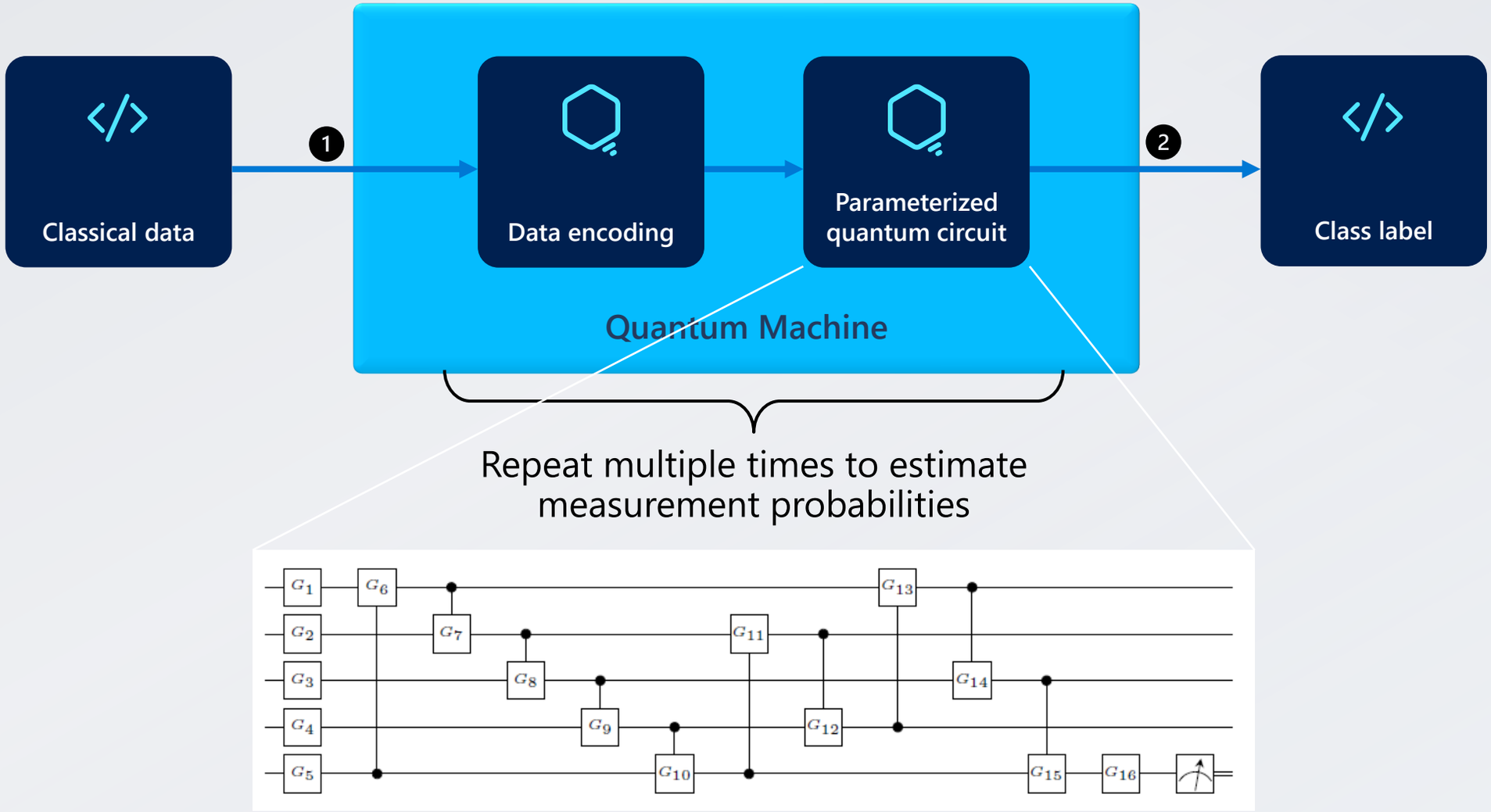
ML algorithms taxonomy based on data generation and processing

Data generation \ Processing	Classical	Quantum
Classical	Classical machine learning Quantum-inspired algorithms	"Traditional" QML: using quantum computer to solve classical problems better than a classical computer can
Quantum	Quantum characterization, verification and validation (QCVV): learning properties of quantum states/processes using measurements and classical ML tools	Less common algorithms such as quantum principal component analysis and generative training of quantum neural networks

Microsoft

Example: Circuit-Centric Quantum Classifiers

Classical data – quantum processing classifier



Data encoding

Bit-encoding: classical data is encoded as bit strings (individual basis states)

Same representation as in classical algorithms => makes comparisons easier

Easy to detect any difference between classical inputs

Can be implemented using QRAM (quantum random access memory) algorithms

Prohibitive space overheads: MNIST dataset (handwritten digits) has 784 pixels = 784 qubits!

