HDSR
HARVARD DATA SCIENCE REVIEW

Volume 4 Issue 1
DOI: 10.1162/99608f92.69c5328d
ISSN: 2644-2353

Assume a Quantum Data Set

Mária Kieferová[†, ‡, *] and Yuval R. Sanders[†]

[†] Centre for Quantum Software and Information,
University of Technology Sydney, NSW 2007, Australia

[‡] Centre for Quantum Computation and Communication Technology,
University of Technology Sydney, NSW 2007, Australia

Abstract. Data-processing algorithms often require that the data is prepared in appropriate structures that are readily accessible or can be prepared on demand. Quantum computers derive their power from storing and manipulating quantum superpositions and could potentially speed up data science tasks. However, they often require input in the form of a quantum state that encodes a nonquantum data set. Here we describe some of the challenges of encoding nonquantum data for use by quantum computers.

Keywords: quantum algorithms, state preparation, quantum machine learning, big data

1. Initialization for quantum algorithms

Quantum computing promises to solve problems that are infeasible for traditional computers. Apart from the simulation of quantum systems, such as in chemistry or material science, there has been a surge in quantum linear algebra algorithms suitable for high-dimensional problems. These algorithms include linear system solvers, regression or machine learning algorithms that have the potential to perform otherwise impossible data science tasks. These otherwise-impossible tasks would likely involve extraordinarily large data sets in which the superior asymptotic complexity scaling of quantum algorithms can prevail over highly optimized supercomputer code.

It is important to emphasize that the 'superior asymptotic complexity scaling' to which we and other quantum computer scientists refer assesses only the complexity of *processing* data. Our aim in this commentary is to elucidate the oft-neglected complexity of *encoding* data into an appropriate format for quantum processing.

We expect the quantum computer to achieve an advantage over classical computers by employing a 'quantum' data encoding, meaning that the data would be presented in some kind of quantum superposition. Thus, the quantum computer can take advantage of entanglement and superposition when processing the data instead of processing them bit-by-bit as classical computers do. The data will be then presented as a quantum state that cannot be copied and need to be measured in order to retrieve classical information that would lead to a collapse of the superposition.

[*] maria.kieferova@uts.edu.au

Published quantum algorithms usually *assume* that the data is accessible in the form required by the quantum algorithm. One might assume that a quantum programmer has access to quantum data from a cloud, however, interacting with such data sets could likely lead to the creation of entanglement between the programmer's quantum computer and the cloud. Alternatively, the quantum data scientist would only have access to a classical database and it will be up to them to transform the classical data into quantum states in an appropriate form.

For clarity's sake, we describe in detail two common methods of quantum data encoding: *bit encoding* and *amplitude encoding* (following the terminology of Wiebe, 2020). In both cases, we consider the task of encoding a classical data set

$$\text{data} = \{x_1, x_2, \ldots, x_N\} \tag{1.1}$$

where each x_ℓ encodes an M-dimensional data point. Focusing on 'big data', we assume that at least one of M or N being very large.

2. Bit encoding

Suppose that each data point x_ℓ ($\ell = 1, 2, \ldots, N$) can be represented by the unique bitstring $\vec{b}^\ell = b_0^\ell b_1^\ell \ldots b_{K-1}^\ell$, where the integer K depends on the kind of data being encoded. If, for example, each M-dimensional data point is a length-M array of 32-bit floating point numbers and we have not compressed the data in any way, then $K = 32M$. Then we can directly encode the data point x_ℓ in a quantum register consisting of K qubits:

$$x_\ell \mapsto |b_0^\ell\rangle\, |b_1^\ell\rangle \ldots |b_{K-1}^\ell\rangle \equiv |\vec{b}^\ell\rangle\,. \tag{2.1}$$

Such encoding can be accomplished for a given data point by first preparing the quantum register in the all-zero state $|0\rangle\, |0\rangle \ldots |0\rangle$ and then applying bitflip operations to the appropriate qubits. However, nothing about this approach is yet 'quantum': the strategy can be understood in purely classical terms. This approach becomes quantum if we choose to encode the data set as a superposition of such bit-encoded datapoints; for example,

$$\text{data} \mapsto \frac{1}{\sqrt{N}} \sum_{\ell=1}^{N} |\vec{b}^\ell\rangle\,, \tag{2.2}$$

where the coefficient of $\frac{1}{\sqrt{N}}$ is present to enforce the technical requirement of state normalization. We refer the reader to chapter 4 of Schuld and Petruccione (2021) for details.

