From Bits to Qubits: An Overview of Data Encoding Techniques in Quantum Machine Learning

Johnny Stuto

Computer Science Dept. University of Idaho

stut2030@vandals.uidaho.edu

Abstract—**This survey paper provides a comprehensive overview of classical data encoding techniques in quantum machine learning (QML), a field that merges quantum computing with machine learning. By harnessing quantum mechanics, QML algorithms aim to solve complex problems more efficiently, presenting new opportunities for innovation across various disciplines.**

The scope of this survey will be the encoding methodology of classical data into in the field of QML. This translation method has implications for the performance and applicability of its algorithms. Effective encoding strategies must not only preserve the integrity of the original data but also exploit the quantum properties to enhance computational power. By outlining the advantages, disadvantages, and applicability of each method, this survey paper hopes to guide researchers and practitioners in choosing appropriate encoding methods for their quantum machine learning projects.

I. Introduction

QML is poised to redefine the boundaries of data analysis, pattern recognition, and predictive modelling. QML leverages the unique properties of quantum systems such as superposition and entanglement to process information in ways that classical systems cannot, potentially leading to more efficient and powerful algorithms, but only if classical data can be encoded into quantum states.

Quantum computers operate on qubits, which behave according to the principles of quantum mechanics. Classical data, on the other hand, is represented in bits. To utilize the advanced computational capabilities of quantum computers for machine learning tasks, this classical data must be converted into a quantum state known as encoding. This step is not merely a formality but a fundamental aspect that determines the efficiency and feasibility of quantum computations. The choice of encoding strategy can influence the performance and applicability of QML algorithms, making it a pivotal area of study. [1]

A. Background

At the heart of quantum computing lies the qubit, the quantum analogue of the classical bit. Unlike a bit, which exists in a state of either 0 or 1, a qubit can exist in a state of 0, 1, or any quantum superposition of these states. This qubit is represented by vector:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are the complex coefficients of the basis states:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \ \& \ |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$

which are normalized as $|\alpha|^2 + |\beta|^2 = 1$. These normalized qubit basis coefficients or amplitudes, represent the probabilities of each basis state realization as $|\alpha|^2$ & $|\beta|^2$. With n qubits, the 2n basis states provide computational breadth. Until measured the system is in superposition, where the qubit is in both basis states at once. [2, 4]

When multiple qubits are entangled, a phenomenon unique to quantum mechanics, the computational power increases exponentially. This entanglement allows quantum computers to process and analyse large datasets much more efficiently than classical computers. [2]

Another key aspect of quantum computing is the quantum gate, the equivalent of a logical gate in classical computing. Quantum gates are unitary matrices that manipulate qubits, to change their state while maintaining quantum properties. The Pauli-X, Pauli-Y, and Pauli-Z gates are rotations of the state about each respective axis, while Hadamar gates place qubits into superposition. [2]

A sequence of quantum gates forms a quantum circuit, which can execute algorithms to solve specific problems. Quantum encoding takes a classical data and translates it into a set of gate parameters in a quantum circuit, creating a quantum state. [1, 2]

Classical encoding is a well-established field, dealing with the representation of information in forms suitable for classical computation. Common methods include binary encoding, one-hot encoding, and grey coding, each with its own advantages in specific uses. Quantum data encoding, by contrast, is not just about representing information but also about maintaining coherence and exploiting entanglement, which are absent in classical systems.[1]

B. Four quadrants of QML

Figure 1 partitions QML into four quadrants crossing quantum and classical processing and data. The upper left quadrant is primarily classical machine learning with novel quantum ideas projected onto it. Both lower quadrants use quantum data as an input and are reserved for applied physics experiments where quantum data can be captured. The upper right quadrant uses classical data on a quantum computing device where the classical data needs to be transformed into quantum states. [2]

