Learning with Quantum Computers

Agnipratim Nag
Sophomore, Engineering Physics
IIT Bombay

***Abstract*—The quantum world has opened up countless possibilities to us - in physics, computer science, information theory, mathematics, you name it! We are going to take our first steps in one tiny subfield of all of this - quantum machine learning. To do this, we first build some initial background in quantum information, followed by some basic demos in Qiskit and Pennylane to get us comfortable with the software. Finally, we end with a small quantum project of our own - a variational quantum classifier!**

***Index Terms*—quantum, computing, machine, learning, variational circuit, neural network, deutsch-josza, neural netwoks**

I. Introduction

Quantum machine learning is the integration of quantum algorithms within machine learning programs. The most common use of the term refers to machine learning algorithms for the analysis of classical data executed on a quantum computer, i.e. quantum-enhanced machine learning. While machine learning algorithms are used to compute immense quantities of data, quantum machine learning utilizes qubits and quantum operations or specialized quantum systems to improve computational speed and data storage done by algorithms in a program.

II. What is quantum computing?

Quantum computing is a type of computation whose operations can harness the phenomena of quantum mechanics, such as superposition, interference, and entanglement. Devices that perform quantum computations are known as quantum computers. Quantum computations allow for speeding up calculations which are otherwise constrained by the laws of classical physics, and this is what we aim to harness! One example of this is the quantum algorithm which solves the Deutsch-Josza Problem. Eventually, we look at how we can make use of quantum neural networks to perform the well known task of clustering but far more efficiently than a classical neural network would.

III. The Deutsch Josza Problem

The Deustch-Jozsa problem is one of the most basic ways to demonstrate the power of quantum computation. Consider a Boolean function f : $\{0,1\}^n$ to $\{0,1\}$ and suppose we have a black-box to compute f. The Deutsch-Jozsa problem is to determine if f is constant (i.e. f(x) = const for all x in $\{0,1\}^n$) or if f is balanced (i.e. f(x) = 0 for exactly half the possible input strings x in $\{0,1\}^n$) using as few calls to the black-box computing f as is possible, assuming f is guaranteed to be constant or balanced. Classically it appears that this requires at least $2^{n-1}+1$ black-box calls in the worst case, but the well known quantum solution solves the problem with probability one in exactly one black-box call. In this report, we will not delve into the details of the Deutsch Josza Algorithm. The reader can learn more about it here.

IV. Variational Quantum Classifiers!

Variational classifiers are a key application of modern day machine learning, performing a task which is crucial to the function of many day to day applications we use all the time. It is simply a computer model that trains itself using labelled data so that it is able to classify new data when provided with it. Usually, this happens on classical computers, but we outline how to build a quantum circuit that uses the same, using the Pennylane library on Python.

A. Fitting a Parity function

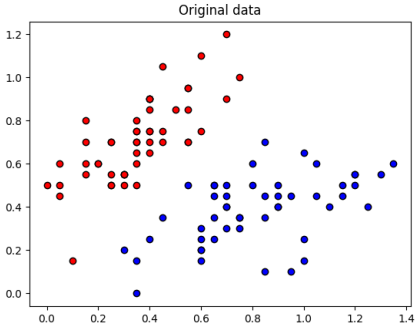
First, we show that a variational quantum classifier can reproduce the parity function where the input is a bitstring of length n, and the output is 1 if it contains an odd number of ones, and 0 otherwise. For this, we setup a quantum device with four wires, each with an arbitrary rotation and an entanglement to the neighbouring qubit using CNOT gates. This is one layer of of the variational circuit we eventually aim to build. Following from our knowledge of classical Machine Learning, we use weights and the cost function to set up our model. To observe how well our model is doing, we set up the cost function to be a usual squared error loss function along with an accuracy function. After this setup is complete, we load our optimizer, choose a batch size and train it. The result of this computation can be seen below.

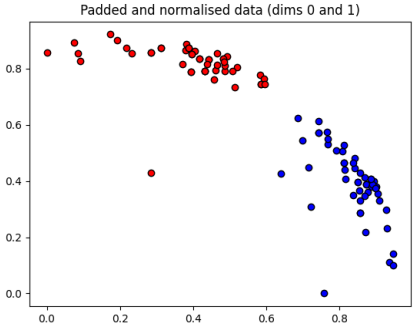
Iter: 1 — Cost: 3.4355534 — Accuracy: 0.5000000
Iter: 2 — Cost: 1.9717733 — Accuracy: 0.5000000
Iter: 3 — Cost: 1.8182812 — Accuracy: 0.5000000
Iter: 4 — Cost: 1.5042404 — Accuracy: 0.5000000
Iter: 5 — Cost: 1.1477739 — Accuracy: 0.5000000
Iter: 6 — Cost: 1.2734990 — Accuracy: 0.6250000
Iter: 7 — Cost: 0.8290628 — Accuracy: 0.5000000
Iter: 8 — Cost: 0.3226183 — Accuracy: 1.0000000
Iter: 9 — Cost: 0.1436206 — Accuracy: 1.0000000

