Kwantowe uczenie maszynowe i kwantowe metody jądrowe

Piotr Gawron

Instytut Informatyki Teoretycznej i Stosowanej Polskiej Akademii Nauk

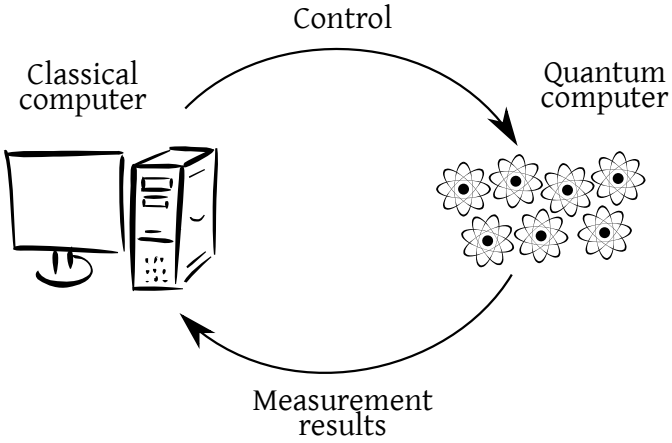
13 marca 2026

AKADEMIA
SZTUKI KWANTOWEJ

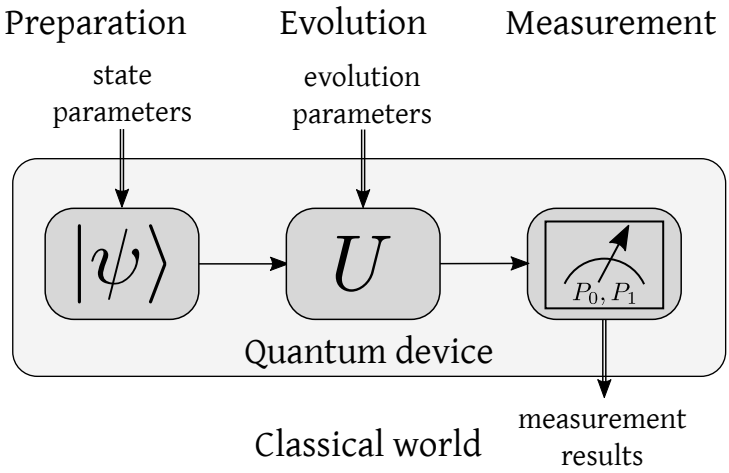
Building blocks of quantum world

Classical and quantum computing

Quantum computers are not self-sufficient



Scheme of a quantum device



Quantum states

Bits and qubits

Bit

0 or 1

Qubit

$\alpha |0\rangle + \beta |1\rangle$

Mathematical properties of qubits

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix}$$

$$|\alpha|^2 + |\beta|^2 = 1$$

$$e^{i\phi}\,|\psi\rangle \equiv |\psi\rangle$$

Dirac bra-ket notation

Let

$$|\psi\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix}, \quad |\phi\rangle = \begin{bmatrix}\gamma\\ \delta\end{bmatrix}, \quad \langle\psi| = |\psi\rangle^\dagger = \begin{bmatrix}\overline{\alpha}, \overline{\beta}\end{bmatrix}.$$

Properties:

- inner (scalar) product

$$\langle\psi|\phi\rangle = \begin{bmatrix}\overline{\alpha}, \overline{\beta}\end{bmatrix} \begin{bmatrix}\gamma\\ \delta\end{bmatrix} = \overline{\alpha}\gamma + \overline{\beta}\delta,$$

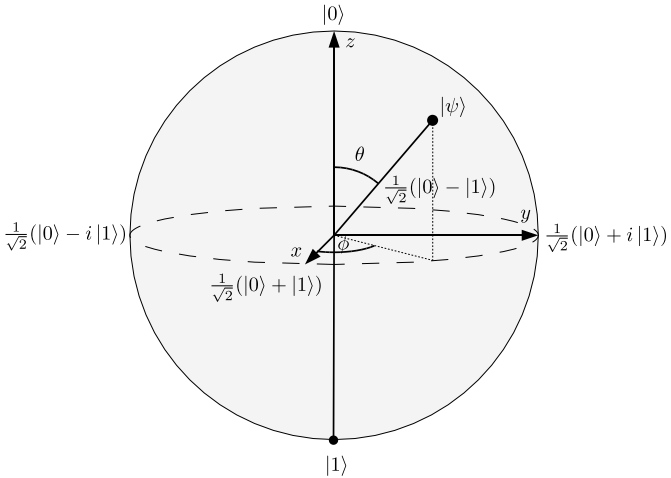
- length

$$\sqrt{\langle\psi|\psi\rangle} = \sqrt{|\alpha|^2 + |\beta|^2} = 1,$$

- outer product

$$|\psi\rangle\langle\phi| = \begin{bmatrix}\alpha\\ \beta\end{bmatrix} \begin{bmatrix}\overline{\gamma} & \overline{\delta}\end{bmatrix} = \begin{bmatrix}\alpha\overline{\gamma} & \alpha\overline{\delta}\\ \beta\overline{\gamma} & \beta\overline{\delta}\end{bmatrix}.$$

Representation of qubits



Quantum measurements

Measurement

Quantum measurement is a function μ:

- from finite set of measurements outcomes $A = \{a_i\}_{i=0}^{n-1}$
- to set the of projection operators $P = \{P_i\}_{i=0}^{n-1}$ (where projections satisfy $P_i^2 = P_i$) such that

$$\sum_{i=0}^{n-1} P_i = \mathbb{I}.$$

Measurement

The **probability of measuring outcome** a_i given the quantum system in in state $|\psi\rangle$ is

$$p(a_i) = \langle\psi| P_i |\psi\rangle .$$

The state of the quantum system **after the measurement** outcome a_i was obtained becomes

$$|\psi\rangle_{a_i} = \frac{P_i |\psi\rangle}{\sqrt{\langle\psi| P_i |\psi\rangle}}.$$

Example 1 — part I

We will measure the state

$$|\psi\rangle = \sqrt{0.7}\,|0\rangle + \sqrt{0.3}\mathrm{i}\,|1\rangle$$

by a measurement

$$A = \{0, 1\}, \quad P = \left\{ P_0 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},\ P_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right\}.$$

Probabilities of obtaining outcomes 0 and 1

$$p(0) = \langle\psi|\, P_0\, |\psi\rangle = \begin{bmatrix} \sqrt{0.7} & -\sqrt{0.3}\mathrm{i} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{0.7} \\ \sqrt{0.3}\mathrm{i} \end{bmatrix} = 0.7,$$

$$p(1) = \langle\psi|\, P_1\, |\psi\rangle = \begin{bmatrix} \sqrt{0.7} & -\sqrt{0.3}\mathrm{i} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{0.7} \\ \sqrt{0.3}\mathrm{i} \end{bmatrix} = 0.3.$$

