Harunobu Ishii

hishii@bu.edu

Abstract

This research intends to introduce a possible application of Quantum Computing to a real-world dataset, binary classification for wine quality. The Quantum model used in this research is based on Bayesian theorem, and its circuit includes quantum gates used widely, such as X-Gate, CNOT Gate, RY Gate and CRY Gate, as well as those that are unique in this research, CCRY Gate and CCCRY Gate. Just like classical machine learning models, addition of attributes enhanced model accuracy on a test dataset, and a quantum machine learning model built for this research achieved slightly better performance than a classical machine learning model, random forest algorithm.

Applied quantum computing

Quantum Machine Learning Model for Wine Quality Classification

1. **Introduction**
   1. Motivation

Quantum computing has been gaining attentions from diverse fields. There has been significant amount of investment for Quantum Computing and significant technological advancement correspondingly. Quantum Computing, though it is still in early stage of development, is proven to overcome complexity problems that classical computing has faced for long time, such as Grover’s searching algorithm and Shor’s factorizing. Those algorithms are substantiated evidence that Quantum Computing will supersede in some areas, such as cryptography and optimization. As such, the author would like to explore areas where quantum computing can enhance computational performance over classical computation and establish applied quantum computations to innovate the real-world.

* 1. Project Overview

Quantum computing uses qubit, which takes a superposition returning 0 or 1 at respective probabilities. A measurement of a qubit can affect other qubit state if two qubits are entangled. Entanglement can be intentionally caused by an application of a gate, which changes state of a qubit. In other words, we can make qubits hold probabilities.

This research project will take advantage of the feature and build a Quantum Machine Learning model based on Bayesian Inference. This model will classify a wine quality, “good” and “bad”, by several sets of variables. As a benchmark, we will use random forest model.

Demonstrating a whole life cycle of quantum modeling, we will explore components of quantum circuit, create new gates, and discover the power of quantum computing.

* 1. Dataset

This research will use a dataset listed in Kaggle, called “Wine Quality Data Set”[[1]](#footnote-1)[1], which consists of 1599 data with 11 features and 1 label:

1 - fixed acidity

2 - volatile acidity

3 - citric acid

4 - residual sugar

5 - chlorides

6 - free sulfur dioxide

7 - total sulfur dioxide

8 - density

9 - pH

10 - sulphates

11 - alcohol

12 - quality ('good' and 'bad' based on score >5 and <5)

1. **Experimentation Plan**

We will build models to classify the set of wine data into “good” and “bad” quality, using varying numbers of features, one classical machine learning model, random forest[[2]](#footnote-2)[2] with Max 2 depth, and four quantum models(QML). The classical model will be used as a benchmark and to find four biggest contributing factors for the binary classification task, using *sklearn.ensemble.RandomForestClassifier(). feature\_importances\_* command.

QML will be built with varying numbers of features, two, three, and four attributes. These features will be re-labelled by either 1 (if >= mean) or 0 (if otherwise). From those labels, likelihood of either value, 1 or 0, will be derived and assigned to a designated qubit. For example, Qubit0 will take a super-position of being either high-alcohol content or low-alcohol content. Qubit1 will take that of being high-sulphates or low-sulphates. More details are discussed later in this paper.

The quantum model will use a hybrid approach of classical and quantum computing, consisting

of three components, pre-processing, training, and post-processing. For data ingestion/Pre-

Processing and Post-Processing, both Classical and Quantum models will use Classical

Computing, i.e., sklearn.model\_selection.train\_test\_split at 20% ratio[[3]](#footnote-3)[3] and

sklearn.metrics.confusion\_matrix[[4]](#footnote-4)[4].

1. **Quantum circuit and Gates**[[5]](#footnote-5)[[6]](#footnote-6)[5][6]

Quantum circuit, a process to assign likelihoods to each of respective qubits, manipulate superpositions through gates, which basically transforms quantum states. There are many different kinds of gate commonly available, but our models use four of them and two other derivatives.