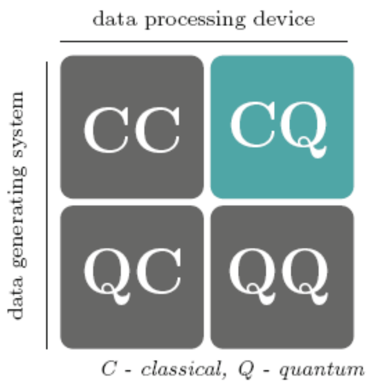

Fig. 1. Quantum/Classical ML

C. QML on Noisy Intermediate Scale Quantum (NISQ)

NISQ, a term coined by John Preskill, is defined as a near term quantum computer with at least 1000 qubits with poor fault-tolerance, but near quantum supremacy.[3] Quantum supremacy is another term from the physicist above which refers to the ability of a quantum device to solve a problem that no classical computer can solve in some predetermined computation timeframe.[8]

Figure 2 breaks down the stages of QML. [2] Most algorithmic improvements touted by QML mandates less than exponential complexity of it encoding stage. It is the second stage of encoding that gatekeeps any novel algorithm and is the scope of this survey.

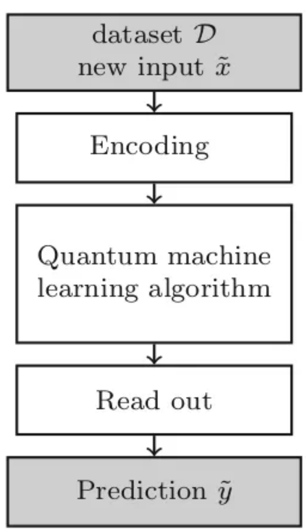

Fig. 2. QML framework [2]

II. Current QML Data Encoding Techniques

Data encoding in QML is a rapidly evolving area, with several techniques developed to translate classical data into quantum states. Mathematically, this encoding project each classical datapoint onto a complex point on the Bloch sphere as shown in figure 3 [6].

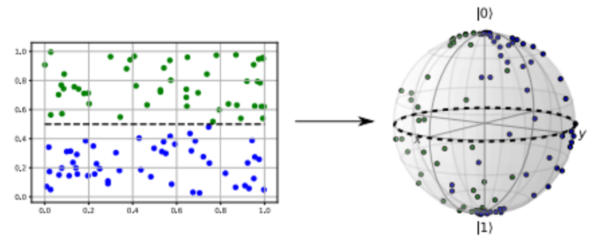

Fig. 3. Encoding Projection [6]

Loading classical datasets onto quantum devices can be a major bottleneck in QML as several encoding methods can be used depending on the algorithm. Each method has its unique characteristics, advantages, and limitations depending on the required number of qubits, the number of parallel encoding operations and the required data representation.[5] Here we discuss some of the prominent data encoding techniques currently in use, where the first three methods encode element by element of a dataset, the fourth method encodes the entire dataset simultaneously. [12]

A. Basis Encoding

Basis or digital encoding involves directly mapping classical binary data to quantum states. This first use of this method was Vedral et al. [14] give multiple examples for algorithms and another formal solution is also given in Leymann et al. [8]. As only one quantum gate is needed to obtain this encoding, this state preparation routine can be implemented straightforwardly. In this method, each classical bit is represented by a qubit in either the $|0\rangle$ or $|1\rangle$ state, corresponding to the classical bit values 0 and 1, respectively.

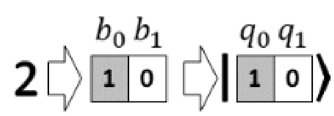

Fig. 4. Basis Encoding of decimal "2" [5]

Mathematically, basis encoding is a binary fraction, where each classical datapoint with translated to the interval $[0,\ 1)$ and represented by a τ -bit string per equation:

$$|D\rangle \ = \sum_{N=1}^{\tau} B_N \frac{1}{2^N}$$

$|\boldsymbol{D}\rangle$ represents the quantum state associated with a classical datapoint.
$\boldsymbol{\Sigma}$ represents the summation over all the bits in the binary fraction.
$\boldsymbol{B_n}$ represents each bit of the binary fraction.
$\frac{1}{2^N}$ represents the value of each bit in the binary fraction.

