INTRODUCTION TO PRACTICAL QUANTUM COMPUTING

QUANTUM COMPUTING INITIATIVE

Abstract. Many introductions to quantum computing get drowned in mathematical formalisms or discussions of theoretical phenomena, that is why we aim to share this as a reference document explaining the foundations of quantum computing from a practical standpoint, focusing on abstracting the excess theory to focus on the necessary details for the applications. We hope to demystify quantum computing to make it more intuitive and therefore more accessible. While the underlying phenomena and formalisms are necessary for success in a more serious study of the subject, we restrict the content of this document to an intuitive overview sufficient for the layman's success.

The content covered in this document uses the **qiskit** library all quantum circuits. Its documentation can be found here: `https://quantum.cloud.ibm.com/docs/en/guides`

1. Qubits

1.1. What Are Qubits.

Everyday computers represent information on a hardware level using bits with values of either 0s or 1s, however **qubits** are quantum mechanical systems used to represent information, and introduce special properties and phenomena due to the physics governing their systems. Fortunately, the technology is mature and abstracted enough to where performing computations on qubits does not require a thorough understanding of the physics behind it anymore.

1.2. Superposition.

One of the main reasons qubits are unique is due to their ability to be in a **superposition** of the multiple states. This means that instead of being equal to either 0 or 1, the qubit's state can be represented as a probabilistic combination of being either 0 or 1.

For example a coin can be heads or tails, but while it's flipping in the air, we can describe its state as being a 50% chance of being a 0, and a 50% of it being a 1. Similarly, a qubit in superposition can have its state as being a 50% chance of being a 0, and a 50% of it being a 1.

Later we will introduce other transformations on the state of the qubit that allow it to be in a superposition with different probabilities, for example 25% it's a 0, and 75% it's a 1. These transformations on the qubits are applied through **gates**.

1.3. Entanglement.

Another property of qubits is **entanglement**. This is simply when the state of a qubit depends on the state of the other.

2. Gates

2.1. What Are Quantum Gates.

For those familiar with digital logic, **quantum gates** are analogous to logical gates that act on bits. They are devices that change the state of a qubit in a known way. Many quantum gates exist today but the nuance in quantum gates comes from combining them to create more profound logic.

2.2. Common Quantum Gates.

One of the most important gates is the **Hadamard** gate, which applies superposition to a qubit, and is implemented as `qc.H(<qubit_index>)` in Qiskit.

Figure 1. Circuit View of a Hadamard Gate

Some other common gates include the **X, Y, Z rotation** gates, respectively accessed as `qc.X(<qubit_index>)`, `qc.Y(<qubit_index>)`, `qc.Z(<qubit_index>)`.

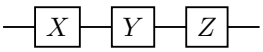


Figure 2. Circuit View of the Rotation Gates

There also exists parameterized versions of these gates which rotates the state of the qubit based on a numerical parameter. This is useful for encoding numerical data into a quantum circuit which will be mentioned later. This form of encoding data through a parameterized rotation gate is called rotation encoding.

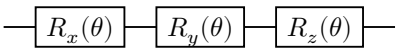


Figure 3. Circuit View of the Parameterized Rotation Gates

2.3. Multi-Qubit Gates.

Another important concept is **multi-qubit gates**, which as the name implies, features gates that operate on multiple qubits. A common example is the **CNOT** gate, which flips one qubit based on the state another qubit. Similar versions exist for the rotation gates and other one qubit gates, where the state of one qubit controls whether or not another qubit is transformed.

These are important for introducing relationships between qubits to potentially mimic relations in the original data. This is beneficial for use cases like quantum machine learning, where the quantum circuit wants to learn the relations in the data.

3. Quantum Circuits

After developing an intuition for qubits and gates, quantum circuits become an obvious next step. The structure of the typical quantum circuit involves preparing the qubits, whether it's a specific quantum state that is required, or encoding classical data into the quantum circuit.

3.1. **Encoding Classical Data.**

For most applications, the data is classical, so encoding techniques will be necessary, notably rotation encoding or amplitude encoding. Rotation encoding is the common go-to technique due to its simplicity, scaling, and general utility. Rotation encoding involves rotating a qubit based on the numerical value in the data.

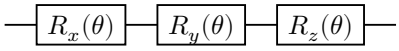


FIGURE 4. Circuit View of Rotation Encoding using Parameterized XYZ Rotations

Since the rotation angles are between 0 and 2π, it helps to scale the data to fit this range before encoding it.

4. HYBIRD ALGORITHMS

While quantum circuits introduce unique and useful operations, the technology is not mature enough to handle the control infrastructure surrounding the quantum circuit. For this reason we embed the quantum circuits into classical algorithms. This could be as simple as introducing a quantum circuit to act as a quantum layer in an algorithm, or having a full quantum algorithm where the encompassing classical algorithms only manages the inputs and outputs. We refer to these algorithms as hybrid algorithms, as they include both classical and quantum parts. This is made feasible by the different encoding methods to transform classical data into a quantum circuit, and that measurements on qubits are classical in nature.

5. INTRODUCTION TO QUANTUM MACHINE LEARNING (QML)

5.1. **Primer on Training Machine Learning Models.**

Supervised machine learning models are parameterized statistical models that:

1. take an input
2. produce an output
3. check the expected output
4. update the parameters in such a way to reduce the difference between the produced and expected outputs

For this reason, we either preprocess the data or improve the model with the aim of having our model better understand the data, allowing it to produce better outputs.

