Imperial College London

Quantum Kernel: Optimisation using Quantum Target Alignment

Arlind Visha - Department of Physics – MSc Quantum Dynamics

Introduction In recent times, there has been growing confidence in the use of quantum computing for machine learning tasks, despite the challenges posed by low coherence times, high error-rates, and limited numbers of qubits. Modern-day quantum computers offer a platform to test the potential of Quantum Machine Learning (QML) applications. A particularly popular and promising approach has been to employ quantum kernel estimations within a hybrid Quantum-Classical computing framework. This method possesses the potential to enhance various computations and tasks. This poster illustrates how such an algorithm is realized and looks at a method of improving accuracy using quantum target alignment [1]. While it does not show any quantum supremacy, it does show that the techniques used improve the quality of the quantum kernel estimation. As a result, we find an improvement in accuracy on a classification task using our techniques on small data sets, tested through a quantum simulator. If applied to larger systems, these methods have the potential to further improve the accuracy of the quantum kernel method. This could lead to uncovering a useful advantage sooner, opening possibilities to explore the potential benefits of quantum kernels in machine learning [2].

Support Vector Machine

Machine learning uses algorithms to detect statistical patterns in data, allowing for predictions or to produce new data with a similar statistical pattern. One of these algorithms is the Support Vector Machine (SVM). An SVM seeks to find a hyperplane that best separates classes of data, to do thi this it employs a similarity measure between all the data points which is called the kernel function, $k(x, x')$. The kernel function map's the data into a higher-dimensional feature space where the data can be linearly separated by a hyper plane and then calculates the inner-product between the data points in this feature space. The inner-product is a similarity measure. This creates a kernel matrix K with the pairwise similarity measures, this is then fed in to a SVM where an optimal hyperplane is found. Although classical kernel methods are a powerful tool, they can be computationally expensive to calculate, especially for high-dimensional data.

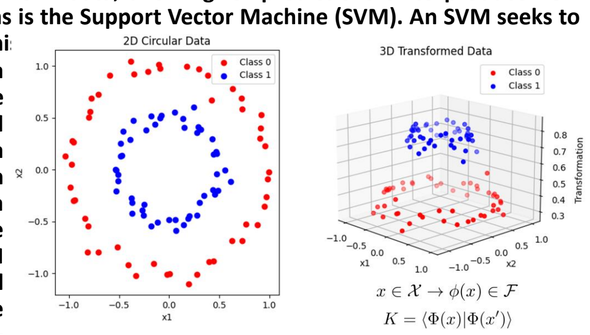


Figure 1: Kernel Transformation, demonstrating how nonlinear data in 2D space (left) can be mapped into 3D space (right) using an RBF kernel, creating a linearly separable structure.

Quantum Kernel Estimation

In quantum kernel estimation (QKE), we use a quantum computer to calculate the kernel matrix K. Quantum kernels can represent and manipulate data in a high-dimensional space, making them well-suited to problems that are classically intractable, such as handling quantum data or finding patterns in data that were not classically realised. To do this, we first embed our data onto a quantum state (if not already quantum) that lives in a 2^n-dimensional Hilbert space $\mathcal{H}$, where n is the number of qubits needed to embed the data point. Then, we need to calculate the inner product between embedded data points. Finally, we measure the value of the inner product.

Embedding the data onto a quantum state is one of the most important steps, as it defines the feature mapping we use (see Quantum feature map). Once we have the data points represented as quantum states, we can calculate the inner product using a swap test (Figure 2), where the probability of measuring the $|0\rangle$ state on the ancilla qubit is proportional to the value of the kernel (Equation 1). Since measuring the probability is statistical, we must take repeated measurements on the same data to have a good estimate of the kernel matrix K.