Classical input data which are approximated by n digits, require n qubits for its representation which can be performed by a single operation which can be prepared in linear time. This form of encoding is straightforward and intuitive but can be qubit-intensive, as it requires one qubit for each classical bit. It is most effective for problems where the binary nature of the data is directly relevant to quantum computation. The amplitudes of each basis state do not encode any information of the classical data but ensure probabilities of the qubit equal one. [2, 4] Example of basis encoding:

$$\vec{x} = (0.1, -0.6, 1.0)$$

Convert $\vec{x}$ to a binary string, where the binary fraction is $\tau = 4$ (first bit is the sign). $\vec{x}$ then maps to:

$$b = [00001\ 11001\ 01111]$$

$$\psi = |00001\ 11001\ 01111> [2]$$

$\vec{x}$ in terms of b and φ:
$\vec{x}$ = [(-1)^φ * (0.0000), (-1)^φ * (0.1100), (-1)^φ * (0.1000)]

Where:
φ is the sign bit, which is 0 for a positive sign, and (-1)^φ is 1 in this case. Each component is multiplied by (-1)^φ to account for the sign.

So, in terms of b and φ:

$\vec{x}$ = [1 * (0.0000), 1 * (0.1100), 1 * (0.1000)]

$\vec{x}$ = [0.0000, 0.1100, 0.1000]

This is $\vec{x}$ expressed in terms of the binary string b and the sign bit φ.

B. Amplitude Encoding

Amplitude or wavefunction encoding maps classical data into the amplitudes of a quantum state. A classical vector of N real numbers can be encoded into the amplitudes of a quantum state of (N) qubits.

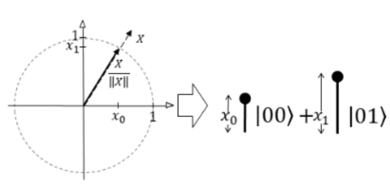

Fig. 5. Amplitude Encoding [5]

This method associates classical data as a real normalized classical vector with its complex amplitudes:

$$x \in C^{2^n}, \quad \sum_k |x_k|^2 = 1$$

These amplitudes can be encoded as quantum states:

$$\vec{x} \Leftrightarrow |\psi_x\rangle = \sum_{j=1}^{2n} x_j \ |j\rangle$$

This method is highly space-efficient, allowing exponential compression of classical data. However, it can be challenging to implement due to the complexity of preparing states with precise amplitude configurations and the difficulty in extracting information from the amplitudes. [6]

C. Angle Encoding

Angle or qubit encoding uses the angles of quantum gates to encode data. Each data point modifies the parameters of a quantum gate, such as rotation angles, effectively encoding the information into the quantum state. This method is less qubit-intensive than basis encoding and can encode multiple data points into a single qubit through different quantum gates. However, the complexity of the quantum circuit can grow rapidly with the size of the data. [5-7,9]

$$|\psi\rangle = \begin{pmatrix} \cos x_0 \\ \sin x_0 \end{pmatrix} \otimes \begin{pmatrix} \cos x_1 \\ \sin x_1 \end{pmatrix} \otimes \ldots \otimes \begin{pmatrix} \cos x_n \\ \sin x_n \end{pmatrix}$$

Fig. 6. Angle Encoding [5]

D. Quantum Random Access Memory (QRAM)

QRAM is theoretically used to encode a superposition of data values at once as opposed to the last three methods. [2,5] After encoding, these quantum states are sensitive to environmental noise leading to decoherence rendering the data useless by a quantum device. The solution would be to place the encoded quantum states onto classical random-access memory (RAM). Classical memory structures cannot store quantum states, as they require a measurement, thus collapsing of the wave function, again rendering the data useless on a quantum device. [11,12]

QRAM mitigates decoherence by encoding noise resistant data this allowing the quantum encoded data to be stored and retrieved with acceptable error. [13]

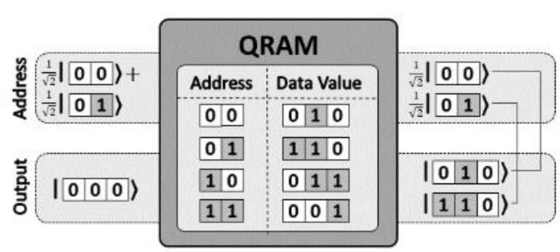

Fig. 7. QRAM [5]

Due to the importance of QRAM, current QRAM Implementation will be reviewed briefly here. The *Bucket Brigade QRAM* design utilizes some clever physics to set up its quantum memory paths. It encodes address values in photons, which are particles of light, that get sent through the system one-by-one. These photons encounter tiny, trapped atoms along the way, which can absorb the photon and become excited into higher energy states. There are specially designed left and right excited states that can steer the next incoming photon on a left or right path, like a railroad switch directing trains. By having the photon states encode the address, these atoms tracing out a branching path can guide the photons to locate the desired quantum memory cell. Once there, the stored data gets transmitted back along that path out to the user.

