Quantum Reservoir Computing with Correlation Matrix

Gilberto Juárez Rangel
(Dated: December 24, 2024)

I. INTRODUCTION

Quantum Reservoir Computing (QRC) is an emerging computational paradigm that integrates quantum mechanics with reservoir computing, a machine learning framework designed to simplify the training of recurrent neural networks. In QRC, quantum systems serve as reservoirs, leveraging quantum properties such as superposition and entanglement to process information in high-dimensional spaces [1]. This approach has shown promise in handling complex temporal and non-linear data tasks, including time series prediction and chaotic system modeling [2, 3].

Recent advancements have demonstrated the scalability of QRC, with implementations utilizing analog quantum computers to process data efficiently [4]. For instance, researchers have developed scalable quantum reservoir learning algorithms that harness the dynamics of neutral-atom analog quantum computers, achieving competitive performance across various machine learning tasks, including binary and multi-class classification, as well as time-series prediction.

The application of QRC to binary classification tasks is particularly noteworthy. By embedding input data into high-dimensional quantum states, QRC systems can effectively separate data into distinct categories, enhancing classification performance. Studies have shown that QRC can outperform classical methods in binary classification by exploiting quantum correlations to capture intricate patterns within the data.

Implementations of QRC have been explored using various quantum platforms. For example, quantum optical reservoirs composed of two-level atoms within optical cavities have been proposed, offering improved memory retention and nonlinear processing capabilities. Such systems have been evaluated for tasks like time-series prediction and waveform classification, demonstrating significant performance enhancements with increased numbers of atoms [5].

The impact of QRC extends across multiple domains, including finance, materials science, and biomedical applications, where efficient and accurate classification is essential. Its robustness against noise and potential for energy-efficient computation make QRC a promising tool for developing sustainable and scalable machine learning solutions. However, challenges such as decoherence and optimization of read-out mechanisms remain, necessitating further research to enhance the practical applicability of QRC in real-world scenarios.

II. DATA EXPLORATION AND PREPROCESSING

A. Exploration

To evaluate the efficiency of the proposed quantum reservoir computing method, the Statlog (German Credit Data) dataset was utilized, composed of 20 features, of which 7 are numerical and the remaining categorical, with a total of 1,000 rows of data. Additionally, the dataset is unbalanced, with 70% of the entries representing good clients (= 1) and 30% representing bad clients (= 2).

Before any further manipulation of the data, the dataset was examined to ensure the absence of duplicate entries and null values. Furthermore, the distribution of numerical and categorical features was visualized using histograms and bar plots. This analysis revealed two categorical features that were highly biased, with over 90% of their values concentrated in a single category. These observations are illustrated in Fig. 1.

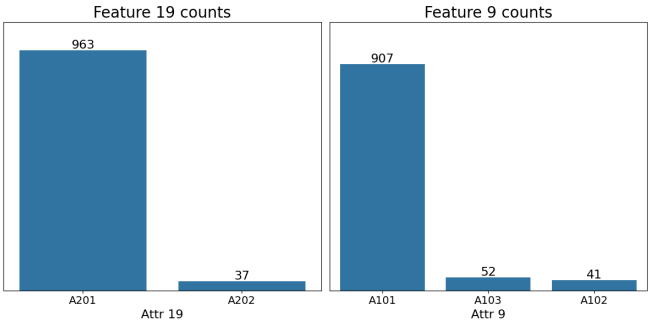

FIG. 1. Distribution of categorical features attr_9 and attr_19,

Moreover, the histogram distribution of the numerical features was plotted, as shown in Fig. 2. This revealed the presence of some biased data, which could be adjusted to approximate a normal distribution through transformations such as logarithmic, square root, or Box-Cox. Among the numerical features, attributes 1, 4, and 12 were identified as suitable candidates for correction using these transformations. The adjustments for these features will be applied during the preprocessing stage.