The kernel matrix K is then used in a hybrid quantum-classical computing framework. In this framework, the quantum computer is used to calculate the kernel matrix, while the classical computer is used to train a Support Vector Machine. This hybrid approach can achieve better accuracy than using a classical computer alone, as the quantum computer can handle the computationally expensive task of calculating the kernel matrix, while the classical computer can use its strengths in training machine learning model [3].

$$P(\text{First qubit} = 0) = \frac{1}{2}\left(\langle\phi|\langle\psi| + \langle\psi|\langle\phi|\right)\frac{1}{2}\left(|\phi\rangle|\psi\rangle + |\psi\rangle|\phi\rangle\right) = \frac{1}{2} + \frac{1}{2}|\langle\psi \mid \phi\rangle|^2 \quad (1)$$

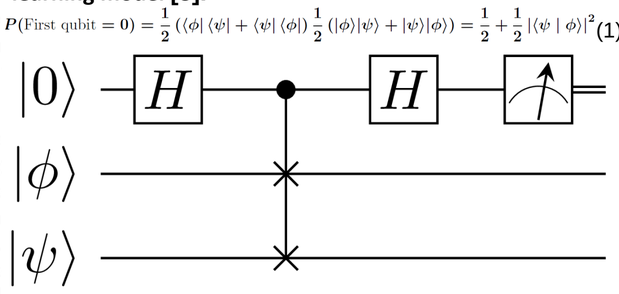


Figure 2: The Swap test, a quantum algorithm that calculates the inner product between two quantum states. The inner product is a measure of the similarity between two vectors, and it can be used to find patterns in data The swap test works by entangling two qubits and then measuring one of the qubits. The probability of measuring the qubit in the /0⟩ state is proportional to the square of the absolute value of the inner product between the two states. (Equation 1).

Quantum feature map

When mapping classical data onto a quantum state, we are transforming data from the data space $\mathcal{X}$ to a finite complex vector space known as a Hilbert space $\mathcal{H}$. This mapping follows the relation: $x \in \mathcal{X} \rightarrow \boldsymbol{\phi}(x) \in \mathcal{C}^{\mathbb{2}^n} \in \mathcal{H}$, where n represents the number of qubits needed to represent the data or the number of features the data possesses.

Quantum embedding is important, as once the data is embedded into the quantum state, we can apply quantum gates that are intractable classically. In essence, we can put the data into superposition states, entangle the data, and harness properties unique to quantum mechanics. These properties become challenging, if not impossible, to simulate for large qubit systems classically since the memory required to store the states grows exponentially with 2^n, but scales linearly with n on a quantum computer.

It remains an active area of research to discover feature mappings that show advantages in real-world applications. However, limitations such as limited qubits, noisy devices, and low coherence times make it challenging to test these applications. I have focused on a method that enhances the accuracy of a kernel using a Y rotation gate as a training parameter and a ZZFeatureMap for data embedding (Figure 3). The ZZFeatureMap utilised incorporates two encoding steps per register, with each feature of the data point being applied three times. By employing circular entanglement, the circuit becomes challenging to simulate classically (if scaled). To realize the quantum kernel matrix K, we can prepare two circuits like Figure 3, where each data point consists of 4 features (4-D data point), so each feature is $x[i]$. We can then employ the swap test, as detailed in the Quantum Kernel Estimation section to estimate the kernel values and build the kernel matrix K [4].

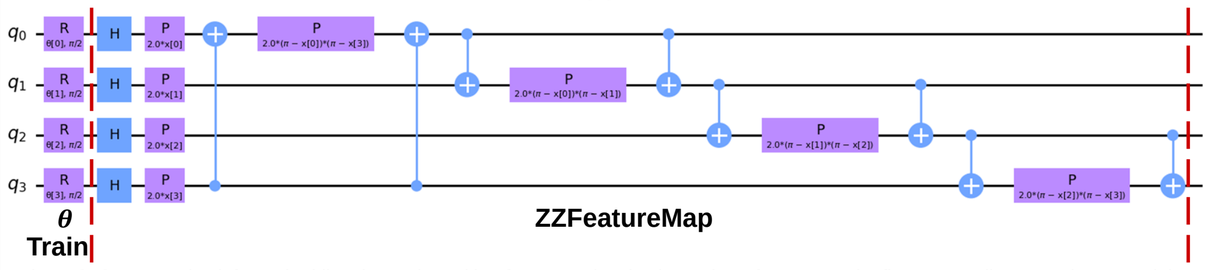


Figure 3: Quantum circuit for embedding data points with 4 features. The circuit consists of two parts: the first part applies a rotation around the Y axis (R - Rotation gate) to each qubit, governed by the training parameter θ, and the second part is a ZZFeatureMap with 1 repetition and circular entanglement. In this setup, each feature has its own training parameter within the vector θ. In the ZZFeatureMap, the data of the features are encoded as shown by applying a phase around the Z axis (P –Phase gate). Figure made using Qiskit.

