IEICE TRANS. INF. & SYST., VOL.E102–D, NO.4 APRIL 2019

LETTER

# Quantum Algorithm on Logistic Regression Problem

**Jun Suk KIM**[†a)], ***Nonmember and*** **Chang Wook AHN**[†b)], ***Member***

**SUMMARY** We examine the feasibility of Deutsch-Jozsa Algorithm, a basic quantum algorithm, on a machine learning-based logistic regression problem. Its major property to distinguish the function type with an exponential speedup can help identify the feature unsuitability much more quickly. Although strict conditions and restrictions to abide exist, we reconfirm the quantum superiority in many aspects of modern computing.

***key words:*** *quantum machine learning, Deutsch-Jozsa Algorithm, logistic regression, feature selection*

## 1. Introduction

Quantum computing has already been in the center of arguments and concerns by computer scientists for nearly two decades, primarily due to its exotic complexity and potential to significantly outperform existing machines for expensive problems, which are inaccessibly costly for classical computers [1]. In spite of several technological difficulties, endeavors to press beyond the horizon are steadily yielding remarkable advances [2]. Consequently, it is not too early to bring the quantum ideas into artificial intelligence, more specifically, machine learning, a versatile tool that can help integrate massive data into essential and precise information. Arguably, quantum advantages can provide higher time-efficient means to machine learning processing in various aspects. In this paper, we analyze their usefulness on machine learning using the simple quantum algorithm, Deutsch-Jozsa Algorithm.

## 2. Quantum Context

The basic computation unit of a quantum computer is a qubit, theoretically implementable with a polarized photon. Unlike a bit, its classical analog, a qubit can be represented as a mixture of probabilities of its two definite states of 0 and 1. In other words, the law of quantum superposition imposes single qubit to reside in multiple states with certain probabilities at the same time as if it exists in numbers, implying the feasibility of simultaneous and parallel computation [3].

Deutsch-Jozsa Algorithm, one of the earliest and simplest forms of quantum algorithm, utilizes superposition to enable function evaluation speedup. It is a generalized variant of Deutsch's Algorithm [4], which is illustrated in Table 1. With a binary digit as a given input, if $f(0) = f(1)$, call the function "constant"; if $f(0) \neq f(1)$, call it "balanced." The problem to be solved with the algorithm is to determine whether a chosen function is constant or balanced. In classical evaluation, one needs to perform two queries of calculation, i.e., calculate $f(0)$, then calculate $f(1)$. Deutsch's Algorithm, on the other hand, shows that the function can be identified with single quantum query by exploiting a superposition of qubits. Deutsch-Jozsa Algorithm also completes the evaluation in one query, but this time with its input as a binary string, capable of evaluating multiple qubits simultaneously. With an input string in length of $n$, classical evaluation would require $2^{(n-1)} + 1$ times of query to solve the problem, so the quantum application provides an exponential speedup.

**Table 1** Constant and balanced functions of Deutsch's Algorithm.

| $x \rightarrow f(x)$ | Function Type |
|---|---|
| $0, 1 \rightarrow 0, 0$ | Constant |
| $0, 1 \rightarrow 1, 1$ | Constant |
| $0, 1 \rightarrow 0, 1$ | Balanced |
| $0, 1 \rightarrow 1, 0$ | Balanced |

The basic structure of quantum circuit for Deutsch-Jozsa Algorithm consists of Hadamard operators ($H$) and a unitary "black-box" oracle ($U_f$). The top qubit, initially in the state of $|000\ldots0\rangle$, is the input, and the bottom qubit, $|1\rangle$, is the auxiliary qubit that controls the input. Both qubits pass through the Hadamard operators to become superposed and enter the oracle, which is fundamentally a query to conduct function evaluation. Finally, the top qubits proceed with another Hadamard operation to exit the superposition, ready for measurement. Mathematically, the overall qubit states at the final stage of the circuit can be written as

$$[\frac{\sum(-1)^{f(x)}|0\rangle}{2^n}][\frac{|0\rangle - |1\rangle}{\sqrt{2}}], \tag{1}$$

So we can determine the function by measuring the first part of (1); it is $(\pm1)|0\rangle$ if $f$ is constant; it is $0|0\rangle$ if $f$ is balanced. All the $|0\rangle$ state terms become either $+1$ or $-1$ in the constant case because every outcome must be the same. Exactly half of them cancel the other half out and result $0|0\rangle$ in the balanced case. For more thorough mathematical explanation, refer to [3]. Note that we can identify the function with single query as promised, regardless of amount of the input.

Manuscript received October 25, 2018.
Manuscript publicized January 28, 2019.

