VQC Class Implimentation using Estimator QNN

A Project Report Submitted by

Shisheer S Kaushik

Mohammad Haris Ansari

Thirumalai M

Harsh Mehta

Indian Institute of Technology Jodhpur

IDRP-QIC

May, 2024

Contents

1 Introduction

Quantum machine learning (QML) is a rapidly growing area that combines quantum computing principles with classical data analysis methods. This integration has the potential to improve computational efficiency and reveal novel insights. The objective of our project is to investigate the merging of quantum algorithms with classical datasets in order to enhance classification accuracy.

We began our journey by choosing binary datasets, which formed the basis for our exploration. In order to establish a standard for comparison, we utilized traditional algorithms to evaluate the initial levels of accuracy. Afterwards, we explored the use of the Variational Quantum Classifier (VQC) class, carefully adjusting parameters such as entanglement types and Ansatz repetitions to determine their impact on classification performance.

Simultaneously, we worked on creating personalized feature maps and Ansatz designs, utilizing the adaptability of Estimator Quantum Neural Networks (QNN) to customize classification models according to the distinct attributes of our datasets. By following this repetitive process, our goal was to clarify the possibility of quantum-inspired approaches in improving the accuracy of classification and the efficiency of computation.

1.1 Data Sets

Two Binary Data sets are utilized for Classification purpose.

1.1.1 Citrus Data Set

The dataset employed in this project mimics the task of differentiating between oranges and grapefruits by analyzing their attributes. Although this task may appear simple to humans, even the act of manually observing it can introduce a certain level of inaccuracy. In order to tackle this issue, this dataset includes crucial characteristics such as color, weight, and diameter of typical oranges and grapefruits. This results in a comprehensive dataset that encompasses a wide range of values, accurately representing both types of fruits.

The data generation process utilized a script that was initially populated with real-time measurements obtained from grapefruits and oranges. The initial values were subsequently modified using NumPy to incorporate deviations according to a normal distribution. This methodology guaranteed that the produced dataset includes a diverse set of values that accurately reflect real-life situations, facilitating thorough analysis and assessment of classification algorithms.

This dataset can be found at: https://www.kaggle.com/datasets/joshmcadams/oranges-vs-grapefruit.

1.1.2 MNIST Data Set

The dataset utilized in our project, known as MNIST [1], serves as a quintessential benchmark in the realm of image classification. While discerning between oranges and grapefruits may be intuitive for humans, distinguishing between handwritten digits presents a classic challenge in machine learning. MNIST comprises a vast collection of grayscale images depicting handwritten digits from 0 to 9, each measuring 28x28 pixels.Similar to the Citrus dataset, our MNIST dataset was processed to cater to our binary classification task. Since our focus is on binary classification, we exclusively selected digits 4 and 9 from the MNIST dataset. These digits were chosen deliberately due to their similarity in appearance, thereby posing a challenging classification task 1.1. We extract only the datapoints corresponding to digits 4 and 9 from the training dataset, which are normalized to ensure consistency in feature scaling. The normalized datapoints are shuffled to randomize the order, which helps prevent any inherent biases during model training.

This dataset can be found at: http://yann.lecun.com/exdb/mnist/.