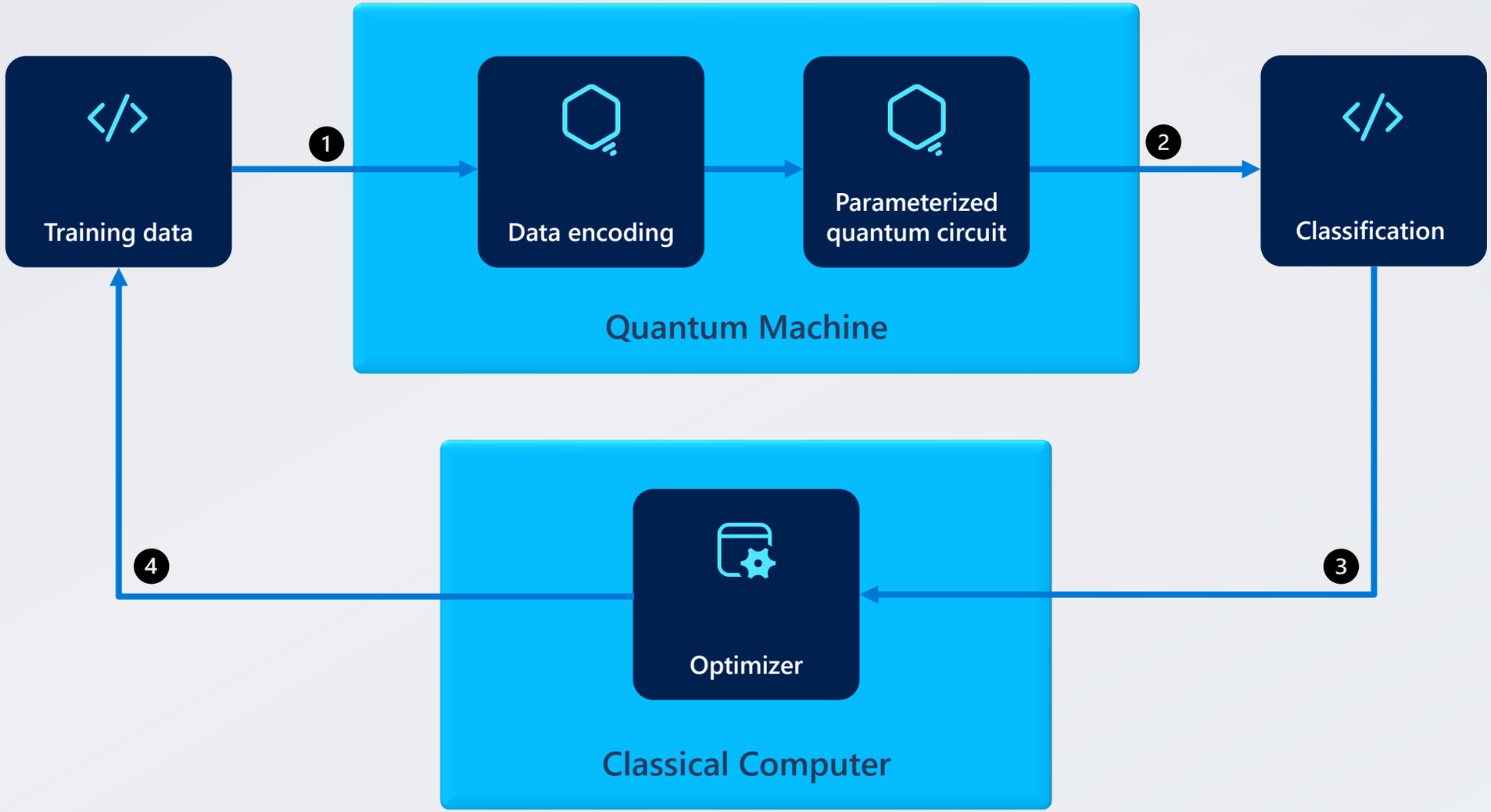
Amplitude-encoding: classical data is encoded as amplitudes of basis states (with normalization)

Hard to detect small differences between classical inputs

N qubits encode 2^N amplitudes => logarithmic space requirements (10 qubits for MNIST dataset)

Used in most QML experiments

Training the classifier (that's a variational algorithm!)