[†]The authors are with School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology, Korea.
a) E-mail: junsuk89@gmail.com
b) E-mail: cwan@gist.ac.kr

DOI: 10.1587/transinf.2018EDL8223

## 3. Adaptation to Machine Learning

In supervised machine learning, one of widely used techniques for data classification is Logistic Regression. It is intuitively clear and comparatively simple to implement, yet provides a decent outcome utilizable in many branches of industries. The basic idea is grouping: it assigns an appropriate amount of weight to each feature of input training set values so that the features' composition, or hypothesis, draws a proper border that divides a set of training individuals into their respective groups. Whether a regression hypothesis is good or bad depends on how well it fits with both the training and test sets. Although avoiding underfitting by obtaining precise hypotheses that accurately fit with their training sets is prioritized, overfitting raises risk of inflexibility, failing to generalize to new, other data sets. To avoid overfitting, two main options exist: reduce the number of features of input set, or conduct regularization.

Here, we examine how Deutsch-Jozsa algorithm can help analyze overfitting issues by feature reduction. Consider a simple logistic regression problem with an overfit hypothesis function that shows high accuracy over training sets and low accuracy over validation sets. We would like to discard several features so that the hypothesis becomes more general and thus more likely to fit with new, arbitrary data. For this particular problem, we decide to manually choose features to get rid of. Figure 1 represents our newly defined "fitting status" function. $x_1$ and $x_2$ are the binary indication of inclusion of two features that we suspect are contributing to the overfitting the most. For example, $x_1x_2 = 01$ means that we keep one feature and dispose of the other from our hypothesis. $f(x)$, on the other hand, shows if the fitting turns better with respect to the input string $x_1x_2$. $f(x) = 0$ indicates that the hypothesis now shows the better fitting, while $f(x) = 1$ means that the fitting is still unsatisfactory either by overfitting or by underfitting. We expect that the degree of such dissatisfaction can be tracked down by setting proper ranges of measures such as root mean squred error (RMSE). Note that the terms "better" and "unsatisfactory" above are not strictly defined and would thus mean various degrees in different cases.

The size of the input for our function will grow exponentially as the number of the considered features increases; with just 10 features there are 1024 inputs to go over. Apparently, checking them one by one is virtually impracticable with classical computing. If such work is done in single query regardless of the amount of input, however, we can

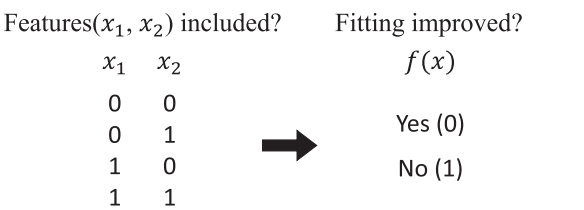


**Fig. 1** Indication of feature inclusion and fitting change.

attempt a few simple, disposable "blind shootings" which verify some of our rough guesses with trivial costs. We demonstrate how Deutsch-Jozsa Algorithm achieves it with the following trick: $x_1x_2 = 11$ means that we do not get rid of any features from our original list, so the fitting should remain unchanged. Therefore, the input 11 always results the output 1 because any unchanged fitting stays unsatisfactory. Now suppose that neither of the two features we have chosen is the main cause of the overfitting, meaning that the input 00 should give us the output 1. Nonetheless, we can't hastily conclude that these features have nothing to do with our feature selection yet because even if they do not mainly contribute to the overfitting, their absence might cause underfitting, in which case the fitting should still be labeled unsatisfactory. Perhaps getting rid of only one could work, and figuring that out requires further computations regarding the inputs 01 and 10, in classical computing.

In quantum computing, let us apply Deutsch-Jozsa algorithm upon the problem. Suppose that its output tells us that the unknown relationship between the feature inclusion and the fitting status - namely, the function - is constant. Remember that the input 11 always goes to the output 1, which means in the constant case every input goes to the output 1. With that logic kept, we can make our conclusion right away: no matter how we alter the binary indication of the features we have chosen, the corresponding fitting change will always stay unsatisfactory. We acquire this result with single computation query.

Overall, the whole procedure can be generalized into the following steps:

1. In case an overfit logistic regression hypothesis has been constructed, select $n$ input features that are suspected to be contributing to the hypothesis fitting problem the most.
2. Assign binary indication variables to the selected features in forms of $x_1, x_2, \ldots x_n$. For each, set 0 if we exclude the corresponding feature from our feature list; set 1 if we have decided to keep it.
3. Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a function that computes whether the fitting improves by adjusting the features. $f(x = x_1x_2 \ldots x_n) = 0$ means that it has improved, and $f(x) = 1$ means that it stays unsatisfactory.
4. Now that the input and output are set up, apply Deutsch-Jozsa Algorithm on computing $f$ with given inputs. Recall that the circuit needs binary input. Because we set the function input as binary, no alteration from the original configuration is needed.
5. Measure the result. The best possible outcome is expected to take place when the algorithm tells that the function is constant; it shows that any combination of the features we have chosen would not help alleviate the fitting pressure.
6. If the result is balanced or not constant, it does not allow a user to make any meaningful conclusion. One can either try a new feature combination or use other feature reduction methods.