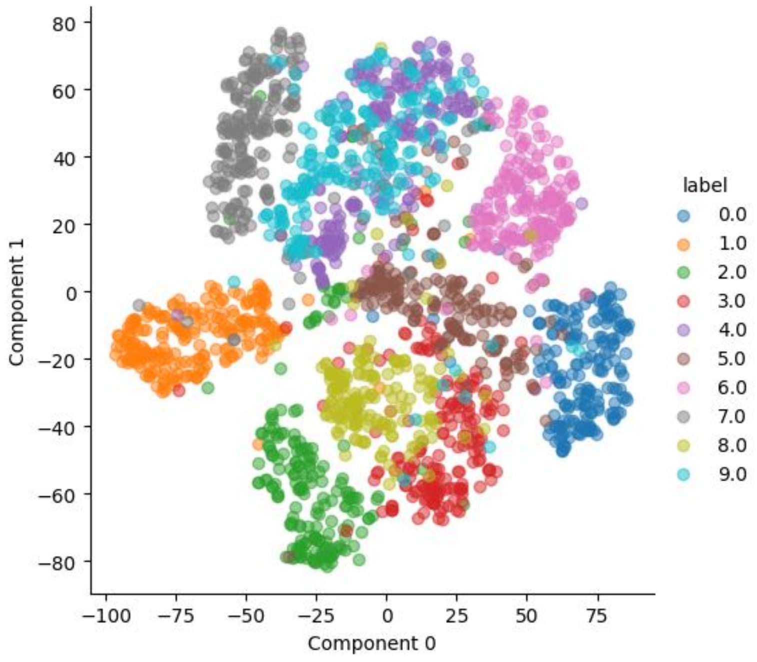

Figure 1.1: 2D scatter plot of the reduced-dimensional representation of the MNIST dataset.

1.2 Implementation and Methodology

An essential component of the re-implementation process entailed developing encoding schemes that serve as a connection between classical data and the quantum domain. The project investigated two primary methodologies: Amplitude Encoding and Angle Encoding. Amplitude Encoding is particularly effective in situations with a large number of dimensions, as it is highly proficient at representing classical data by utilizing the amplitudes of qubits in a resourceful manner. However, this method was not the most appropriate choice for the project's datasets that had a moderate number of features.

We also investigated alternative Encoding Schemes such as ZZFeatureMap and Angle encoding based custom Feature Map. However, the most favorable outcomes were achieved using the Z feature Map. Therefore, we chose to utilize the pre-existing Z feature map. The chosen approach not only led to enhanced accuracy in classification but also led to the creation of less complex quantum circuits, necessitating fewer quantum operations and resulting in faster execution times.

An ansatz is a fundamental concept that determines the configuration of a quantum circuit. It precisely outlines the order in which quantum gates are applied to the qubits. We have also created two distinct variational circuits (ansatzes). These specifically designed approaches were carefully created to accurately capture the fundamental connections (correlations) between the quantum states.

Aside from the custom-designed approaches, we also examined the effectiveness of conventional approaches such as RealAmplitude, TwoLocal, and EfficientSU2. By incorporating these established approaches, we were able to establish a useful standard for assessing the efficacy of the customized designs. After comparing the classification performance achieved using both custom and standard approaches, we observed that the custom designed approaches had higher accuracy.

In order to attain the highest level of performance from the customized Variational Quantum Classifier (VQC), we employed the Estimator Quantum Neural Network (QNN). This quantum neural network (QNN) played a pivotal role in the optimization process. It served as a guide, assisting in the precise adjustment of the parameters of the quantum circuit defined by the ansatz.

In addition, we have implemented supplementary strategies with the specific objective of improving the classification outcomes. This involved integrating both backward and forward propagation methods. In addition, we evaluated the performance of the classification model in practical situations by testing it on a noisy simulator. This allowed us to assess how effectively the custom VQC would handle real-world imperfections.

2 Variational Quantum Classifier (VQC)

Quantum systems have the ability to efficiently represent and process information in an exponential manner compared to classical systems by utilizing quantum phenomena like entanglement and superposition. This allows for Variational Quantum Computing (VQC) to effectively handle large and intricate datasets using fewer computational resources which could result in more resilient and expressive feature representations[2]. This also enables Quantum circuits to simultaneously process multiple computational paths, leveraging quantum parallelism to explore a vast solution space more effectively during optimization. Variational Quantum Classifier (VQC) is a quantum-inspired classification algorithm designed to categorize input data into predefined classes or categories. Unlike classical classifiers that rely solely on classical computation, VQC harnesses the computational power of quantum systems to perform classification tasks more efficiently and potentially with higher accuracy. The Key components of VQC Include:

- **Quantum Feature Maps:** These maps encode classical data features into quantum states, by encoding which typically involves mapping classical data features onto quantum states, effectively representing the input data in a quantum mechanical framework.