Single-qubit classification example

Data with 2 features (x_0, x_1)
Class 0 = $|x_0| < |x_1|$ (points closer to the vertical)
Class 1 = $|x_0| > |x_1|$ (points closer to the horizontal)

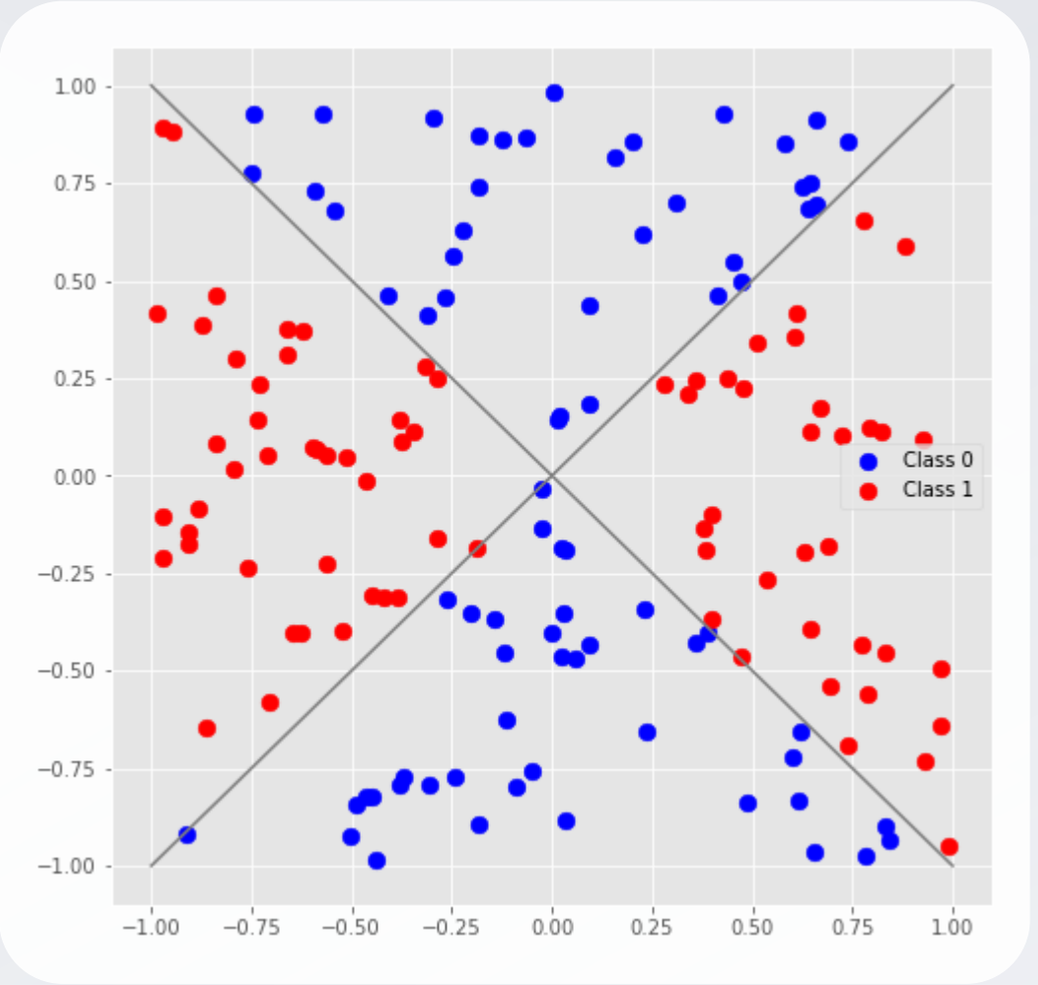
Data encoding: the data is normalized and encoded as a single-qubit quantum state

$$\frac{1}{\sqrt{x_0^2+x_1^2}}\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

Circuit **architecture**: a single gate $R_y(\theta)$

Numeric **parameters**:

- rotation angle θ
- bias b (see next slide)



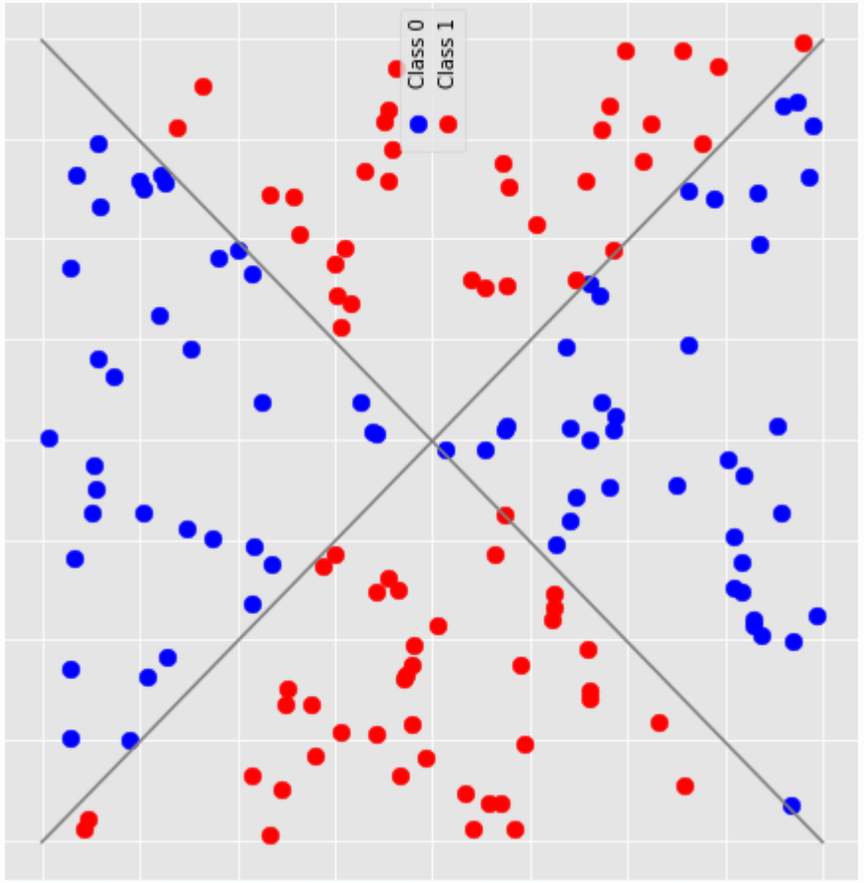
Single-qubit classification example (continued)