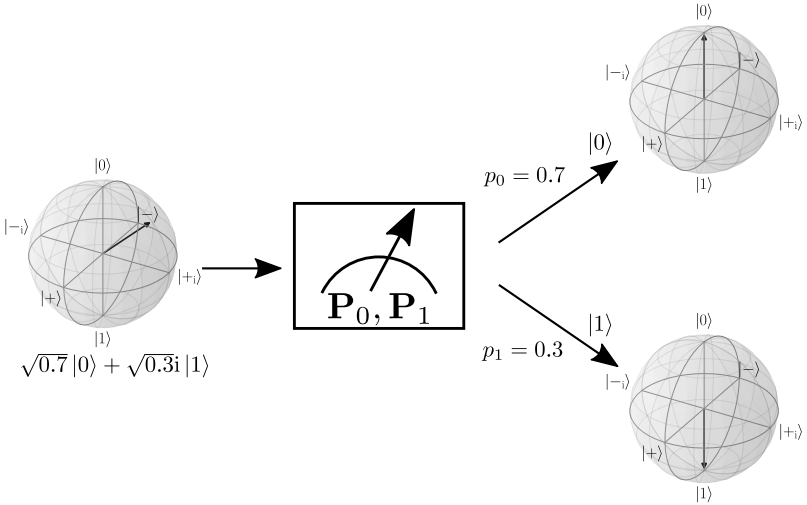
Example 1 — part II

Conditional states after the measurement

$$|\psi\rangle_0 = \frac{P_0\,|\psi\rangle}{\sqrt{\langle\psi|\,P_0\,|\psi\rangle}} = \frac{1}{\sqrt{0.7}}\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\sqrt{0.7}\\ \sqrt{0.3}\mathrm{i}\end{bmatrix} = \begin{bmatrix}1\\ 0\end{bmatrix}$$

$$|\psi\rangle_1 = \frac{P_1\,|\psi\rangle}{\sqrt{\langle\psi|\,P_1\,|\psi\rangle}} = \frac{1}{\sqrt{0.3}}\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\sqrt{0.7}\\ \sqrt{0.3}\mathrm{i}\end{bmatrix} = \begin{bmatrix}0\\ 1\end{bmatrix}$$

Example 1 — part III



Example 2 — part I

We will measure the state

$$|\psi\rangle = \sqrt{0.7}\,|0\rangle + \sqrt{0.3}\mathrm{i}\,|1\rangle$$

by a measurement

$$A = \{-_\mathrm{i}, +_\mathrm{i}\}, \quad Q = \left\{ Q_{-_\mathrm{i}} = \frac{1}{2}\begin{bmatrix} 1 & \mathrm{i} \\ -\mathrm{i} & 1 \end{bmatrix},\ Q_{+_\mathrm{i}} = \frac{1}{2}\begin{bmatrix} 1 & -\mathrm{i} \\ \mathrm{i} & 1 \end{bmatrix} \right\}.$$

Probabilities of obtaining outcomes $-_\mathrm{i}$ and $+_\mathrm{i}$

$$p(-_\mathrm{i}) = \langle\psi|\, Q_{-_\mathrm{i}}\, |\psi\rangle = \begin{bmatrix} \sqrt{0.7} & -\sqrt{0.3}\mathrm{i} \end{bmatrix} \frac{1}{2}\begin{bmatrix} 1 & \mathrm{i} \\ -\mathrm{i} & 1 \end{bmatrix} \begin{bmatrix} \sqrt{0.7} \\ \sqrt{0.3}\mathrm{i} \end{bmatrix} = \frac{1}{10}(5 - \sqrt{21}) \approx 0.042,$$

$$p(+_\mathrm{i}) = \langle\psi|\, Q_{+_\mathrm{i}}\, |\psi\rangle = \begin{bmatrix} \sqrt{0.7} & -\sqrt{0.3}\mathrm{i} \end{bmatrix} \frac{1}{2}\begin{bmatrix} 1 & -\mathrm{i} \\ \mathrm{i} & 1 \end{bmatrix} \begin{bmatrix} \sqrt{0.7} \\ \sqrt{0.3}\mathrm{i} \end{bmatrix} = \frac{1}{10}(5 + \sqrt{21}) \approx 0.958.$$

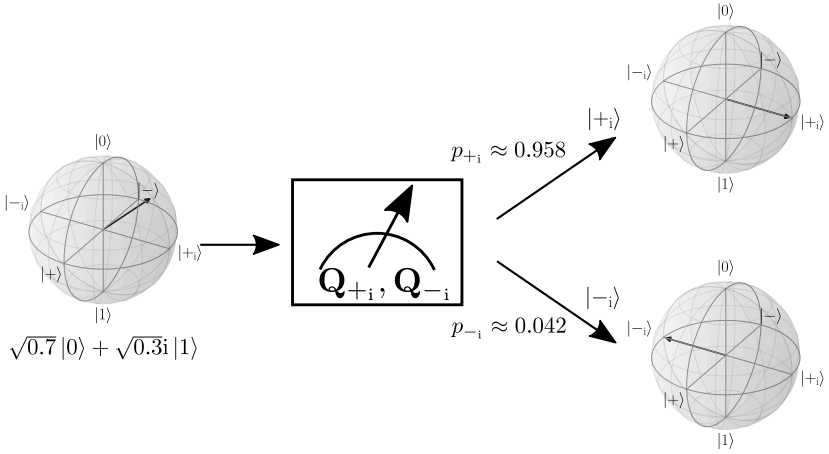
Example 2 — part II

Conditional states after the measurement

$$|\psi\rangle_{-_{\mathrm{i}}} = \frac{Q_{-_{\mathrm{i}}}|\psi\rangle}{\sqrt{\langle\psi|Q_{-_{\mathrm{i}}}|\psi\rangle}} = \frac{1}{\sqrt{\frac{1}{10}(5-\sqrt{21})}}\frac{1}{2}\begin{bmatrix}1 & \mathrm{i}\\ -\mathrm{i} & 1\end{bmatrix}\begin{bmatrix}\sqrt{0.7}\\ \sqrt{0.3}\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ -\mathrm{i}\end{bmatrix} = |-_{\mathrm{i}}\rangle$$

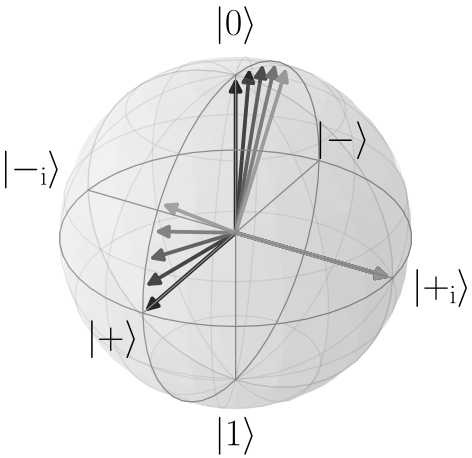
$$|\psi\rangle_{+_{\mathrm{i}}} = \frac{Q_{+_{\mathrm{i}}}|\psi\rangle}{\sqrt{\langle\psi|Q_{+_{\mathrm{i}}}|\psi\rangle}} = \frac{1}{\sqrt{\frac{1}{10}(5+\sqrt{21})}}\frac{1}{2}\begin{bmatrix}1 & -\mathrm{i}\\ \mathrm{i} & 1\end{bmatrix}\begin{bmatrix}\sqrt{0.7}\\ \sqrt{0.3}\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ \mathrm{i}\end{bmatrix} = |+_{\mathrm{i}}\rangle$$

Example 2 — part III



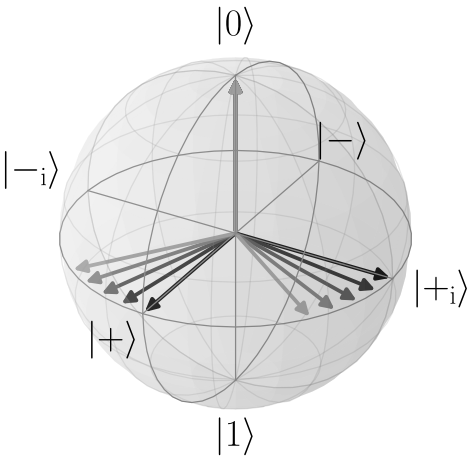
Quantum evolution

Quantum evolution



Rotation around y-axis

Quantum evolution



Rotation around z-axis

Entanglement and Bell states

Composition of quantum systems

Tensor product for **two qubit** states

$$|\psi\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \alpha\,|0\rangle + \beta\,|1\rangle\,,$$

$$|\phi\rangle = \begin{bmatrix}\gamma\\ \delta\end{bmatrix} = \gamma\,|0\rangle + \delta\,|1\rangle\,,$$

and their joint state in $\mathbb{C}^2 \otimes \mathbb{C}^2$

$$|\psi\rangle \otimes |\phi\rangle = \begin{bmatrix}\alpha\gamma\\ \alpha\delta\\ \beta\gamma\\ \beta\delta\end{bmatrix},$$

$$|\psi\rangle \otimes |\phi\rangle = \alpha\gamma\,|0\rangle \otimes |0\rangle + \alpha\delta\,|0\rangle \otimes |1\rangle + \beta\gamma\,|1\rangle \otimes |0\rangle + \beta\delta\,|1\rangle \otimes |1\rangle\,,$$

$$|\psi\phi\rangle = \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle\,.$$