X-Gate

X Gate is used to flip a state of a designated qubit. Qubit0 was initiated with the state |0> at the beginning. X gate qubit0 flips to the state |1> as depicted in the following diagram.

Chart

Description automatically generated

CNOT-Gate

CNOT-Gate, denoted as CX in qiskit API[[7]](#footnote-7)[7], stands for control-NOT gate, which is used to flip a state of a controlled qubit (Qubit1 in this case) if a state of a controlling qubit (Qubit0 in this case) is |1>. As you can see, the Qubit1 was previously |0>, which, CNOT gate, was flipped to the state|1>.

Chart

Description automatically generated

RY-Gate

RY-Gate rotates a state of a qubit by angle, which is converted from probability beforehand, in a help method, prob\_to\_angle function[[8]](#footnote-8)[8], in our case. In the following example, qubit0 was rotated by 50% by RY Gate in the following case. As you can see, qubit0 superposition changed from |0> to 0.5|0> + 0.5|1>.

A picture containing chart

Description automatically generated

CRY-Gate

CRY-Gate, whose “C” represents Controlled, also rotates a state of a qubit by angle for those whose controlling qubit is in state |1>. Qubit1 was flipped by X gate and is taking |1> in the beginning, and Qubi1 was rotated by 50% by RY Gate in the following case. Hence, the qubit1 superposition became 0.5|0> + 0.5|1>.

Chart

Description automatically generated

Derivative gates of CRY gate , CCRY gate and CCCRY gate are discussed in depth in the following section.

1. **Model**

The following flow chart outlines a blueprint of this quantum machine learning model for the wine quality binary classification.

Diagram

Description automatically generated

The first step is data ingestion, conducted in the classical process such as data validation, data cleaning, data conversion and data split into train and test data. Since the later steps in of the model will need probabilities of the occurrence of certain variables and conditional probabilities of combinations of variables, such as alcohol content, sulphates content, volatile acidity, and sulphury dioxide content, those values are converted into binary here. As mentioned earlier, 1 is assigned when a value is beyond average of the entire dataset, and 0 if otherwise. Afterwards, the dataset is split into a training dataset and a test dataset at 80:20 ratio.

The second step calculates probabilities of each of those four biggest contributing variables. To build a Bayesian model with two variables, alcohol content and sulphates content, for instance, the model needs marginal probability of alcohol content being higher/lower than average and that of sulphates content being higher/lower than average, as well as joint probabilities of alcohol content being higher/lower than average and sulphates content being higher/lower than average. There are four joint probabilities for two variables.

In the third step, a “Norm” value is calculated. The Norm value represents an estimated conditional probability, i.e., a probability of “good” wine quality given higher/lower alcohol and higher/lower sulphates contents. This value is updated through a recursive training process in the “train\_qbn\_wine” function, largely driven by “to\_params” and “calculate\_qual\_params” functions, discussed in the fifth step.

The fourth step is where the quantum computing begins.

It first assigns likelihood of alcohol content and sulphates content into first two qubits, the third qubit for norm values, and the fourth qubit for quality. After this operation, super positions of the qubit0 (alcohol) and qubit1(sulphates) becomes 0.57\*|0> +0.43\*|1> and 0.63\*|0> +0.37\*|1>, respectively.

Text

Description automatically generated Text

Description automatically generated

It then entangles these qubits to apply norm parameters to the third qubit.

Text

Description automatically generated

Lastly, it applies the quality parameter to the fourth qubit.

Text

Description automatically generated

To apply the norm parameter in accordance with qubit0 and qubit1, we created a supportive function, CCRY[[9]](#footnote-9)[9]. Its structure follows as below:

1. Apply the parameter to the qubit2 only if qubit0 and qubit1 are 1. It first rotates qubit2 by half of the probability if the qubit1 is 1.
2. Flip the qubit1 when qubit0 is 1.
3. Rotate the qubit2 back by half of the probability if qubit1 is 1. Since the state of qubit1 was flipped in the previous operation, it reversed the qubit2 to the original state if qubit0 is 1 but qubit1 is 0. We do not want to apply any probability for this case because we want to apply it only if both qubit0 and qubit1 are 1.
4. Flips qubit1 if qubit0 is 1.
5. And apply the remaining half probability to the qubit2 I qubit0 is 1.