Classification process:

1. Encode the classical data into a quantum state by applying $R_y(\text{atan2}(x_1, x_0))$
2. Apply $R_y(\theta)$
3. Measure the quantum state to get 0 or 1
4. Repeat steps 1-3 multiple times to estimate measurement probabilities $P(0)$ and $P(1)$
5. The classification label is 1 if $P(1) + b > P(0)$ and 0 otherwise

   E.g., for bias $b = 0$, the classification label is the more frequent measurement result

For this example, the optimal parameters are $\theta = \pi$ (rotate the picture by $\frac{\pi}{2}$) and $b = 0$

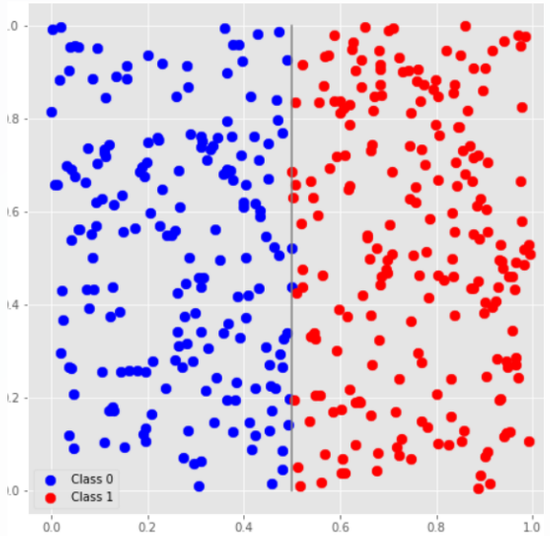


What can this classifier architecture do?

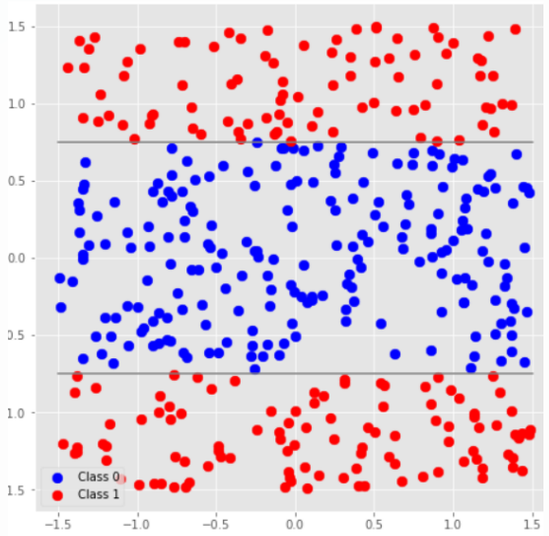
One-qubit classifier with two parameters can classify only this kind of angular data

Classification angle θ affects the rotation of the classes, bias b – the angle of the classes

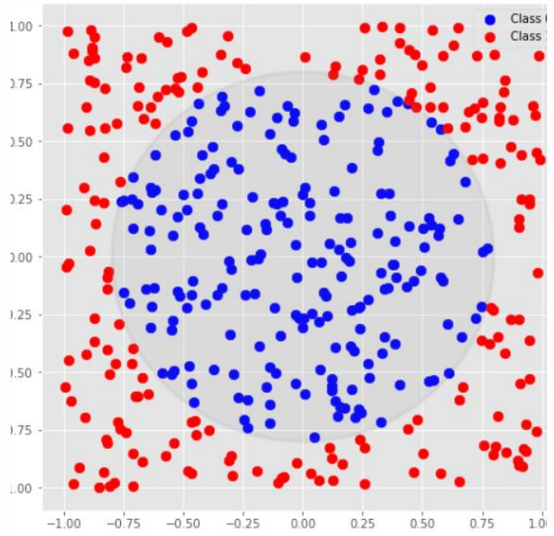
What other shapes of data can it classify?