Another approach called the *Fanout QRAM* also sets up routing paths but gives each address bit photon many options to propagate down by rapidly splitting into multiple photon copies. Atoms help copy and direct these photons by tweaking their polarization, which is the orientation of their light wave vibration. The phase gate version instead has microwave cavities that house more helper photons to point the way. In both cases sort of a winner-take-all playoff bracket gets set up so only one complete path lights up for the data retrieval.

The *Flip-Flop QRAM* is not as visually intuitive, using quantum computing circuitry instead. But the key idea is that data gets encoded in rotation operations applied to the quantum data bits, with carefully choreographed address-controlled corrections to cancel things out. This lets them impress the right data angle encodings in sequence while preserving overall quantum coherence. [11]

III. Comparative Analysis: Applicability & Challenges

A. Basis encoding is most effective for problems where binary data plays a crucial role. Basis embedding is suitable for algorithms where the distinction between 0 and 1 states is directly relevant to the computation. Some use cases are cryptography, binary classification tasks, and simulations that require binary input. While conceptually straightforward, this method often fails to exploit the full quantum computational advantage due to its quasi-classical nature and may not effectively harness quantum phenomena such as superposition and entanglement.

A realized application of binary encoding are variational quantum algorithms that solve unconstrained binary optimization. This problem inputs n_c classical datapoints which are encoded on O(logn_c) qubits. This provides an increase in correlations among the classical datapoints captured by a variational quantum basis state. [15]

B. Amplitude encoding is most beneficial for encoding large numerical datasets due to its space efficiency. Also useful for datasets where parameters can be effectively translated into rotation angles of quantum gates. Some use cases including algorithms that involve parameter tuning such as certain types of neural networks and optimization problems. [7, 14]

A realized implementation of amplitude encoding is Approximate Amplitude Encoding (AAE). This specialized encoding method has been applied to compute the singular value decomposition entropy, a financial market indicator, of stock market classical datasets. [16]

C. Angle encoding is useful for datasets where parameters can be effectively translated into rotation angles of quantum gates. Some use cases including algorithms that involve parameter tuning such as certain types of neural networks and optimization problems.

D. QRAM is theoretically suitable for scenarios requiring the handling of large datasets, offering a way to encode and retrieve vast amounts of data efficiently. Uses of QRAM include large data analysis, pattern recognition in large datasets, and complex computational tasks that require access to a broad range of data points.[17]

Many researchers regard the realization of cheap, large scale, high quality QRAM is unlikely. Passively corrected quantum memory and highly complex, high-fidelity ballistic computation. [17]

One recent paper discusses a concept in physics known as "gentle measurement." In quantum mechanics, any measurement of a qubit is an inherently destructive process, which would disallow recovery of classical ram structures. Researchers have found that a measurement applied to a qubit necessarily causes a collapse if and only the outcome of that observation is highly unpredictable. [18]

IV. Conclusions

Quantum Machine Learning presents a novel approach to handling and processing data, offering potential breakthroughs in computational capabilities. The key to leveraging these capabilities lies in the effective encoding of classical data into quantum states. Many quantum algorithms using n datapoints assume encoding with complexity O ($log_2 n$) which provides an exponential saving in qubit space at the cost of an exponential increase in computational time. In reality, this quantum state would take time O(2^n) to prepare for computation.[14]

As this survey highlights, different encoding techniques offer varied advantages and challenges, making the choice of encoding strategy a critical decision in the design of QML algorithms. The future of QML is promising, with ongoing research and development expected to address current limitations and unlock the full potential of quantum computing in machine learning. Quantum Machine Learning represents a frontier in computational technology, blending the principles of quantum mechanics with the versatility of machine learning. This survey underscores the critical role of data encoding in harnessing the potential of quantum computing for machine learning applications. [8]

V. References

[1] Biamonte, Jacob, et al. “Quantum Machine Learning.” Nature, vol.549,Sept.2017,pp.195–202.NASAADS, https://doi.org/10.1038/nature23474.