Entanglement

Recall the product state

$$|\psi\phi\rangle = \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle\,,$$

and consider another state

$$|\Psi\rangle = a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle\,.$$

Question

Can we always find coefficients which satisfy

$$\begin{aligned} a &= \alpha\gamma \\ b &= \alpha\delta \\ c &= \beta\gamma \\ d &= \beta\delta? \end{aligned}$$

Entanglement

Bell State

$$\left|\Phi^{+}\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

$$\left|\Phi^{+}\right\rangle \stackrel{?}{=} \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle\,,$$

Contradiction

$$\alpha\gamma = \beta\delta = \frac{1}{\sqrt{2}},$$
$$\alpha\delta = \beta\gamma = 0.$$

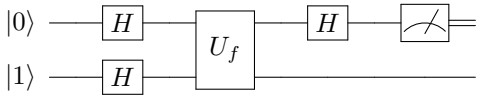
Gate model of computation

Gate model

How to perform computations?

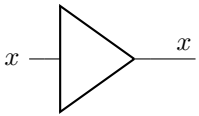
- Prepare the system in a known state, eg.: $|0\rangle \otimes |1\rangle$.
- Apply quantum gates and partial measurements on the state.
- Perform the measurement.
- Analyze the result and post-processing.

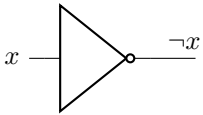
Example of a quantum circuit

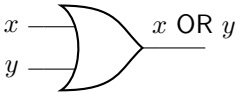


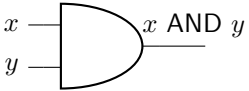
Classical gates

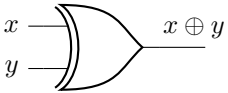
Classical gates







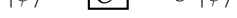




Single qubit gates

Single qubit gate — unitary gate

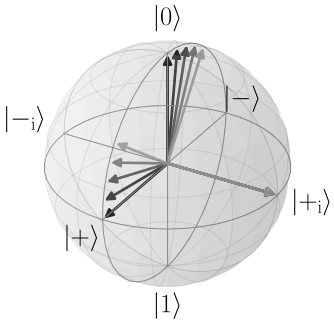
How to denote a quantum single-qubit gate?

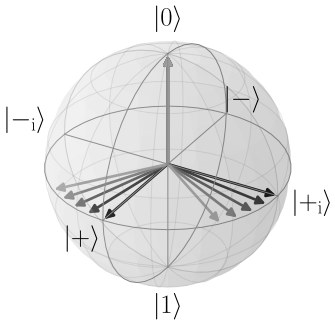


Quantum evolution is time-reversible!

$$U|\psi_{t=0}\rangle = |\psi_{t=1}\rangle$$
$$U^\dagger|\psi_{t=1}\rangle = |\psi_{t=0}\rangle$$

Rotations





$$R_y(\gamma) = \begin{bmatrix} \cos\left(\frac{\gamma}{2}\right) & \sin\left(\frac{\gamma}{2}\right) \\ -\sin\left(\frac{\gamma}{2}\right) & \cos\left(\frac{\gamma}{2}\right) \end{bmatrix}$$

$$R_z(\beta) = \begin{bmatrix} \mathrm{e}^{\frac{\mathrm{i}\beta}{2}} & 0 \\ 0 & -\mathrm{e}^{\frac{\mathrm{i}\beta}{2}} \end{bmatrix}$$

Important examples of single-qubit gates

$$\alpha|0\rangle + \beta|1\rangle \xrightarrow{\boxed{I}} \alpha|0\rangle + \beta|1\rangle \qquad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\alpha|0\rangle + \beta|1\rangle \xrightarrow{\boxed{X}} \beta|0\rangle + \alpha|1\rangle \qquad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

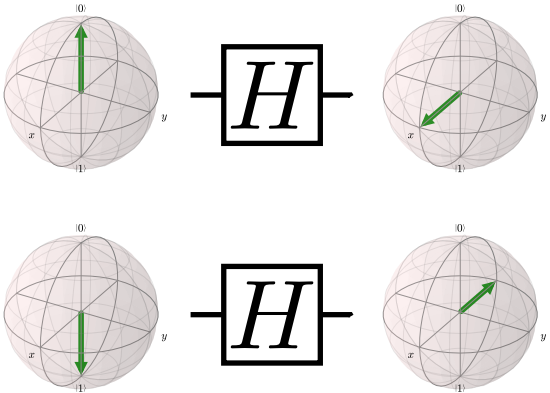
$$\alpha|0\rangle + \beta|1\rangle \xrightarrow{\boxed{Y}} -\beta\mathrm{i}|0\rangle + \alpha\mathrm{i}|1\rangle \qquad \begin{bmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{bmatrix}$$

$$\alpha|0\rangle + \beta|1\rangle \xrightarrow{\boxed{Z}} \alpha|0\rangle - \beta|1\rangle \qquad \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\alpha|0\rangle + \beta|1\rangle \xrightarrow{\boxed{P(\phi)}} \alpha|0\rangle + e^{i\phi}\beta|1\rangle \qquad \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\phi} \end{bmatrix}$$

Hadamard gate

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H|0\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}, \quad H|1\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}.$$



Decomposition of a unitary gate

Definition

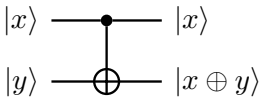
A square matrix is called *unitary* iff

$$UU^\dagger = U^\dagger U = \mathbb{1}.$$

Every unitary gate U can be written as

$$U = R_Z(\psi)R_Y(\theta)R_Z(\Delta) = e^{i\varphi/2}\begin{bmatrix} e^{i\psi} & 0 \\ 0 & e^{-i\psi} \end{bmatrix}\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} e^{i\Delta} & 0 \\ 0 & e^{-i\Delta} \end{bmatrix}.$$

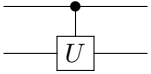
CNOT gate



in_1	in_2	out_1	out_2
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} : \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} \rightarrow \begin{bmatrix} \alpha \\ \beta \\ \delta \\ \gamma \end{bmatrix}$$

Controlled unitary

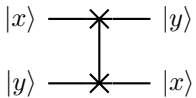


$$U = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}$$

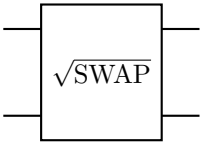
$$CU_1^2 = |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix}$$

2-qubit gates

Swap gate

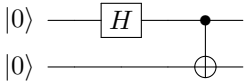


$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} : \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} \rightarrow \begin{bmatrix} \alpha \\ \gamma \\ \beta \\ \delta \end{bmatrix}$$



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1+i}{2} & \frac{1-i}{2} & 0 \\ 0 & \frac{1-i}{2} & \frac{1+i}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} : \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} \rightarrow \begin{bmatrix} \alpha \\ \frac{1+i}{2}\beta + \frac{1-i}{2}\gamma \\ \frac{1-i}{2}\beta + \frac{1+i}{2}\gamma \\ \delta \end{bmatrix}$$