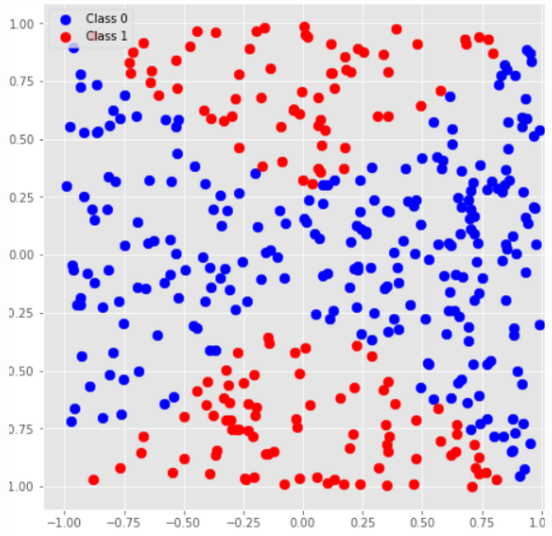
Vertically separable classes



Horizontally separable classes



Classes separated by a circle



Classes separated by a hyperbola

Feature engineering examples

Padding: new features append the parameters array to the original features

- Parameter a_0 + original features $\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$ = new features $\begin{pmatrix} a_0 \\ x_0 \\ x_1 \\ 0 \end{pmatrix}$
- Parameters $\begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$ + original features $\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$ = new features $\begin{pmatrix} a_0 \\ a_1 \\ x_0 \\ x_1 \end{pmatrix}$

Split fanout: new features are a tensor product of "concatenation of the left halves of parameters and original features" and "concatenation of the right halves of parameters and original features"

- Parameters $\begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$ + original features $\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$ = new features $\begin{pmatrix} a_0 a_1 \\ a_0 x_1 \\ x_0 a_1 \\ x_0 x_1 \end{pmatrix} = \begin{pmatrix} a_0 \\ x_0 \end{pmatrix} \otimes \begin{pmatrix} a_1 \\ x_1 \end{pmatrix}$

Feature engineering 1: vertically separable classes

Two classes divided by a vertical line $x_0 = 0.5$

- Classification can use only the first feature
- Split fanout allows to isolate the two features!

  Parameters $\begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$ + features $\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$ = new features

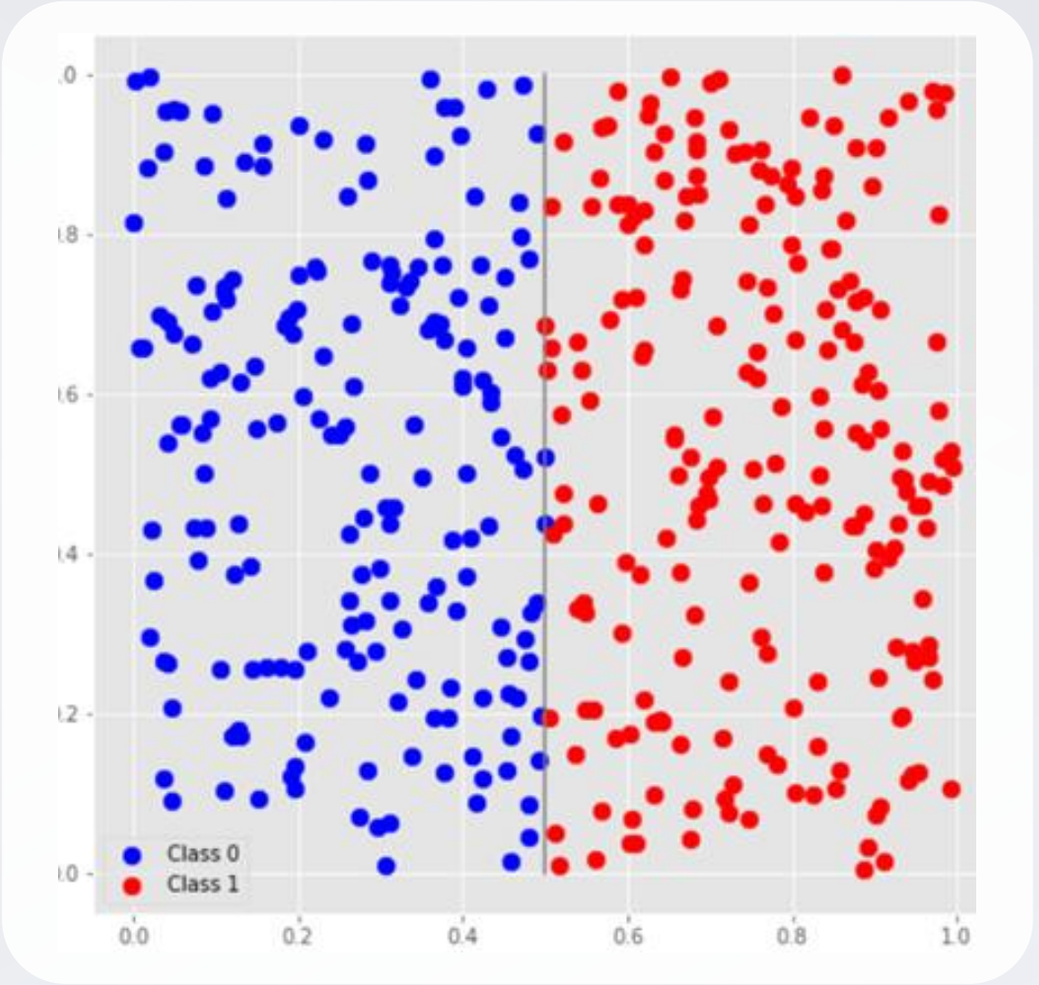
$$\begin{pmatrix} a_0 \\ x_0 \end{pmatrix} \otimes \begin{pmatrix} a_1 \\ x_1 \end{pmatrix}$$