## 4. Discussion

In Deutsch-Jozsa Algorithm, if a function is balanced, its inputs are divided into exact halves, meaning that the number of the inputs must be even. Odd numbers of the inputs would never produce a balanced function and cause the algorithm produce a "vague" outcome, out of which one can't make any meaningful conclusion. In order to avoid the discrepancy, we designed the input as a binary combination of "acceptance (0) or rejection (1)" of each feature. We then can guarantee that with $n$ features the number of the inputs always stays even ($2^n$) no matter what $n$ is, so the method is generalized to diverse data sets.

Recall that increase in computational cost with respect to the size of input shouldn't be a primary concern, thanks to the quantum oddities. Placing our method in the middle of the computing process should not significantly increase the time or computation cost, even for operations with enormous feature scale. Consequently, we can play a series of giveaways to see if we're lucky with our guess and know which features wouldn't lift the fitting pressure, or there is no luck but the computational efficiency is not seriously harmed and thus can still afford trying other methods.

As far as the circuit configuration is concerned, we would only need to switch the Hadamard operators for different inputs. Specifically, as a function does not necessarily change accordingly with different inputs in classical systems, a quantum black-box oracle wouldn't require any modification as long as the regression setting, including training, testing, and validating sets, remains the same. This is worth mentioning because it implies that the input to our suggested application can be changed without altering the whole problem. One might want to choose and try different combinations of features prior to actually performing their removal.

With the outcome from the previous section in our hand, one question arises: what if the algorithm tells you that the function is balanced, not constant? Again, we know that the input 11 goes to the output 1 regardless, but what would happen to the inputs 00, 01, and 10 remains enigmatic. We can only know that one goes to the output 1, and the other two go to the output 0. One major obstacle in seeking usefulness of Deutsch-Jozsa Algorithm is that it can't specify a particular element from the set of output; every state of an input has its evaluation via superposition, but it is "quarantined" from any external observance. Because Hadamard matrices are reversible, passing through them twice puts the top qubit back to its original canonical state, in which the superposition is no longer in effect. Measuring qubits before the second Hadamard operation wouldn't help either; any attempts to measure them in superposition are expected to cause wave function collapse, resulting all but one state losing their probabilities of existence. Furthermore, the fact that the function is promised to be either constant or balanced [3] causes another problem when the function is neither constant nor balanced. It is highly unlikely that we would find any effective outcome from the algorithm. After all, even though Deutsch-Jozsa Algorithm is a powerful engine, it is not surprising to realize that its actual exploitation is considerably limited.

## 5. Conclusion

Deutsch-Jozsa Algorithm is known as the most basic form of quantum algorithm, applicable presumably only to contrived problems. We showed, however, that we can use it to aid the faster feature selection process in logistic regression problems. More specifically, the algorithm helps quickly identify if the chosen features are unacceptable for our list. Although the definite outcome is expected by chance along with several conditions in strict setting, we found a technique to put into a decent use. The next generation of quantum algorithms, such as Grover's and Shor's Algorithms, are believed to be much more suited and exploitable to quantum machine learning problems [5], but it is without doubt that Deutsch-Jozsa Algorithm has inspired many to proceed with the potential of quantum computing, and its applications are still noteworthy.

## Acknowledgments

This work was supported by GIST Resarch Institute (GRI) grant funded by the GIST in 2019.

### References

[1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum Machine Learning," Nature, vol.549, pp.195–202, Sept. 2017.

[2] D. Ristè, M.P. da Silva, C.A. Ryan, A.W. Cross, A.D. Córcoles, J.A. Smolin, J.M. Gambetta, J.M. Chow, and B.R. Johnson, "Demonstration of quantum advantage in machine learning," npj Quantum Information 3, no.16, pp.1–5, 2017. DOI:10.1038/s41534-017-0017-3

[3] N.S. Yanofsky, M.A. Mannucci, ed., Quantum Computing for Computer Scientists, Cambridge University Press, Cambridge, 2008.

[4] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," Royal Society Proceedings: Mathematical and Physical Sciences, vol.439, no.1907, pp.284–290, Dec. 1992. DOI:10.1098/rspa.1992.0167

[5] W.Z. Chung and S.-W. Lee, "The Present and perspective of quantum machine learning," Journal of KIISE, vol.43, no.7, pp.751–762, 2016. DOI:10.5626/JOK.2016.43.7.751