5.2. **Types of Tasks & Models.**

Classifiers are models that label a given input as one of the groups it was trained on, aka perform classification. Regressors are models that predict a numerical value based on the given input, aka perform regression.

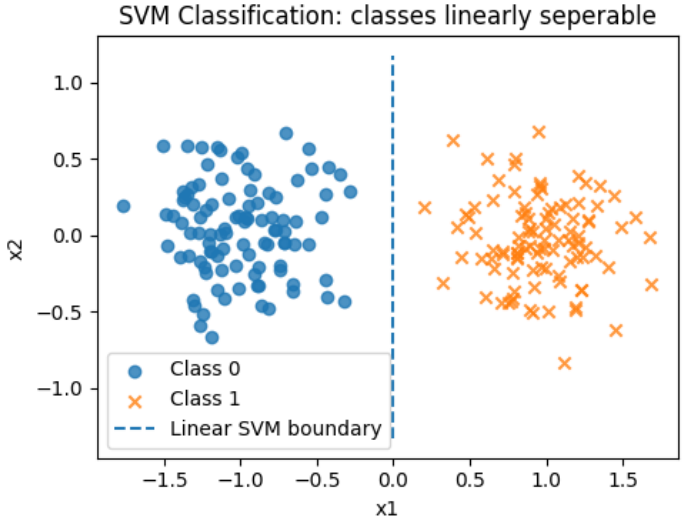
5.3. **Hybrid Algorithms in QML.**

A popular example of how hybrid algorithms are used can be seen in quantum machine learning. Regardless of the model and data involved, classical control systems are used to pass input into the algorithms and handle the output.

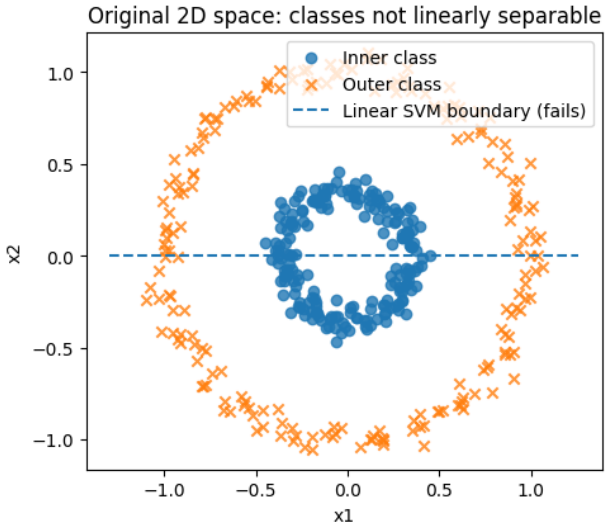
There are two main paradigms for applying quantum circuits in machine learning: quantum feature maps or quantum neural networks.

5.4. **Quantum Feature Maps.**

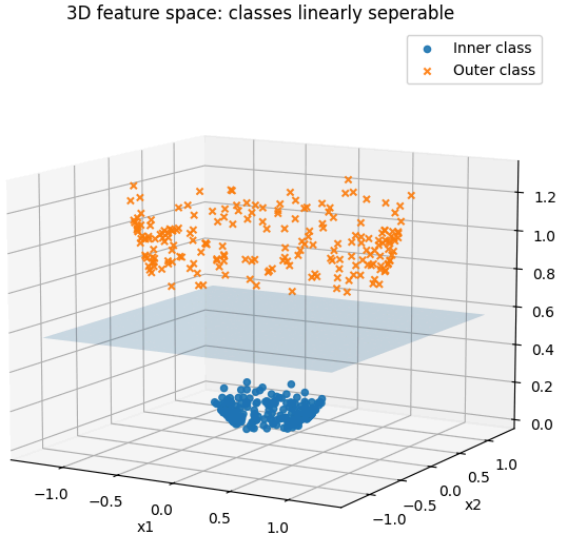
Before discussing quantum feature maps, it is necessary to form the intuition on why feature maps are used. We can take Support Vector Machines (SVM), which is a classification model that learns a linear boundary between the data, and classifies new entries based on what side of boundary they are on. Take for example the data classified in the graph below.



This works well for linearly separable data, however it struggles more as the variation in the data increases. Therefore for another dataset it may be harder to find a linear boundary that separates the data. Take the following graph as an example.



When we plot the data, it is impossible to find a linear boundary that separates the data. This makes the SVM model ineffective for this data. However, if we were to apply an algorithm to that data, such that each point is mapped on a 3 dimensional plot instead of remaining 2 dimensional, we get the following graph.



Mapping the 2D data into 3D unlocks a new linear boundary that was previously invisible to us. This is not an optical trick, but instead a consequence of the data being better represented and more expressive in a 3D space compared to a 2D space.

What makes quantum circuit useful here, is a key property of quantum computing. A quantum circuit space with n qubits operates in a 2^n dimensional space.

This means a quantum circuit with 2 qubits can represent information in $2^2 = 4$ dimensional spaces, and a 3 qubit circuit utilizes $2^3 = 8$ dimensional spaces, and so on. While that many dimensions isn't easy to conceive intuitively, what matters is to best represent the data, so that the ML model can effectively explore the data and determine correlations with the expected output. For this reason, a more expressive quantum circuit for any data imporves its modelling capacity.