Measure the qubit $\begin{pmatrix} a_0 \\ x_0 \end{pmatrix}$:

0 with probability a_0^2 and 1 with probability x_0^2

For $\theta = 0$ (no rotation) and $b = 0$ the data point is classified as 1 if $x_0^2 > a_0^2$; $a_0 = 0.5$

With rotation, can handle classes separable by a line



Feature engineering 2: horizontally separable classes

Two classes divided by horizontal lines $|x_1| = 0.75$

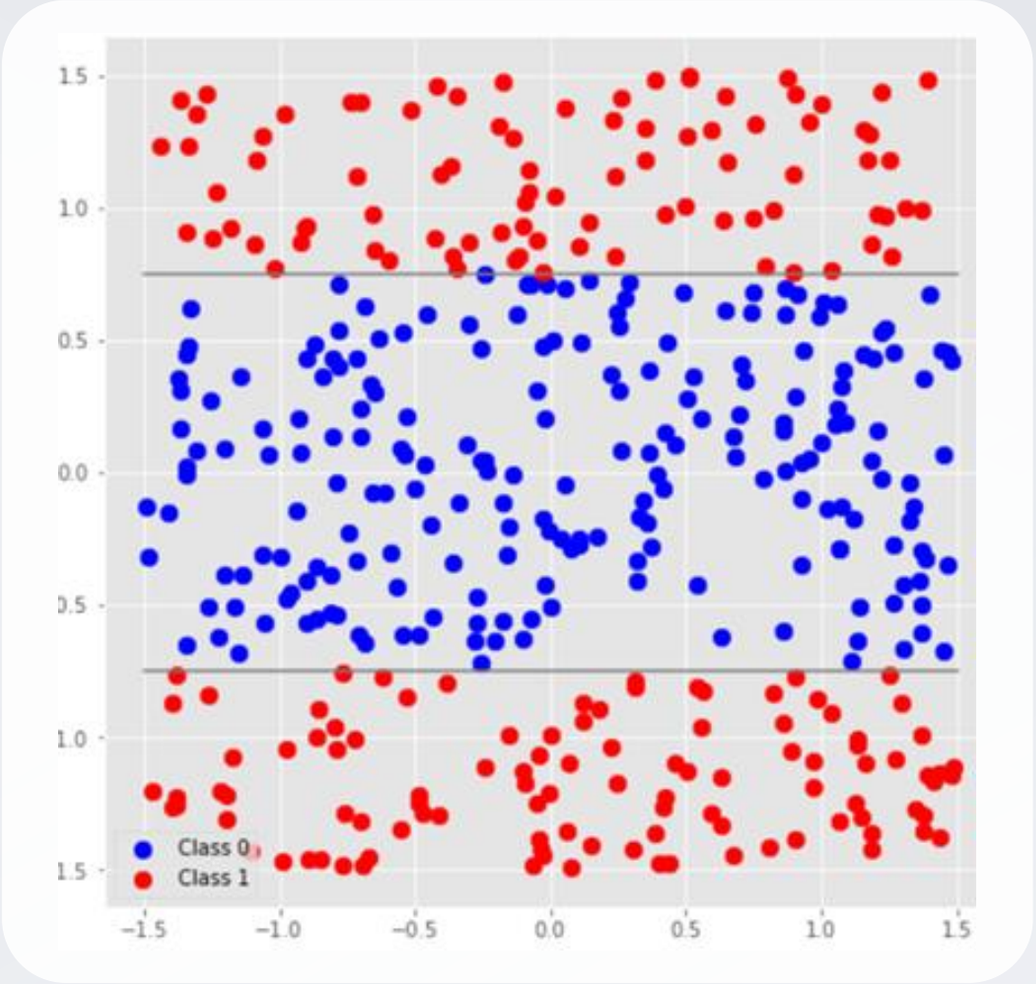
- Classification can use only the second feature
- Split fanout allows to isolate the two features!