Example of a circuit



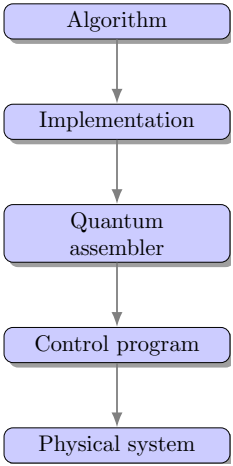
1. $(H \otimes I)\,|0\rangle \otimes |0\rangle = \frac{1}{\sqrt{2}}\,(|0\rangle + |1\rangle) \otimes |0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

2. $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}\,(|00\rangle + |11\rangle)$

From algorithm to physical realization

Programming stack

Programming stack



QuTiP implementation

Implementation

QuTiP implementation of Deutsch's algorithm for function $f(0) = 1, f(1) = 0$.

```
qc = QubitCircuit(2, num_cbits=1)
qc.add_gate("X", 1)
qc.add_gate("SNOT", 1)
qc.add_gate("SNOT", 0)
qc.add_gate("CNOT", 1, 0)
qc.add_gate("X", 1)
qc.add_gate("SNOT", 0)
qc.add_measurement("M0", targets=[0], classical_store=0)
```

Quantum assembler

Quantum assembler

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[1];

x q[1];
h q[1];
h q[0];
cx q[0],q[1];
x q[1];
h q[0];
measure q[0] -> c[0]
```

Quantum physical system

Physical system

Schrödinger equation

$$i\hbar\frac{\partial\Psi(t)}{\partial t} = \hat{H}\Psi(t)$$

- H is a Hermitian operator (*i.e.* $H = H^\dagger$) called **Hamiltonian** of the system.
- $\Psi(t)$ is a *wave function* of the system
- $\hbar$ is the reduced Planck constant

Physical system

Solution to the Schrödinger equation:

$$|\psi_{t_1}\rangle = U(t_0, t_1)\,|\psi_{t_0}\rangle\,,$$

where

- $|\psi_{t_0}\rangle$ — initial state of the system,
- $|\psi_{t_1}\rangle$ — final state of the system
- $U(t_0, t_1)$ — **unitary operator** driving the system from t_0 to t_1.

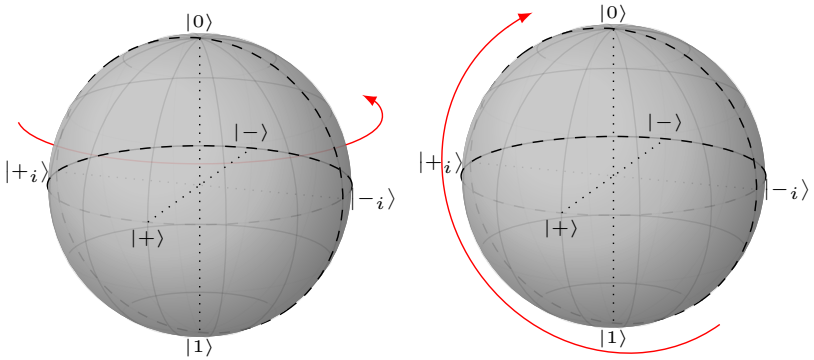
- An operator is unitary iff $UU^\dagger = U^\dagger U = \mathbb{1}$
- If Hamiltonian H is constant between time t_0 and t_1, then:

$$U(t_0, t_1) = e^{-i(t_1 - t_0)H},$$

where $e^M = \sum_{k=0}^{\infty} \frac{M^k}{k!}$.

Physical system

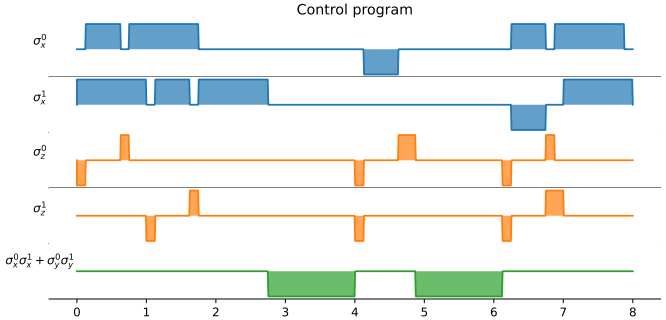
$$H(t) = h_{x_0}(t)\sigma_x^0 + h_{x_1}(t)\sigma_x^1 + h_{z_0}(t)\sigma_z^0 + h_{z_1}(t)\sigma_z^1 + J(t)\left(\sigma_x^0\sigma_x^1 + \sigma_y^0\sigma_y^1\right)$$



Steering the program

Control program

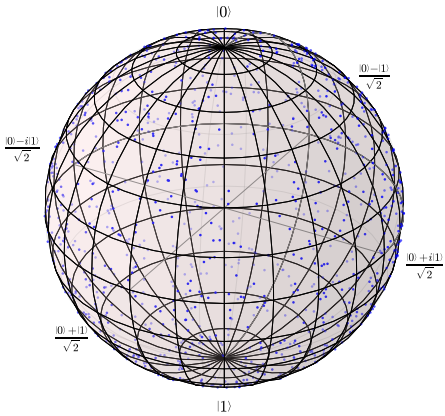
$$H(t) = h_{x_0}(t)\sigma_x^0 + h_{x_1}(t)\sigma_x^1 + h_{z_0}(t)\sigma_z^0 + h_{z_1}(t)\sigma_z^1 + J(t)\left(\sigma_x^0\sigma_x^1 + \sigma_y^0\sigma_y^1\right)$$



Problem of noisy devices

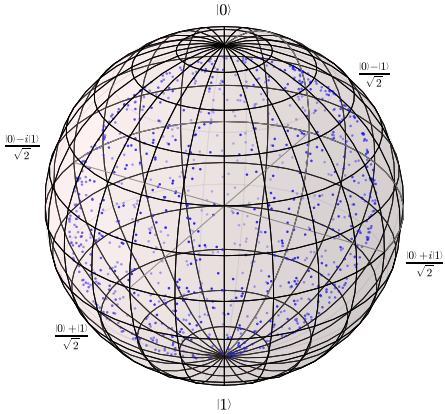
A zoo of quantum noisy channels

Qubit channels — zoo



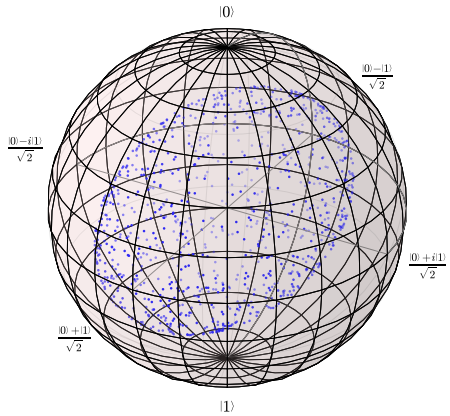
Rysunek: Uniformly chosen 1000 random pure states.

Qubit channels — zoo



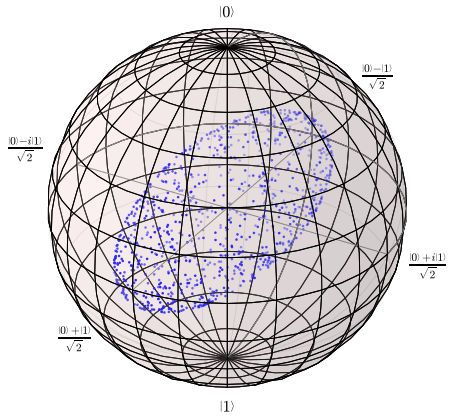
Rysunek: Action of bit flip channel: $p = 0.1$.

Qubit channels — zoo



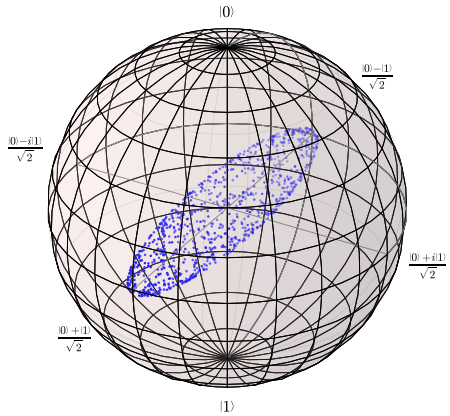
Rysunek: Action of bit flip channel: $p = 0.2$.

Qubit channels — zoo



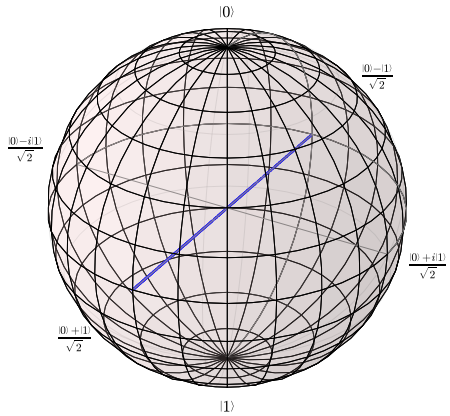
Rysunek: Action of bit flip channel: $p = 0.3$.

Qubit channels — zoo



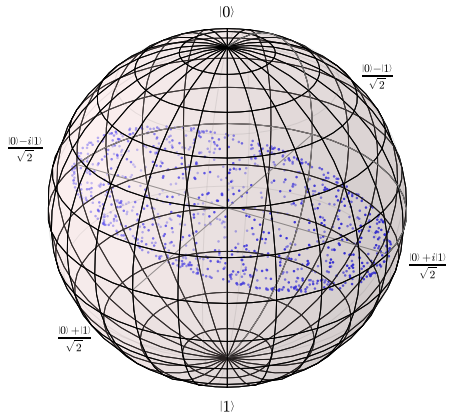
Rysunek: Action of bit flip channel: $p = 0.4$.

Qubit channels — zoo



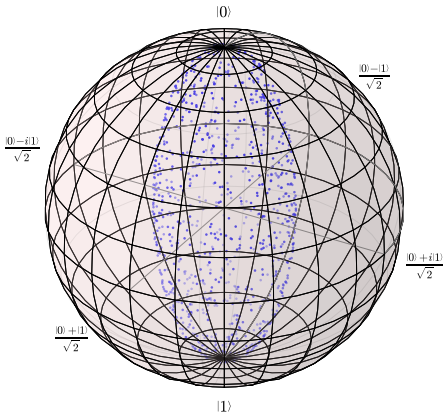
Rysunek: Action of bit flip channel: $p = 0.5$.

Qubit channels — zoo



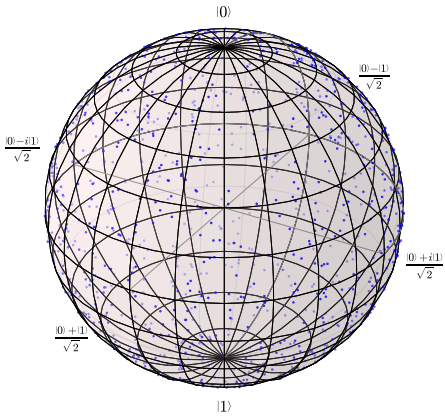
Rysunek: Action of phase flip channel: $p = 0.3$.