Text

Description automatically generated **A picture containing text, clock

Description automatically generated**

We also created the CCCRY function to control three qubits for three variables.

**A screenshot of a computer

Description automatically generated with medium confidence**

**A picture containing clock, clipart

Description automatically generated**

The above explanation used to apply probability and “rotate” interchangeably, because those gates take only angles but probability. So, another function is established to convert probability into angle[[10]](#footnote-10)[10].

****

The complete model for two variables looks like the following diagram.

Chart

Description automatically generated

The model is trained in the fifth step. As mentioned in the third step, the model training is run by the “train\_qbn\_wine” function. The role of the “train\_qbn\_wine” function is to generate recursions, generating “results” (quantum states at each recursive training).

The “results” is taken into the “to\_params” function as its argument. The “to\_params” function computes 2^(n+1) numbers of parameters (Norm parameters). The parameters are computed category-wise (higher/lower alcohol, higher/lower sulphates, and high/low quality) by the sum of the previously given parameters of the data favored by Norm (= 1) divided by the sum of all data in the category. This set of parameters is applied to the model in the next iteration.

The “calculate\_qual\_params” function returns a set of two parameters, given by sum of the Norm parameter of good quality wine divided by the sum of the Norm parameter of all data. Conceptually, the parameter can be interpreted as how accurate the Norm parameter was previously estimated.

1. **Result**

We experimented total 5 models, 1 classical machine learning model, random forest model, and four quantum models. The following sections shows results and model specifications.

* 1. Random Forest model

The random forest model used Scikit-learn API. With two layers/depths, it achieved 72.5% of Accuracy. Though this is a basic implementation and there shall be much space to improve its accuracy, we will benchmark 72.5% and move on because the primary purpose of this research is to investigate the power of quantum computing.

Through this implementation, we identified four biggest contributing variables in this dataset, which are alcohol, sulphates, volatile acidity, and sulfur dioxide contents. The following quantum computing will be built on these variables.

Diagram

Description automatically generatedChart, bar chart

Description automatically generated

* 1. Quantum Model with Two Variables

The first quantum model was built with two variables, Alcohol and Sulphates, from which Norm parameters were derived to predict Quality. This model achieved the highest score at 71 % of accuracy after 20 iterative trainings.

Diagram

Description automatically generated Chart, line chart

Description automatically generated

* 1. Quantum Model with Three Variables

The second set of quantum models was built with three variables, Alcohol, Sulphates, and Volatile Acidity. There are two types of models with three variables. One of which derives Norm parameters from three of variables, while the other derives from two Alcohol and Sulphates, and apply conditional probability to Quality qubit based on the norm parameter and Volatile Acidity.

5.3-1 Bayesian Inference with Three Variables

This is the model in which the norm parameter is calculated by three of the variables. This model achieved the highest score, 52 % of accuracy after 40 iterative trainings.

Diagram

Description automatically generated Chart, line chart

Description automatically generated

5.3-2 Bayesian Inference with Two Variables and One independent variable

This is the model in which the norm parameter is calculated by two variables and the probability of volatile Acidity was applied directly to Quality qubit. This model achieved the highest score 74 % of accuracy after 30 iterative trainings.

Diagram

Description automatically generated Chart, line chart

Description automatically generated

* 1. Quantum Model with Four Variables

Considering the model performances with three variables, it appears to be more worthwhile exploring the model with the norm parameters derived with two variables than three. This model derives the norm parameter from Alcohol and Sulphates and apply probabilities of Volatile Acidity and Sulphur dioxide contents directly to Quality qubits as conditional probabilities. This model achieved the highest score 74 % of accuracy after 30 iterative trainings.