In Fig. 3, the kernel density estimate between the

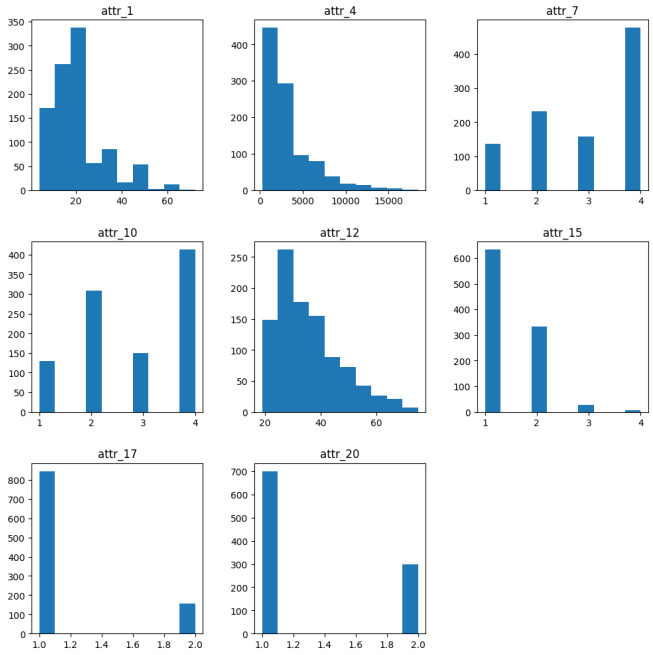

FIG. 2. Distribution of numerical features

features and the target is plotted. From the pair plots, it can be observed that the relationships between features within each category exhibit similar behavior. Moreover, the diagonal of the plot highlights that the distribution of each attribute is largely uniform across categories, further emphasizing the challenge of correctly classifying this dataset.

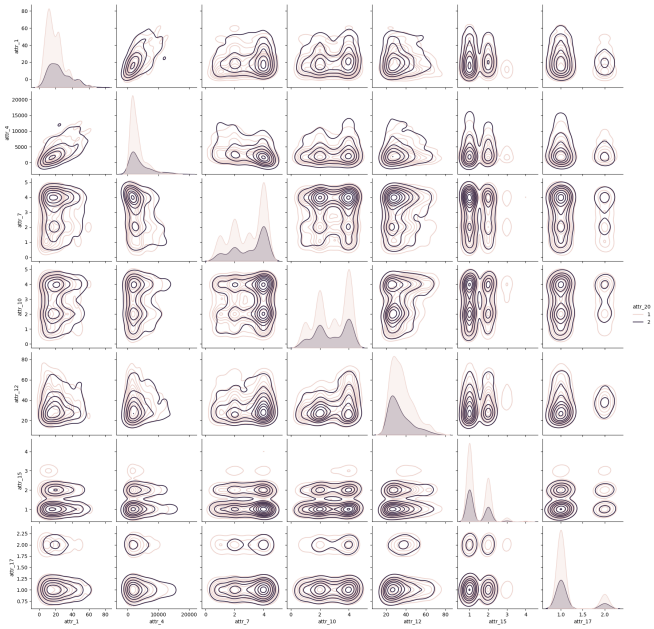

FIG. 3. Kernel density estimation pairplot between features for each category.

B. Preprocessing

Having identified key characteristics of the dataset, we proceeded to remove, split, and transform the data. During the removal stage, features 9 and 19 were eliminated, as they were highly biased and did not provide sufficient information for the model. Additionally, the interquartile range (IQR) method was applied for outlier detection, identifying 147 rows as containing outliers. Before removing these rows, we verified that they did not significantly affect the minority category of bad clients. After this verification, removing the 147 rows resulted in the proportion of bad clients decreasing from 30% to 26.38%.

Before transforming the dataset, it was necessary to split it into training and test subsets. This step is crucial to avoid the risk of data leakage. The splitting was performed with a test size of 40% and using stratification to ensure that both the training and test subsets maintained the same proportion of good and bad clients.

As previously mentioned, features 1, 4, and 12 from the training split were transformed to approximate a normal distribution. Among these, only feature 4 achieved a p-value greater than 0.05 after transformation, with a p-value of 0.306 following a logarithmic transformation. For a more detailed summary of the transformations and results, refer to Table II B.

	Attribute		
Transformation	1	4	12
None	1.3e-38	1e-86	8.44e-30
Log	1.6e-4	0.31	7.84e-09
Square Root	1.81e-07	1.42e-07	2.16e-07
Box-Cox	2.57e-06	0.25	1e-12

TABLE I. P-value of the normal test for attributes 1, 4 and 12.

Finally, after applying the transformations, a Min-Max scaling was performed to ensure all features and target were normalized to a range between 0 and 1. This step prevents subsequent algorithms from prioritizing features with larger values. As before, the scaling transformation was fitted to the training set, and the same scaling function was applied to the test set to avoid data leakage.