The bit encoding method would play an important role in data science techniques that depend on Grover's search algorithm (Grover, 1996), or closely related techniques, due to the need for a *black-box quantum subroutine*. This means the user of Grover's algorithm is expected to construct a quantum program that assigns to each possible input ℓ a mark that indicates to the search algorithm whether or not the marked item is that which is being searched for: the program should perform $|\ell\rangle\, |0\rangle \mapsto |\ell\rangle\, |\text{mark}(\ell)\rangle$. One could imagine programming some function that assigns a mark to each possible data point (i.e., performs $|\vec{b}\rangle\, |0\rangle \mapsto |\vec{b}\rangle\, |\text{mark}(\vec{b})\rangle$ for any bitstring $\vec{b}$), but this would be an incomplete solution. We also need a data encoding routine of the form $|\ell\rangle\, |\vec{0}\rangle \mapsto |\ell\rangle\, |\vec{b}^\ell\rangle$, where $\vec{0}$ refers to the length-K bitstring consisting entirely of zeroes. Then we could combine the two into the sort of black-box quantum subroutine required by Grover search.

The bit encoding method then arises because Grover's search strategy involves preparing superpositions like $\frac{1}{\sqrt{N}} \sum_{\ell=1}^{N} |\ell\rangle$ using $O(\log N)$ quantum operations. The user-provided quantum

subroutine would then be used *once* to create a superposition of the form

$$\frac{1}{\sqrt{N}} \sum_{\ell=1}^{N} |\ell\rangle \, |\vec{b}^{\ell}\rangle \, |\mathrm{mark}(\vec{b}^{\ell})\rangle \,, \tag{2.3}$$

at which point Grover's algorithm prescribes a technique to boost the amplitude of marked items at the expense of unmarked items with $O(\sqrt{N})$ uses of the black-box quantum subroutine that, to repeat, must be supplied by the user. By contrast, we require $O(N)$ calls to that user-provided subroutine in nonquantum computing.

This is an enticing *potential* speedup, but one must remember that the overall cost of the algorithm depends heavily on the computational cost of the user-provided quantum subroutine, which depends in a potentially complicated way on the parameter K and may turn out to dominate the computational complexity. It is this *overall* cost that must be assessed when evaluating quantum versus classical approaches. We caution data scientists against the tempting habit to ignore the cost of the user-provided subroutine and counting only the number of uses of that subroutine.

3. Amplitude encoding

In contrast with bit encoding, the amplitude encoding method seeks to encode each individual data point in the *amplitudes* of a quantum superposition, rather than directly in the quantum register. In this case, we assume each data point x_ℓ can be represented with a length-M vector $\vec{v}^{\ell} = \left(v_1^{\ell}, v_2^{\ell}, \ldots, v_M^{\ell}\right)$ that, for technical reasons, we further assume to be normalized ($\vec{v}^{\ell} \cdot \vec{v}^{\ell} = 1$) and positive ($\vec{v}_k^{\ell} \geq 0$ for each k, ℓ)—see Schuld and Petruccione (2021) for a detailed discussion. The amplitude encoding can encode each data point in quantum superposition using a procedure like

$$x_\ell \mapsto \sum_{k=1}^{M} v_k^{\ell} \, |k\rangle =: |\vec{v}^{\ell}\rangle \,. \tag{3.1}$$

The amplitude encoding is then analogous to storing the vector $\vec{v}^{\ell}$ by creating an M-sided die such that, when rolling that M-sided die, the probability of side k showing is equal to the square of v_k^{ℓ}. In other words, we perform *readout* of the data x_ℓ from the M-sided die by (1) rolling the die many times, (2) recording the relative frequencies f_k of each outcome k, and (3) calculating the square root $\sqrt{f_k} \approx v_k^{\ell}$; the more we roll the die, the more accurate the readout. Note that step (3) is necessary because a quantum amplitude is related to the *square root* of a probability.