- **Parameterized Quantum Circuits (Ansatz):** The encoded quantum state undergoes a series of quantum operations, parameterized by a set of adjustable parameters known as variational parameters (Ansatz). These parameters are iteratively adjusted to minimize a predefined cost function, often based on the discrepancy between the predicted and actual labels of the input data.

- **Variational Optimization Algorithms:** VQC employs variational optimization algorithms, such as gradient descent or variational quantum algorithms, to iteratively adjust variational parameters and optimize classification performance. After optimization, the quantum circuit is measured to extract classical information, which is subsequently used to make classification decisions. The outcome of the measurement provides probabilistic information about the class membership of the input data, allowing for classification based on the highest probability outcome.

Quantum systems have the ability to efficiently represent and process information in an exponential manner compared to classical systems by utilizing quantum phenomena like entanglement and superposition. This allows for Variational Quantum Computing (VQC) to effectively handle large and intricate datasets using fewer computational resources which could result in more resilient and expressive feature representations. This also enables Quantum circuits to simultaneously process multiple computational paths, leveraging quantum parallelism to explore a vast solution space more effectively during optimization.

3 Data Encoding Schemes

3.1 Amplitude Encoding Scheme

The Amplitude Encoding Scheme operates by translating classical data into a quantum state through a systematic process. This scheme involves mapping the binary representation of classical data onto the amplitudes of a quantum state. Each classical data point is encoded as a specific amplitude of the quantum state, with the relative magnitudes of these amplitudes corresponding to the probability of measuring each data point. Quantum gates and operations are then applied to manipulate the quantum state and perform computations on the encoded data.

3.2 Angle Encoding Scheme

The Angle Encoding Scheme operates by representing classical data as angles in a quantum state, facilitating its processing on quantum computers. This encoding mechanism involves mapping the features of classical data onto the rotation angles of quantum gates, typically single-qubit rotations. Each feature of the classical data corresponds to a specific angle, with the magnitude of the angle encoding the magnitude or significance of the corresponding feature. Quantum gates, such as rotation gates (e.g., RX, RY, RZ), are then applied to the qubits in the quantum circuit, with the rotation angles determined by the encoded classical data.

3.3 ZFeatureMap :

ZFeatureMap is a subclass of pauli feauture map where the pauli strings are fixed as as ['Z']. This has been implemented in this model since we have very less feature and it shows significantly better results than all the other encoding system. Circuit depth reduces to 26 and feature map depth reduces to constant as one. And it also helps to gain better accuracy and F1 score.

4 Customized Ansatz :

In this model, customized ansatz has been used to replace variational quantum classifier function to show better performance. This generates a parameterized quantum circuit with number of qubits qubits and the desired number of repetition layers (reps). The circuit employs a ParameterVector to facilitate parameterization, making it suitable for use in variational quantum algorithms.The circuit employs a layered approach, with a specified number of repetitions. Each repetition involves a sequence of operations that include rotation gates and entanglement gates, creating a variational pattern that can be adjusted by tuning the parameters

5 Sampler QNN :

The SamplerQNN is a neural network that transforms the quasi-probabilities of the Sampler primitive into predicted classes by taking in a parametrized quantum circuit with predefined parameters for incoming data and/or weights. Used often is a mixed quantum circuit. Such a circuit consists of two circuits: an ansatz (weight parameters) and a feature map (which provides the network input parameters). In such case, a QNNCircuit can be employed as a circuit to facilitate the development of an ansatz and feature map. The input and weight parameters are obtained from the QNNCircuit, hence they are not necessary if a QNNCircuit is passed as a circuit.

6 Estimator QNN

The EstimatorQNN is a neural network that returns the expected value(s) of a parametrized quantum circuit with given parameters for input data and/or weights and optional observable(s). Used often is a mixed quantum circuit. Such a circuit consists of two circuits: an ansatz (weight parameters) and a feature map (which provides the network input parameters). In such case, a QNNCircuit can be employed as a circuit to facilitate the development of an ansatz and feature map. The input and weight parameters are obtained from the QNNCircuit, hence they are not necessary if a QNNCircuit is passed as a circuit.