$$\binom{a_0}{x_0} \otimes \binom{a_1}{x_1}$$

- Swap the two qubits as part of the classifier circuit

Measure the qubit $\binom{a_1}{x_1}$:
0 with probability a_1^2 and 1 with probability x_1^2

For $\theta = 0$ (no rotation) and $b = 0$ the data point is classified as 1 if $x_1^2 > a_1^2$; $a_1 = 0.75$



Feature engineering 3: classes separated by a circle

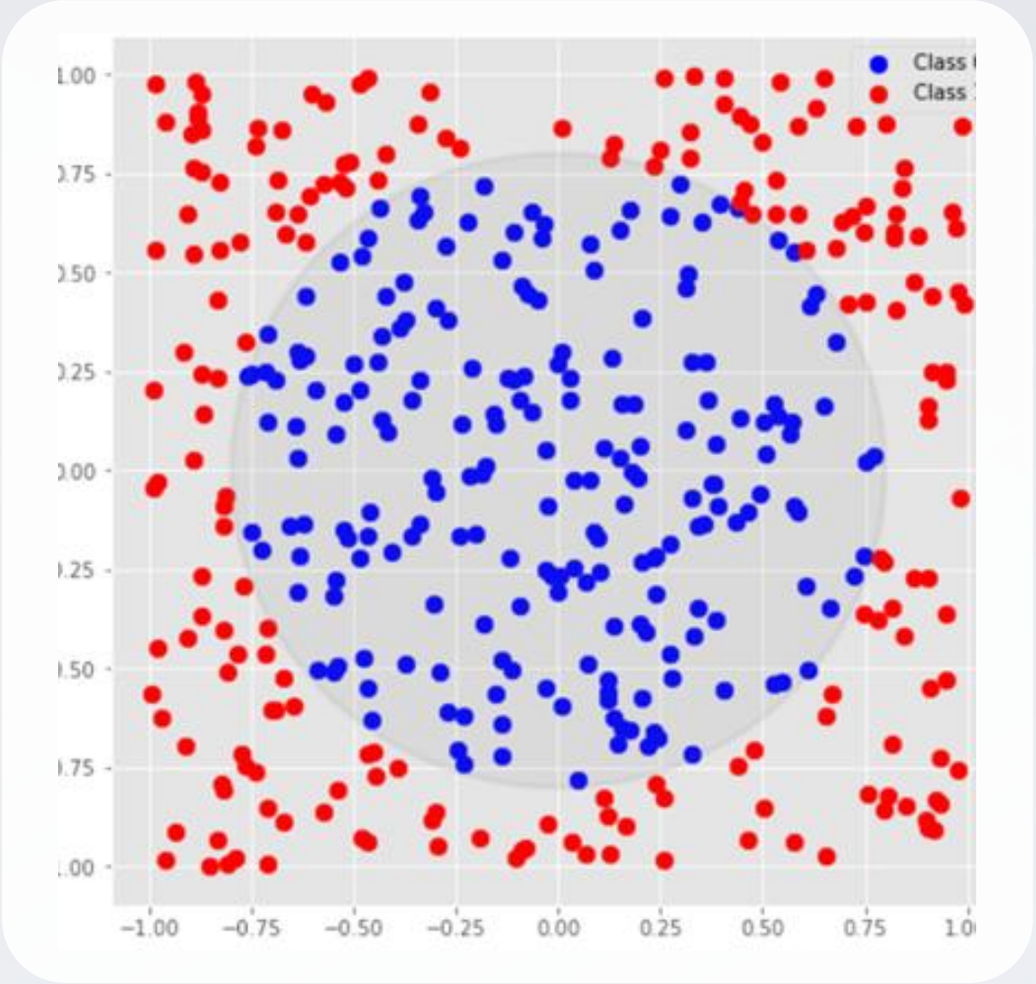
Two classes divided by a circle $x_0^2 + x_1^2 = 0.8^2$

- The squares of features look like the squares of amplitudes in measurement probabilities
- Padding allows us to get the state

$$\begin{pmatrix} 0.8 \\ 0 \\ x_0 \\ x_1 \end{pmatrix} = 0.8|00\rangle + x_0|01\rangle + x_1|11\rangle$$

