Classical Kernel in SVM

Support Vector Machines (SVM) are used for classification and regression. The mathematical optimization problem for SVM with a linear kernel is given by:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 \tag{1}$$

Subject to:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i \tag{2}$$

When the data is not linearly separable, we use a kernel function K to map the input space into a higher-dimensional feature space:

$$\min_{\alpha} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_i \alpha_i \tag{3}$$

Subject to:

$$0 \leq \alpha_i \leq C \tag{4}$$

$$\sum_i \alpha_i y_i = 0 \tag{5}$$

The Gaussian Radial Basis Function (RBF) is a common kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2\right) \tag{6}$$

Quantum Kernel in SVM

Quantum Kernel Methods use quantum computing principles to calculate the inner product in a Hilbert space. Data points x and y are encoded into quantum states $|\psi_x\rangle$ and $|\psi_y\rangle$:

$$|\psi_x\rangle = \Phi(x)|0...0\rangle \tag{7}$$

$$|\psi_y\rangle = \Phi(y)|0...0\rangle \tag{8}$$

The quantum kernel is the inner product of these states:

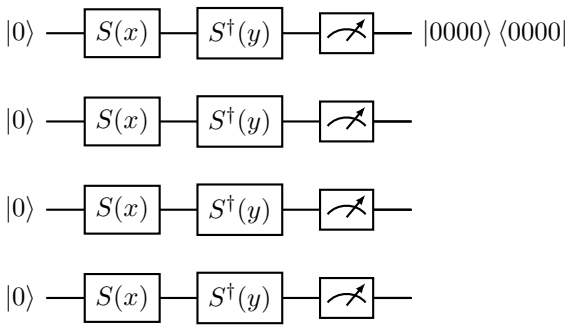
$$K_Q(x, y) = |\langle\psi_x|\psi_y\rangle|^2 \tag{9}$$

This is computed using a quantum circuit that involves a unitary operation $U(x)$ and its conjugate transpose $U(y)^\dagger$:

$$K_Q(x, y) = |\langle 0...0|U(x)^\dagger U(y)|0...0\rangle|^2 \tag{10}$$

Comparison and Advantages of Quantum Kernel

- **Dimensionality:** Quantum kernels map data into a space of dimension 2^n.
- **Entanglement:** Quantum states can exhibit entanglement, a resource with no classical analogue.
- **Quantum Parallelism:** Superposition allows for simultaneous computations.
- **No Explicit Feature Map:** Quantum kernels avoid the need to explicitly compute the feature map.



Literature Review

Quantum Machine learning in Hilbert Feature Spaces

The central thesis posits that encoding inputs into quantum states acts as a nonlinear feature map, mapping data into a quantum Hilbert space where a quantum computer can perform analysis. Two main approaches for quantum machine learning models are proposed:

- An implicit model where a quantum device estimates inner products in Hilbert space to compute kernels for classical kernel methods like SVMs. The quantum device estimates inner products of quantum states in feature Hilbert space, which is then used to compute a classically intractable kernel. These kernels are fed into classical kernel methods such as a support vector machine (SVM). The authors demonstrate the classification power of an SVM with a custom quantum kernel by visualizing decision boundaries on various datasets and showing the kernel's performance in classifying these datasets.
- An explicit model using a variational quantum circuit as a linear classifier within the Hilbert space. A variational quantum circuit is used as a linear model to classify data directly in the Hilbert space. The paper explores this approach by considering a model circuit with various quantum gates, indicating how this circuit could be trained to find an optimal decision boundary. They use a perceptron classifier in Fock space to show that data becomes linearly separable with the squeezing feature map, achieving 100 percent training accuracy on the presented data. The explicit approach shows potential benefits when classically intractable kernels are used, which may be essential for quantum machine learning.

The Need for RKHS in Quantum Kernel Methods

Instead of asking what kernel is associated with a quantum Hilbert space, they associate a quantum Hilbert space with a feature space and derive a kernel that is given by the inner product of quantum states.According to the authors, this automatically gives rise to an RKHS, and then the entire apparatus of kernel theory can be applied.