Quantum Target Alignment

Finding the right model for the quantum feature map is a difficult task. One method that could improve how the model performs is by using quantum target alignment (QTA). QTA measures the similarities between two kernel matrices and is given by:

$$\mathrm{KA}\left(K_1, K_2\right) = \frac{\mathrm{Tr}\left(K_1 K_2\right)}{\sqrt{\mathrm{Tr}\left(K_1^2\right)\mathrm{Tr}\left(K_2^2\right)}} \quad (2)$$

In the context of quantum kernels, K_1 is the kernel matrix which is calculated using the training data on a quantum kernel circuit (with random parameters) and K_2 is the "ideal" kernel function, which is the outer product of the vector with the training labels y. This returns a matrix where labels of the same class are given a value 1 and labels with different classes are labelled -1. So, the QTA is defined by the alignment of the quantum kernel matrix with the training data with the kernel yy^T. It is only a necessary but not sufficient condition for a good kernel. This means that a kernel with a high QTA value may not always perform well on unseen data.

In the quantum circuit, we have defined different parameters $\theta_0, \theta_1, \theta_2, \theta_3$, one for each feature. By adjusting these parameters, we can try to maximize $KA(K_1, K_2)$. One method of doing this is a parameter search through $0 - 2\pi$, but for 4 different parameters, this is computationally expensive as we would need $n\text{^}4$ evaluations for all combinations assuming we vary θ_i by $2\pi/n$. Instead, we use a stochastic gradient ascent method. By finding the gradient of $KA(K_1, K_2)$ using the finite difference method, we can adjust the parameters to maximize the QTA [1], [5].

Method – Workflow

Quantum kernel classification using the CIFAR-10 data sets with images of cats and dogs. Firstly, I pre-processed the data reading the features to 4 using Principal Component Analysis PCA [6] to match the number of quantum registers, defined in the quantum feature map. I used 20 data points and ran the simulation using IBM's Qiskit Python Library. A workflow of the process is shown below.

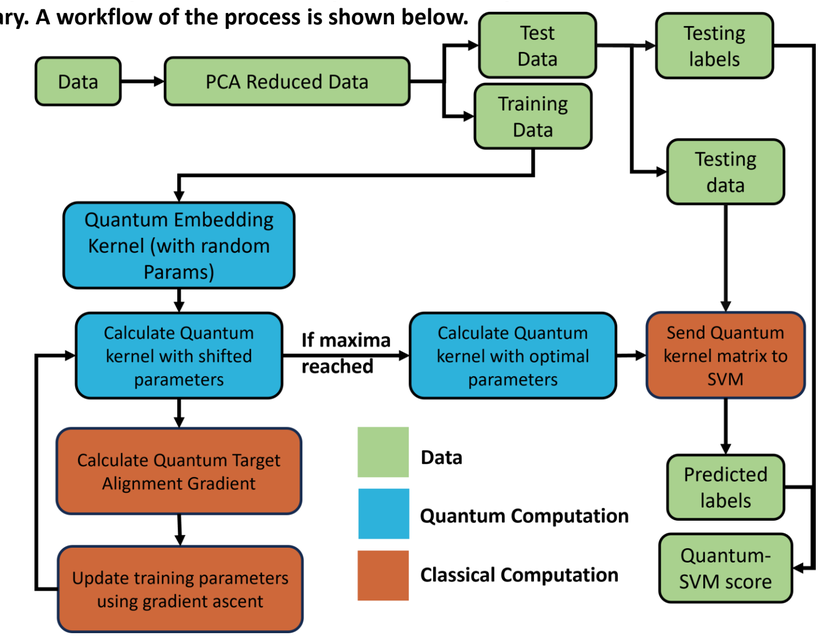
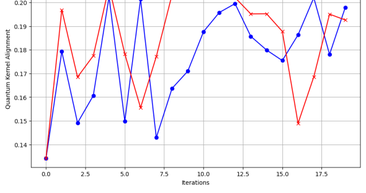
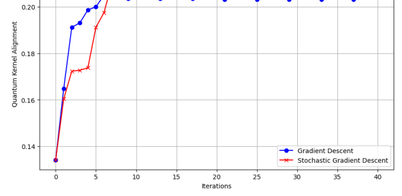


