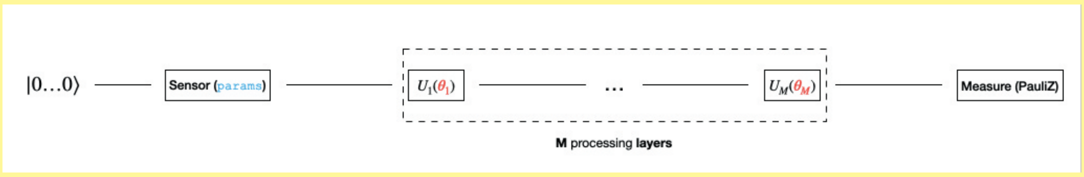
Telling quantum DoQs and quantum Qats apart

Team Quant'ronauts

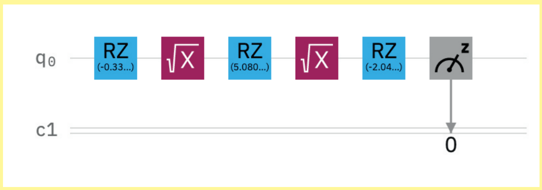
Idea: we classify regions of the Hilbert space, of quantum states of n qubits. There are 2 categories, "Qat" and "DoQ". As an example, for n=1, one hemisphere of the Bloch sphere could be labelled "Qat", the other hemisphere "DoQ". The state vectors to classify are generated as the output of a sensor, which is then fed into a classifier circuit of M layers. Note that we are NOT classifying the classical params vector of the sensor, as we could use any other sensor with different parameterization as long as it's capable of producing Qat and DoQ states. Also, we take the sensor as is, we don't try to "optimize" it.

Catch: during operation, the sensor can only produce its output once. Thus, when we calculate the accuracy on the test set, we are not allowed to make use of expected values resulting from many shots. There is only 1 shot (in the training phase, we can optimize using expected values, as training is done in our laboratory where we can recreate the sensor outputs of the training set at will). We'd like to experiment how much the accuracy drops due to this 1-shot limitation, whether it's different using simulator vs real quantum hardware, and what kind of cost function would reduce this impact.

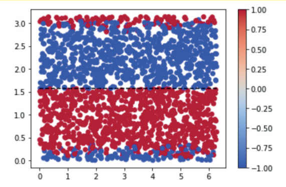
Extra: if multiple shots are allowed, how much would a data re-uploading scheme improve the accuracy? E.g. imagine there are M identical sensors located very close to each other. When a certain physical event happens, it sets all the parameters of the M sensors at once, identically for each sensor. Then, the parameters don't change until the next event. Furthermore, there may be exponentially many parameters of the sensor, inaccessible to us. So again, we are classifying quantum states.



Full circuit training



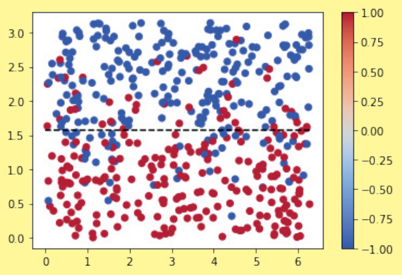
Graph training data set



```
Error rate : 12.156 %

Resources used for 1999 datas :
Circuit run : 44538, Optimize run : 42539, Deph 2, Shot(s) : 1, Qubit(s) : 1
```

Graph testing



3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
5
6
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00