7 Results and Discussions

In the case of citrus dataset of customized feature map along with customized ansatz yielded 39.21% as accuracy. Wheraeas ZZFeaturemap pushed it to 41.28%. Then ZFeaturemap showed better accuracy then all the other encoding system which is of 48.09%. Post altering the parameters like maxiter normalized to 150 and the accuracy hiked to 49.62% and F1 score 33.24% and when maxiter to 400 results 50.0% and 33.33% as F1 score. This is because in previous encoding systems there exist entanglement circuit, which increases circuit depth and leads to less in accuracy. However in ZFeaturemap there is no entanglement circuit which reduces the circuit depth to constant and increases in accuracy. Also F1 score has been increased much better in ZFeaturemap while comparing to other encoding scheme Same goes for MNIST dataset when maxiter to 300 and accuracy to 81%., accuracy and F1 score improved significantly in ZFeauturemap. Parameters like max iteration has been altered from 50 to 100 then to 400 and it results in increase in accuracy more. Please find more details of code here

7.1 Benchmarking using Classical Algorithms

7.1.1 Random Forest Classifier

The Random Forest Classifier is a powerful and versatile ensemble learning method widely used in classification tasks. It belongs to the family of decision tree algorithms. The Random Forest Classifier constructs a large number of decision trees during training. Each tree is trained on a random subset of the training data and uses a random subset of features at each node split. This randomness introduces diversity among the trees, which helps in reducing the model's variance. During prediction, each decision tree in the ensemble independently classifies the input data. The final prediction is determined by aggregating the predictions of all the trees. In classification tasks, the class with the most votes among the trees is selected as the final prediction. The Random Forest Classifier employs a technique called bagging (bootstrap aggregating), which involves sampling the training data with replacement to create multiple subsets for training individual trees. This helps in ensuring that each tree in the ensemble is exposed to different subsets of the data, leading to diverse and robust models. In our experimentation with the Random Forest Classifier using different data splits (0.6, 0.7, and 0.8), we observed that the classifier achieved the highest accuracy when the data split was set to 0.8. This indicates that providing a larger proportion of the data for training (80%) resulted in a more effective model, capable of better generalization and classification performance.

7.1.2 Decision Tree Classifier

The Decision Tree Classifier begins by selecting the most informative feature from the input dataset. It then splits the data based on the selected feature into subsets that are as pure as possible in terms of class labels. This splitting process continues recursively for each subset until a stopping criterion is met. Common stopping criteria include reaching a maximum tree depth, having a minimum number of samples in a node, or when all samples in a node belong to the same class. Once the tree is constructed, predictions are made by traversing the tree from the root node to a leaf node based on the feature values of the input data. The class label associated with the majority of samples in the leaf node is assigned as the predicted class. Similar to Random Forest Classifier, we observed that the classifier achieved the highest accuracy when the data split was set to 0.8.

7.1.3 Naives Bayes Classifier

The Naive Bayes Classifier is based on Bayes' theorem, which calculates the probability of a class given some observed features. The "naive" assumption in Naive Bayes is that all features are independent of each other given the class label. This simplifies the computation of conditional probabilities and makes the classifier computationally efficient. The classifier estimates the probabilities for each feature given each class from the training data. It also calculates the prior probabilities for each class. Given a new instance with feature values the classifier calculates the posterior probability for each class using Bayes' theorem. The class with the highest posterior probability is then assigned as the predicted class for the instance. Unlike previous two classifier, the heightest accuracy was observed when the data split was set to 0.6.