Figure 4: Quantum – Support Vector Machine workflow chart: Green boxes represent data, blue boxes are quantum computations and orange boxes are classical computations. It starts with a reduction of features of the data to match the qubits in the simulation, this is done using Principal Component Analysis PCA [6]. Then the data is split into training and testing data, the training data is then fed into the quantum kernel and a kernel matrix K is calculated. The training parameters are then shifted by $+1e^{-6}$, and the Quantum Kernel Alignment (QKA) is calculated with the new kernel matrix is calculated. This then allows the calculation of the gradient of the QKA, where we update the value of the training parameter θ using gradient ascent. A stochastic approach is used where, only two out of four of the θ_i parameters is updated at a time. This has reduced the computational time and θ converges to a maximum faster. Once a maxima has been reached the kernel with the optimised thetas is then sent to a SVM where classification takes place, the SVM then predicts the labels given test data, this is compared with the actual labels and a Q-SVM score is calculated.

Results





Method	Accuracy	Max QTA	Iterations	Learning Rate
Classical SVM	0.8	n/a	n/a	n/a
Random Parameters	0.5 (random)	0.13567	1	n/a
Gradient Descent	0.65	0.20667	40	30
Stochastic Gradient Descent	0.75	0.21634	40	30
Gradient Descent	0.60	unstable	20	300
Stochastic Gradient Descent	0.55	unstable	20	300

Conclusions and Future Work We have shown how a quantum computer can be used to perform machine learning tasks. We have shown how to implement this from embedding classical data onto quantum states to calculating the kernel matrix K using the swap test. We then went through a method for potentially improving the kernel matrix using quantum target alignment (QTA). Our results show that using QTA, we can improve our classification accuracy on the CIFAR-10 dataset, which contains images of cats and dogs. We used 20 data points for this experiment. While we did not achieve a quantum advantage, as the classical support vector machine (SVM) scored 0.8, we did see an improvement over randomly guessing the parameters. Random parameters led to an average score in the range of 0.45-0.55, which is the range for random guessing. However, using a gradient descent method to optimize the QTA, we achieved a classification accuracy of 0.65 on the test data. Furthermore, using the stochastic gradient descent method yielded further improvement to 0.75 accuracy on the test data, suggesting that the original gradient descent method gets stuck at some local maxima points. Moving forward, I would like to test this with more data points, iterations and find a method to get out of local maxima. Further improvements could be conducting experiments on a noisy quantum simulator and a real quantum device. One could also explore using a stochastic method for evaluating the kernel function, such that a n random pairs of kernels is calculated at each step. Quantum kernel methods show great promise, and by utilising Quantum Target Alignment (QTA), we could improve the training of quantum kernels for enhanced accuracy. This innovative approach may speed-up the achievement of a valuable quantum advantage in the field.

References:
[1] An overview of kernel alignment and its applications, Wang, T., Zhao, D. & Tian, S, 2015, https://doi.org/10.1007/s10462-012-9369-4
[2] The Inductive Bias of Quantum Kernels, Jonas M. Kübler and Simon Buchholz and Bernhard Schölkopf, 2021, https://doi.org/10.48550/arXiv.2106.03747
[3] Quantum Kernel Estimation With Neutral Atoms For Supervised Classification: A Gate-Based Approach, Marco Russo and Edoardo Giusto and Bartolomeo Montrucchio, 2023, https://doi.org/10.48550/arXiv.2307.15840
[4] Quantum Text Encoding for Classification Tasks, Aaranya Alexander and Dominic Widdows, 2022, https://doi.org/10.1109%2Fsec54971.2022.00052
[5] Training quantum embedding kernels on near-term quantum computers, Thomas Hubregtsen and David Wierichs and Elies Gil-Fuster, 2022, https://doi.org/10.1103%2Fphysreva.106.042431
[6] Principal component analysis, Svante Wold, Kim Esbensen, Paul Geladi, 1987, https://doi.org/10.1016/0169-7439(87)80084-9

Any Questions please don't hesitate to email me at av722@ic.ac.uk