- **Stability:** RKHS ensures that small perturbations in the quantum state translate to minor changes in the function evaluations, vital for stable quantum computing.
- **Feature Maps:** Quantum feature maps encode classical data into quantum states, with RKHS providing a structure where inner products (kernels) reflect data similarity meaningfully.
- **Quantum Classifiers:** In both implicit and explicit quantum classifiers, RKHS affords a controlled optimization landscape, which is critical given the probabilistic nature of quantum measurements.
- **Generalization:** The RKHS framework supports better model generalization, ensuring that classifiers do not overfit to training data and maintain performance on new, unseen data.
- **Mathematical Rigor:** RKHS formalizes the feature space mathematics, allowing rigorous development of algorithms that harness the computational advantages of quantum systems.

Results obtained by the authors

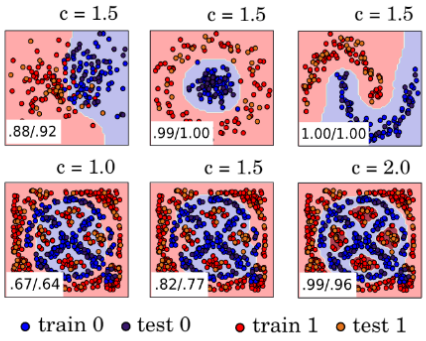


FIG. 5. Decision boundary of a support vector machine with the custom kernel from Eq. (7). The shaded areas show the decision regions for Class 0 (blue) and Class 1 (red), and each plot shows the rate of correct classifications on the training set/test set. The first row plots three standard 2-dimensional datasets: 'circles', 'moons' and 'blobs', each with 150 test and 50 training samples. The second row illustrates that increasing the squeezing hyperparameter c changes the classification performance. Here we use a dataset of 500 training and 100 test samples. Training was performed with python's *scikit-learn* SVC classifier using a custom kernel which implements the overlap of Eq. (8).

Figure 1: First image

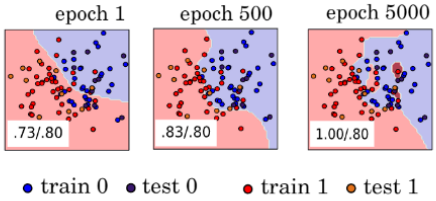


FIG. 6. Decision boundary of a perceptron classifier in Fock space after mapping the 2-dimensional data points via the squeezing feature map with phase encoding from Eq. (6) (with $c = 1.5$). The perceptron only acts on the real subspace and without regularisation. The 'blobs' dataset has now only 70 training and 20 test samples. The perceptron achieves a training accuracy of 1 after less than 5000 epochs, which means that the data is linearly separable in Fock space. Interestingly, in this example the test performance remains exactly the same. The simulations were performed with the Strawberry Fields simulator as well as a scikit-learn out-of-the-box perceptron classifier.

Figure 2: Second image

Supervised Learning with Quantum Enhanced Feature Spaces

The paper "Supervised Learning with Quantum Enhanced Feature Spaces," explores the intersection of quantum computing and machine learning, specifically focusing on improving machine learning tasks with quantum computing techniques. The study introduces two novel methods implemented on a superconducting processor that utilize the high-dimensional quantum state space for data classification, aiming to harness the potential computational advantages of quantum systems.

Quantum Feature Map

Imagine you have a way to translate (encode) regular data (like images, text, or any information) into a quantum state, which is like a super-dense version of the data that can exist in multiple states simultaneously. This translation process is done through something called a quantum feature map. The idea is to take advantage of the quantum world's high-dimensional space to represent data in ways that classical computers can't efficiently replicate. A feature map on n-qubits is generated by the unitary

$$U_\Phi(\vec{x}) = U_\Phi(\vec{x}) H^{\otimes n} U_\Phi(\vec{x}) H^{\otimes n} \tag{11}$$

where H denotes the conventional Hadamard gate and

$$U_\Phi(\vec{x}) = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{i \in S} Z_i \right) \tag{12}$$

is a diagonal gate in the Pauli Z - basis.