One could then encode the entire data set as a 'superposition of superpositions':

$$\text{data} \mapsto \frac{1}{\sqrt{N}} \sum_{\ell=1}^{N} \sum_{k=1}^{M} v_k^{\ell} \, |\ell\rangle \, |k\rangle \equiv \frac{1}{\sqrt{N}} \sum_{\ell=1}^{N} |\ell\rangle \, |\vec{v}^{\ell}\rangle \,. \tag{3.2}$$

Although this strange encoding has important limitations, it has one powerful feature: the number of qubits needed to store this state is $O(\log(NM)) = O(\log N + \log M)$ because we need only $O(\log N)$ qubits to store the index ℓ and $O(\log M)$ qubits to store the index k. This contrasts with the $O(MN)$ space cost one would expect for storing such a data set in a classical register. The exponential improvement to space cost underlies the potentially exponential gains from applications such as quantum linear system solvers (Harrow et al., 2009) and quantum data fitting algorithms (Wiebe et al., 2012), however, defining the encoding could be quite computationally difficult (see, e.g., Aharonov and Ta-Shma, 2007) and thereby wipe out any advantage to using the quantum computer.

4. Conclusion

The implication for data scientists is that there are enticing quantum speedups to be investigated, but it is not easy to translate those *potential* speedups into *actual* improvements for practical problems. This challenge is not unique to data science and the computational cost of data encoding has been raised in the context of quantum machine learning in the work of Aaronson (2015) and Wiebe (2020).

The proper choice of quantum data encoding methodology depends on the kind of data being analyzed and the kind of algorithm to be applied. We should expect that the quantum data encoding methodology has meaningful and potentially detrimental effects on the overall efficiency of the algorithm, and that the complexity analysis of that quantum data encoding methodology is a potentially challenging intellectual exercise.

The difficulty of encoding data for quantum processing indicates a larger need to consider the management of quantum memory. While the immediate development of quantum computers focuses on perfecting and scaling the quantum processing unit (QPU), quantum memory management could soon become an important and distinct research area. Researchers are already starting to consider the role of quantum RAM (Arunachalam et al., 2015; Giovannetti et al., 2008) as well as quantum ROM (Berry et al., 2019; Low et al., 2018) within the analysis of quantum computer applications.

Our key message is that it is necessary to consider the complete quantum algorithm—input, data processing, and output—in order to compare it with its classical counterpart. Since quantum information cannot be copied and is destroyed by measurement, the complexity of input is a pervasive cost that needs to be factored into the computation. We therefore advise data scientists to temper their excitement about the promise of quantum algorithms by heeding the challenge of presenting data to the quantum computer.

Disclosure Statement. The authors have no conflicts of interest to declare.

Acknowledgments. We thank Michael Bremner for insightful discussions. MK was supported by the Sydney Quantum Academy, Sydney, NSW, Australia and ARC Centre of Excellence for Quantum Computation and Communication Technology (CQC2T), project number CE170100012. YRS is supported by Australian Research Council Grant DP200100950

Contributions. Both authors contributed to the work equally.

References

Aaronson, S. (2015). Read the fine print. *Nature Physics*, *11*(4), 291–293. https://doi.org/10.1038/nphys3272

Aharonov, D., & Ta-Shma, A. (2007). Adiabatic quantum state generation. *SIAM Journal on Computing*, *37*(1), 47–82. https://doi.org/10.1137/060648829

Arunachalam, S., Gheorghiu, V., Jochym-O'Connor, T., Mosca, M., & Srinivasan, P. V. (2015). On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, *17*(12), Article 123010. https://doi.org/10.1088/1367-2630/17/12/123010

Berry, D. W., Gidney, C., Motta, M., McClean, J. R., & Babbush, R. (2019). Qubitization of arbitrary basis quantum chemistry leveraging sparsity and low rank factorization. *Quantum*, *3*, 208.

Giovannetti, V., Lloyd, S., & Maccone, L. (2008). Quantum random access memory. *Physical Review Letters*, *100*(16), Article 160501. https://doi.org/10.1103/physrevlett.100.160501

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, 212–219. https://doi.org/10.1145/237814.237866

Harrow, A. W., Hassidim, A., & Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical Review Letters*, *103*(15), Article 150502. https://doi.org/10.1103/physrevlett.103.150502

Low, G. H., Kliuchnikov, V., & Schaeffer, L. (2018). Trading t-gates for dirty qubits in state preparation and unitary synthesis. *ArXiv e-print*.

Schuld, M., & Petruccione, F. (2021). *Machine learning with quantum computers*. Springer International Publishing. https://doi.org/10.1007/978-3-030-83098-4

Wiebe, N. (2020). Key questions for the quantum machine learner to ask themselves. *New Journal of Physics*, *22*(9), Article 091001. https://doi.org/10.1088/1367-2630/abac39

Wiebe, N., Braun, D., & Lloyd, S. (2012). Quantum algorithm for data fitting. *Physical Review Letters*, *109*(5), Article 050505. https://doi.org/10.1103/physrevlett.109.050505