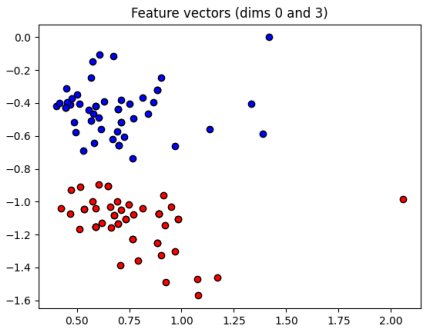
Iter: 10 — Cost: 0.2982810 — Accuracy: 1.0000000
Iter: 11 — Cost: 0.3064355 — Accuracy: 1.0000000
Iter: 12 — Cost: 0.1682335 — Accuracy: 1.0000000
Iter: 13 — Cost: 0.0892512 — Accuracy: 1.0000000
Iter: 14 — Cost: 0.0381562 — Accuracy: 1.0000000
Iter: 15 — Cost: 0.0170359 — Accuracy: 1.0000000
Iter: 16 — Cost: 0.0109353 — Accuracy: 1.0000000
Iter: 17 — Cost: 0.0108388 — Accuracy: 1.0000000
Iter: 18 — Cost: 0.0139196 — Accuracy: 1.0000000
Iter: 19 — Cost: 0.0123980 — Accuracy: 1.0000000
Iter: 20 — Cost: 0.0085416 — Accuracy: 1.0000000
Iter: 21 — Cost: 0.0053549 — Accuracy: 1.0000000
Iter: 22 — Cost: 0.0065759 — Accuracy: 1.0000000
Iter: 23 — Cost: 0.0024883 — Accuracy: 1.0000000
Iter: 24 — Cost: 0.0029102 — Accuracy: 1.0000000
Iter: 25 — Cost: 0.0023471 — Accuracy: 1.0000000

B. Classification of the Iris dataset

Moving onto the next part, we load the iris dataset on which we will act our classifier. First, we perform some preprocessing by loading the amplitudes of the data into quantum states. State preparation is a slightly non trivial task where every input bitstring has to be converted into a set of angles which can be fed into the subroutine which prepares the corresponding quantum state. For this, we define a function that determines these values and prepares the states so that we can work our quantum magic on them. The variational circuit model and cost function are reused from the previous exercise for the upcoming parts.
At present, we can have a look at what our original dataset, its normalised version and what the feature vectors look like.







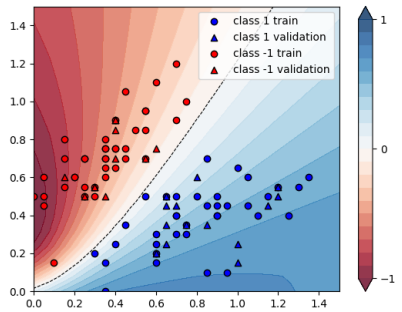
Now, we split our data into the training and validation set. One from which the model learns and the other from which the model measures how well it has learnt. As usual, we carry out optimization by setting up weights, bias and a batch size, and then we run it on our data! Similar to last time, we measure the accuracy of the predictions on both the training and validation set. The result of the computation can be seen below.

Iter: 1 | Cost: 1.4490948 | Acc1: 0.4933333 | Acc2: 0.5600000
Iter: 2 | Cost: 1.3312057 | Acc1: 0.4933333 | Acc2: 0.5600000
Iter: 3 | Cost: 1.1589332 | Acc1: 0.4533333 | Acc2: 0.5600000
Iter: 4 | Cost: 0.9806934 | Acc1: 0.4800000 | Acc2: 0.5600000
Iter: 5 | Cost: 0.8865623 | Acc1: 0.6133333 | Acc2: 0.7600000
Iter: 6 | Cost: 0.8580769 | Acc1: 0.6933333 | Acc2: 0.7600000
Iter: 7 | Cost: 0.8473132 | Acc1: 0.7200000 | Acc2: 0.6800000
Iter: 8 | Cost: 0.8177533 | Acc1: 0.7333333 | Acc2: 0.6800000
Iter: 9 | Cost: 0.8001100 | Acc1: 0.7466667 | Acc2: 0.6800000
Iter:10 | Cost: 0.7681053 | Acc1: 0.8000000 | Acc2: 0.7600000

40 more iterations in between...

Iter: 50 | Cost: 0.4969733 | Acc1: 0.9466667 | Acc2: 1.0000000
Iter: 51 | Cost: 0.4782257 | Acc1: 0.9466667 | Acc2: 1.0000000
Iter: 52 | Cost: 0.4536953 | Acc1: 0.9600000 | Acc2: 1.0000000
Iter: 53 | Cost: 0.4350300 | Acc1: 1.0000000 | Acc2: 1.0000000
Iter: 54 | Cost: 0.4272909 | Acc1: 0.9733333 | Acc2: 0.9600000
Iter: 55 | Cost: 0.4288929 | Acc1: 0.9466667 | Acc2: 0.9200000
Iter: 56 | Cost: 0.4261037 | Acc1: 0.9333333 | Acc2: 0.9200000
Iter: 57 | Cost: 0.4082663 | Acc1: 0.9466667 | Acc2: 0.9200000
Iter: 58 | Cost: 0.3698736 | Acc1: 0.9600000 | Acc2: 0.9200000
Iter: 59 | Cost: 0.3420686 | Acc1: 1.0000000 | Acc2: 1.0000000
Iter: 60 | Cost: 0.3253480 | Acc1: 1.0000000 | Acc2: 1.0000000

Finally, we plot the continuous output of the variational classifier for the first two dimensions of the Iris data set.



ACKNOWLEDGMENT

A very big thank you to the Analytics Club, IIT Bombay for giving me the opportunity to do this project. Additionally, I would like to thank my mentors, Siddhant Midha and Aditya Sriram for their invaluable guidance throughout the duration of this project.

REFERENCES

[1] Quantum Information and Computation, Michael Nielsen and Isaac Chaung
[2] Qiskit Tutorials, IBM Quantum
[3] Pennylane Quantum Machine Learning Demos - Building a Quantum Variational Classifier, Maria Schuld