Qubit channels — zoo



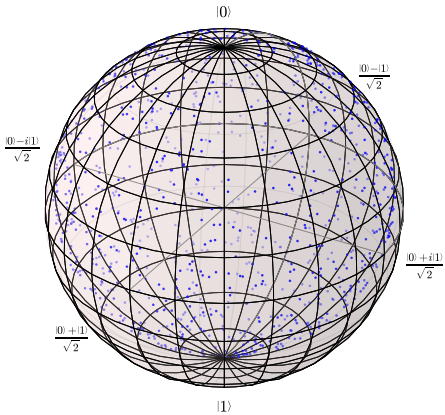
Rysunek: Action of bit-phase flip channel: $p = 0.3$.

Qubit channels — zoo



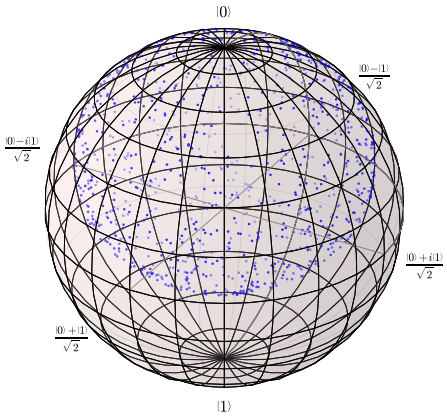
Rysunek: Uniformly chosen 1000 random pure states.

Qubit channels — zoo



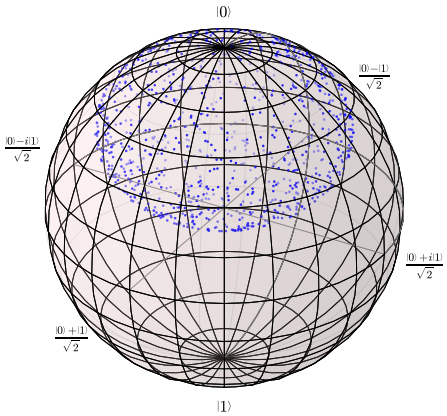
Rysunek: Action of amplitude dumping channel: $p = 0.1$.

Qubit channels — zoo



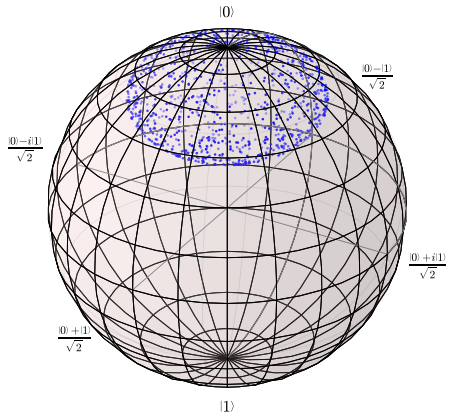
Rysunek: Action of amplitude dumping channel: $p = 0.3$.

Qubit channels — zoo



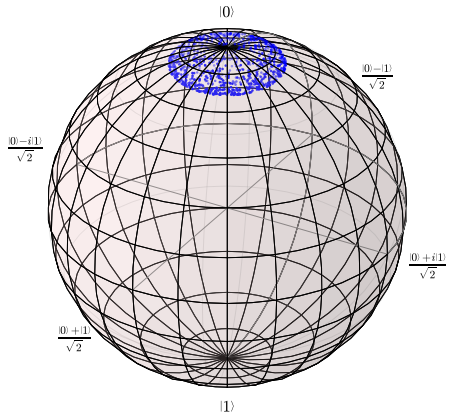
Rysunek: Action of amplitude dumping channel: $p = 0.5$.

Qubit channels — zoo



Rysunek: Action of amplitude dumping channel: $p = 0.7$.

Qubit channels — zoo

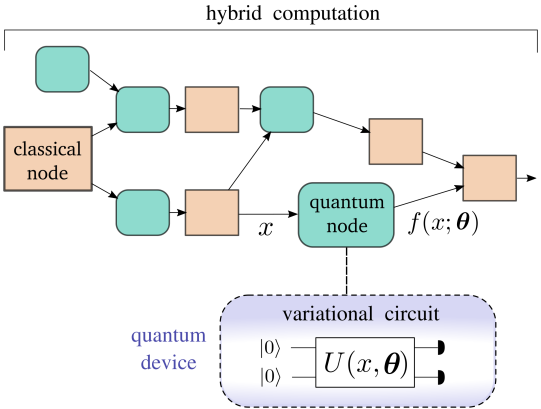


Rysunek: Action of amplitude dumping channel: $p = 0.9$.

Quantum neural networks

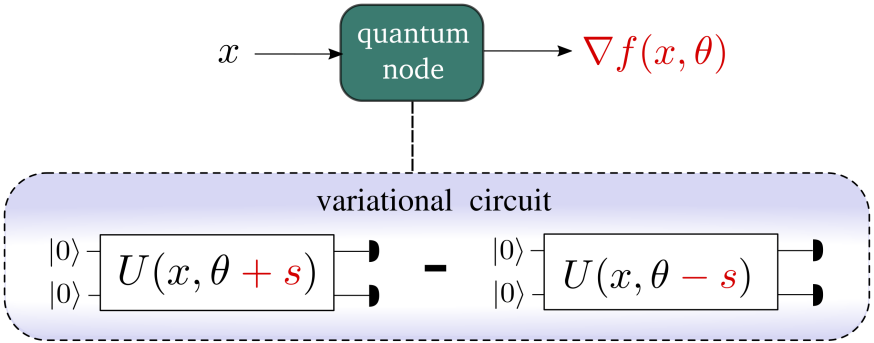
Hybrid quantum–classical computation

Concept of hybrid computation



Source: `https://github.com/XanaduAI/pennylane/` (Distributed under Apache Licence 2.0)

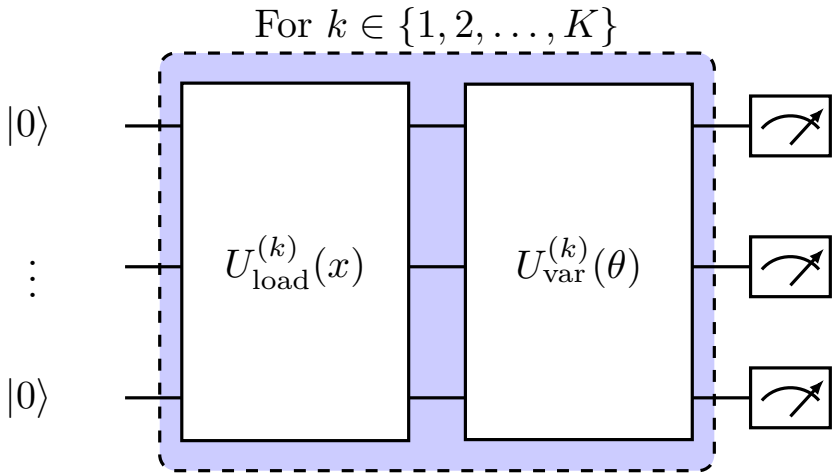
Gradient — parameter shift rule



Source: `https://github.com/XanaduAI/pennylane/` (Distributed under Apache Licence 2.0)

Variational quantum circuits

Variational quantum circuit



Quantum neural network

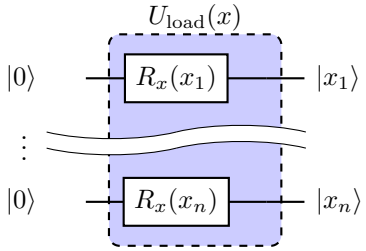
Classification with Quantum Neural Network

Data encoding

A data sample

$$x = (x_1, x_2, \ldots, x_N)$$

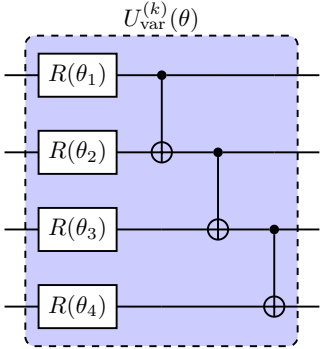
is encoded into a quantum state $|x\rangle$.



Classification with Quantum Neural Network

Variational gate

A variational gate $U_{\mathrm{var}}(\theta)$ represents transformation of data by our quantum neural network.



Mathematical tools of QNNs

Observable, expectation value

- Let O be an **observable**—a quantum measurement whose labels are real numbers.
- Mathematically it is represented as a Hermitian operator i.e.

$$O^\dagger = O.$$

- The **expectation value** of an observable O in a state $|\psi\rangle$

$$\langle O \rangle_{|\psi\rangle} = \langle \psi | O | \psi \rangle .$$

- For example if we observe spin of a system, expectation value tells us about the average spin of this system in a given state.

Gradients, parameter shift rule

- A function

$$f(\theta) = \langle O \rangle_{U(\theta)|\psi\rangle}$$

if $U(\theta) = e^{-i\theta G}$, where G has two distinctive eigenvalues $\{-r, r\}$ has gradient

$$\frac{\partial}{\partial\theta} f = r\left[f(\theta + s) - f(\theta - s)\right],$$

where $s = \pi/4r$.

- For example for $R_x(\theta) = e^{-\theta i \sigma_x/2}$, and $f(\theta) = \langle O \rangle_{R_x(\theta)|\psi\rangle}$ the gradient is

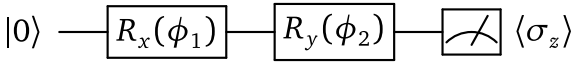
$$\frac{\partial}{\partial\theta} f(\theta) = \frac{1}{2}\left[f(\theta + \pi/2) - f(\theta - \pi/2)\right].$$

Optimizers

- Gradient descent: $\theta^{(t+1)} = \theta^{(t)} - \eta \nabla f(\theta^{(t)})$;
- Nesterov momentum: $\theta^{(t+1)} = \theta^{(t)} - a^{(t+1)}$, $a^{(t+1)} = m a^{(t)} + \eta \nabla f(\theta^{(t)} - m a^{(t)})$;

With η — the step size, m — the momentum.

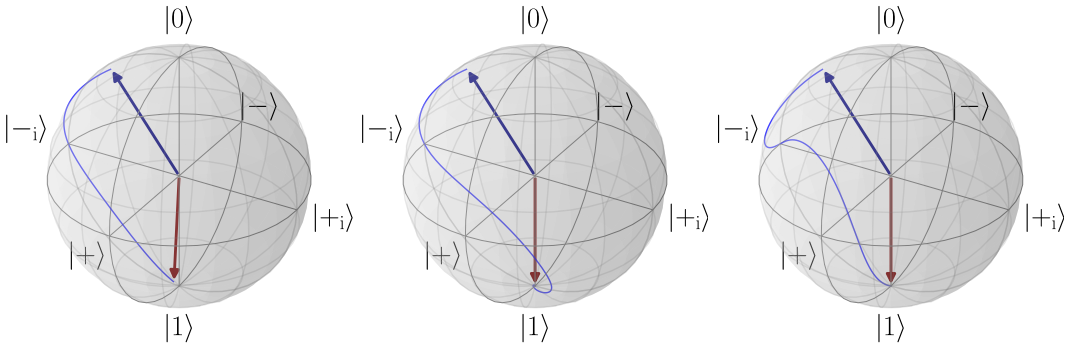
Variational quantum algorithm I