Data

The authors' methods were evaluated using synthetically generated data that is linearly separable via the aforementioned feature map. For $n = d = 2$ qubits, the feature mappings defined are $\phi_{\{i\}}(\vec{x}) = x_i$ and

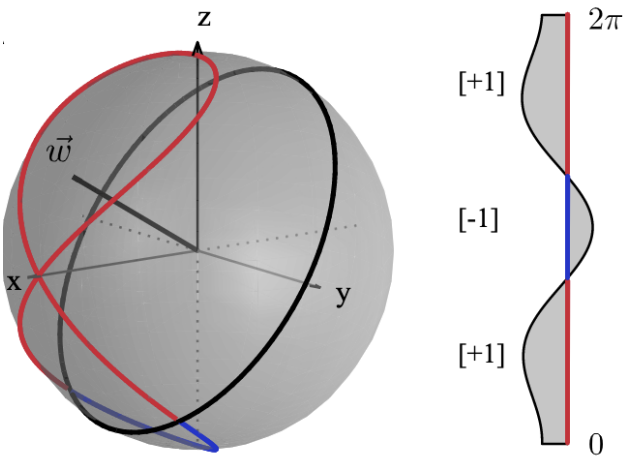


Figure 3: Feature map representation for a single qubit

$\phi_{\{1,2\}}(\vec{x}) = (\pi - x_1)(\pi - x_2)$. Labels for data vectors $\vec{x} \in T \cup S \subset (0, 2\pi]^2$ are assigned based on the parity function $f = Z_1 Z_2$ and a randomly chosen unitary $V \in SU(4)$. A label $m(\vec{x}) = +1$ is given if $\langle \Phi(\vec{x}) | V^\dagger f V | \Phi(\vec{x}) \rangle \geq \Delta$, and $m(\vec{x}) = -1$ if $\langle \Phi(\vec{x}) | V^\dagger f V | \Phi(\vec{x}) \rangle \leq -\Delta$. A separation gap of $\Delta = 0.3$ is and 20 data points per label are used for both training and classification sets.

Proposed Methods

The paper introduces two methods:

- **Quantum Variational Classifier**
    1. **Encoding Data:** First, data is encoded into quantum states using a feature map. It's like converting a readable book into a quantum form where each page can exist in multiple states at once.
    2. **Variational Circuit:** Then, a special quantum circuit (a variational quantum circuit), which can be tweaked and adjusted, is applied. This circuit is trying to find the best way to separate the data into categories within the quantum world.
    3. **Optimization:** Through a series of adjustments and measurements, the parameters of the circuit are fine-tuned to improve the separation accuracy. It's akin to adjusting the lenses of a telescope to get a clear view of a distant planet.
- **Quantum Kernel Estimator**
    1. **Kernel Calculation:** In classical machine learning, kernels are ways to measure how similar two data points are. In QKE, this similarity measurement is taken to the quantum level. We're calculating how close or far two quantum states are, representing the data.
    2. **Direct Estimation:** Instead of tweaking a circuit to find the best separator as in QVC, QKE directly calculates the "closeness" of every pair of data points in their quantum state form. Imagine having a quantum ruler that measures the distance between planets in a galaxy, where each planet represents a piece of data.
    3. **Classification:** Using these quantum-measured similarities, we can then apply classical methods to classify the data into groups, much like sorting a collection of stars into constellations based on their proximity.

Implementation and Results

The methods were implemented on an IBM quantum processor, demonstrating the feasibility of these quantum-enhanced classification techniques. Despite the presence of noise and other practical limitations of current quantum hardware (referred to as Noisy Intermediate-Scale Quantum (NISQ) devices), the experiments showed promising results, including high classification accuracies that suggest potential pathways to achieving quantum advantages in machine learning tasks.