[2] Schuld, Maria, and Francesco Petruccione. “Supervised Learning with Quantum Computers.” Springer Cham, 2018. Springer Nature Switzerland AG, Doi: HTTPs://doi.org/10.1007/978-3-319-96424-9.

[3] Preskill, John. “Quantum Computing in the NISQ Era and Beyond.” Quantum,vol. 2, Aug. 2018, p. 79. arXiv.org, https://doi.org/10.22331/q-2018-08-06-79.

[4] Nielsen, Michael A., et al. “Quantum Computation and Quantum Information.” American Journal of Physics, vol. 70, no. 5, May 2002, pp. 558–59. *DOI.org*, https://doi.org/10.1119/1.1463744.

[5] Vedral, Vlatko, et al. “Quantum Networks for Elementary Arithmetic Operations.” Physical Review A, vol. 54, no. 1, July 1996, pp. 147–53. APS, https://doi.org/10.1103/PhysRevA.54.147.

[6] Weigold, Manuela, et al. “Expanding Data Encoding Patterns for Quantum Algorithms.” 2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C), 2021, pp. 95–101. IEEE Xplore, https://doi.org/10.1109/ICSA-C52384.2021.00025. https://hillside.net/plop/2020/papers/weigold.pdf

[7] LaRose, Ryan, and Brian Coyle. “Robust Data Encodings for Quantum Classifiers.” Physical Review A, vol. 102, no. 3, Sept. 2020, p. 032420. APS, https://doi.org/10.1103/PhysRevA.102.032420.

[8] Leymann, Frank, and Johanna Barzen. “The Bitter Truth about Gate-Based Quantum Algorithms in the NISQ Era.” Quantum Science and Technology, vol. 5, no. 4, Sept. 2020, p. 044007. iopscience.iop.org, https://doi.org/10.1088/2058-9565/abae7d.

[9] Preskill, John. “Why I Called It ‘Quantum Supremacy.’” Quanta Magazine, 2 Oct. 2019, https://www.quantamagazine.org/john-preskill-explains-quantum-supremacy-20191002/.

[10] V. Giovannetti, S. Lloyd, and L. Maccone, “Quantum random access memory,” Physical Review Letters, vol. 100, no. 16, p. 15, 2008.

[11] Phalak, Koustubh, et al. “Quantum Random Access Memory for Dummies.” Sensors, vol. 23, no. 17, Aug. 2023, p. 7462. DOI.org, https://doi.org/10.3390/s23177462.

[12] Schlosshauer, M. “Quantum decoherence” Phys. Rep. 2019, 831, 1–57.

[13] Hann, C.T.; Lee, G.; Girvin, S.; Jiang, L. “Resilience of quantum random access memory to generic noise”. PRX Quantum 2021,2, 020311.

[14] Kieferová, Mária, and Yuval R. Sanders. “Assume a Quantum Dataset.” *Harvard Data Science Review*, Jan. 2022. *DOI.org (Crossref)*, https://doi.org/10.1162/99608f92.69c5328d.

[15] Tan, Benjamin, et al. “Qubit-Efficient Encoding Schemes for Binary Optimisation Problems.” *Quantum*, vol. 5, May 2021, p. 454. *arXiv.org*, https://doi.org/10.22331/q-2021-05-04-454.

[16] Nakaji, Kouhei, et al. “Approximate Amplitude Encoding in Shallow Parameterized Quantum Circuits and Its Application to Financial Market Indicator.” *Physical Review Research*, vol. 4, no. 2, May 2022, p. 023136. *arXiv.org*, https://doi.org/10.1103/PhysRevResearch.4.023136.

[17] Jaques, Samuel, and Arthur G. Rattew. *QRAM: A Survey and Critique*. arXiv:2305.10310, arXiv, 17 May 2023. *arXiv.org*, https://doi.org/10.48550/arXiv.2305.10310.

[18] Aaronson, Scott, and Guy N. Rothblum. “Gentle Measurement of Quantum States and Differential Privacy.” Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, ACM, 2019, pp. 322–33. DOI.org, https://doi.org/10.1145/3313276.3316378.