```
import pennylane as qml
from pennylane.optimize import GradientDescentOptimizer
from pennylane import numpy as np
# Create device
dev = qml.device('default.qubit', wires=1)
# Quantum node
@qml.qnode(dev)
def circuit1(var):
    qml.RX(var[0], wires=0)
    qml.RY(var[1], wires=0)
    return qml.expval(qml.PauliZ(0))
# Create optimizer
opt = GradientDescentOptimizer(0.25)
# Optimize circuit output
var = np.array([0.5, 0.2], requires_grad=True)
for it in range(30):
    var = opt.step(circuit1, var)
    print("Step {}: cost: {}".format(it, circuit1(var)))
```

Source: arXiv:1811.04968

Variational quantum algorithm II



Rysunek: Three optimization strategies: gradient descent, Nesterov momentum optimizer, Adam optimizer ($var_{\mathrm{init}} = (0.5, 0.2)$).

An example of quantum neural network

Dataset

We use an artificial dataset (moons) consisting of two classes.

Data preprocessing

- The original feature vectors (single feature for all samples) X_{orig} are normalized

$$X_{\text{std}} = \frac{X_{\text{orig}} - \min(X_{\text{orig}})}{\max(X_{\text{orig}}) - min(X_{\text{orig}})}$$

$$X_{\text{scaled}} = X_{\text{std}}(\max - \min) + \min,$$

  with $\min = 0$, $\max = \pi$.

- In order to encode as in angle of a qubit rotation.

An example of quantum neural network

Data encoding gate

- Encoding gate $U_{\mathrm{e}}(x)$ transforms data sample $x = (x_1, x_2, \ldots, x_N) \in \mathbb{R}^N$ into data state

$$|x\rangle = U_{\mathrm{e}}(x)\,|0\rangle^{\otimes N} = R_x(x_1) \otimes R_x(x_2) \otimes \ldots \otimes R_x(x_N)\,|0\rangle^{\otimes N},$$

- where

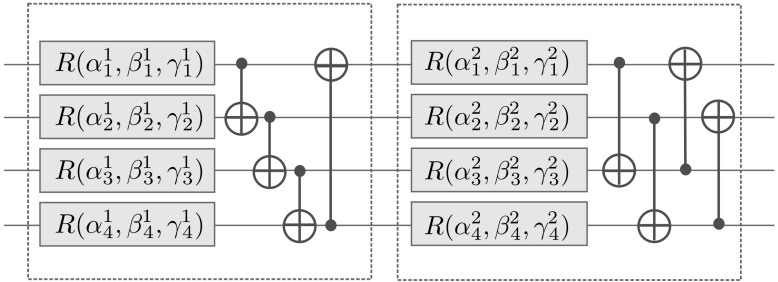
$$R_x(\phi) = e^{-i\phi\sigma_x/2} = \begin{bmatrix} \cos(\phi/2) & -i\sin(\phi/2) \\ -i\sin(\phi/2) & \cos(\phi/2) \end{bmatrix}$$

- and $x_i \in [0, \pi]$.

An example of quantum neural network

Variational gate

- Variational gate $U_{\mathrm{v}}(w)$ entangles the information and is parametrized. Angles $w \in \mathbb{R}^{N\times 3\times L}$ are the parameters we want to learn from the data and L denotes the number of quantum layers.



An example of quantum neural network

Decision function

- The decision function $f : \mathbb{R}^N \to \mathbb{R}$.
- $f(x;\theta) = \langle O \rangle_{U_{\mathrm{v}}(\theta)U_{\mathrm{e}}(x)|0\rangle}$, with

$$\langle O \rangle_{U_{\mathrm{v}}(\theta)U_{\mathrm{e}}(x)|0\rangle} = \langle 0| U_{\mathrm{e}}(x)^{\dagger} U_{\mathrm{v}}(\theta)^{\dagger} O U_{\mathrm{v}}(w) U_{\mathrm{e}}(x) |0\rangle ,$$

- Observable $O = \sigma_z \otimes \mathbb{1}^{\otimes(N-1)}$.
- The classification function $\mathrm{cls} : \mathbb{R}^N \to \{-1, 1\}$: $\mathrm{cls}(x;\theta) = \mathrm{sgn}(f(x;\theta))$.

An example of quantum neural network

Training

- The available data are divided into: train and test.
- Initial θ-s: $w \sim \mathcal{U}(0, 2\pi)$.
- Cost function

$$\text{cost}\left((y_i^{\text{train}})_{i\in\xi_t}, (f(X_{i,:}^{\text{train}};\theta))_{i\in\xi_t}\right) = \sum_{i\in\xi_t}(y_i^{\text{train}} - f(X_{i,:}^{\text{train}};\theta))^2/|\xi_t|.$$

- Batches ξ_t.

Example in python

Source code

Binary classification in pennylane

Imports