Conclusion and Future Directions

The paper concludes with optimism about the role of quantum computing in enhancing machine learning, particularly through developing suitable quantum feature maps and kernels that can leverage the unique capabilities of quantum systems. The work suggests a roadmap for further exploration in quantum machine learning, aiming to identify and implement quantum algorithms that provide significant advantages over classical approaches for real-world data sets.

Code Implementations

Classification using ad-hoc dataset

This implementation (titled *QK-Ad-hoc.ipynb* in our Github repository) adapted from the Qiskit tutorial on Quantum Kernel Machine Learning (tutorial here)is based on the paper "Supervised Learning with Quantum Enhanced Feature Spaces" which has been discussed above. The dataset used is the ad-hoc dataset with the hyperparameters same as those mentioned in the paper (except for Clustering where a higher generation gap of 0.6 between the 2 classes is used). Classification is done using three approaches and the model performances for each are measured, demonstrating perfect classification:

- Kernel as a callable function with SVC
- Precomputed kernel with SVC
- With QSVC

Additionally, it has also been shown that there are use-cases for kernel-based unsupervised algorithms too, for example, in the context of clustering, **Spectral Clustering**, is a technique where data points are treated as nodes of a graph, and the clustering task is viewed as a graph partitioning problem where nodes are mapped to a space where they can be easily segregated to form clusters. A clustering score of 0.73 was obtained for spectral clustering using a precomputed (quantum) kernel. Lastly, Kernel PCA with its comparison on Guassian and Quantum kernel was performed showing that the gaussian kernel based Kernel PCA model fails to make the dataset linearly separable, while the quantum kernel succeeds.

Key Components

- **ZZFeatureMap:** This component encodes classical data into quantum states, creating a feature space where quantum properties like entanglement significantly enhance the data's dimensionality and complexity.
- **Sampler and ComputeUncompute:** These components are crucial for estimating the state overlap (fidelity) between quantum states, which directly relates to computing the kernel matrix elements in the quantum kernel estimation method.
- **FidelityQuantumKernel:** It leverages the ComputeUncompute method for calculating similarities between data points in the quantum-enhanced feature space, enabling advanced machine learning applications.

Kernel Based Training using scikit-learn

This implementation (titled place-holder) is adapted from the Kernel-Based training of quantum models with scikit learn (tutorial here). The dataset used in this implementation is the well-known iris dataset. The tutorial explains the link between quantum models and kernel methods, replacing the common variational approach to quantum machine learning with a classical kernel method that is a small building block of the overall algorithm computed by the quantum device.

For small variational parameters, variational circuits are as efficient as neural networks. However, for quantum machine learning applications with large parameters kernels methods scale better than variational circuits.

Upcoming Plans

In our pursuit of developing a comprehensive understanding of kernel performance in various computational contexts, our endeavors will focus on the following key areas:

1. **Diverse Data Comparison:** We aim to rigorously compare the performance of different kernels using a variety of input data. This includes both real-world data sets and various forms of computationally generated data. The objective is to mitigate the risk of data bias towards any specific kernel. By ensuring the data's diversity, we aspire to achieve a more balanced and objective assessment of kernel performance.

2. **Optimization and Preprocessing:** To facilitate a fair comparison, we will repeat the analysis using kernels that have undergone full optimization. Additionally, we will preprocess the data to eliminate any potential discrepancies that could skew the results.

3. **Quantum vs. Classical Kernels:** Expanding the scope of our comparison, we will use different types of quantum kernels with their classical counterparts. This comparison is aimed at shedding light on the general performance differences between these two categories of kernels.

4. **Complexity and Scalability:** To further our understanding of how complexity affects performance, we will extend our project to test kernels with more features and more than two classes. This exploration will help us ascertain the scalability of these kernels and their efficacy in handling complex data structures.

5. **Clustering Applications:** Lastly, we will attempt to explore the application of Quantum Kernels for various unsupervised clustering tasks (not just Spectral Clustering) in order to gauge their feasibility and usability for the same.

References

All the resources used by us for understanding the various concepts at play, for exploring the varied applications of quantum kernels and for understanding their code implementations can be found in our Github repository in the file *Useful-Links.txt* in our main directory.