III. PROPOSITION OF QRC

For binary classification, Quantum Reservoir Computing (QRC) is typically implemented using a feature map $\mathcal{F}$ and a static quantum operation known as the reservoir $\mathcal{R}$. The feature map $\mathcal{F}(x)$, where x represents a sample

from the training dataset, is defined as follows:

$$\mathcal{F}(x) = \bigotimes_{i=1}^{N} Ry_i(x_i \times \pi) \tag{1}$$

where x_i is the i-th feature value of the sample x, and $Ry_i(\theta)$ denotes the rotation around the y-axis of the Bloch sphere for the i-th qubit, given by $e^{-i\frac{\theta}{2}\sigma_y}$, where σ_y is the Pauli operator. Furthermore, the reservoir is represented by m Hamiltonian time evolutions, where $0 < m \leq$ the number of features. Each step $\hat{H}i$ is represented by combinations of $\sigma x_j \otimes \sigma_{x_k}$ for odd i and $\sigma_{y_j} \otimes \sigma_{y_k}$ for even i, where k represents the i-th strongest correlation of feature j. Each combination is accompanied by a weight ω_j, which is the normalized correlation of feature i with all other features, excluding itself.

$$\hat{H}_i = \sum_{j=1}^{N} \omega_j \sigma_{x_j} \otimes \sigma_{x_k}, \qquad i = 1, 3, 5, ... \tag{2}$$

$$\hat{H}_i = \sum_{j=1}^{N} \omega_j \sigma_{y_j} \otimes \sigma_{y_k}, \qquad i = 2, 4, 6, ... \tag{3}$$

$$\tag{4}$$

Additionally, if both features j and k exhibit the strongest mutual correlation, only one of them will be considered. Moreover, note that for each H_i, all operations commute, allowing them to be represented by the two-qubit gates $R_{xx}(\theta) = e^{-i\frac{\theta}{2}\sigma_x \otimes \sigma_x}$ and $R_{yy}(\theta) = e^{-i\frac{\theta}{2}\sigma_y \otimes \sigma_y}$. Since we prioritize the strongest relationships first, the evolution can be expressed as follows:

$$\hat{U}^{(m)}(t) = e^{-i\hat{H}_m t} ... e^{-i\hat{H}_2 t} e^{-i\hat{H}_1 t}. \tag{5}$$

For each step, the evolution time is set to $t = \pi$. Since the dataset includes both categorical and numerical data, the correlation matrix will be computed using distinct correlation metrics. For numerical-numerical features, we will apply Pearson correlation; for categorical-categorical features, we will use Cramér's V correlation; and for numerical-categorical features, we will utilize the point-biserial correlation.

To better illustrate how the proposed method proceeds, we will use a sample feature correlation matrix. Suppose we have 4 features, and the correlation matrix is as follows (with the diagonal values removed, as the strongest relationship is always with itself).

$$corr = \begin{pmatrix} 0 & 0.6 & 0.4 & 0.1 \\ 0.6 & 0 & 0.7 & 0.5 \\ 0.4 & 0.7 & 0 & 0.8 \\ 0.1 & 0.5 & 0.8 & 0 \end{pmatrix} \tag{6}$$

The next step is to normalize the values by rows, ensuring that the sum of all values in each row equals 1. As observed, the matrix is symmetric; however, if the strongest correlation for feature i is with feature j, it does not necessarily imply that the strongest correlation for feature j is with feature i.

Therefore, the first Hamiltonian $\hat{H}_1$ is represented as:

$$\hat{H}1 = 0.6\sigma y_1 \otimes \sigma_{y_2} + 0.7\sigma_{y_2} \otimes \sigma_{y_3} + 0.8\sigma_{y_3} \otimes \sigma_{y_4} \tag{7}$$

Notice that features 3 and 4 are mutually the strongest correlations; thus, the term is included only once. Finally, the X matrix is obtained by calculating the expected value of the σ_z Pauli operator for each qubit. For m data points, the matrix is structured as follows:

$$X = \begin{pmatrix} \langle\sigma_{z_{1,1}}\rangle & \langle\sigma_{z_{1,2}}\rangle & \dots & \langle\sigma_{z_{1,18}}\rangle \\ \langle\sigma_{z_{2,1}}\rangle & \langle\sigma_{z_{2,2}}\rangle & \dots & \langle\sigma_{z_{2,18}}\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle\sigma_{z_{m,1}}\rangle & \langle\sigma_{z_{m,2}}\rangle & \dots & \langle\sigma_{z_{m,18}}\rangle \end{pmatrix} \tag{8}$$

Here, $\sigma_{z_{i,j}}$ represents the expected value of σ_z for the i-th quantum circuit, and j denotes the j-th qubit in the circuit. These expected values can be approximated simultaneously by measuring all M times and calculating the marginal probability for each qubit:

$$\sigma_{z_{i,j}} = P_i(0) - P_i(1), \tag{9}$$

where $P_i(j)$ is the probability of measuring j at the i-th qubit. After constructing the X matrix, a linear regression is performed to best fit the y_{target}. The predictions are calculated as:

$$y_{pred} = X\omega, \tag{10}$$

where the weights ω minimize $||y_{target} - y_{pred}||$ using the pseudo-inverse matrix:

$$\omega = (X^T X)^{-1} X^T y_{target}. \tag{11}$$

Since the dataset is unbalanced, using a threshold of 0.5 to classify y_{pred} values as either good or bad customers would result in poor performance. Instead, the optimal threshold is determined by maximizing the $F1$-score. Using recall alone is not ideal, as it would bias the threshold toward predicting all bad customers correctly, which is not the desired outcome.