Measure the second qubit:
0 with probability 0.8^2 and 1 with probability $x_0^2 + x_1^2$

$\theta = 0$ (no rotation) and $b = 0$



Feature engineering 4: classes separated by a hyperbola

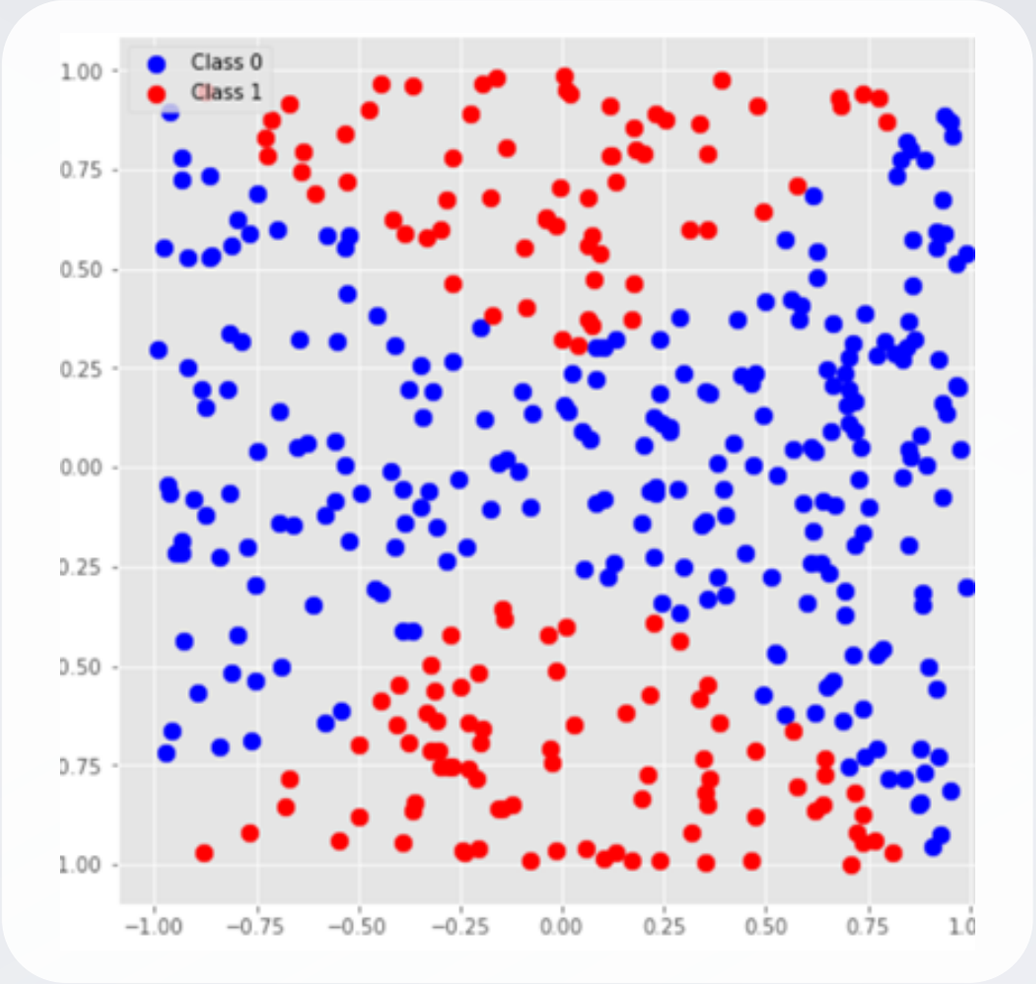
Two classes divided by a hyperbola $0.3^2 + x_0^2 = x_1^2$

- The squares of features look like the squares of amplitudes in measurement probabilities
- Padding allows us to get the state

$$\begin{pmatrix} 0.3 \\ x_0 \\ x_1 \\ 0 \end{pmatrix} = 0.3|00\rangle + x_0|10\rangle + x_1|01\rangle$$

Measure the second qubit:
0 with probability $0.3^2 + x_0^2$ and 1 with probability x_1^2

$\theta = 0$ (no rotation) and $b = 0$



Microsoft

Quantum Machine Learning: Challenges and State of Research

QML: Challenges

Input problem: loading classical data into quantum algorithm is very inefficient

Output problem: extracting classically meaningful data from quantum state output is hard

Finding a classical analogue accurately and comparing performance is hard

Lack of sufficient computing power to train and validate models

Lack of standardized datasets and metrics for performance evaluation

Lack of standardized software tools used for implementing QML algorithms

Plus all the same challenges that more general variational quantum algorithms have!

QML: State of Research

QML attracted a lot of attention (and funding) as a combination of Q and ML

The last decade saw a lot of research on QML

Lately the consensus starts to lean towards QML not yielding practical advantage

Same as VQA in general