```
from importlib.metadata import requires
from itertools import chain

from sklearn import datasets
from sklearn.utils import shuffle
from sklearn.preprocessing import minmax_scale
from sklearn.model_selection import train_test_split
import sklearn.metrics as metrics


import pennylane as qml
from pennylane import numpy as np
from pennylane.templates.embeddings import AngleEmbedding
from pennylane.templates.layers import StronglyEntanglingLayers
```

Binary classification in pennylane

Data import, preprocessing and splitting

```
np.random.seed(42)

# create the dataset
X, y = datasets.make_moons(100, noise=0.1, random_state=42)

# shuffle the data
X, y = shuffle(X, y, random_state=42)


# normalize data
X = minmax_scale(X, feature_range=(0, np.pi))

# split data into train and test
```

Binary classification in pennylane

Building the quantum classifier

```
# number of qubits is equal to the number of features
n_qubits = X.shape[1]

# quantum device handle
dev = qml.device("default.qubit", wires=n_qubits)

# quantum circuit
@qml.qnode(dev)
def circuit(weights, x=None):
    AngleEmbedding(x, wires = range(n_qubits))
    StronglyEntanglingLayers(weights, wires = range(n_qubits))
    return qml.expval(qml.PauliZ(0))

def cost(theta, X, expectations):
    e_predicted = np.array([circuit(theta, x=x) for x in X])

    loss = np.mean((e_predicted - expectations)**2)
    return loss

# helper function: returns 1 if x>0 else 0
def sgn(x):
    return (x>0)*1
```

Binary classification in pennylane

Training preparation

```
# number of quantum layers
n_layers = 3

# convert classes to expectations: 0 to -1, 1 to +1
e_train = np.empty_like(y_train)
e_train[y_train == 0] = -1
e_train[y_train == 1] = +1

# select learning batch size
batch_size = 5

# calculate number of batches
batches = len(X_train) // batch_size

# select number of epochs
n_epochs = 5

# draw random quantum node weights
shape = qml.StronglyEntanglingLayers.shape(n_layers=n_layers, n_wires=n_qubits)
theta = np.random.random(size=shape, requires_grad=True)
```

Binary classification in pennylane

Training

```
# train the variational classifier

# start of main learning loop
# build the optimizer object
pennylane_opt = NesterovMomentumOptimizer()

log = []
# split training data into batches
X_batches = np.array_split(np.arange(len(X_train)), batches)
for it, batch_index in enumerate(chain(*(n_epochs * [X_batches]))):
    # Update the weights by one optimizer step
    batch_cost = \
        lambda t: cost(t, X_train[batch_index], e_train[batch_index])
    theta = pennylane_opt.step(batch_cost, theta)
    log.append({"theta":theta})
```

Binary classification in pennylane

Inference

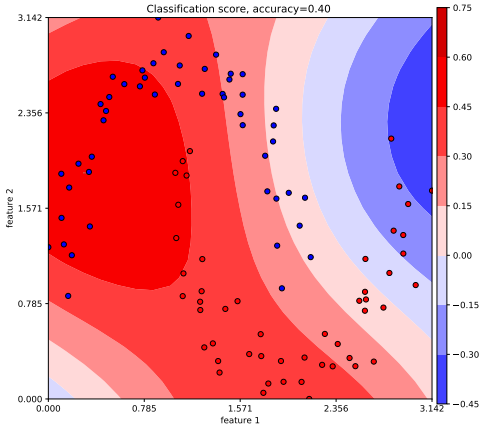
```
# convert scores to classes
scores = np.array([circuit(theta, x=x) for x in X_test])
y_pred = sgn(scores)