Diagram

Description automatically generated Chart, line chart

Description automatically generated

1. **Conclusion and Future Directions**

The quantum machine learning model achieved 74% of accuracy with three variables and four variables, slightly better performance than the classical random forest model. Amongst four quantum models, the second batch with 3 variables constituting norm value is an exception. Its performance was rather more volatile than three other models. It even deteriorated after 20 and 30 iterations. Amongst other three quantum models, the number of training iterations showed impact on the result when it was increased from 1 to 10 iterations, but only minimal change was observed after afterwards. Comparing the result of the model with two variables, the result of the model with four variables improved by 3%. The biggest difference in these two models is the starting accuracy, observed when only one iterative training was implemented – the one with two variables was only 53%, while the one with four was 61% of accuracy.

This research relied on a classical computation to find big contributing factors/variables, however, in my future research I would like to explore a way to conduct searching biggest contributing variables with Quantum Computing, by such means as Grover’s Search Algorithm[[11]](#footnote-11)[11], which, if successfully constructed, could search a desired objective in quadratic time sqrt(n). It would also be interesting to explore applications of Shor’s algorithm[[12]](#footnote-12)[12], since the author believes that it can enhance time complexity of optimization problems.

1. **References**

[1] Naresh Bhat, Wine Quality Classification - Data set for Binary Classification [https://www.kaggle.com/nareshbhat/wine-quality-binary-classification?select=wine.csv]

[2] Scikit-learn, sklearn.ensemble.RandomForestClassifier [https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html]

[3] Scikit-learn, sklearn, sklearn.model\_selection.train\_test\_split [https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html]

[4] Scikit-learn, sklearn, sklearn.metrics.confusion\_matrix¶ [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion\_matrix.html]

[5] Chris Bernhardt, QUANTUM COMPUTING FOR EVERYONE (P.118)

[6] Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P. 147)

[7] Qiskit [https://qiskit.org/textbook/ch-gates/multiple-qubits-entangled-states.html#cnot] (3. Multi-Qubit Gates 3.1 The CNOT-Gate)

[8] Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P. 141)

[9] Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P.276)

[10] Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P.260)

[11][12] Chris Bernhardt, QUANTUM COMPUTING FOR EVERYONE (P.176)

1. Naresh Bhat, Wine Quality Classification - Data set for Binary Classification [https://www.kaggle.com/nareshbhat/wine-quality-binary-classification?select=wine.csv] [↑](#footnote-ref-1)
2. Scikit-learn, sklearn.ensemble.RandomForestClassifier [https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html] [↑](#footnote-ref-2)
3. Scikit-learn, sklearn, sklearn.model\_selection.train\_test\_split [https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html] [↑](#footnote-ref-3)
4. Scikit-learn, sklearn, sklearn.metrics.confusion\_matrix¶ [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion\_matrix.html] [↑](#footnote-ref-4)
5. Chris Bernhardt, QUANTUM COMPUTING FOR EVERYONE (P.118) [↑](#footnote-ref-5)
6. Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P. 147) [↑](#footnote-ref-6)
7. Qiskit [https://qiskit.org/textbook/ch-gates/multiple-qubits-entangled-states.html#cnot] (3. Multi-Qubit Gates 3.1 The CNOT-Gate) [↑](#footnote-ref-7)
8. Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P. 141) [↑](#footnote-ref-8)
9. Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P.276) [↑](#footnote-ref-9)
10. Dr. Frank Zickert - Hands-On Quantum Machine Learning With Python Volume 1\_ Get Started-PYQML (2021) (P.260) [↑](#footnote-ref-10)
11. Chris Bernhardt, QUANTUM COMPUTING FOR EVERYONE (P.176) [↑](#footnote-ref-11)
12. Chris Bernhardt, QUANTUM COMPUTING FOR EVERYONE (P.176) [↑](#footnote-ref-12)