IV. REAL HARDWARE SIMULATION

To ensure the feasibility of the proposed method, we evaluate the circuit depth at each step and compare the expected values across different numbers of layers using the Mean Squared Error (MSE). As shown in Fig. **??**, the circuit depth increases linearly with the number of steps. This observation aligns with expectations, as each step

involves an equal or lesser number of two-qubit gates, contributing proportionally to the overall circuit depth.

Since running circuits on real quantum hardware can be time-consuming, primarily due to queue times, we instead execute the circuits on a simulator. In Qiskit, this is achieved by utilizing noise models and simulation backends. First, the logical quantum circuit is transpiled onto the physical qubits of the hardware, and then it is further transpiled into the noise simulator. As shown in Fig. **??**, the Mean Squared Error (MSE) increases significantly with the number of steps, highlighting the impact of noise on circuit performance.

V. RESULTS

From the previous section, we observed that after *NUMBER* steps of correlations, the noise exceeds *NUMBER*. Therefore, we will only consider up to *NUMBER* steps for the QRC. This analysis is conducted in a noise-free simulation for simplicity. While running on actual quantum hardware might not take excessively long, noisy simulations require significantly more time to execute.

In Fig. **??**, we observe both recall and accuracy of predictions across different numbers of steps. At $\#n$ steps, the performance metrics indicate improved results, suggesting an optimal balance between accuracy and recall reliability.

VI. CONCLUSION

In this work, we proposed a novel Quantum Reservoir Computing (QRC) approach tailored to binary classification tasks, integrating feature correlations into the reservoir's design. By leveraging Hamiltonian time evolutions based on feature relationships, the method aims to encode both numerical and categorical data effectively within a quantum framework. The feature map and reservoir construction were carefully designed to ensure efficient representation of input data while keeping circuit depth manageable. Preliminary analysis suggests that balancing circuit complexity with noise sensitivity is critical to achieving optimal performance. While the proposed method demonstrates potential for improving classification tasks, further exploration is needed to evaluate its robustness under realistic noise models and its scalability to larger datasets. The results from this study could provide valuable insights into the feasibility of QRC for practical machine learning applications, paving the way for future advancements in quantum-enhanced data analysis.

[1] K. Fujii and K. Nakajima, Phys. Rev. Appl. **8**, 024030 (2017).

[2] C. Zhu, P. J. Ehlers, H. I. Nurdin, and D. Soh, Practical and scalable quantum reservoir computing (2024), arXiv:2405.04799 [quant-ph].

[3] K. Nakajima, Japanese Journal of Applied Physics **59**, 060501 (2020).

[4] M. Kornjača, H.-Y. Hu, C. Zhao, J. Wurtz, P. Weinberg, M. Hamdan, A. Zhdanov, S. H. Cantu, H. Zhou, R. A. Bravo, K. Bagnall, J. I. Basham, J. Campo, A. Choukri, R. DeAngelo, P. Frederick, D. Haines, J. Hammett, N. Hsu, M.-G. Hu, F. Huber, P. N. Jepsen, N. Jia, T. Karolyshyn, M. Kwon, J. Long, J. Lopatin, A. Lukin, T. Macrì, O. Marković, L. A. Martínez-Martínez, X. Meng, E. Ostroumov, D. Paquette, J. Robinson, P. S. Rodriguez, A. Singh, N. Sinha, H. Thoreen, N. Wan, D. Waxman-Lenz, T. Wong, K.-H. Wu, P. L. S. Lopes, Y. Boger, N. Gemelke, T. Kitagawa, A. Keesling, X. Gao, A. Bylinskii, S. F. Yelin, F. Liu, and S.-T. Wang, Large-scale quantum reservoir learning with an analog quantum computer (2024), arXiv:2407.02553 [quant-ph].

[5] J. Dudas, B. Carles, and E. Plouet, npj Quantum Inf **9**, 10.1038/s41534-023-00734-4 (2023).