print(metrics.accuracy_score(y_test, y_pred))
```

Visualization of learning process

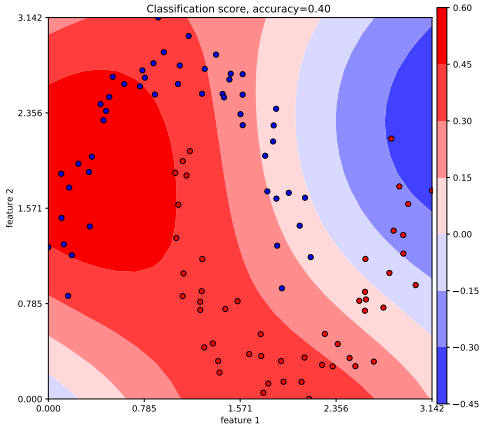
Learning process

Learning step 1



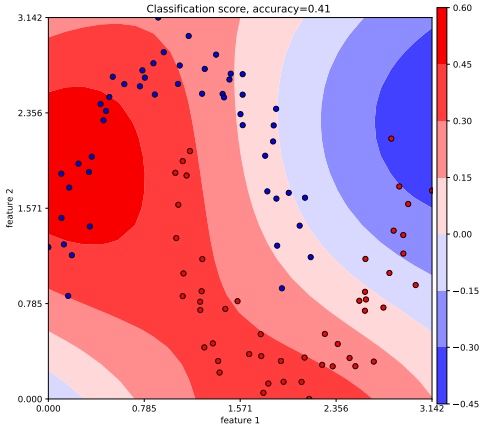
Learning process

Learning step 2



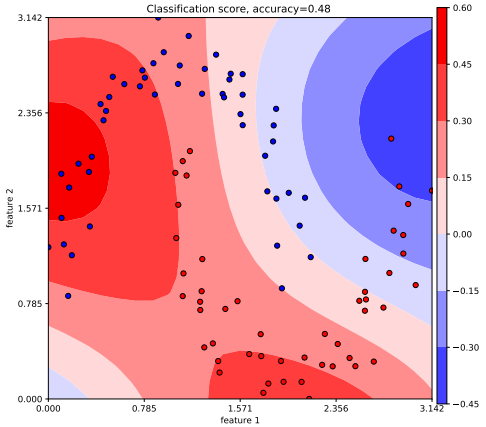
Learning process

Learning step 3



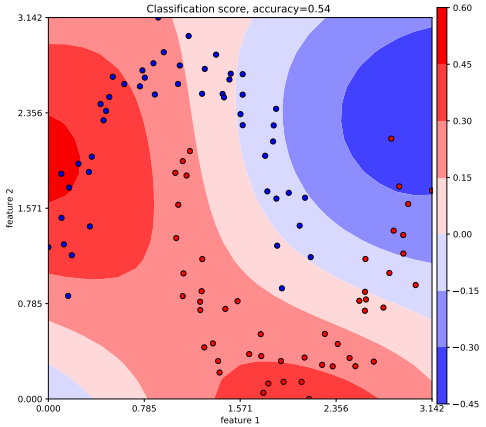
Learning process

Learning step 4



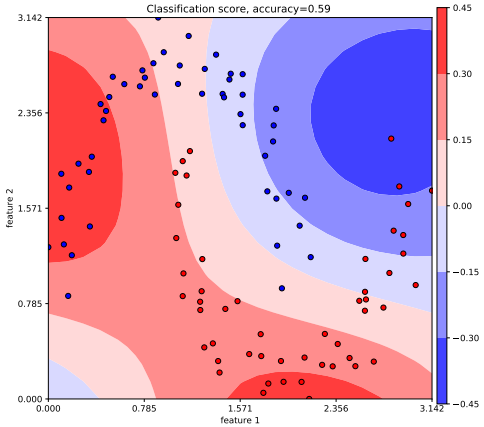
Learning process

Learning step 5



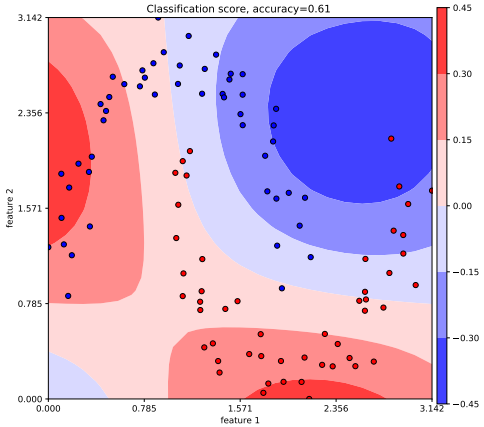
Learning process

Learning step 6



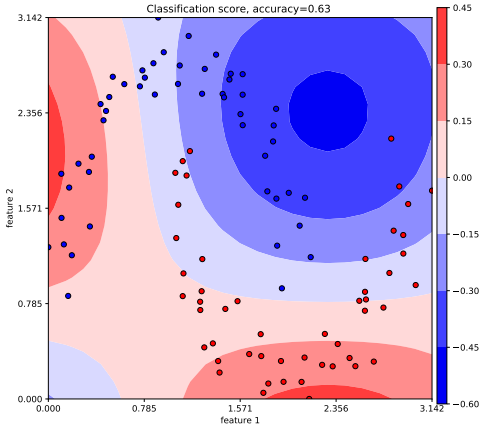
Learning process

Learning step 7



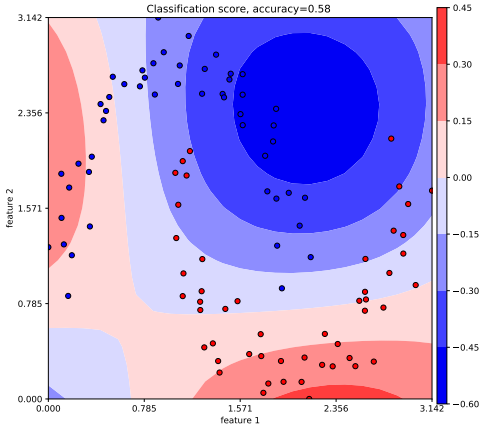
Learning process

Learning step 8



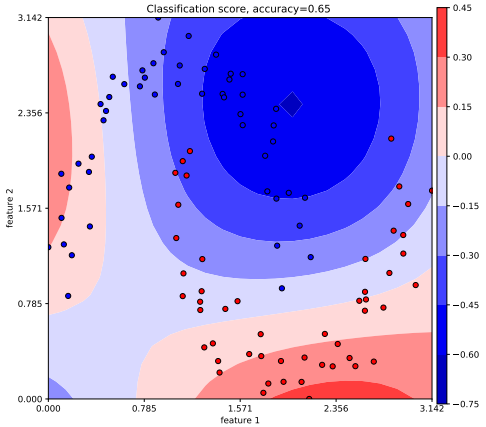
Learning process

Learning step 9



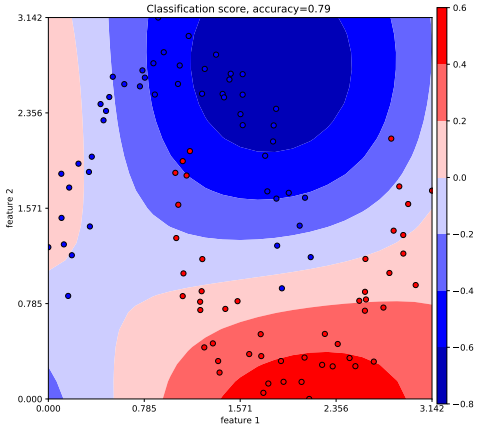
Learning process

Learning step 10



Learning process

Learning step 15



Learning process

Learning step 20