Munich Network Management Team

Ludwig-Maximilians-Universität München

Prof. Dr. D. Kranzlmüller, Korbinian Staudacher, Xiao-Ting Michelle To

Introduction to Quantum Computing – Exercise Sheet 2

You can find the Jupyter notebook in the following GitHub repository:
`https://github.com/mnm-team/qc-focused-tutorial-23`.
To run the Jupyter notebook, you have several options:

1. My Binder (gesis): `https://notebooks.gesis.org/binder/` (No registration necessary)
2. Google Collab: `https://colab.research.google.com/` (You need a Google account for this!)
3. IBM Quantum: `https://quantum-computing.ibm.com/` (You need an IBM account for this!)
4. Locally with Python and Jupyter notebooks (`https://pennylane.ai/install`)

Exercise 2.1: *Teleportation in PennyLane*

This exercise is designed to help you getting familiar with the PennyLane Quantum programming environment.[4] We provide you a jupyter notebook `Exercise-2_1-teleportation-template.ipynb` with a basic framework for simulating quantum teleportation. Your job is to teleport a QuBit in state $R_y(\frac{\pi}{3})\,|0\rangle$ using the circuit from the lecture. Therefore, you need to add the correct gates to the places marked with `# TODO`. You can verify your solution by plotting a histogram of the measurement results from the teleported QuBit. For assistance, we recommend the following pages from the PennyLane documentation:

- `https://docs.pennylane.ai/en/stable/introduction/operations.html`
- `https://docs.pennylane.ai/en/stable/introduction/measurements.html` (especially mid-circuit measurements)

Exercise 2.2: *kNN Classification*

One of the simplest classification algorithms is the k-nearest neighbor (kNN) algorithm. It classifies a given object o, represented as a point in a coordinate system, depending on the k nearest neighbors, i.e. objects $o_0, \dots, o_{k-1}$, and their respective classes.
In this exercise we will try to find out whether a person likes the Beatles. For a person, two features are given:

- Their age
- The number of minutes that they watch television per day

To find out which objects $o_0, \dots, o_{k-1}$ are the nearest to object o, we need a distance measure. The most-used one is the Euclidean norm

$$d(a,b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

where $a = (x_a, y_a)$ and $b = (x_b, y_b)$ are feature vectors.
If a and b are unit vectors, then the Euclidean norm can be written as

$$d(a,b) = \sqrt{2(1 - |cos(\theta_{ab})|)}$$

where θ_{ab} is the angle between a and b.
Given the fact that qubit states are always unit vectors, we can translate this formula to the Dirac Notation using $|a\rangle$ and $|b\rangle$:

$$d(a,b) = \sqrt{2(1 - |\langle a|b\rangle|)}$$

[4] `https://pennylane.ai`

This means the distance between two vectors can be determined if $| \langle a|b \rangle |$ can be calculated. For this, you can use the knowledge of the SWAP test with which $| \langle a|b \rangle |^2$ can be calculated.
Given the code in `Exercise-2_2-beatles-template.ipynb`, calculate the distance between two objects A and B with the SWAP test. You can use the code at the end of the Jupyter notebook to print the quantum circuit (e.g. if you want to check if the circuit looks like it is supposed to).
Only add code to the places marked with `# TODO`!

Exercise 2.3: *Binary Classification*

In this task you will learn how to program and train a quantum classifier. The goal is to train a variational circuit with 4 QuBits on a subset of the MNIST database containing only the digits 0 and 1. Again we provide you a notebook `Exercise-2_3-MNIST-classifier-template.ipynb` where we already programmed the main framework for the task. You job is to add the following:

1. The data encoding of the feature vectors using *angle encoding.*
2. The variational model with parametrized gates using three rotations per QuBit plus *circular entanglement* like in the slides.
3. A cost and accuracy function for training the parametrized gates using the mean squared error (MSE) function.

Finally you test your trained model against a test dataset. When you programmed everything correctly, the final accuracy should be above 0.75.