7.1.4 K-Nearest Neighbour Classifier

The K-Nearest Neighbors (KNN) Classifier is instance-based learning algorithm used for classification tasks. KNN relies on a distance metric (e.g., Euclidean distance) to measure the similarity between data points in the feature space. The choice of distance metric can significantly impact the performance of the classifier. Given a new data point, KNN identifies the K nearest neighbors to this point based on the chosen distance metric. The value of K is a hyper-parameter that needs to be specified beforehand. Once the K nearest neighbors are identified, the predicted class for the new data point is determined by a majority vote among these neighbors. The class with the highest frequency among the neighbors is assigned as the predicted class. IN this classifier, the highest accuracy was observed when the data split was set to 0.6.

7.1.5 Support Vector Classifier

The Support Vector Classifier (SVC), also known as Support Vector Machine aims to find the hyperplane that maximizes the margin between the classes in the feature space. The margin is defined as the distance between the hyperplane and the nearest data points from each class, known as support vectors. In cases where the classes are not linearly separable, SVC can employ a kernel function to map the input data into a higher-dimensional space where linear separation becomes possible. Common kernel functions include linear, polynomial, and radial basis function (RBF) kernels. Once the hyperplane is determined, SVC assigns new data points to one of the two classes based on which side of the hyperplane they fall on. Data points lying on opposite sides of the hyperplane are assigned to different classes. It was observed that the classifier achieved the highest accuracy when the data split was set to 0.8.

Table 7.1: Performance of Different Classification Models

Data Split	Model Name (Classifier)	Accuracy	F1 Score	ROC curve area	PRC Curve Area
0.6	Random Forest Classifier	95.00	95.00	1.00	0.99
	Decision Tree Classifier	93.30	93.30	1.00	0.91
	Naives Bayes Classifier	92.12	92.12	0.98	0.98
	K-Nearest Neighbors Classifier	92.12	92.12	0.96	0.95
	Support Vector Classifier	92.37	92.37	0.98	0.98
0.7	Random Forest Classifier	95.3	95.30	1.00	0.99
	Decision Tree Classifier	94.33	94.33	0.94	0.92
	Naives Bayes Classifier	91.96	91.96	0.98	0.98
	K-Nearest Neighbors Classifier	91.36	91.36	0.96	0.95
	Support Vector Classifier	92.5	92.49	0.98	0.98
0.8	Random Forest Classifier	95.65	95.65	1.00	0.99
	Decision Tree Classifier	94.39	94.40	0.94	0.92
	Naives Bayes Classifier	92.00	92.00	0.98	0.98
	K-Nearest Neighbors Classifier	91.90	91.90	0.96	0.95
	Support Vector Classifier	92.90	92.90	0.98	0.98

The superior performance of the Decision Tree Classifier with a data split of 0.8 can be attributed to its ability to utilize a larger amount of training data for constructing the decision tree. With more data available for training, the classifier can learn more complex decision boundaries, leading to improved classification accuracy 7.1.

7.1.6 VQC and Custom VQC Class

We performed a comparative analysis of the outcomes derived from the VQC and our customized re-implementation of the VQC using Estimator QNN. The comparison was performed across various dimensions, encompassing classification accuracy, F1 score, and other pertinent evaluation metrics. By implementing PCA methods to reduce the number of features from 5 to 2 and increasing maxiter with the COBYLA optimizer to 200, the accuracy of the Citrus data set can be improved.
Detailed Comparison can be found here: Google Sheets

8 Future Work and Conclusion

This project demonstrated variatioanl quantum classifier with customized ansatz and encoded dataset using customized encoding system. However ZFeaturemap significantly improved accuracy while comparing to all the other encoding mechanism. Customized ansatz need to modified thereby make the moedl more efficient. Best way is to implement dynamic circuit wherever required which in turn reduces circuit depth and move towards more accurate. COBYLA is the optimizer used in this model, including different optimizer also helps as better optimization technique.

References

[1] Y. LeCun, C. Cortes, and C. Burges, “Mnist handwritten digit database,” *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[2] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, “Supervised learning with quantum-enhanced feature spaces,” *Nature*, vol. 567, no. 7747, p. 209–212, Mar. 2019. [Online]. Available: http://dx.doi.org/10.1038/s41586-019-0980-2