COMPLUTENSE UNIVERSITY OF MADRID

ARCHITECTURE AND PROGRAMMING OF QUANTUM COMPUTERS

Q-SVM for Handwritten Digit Classification using MNIST dataset

Course Instructor: Dr. Guillermo Botella

May 18, 2025

Abstract

Quantum computing offers a different perspective on how to construct new types of kernels that may improve classification performance compared to existing classical versions. In this work, we evaluate Quantum Support Vector Machines (QSVMs) on MNIST-derived handwritten digit recognition benchmarks, contrasting them with classical SVMs using linear and RBF kernels.

After reducing each 28×28 image to four PCA-derived features, we implement different feature maps and estimate kernel entries via the Qiskit AerSimulator. While quantum kernels can implicitly represent exponentially large feature spaces, we find that for most practical digit-classification tasks such high-dimensional representations are unnecessary to achieve high accuracy. Classical SVM's record a +95% accuracy on all problems and QSVM's reach +90% accuracy on binary classification tasks but only 50% on the four-class problem.

These results demonstrate that QSVM can match or slightly outperform classical SVM in narrow cases but shows no consistent advantage, underscoring the critical importance of quantum feature-map design and of choosing scenarios where the quantum dimensional representation advantage offers a real benefit.

Contents

INTRODUCTION

1.1 Motivation

Quantum computing has emerged as a promising frontier for advancing machine learning, offering potential speed-ups and enhanced capabilities through quantum phenomena like superposition and entanglement. One area of active research is Quantum Support Vector Machines (QSVM), the quantum counterpart of classical SVMs, which aim to leverage quantum feature spaces for improved classification performance. Handwritten digit recognition (e.g., on the MNIST dataset) provides an excellent test-bed for comparing classical and quantum approaches, as it involves high-dimensional image data and is a well-studied problem in machine learning. This project explores a QSVM for classifying handwritten digits, comparing its performance against classical SVM baselines on several tasks: binary classification of digit 6 vs 9, binary classification of 1 vs 7, and multi-class classification of digits 1, 2, 3, 4. We investigate different kernel "feature maps" for the QSVM, analyze their impact on accuracy, and discuss whether QSVM can achieve any advantage over classical SVM in these experiments.

The appeal of QSVM lies in its ability to implicitly embed data into exponentially large Hilbert spaces, potentially enabling separations that are intractable for classical kernels. In theory, a properly chosen quantum kernel could uncover intricate patterns in data by exploiting high-order correlations that classical kernels might miss. However, despite this theoretical promise, it remains unclear under what conditions quantum kernels can deliver tangible improvements on real-world tasks where classical methods already perform exceptionally well. Handwritten digit recognition, with its low-noise, high-resolution images and well-characterized structure, offers a very high benchmark: classical SVMs routinely exceed 95 % accuracy, raising the bar for any quantum-enhanced alternative.

At the same time, the current era of noisy intermediate-scale quantum (NISQ) devices imposes severe constraints on circuit depth, qubit count, and noise tolerance. Understanding which aspects of quantum kernel design, such as the degree of entanglement or the order of feature interactions, actually contribute to improved generalization, versus which simply are overfitting, is crucial for future developments.

By focusing on a well-understood task like digit classification, this work aims to illuminate the broader question of when and how quantum machine learning can overcome classical limits.

1.2 Scope and Contributions

Recent studies underline both the potential and challenges of QSVMs. Quantum kernel methods, like QSVM, embed data into the Hilbert space of a quantum system, where the inner products (kernels) between data points can be estimated by quantum circuits. In principle, this allows accessing extremely large (even exponentially large) feature spaces that are infeasible to simulate classically. If successful, a QSVM could separate classes that are hard to separate with any classical kernel, potentially achieving higher accuracy or using fewer resources. Indeed, prior work demonstrates the feasibility of QSVM for tasks like handwriting recognition – e.g., Bindhani et al. (2020) implemented SVM on a quantum computer to recognize handwritten digits using various quantum feature maps. Their goal was to demonstrate the advantage of applying SVM in quantum feature space and to compare the accuracy of QSVM vs classical SVM. These studies indicate that while QSVMs are conceptually powerful, realizing a tangible performance boost over classical methods is non-trivial given present quantum hardware constraints.

Against this backdrop, our project serves as a focused case study of QSVM on a small-scale handwritten digit classification problem. We use simulations to implement the QSVM, allowing us to isolate the algorithmic performance without noise. We design quantum feature map circuits inspired by the literature (including a simplified quantum feature map and a more complex one proposed by Havlíček et al. [1]) and evaluate their classification accuracy. Classical SVMs with linear and RBF kernels are used as benchmarks. By analyzing results on the chosen binary and multi-class tasks, we aim to assess: (1) how QSVM performance compares to classical SVM on equivalent data, (2) how the choice of quantum kernel (feature map) influences QSVM accuracy, and (3) what limitations are evident (e.g., data size, feature dimensionality, lack of quantum hardware) that impact QSVM's effectiveness. Ultimately, we will assess through the findings whether QSVM offers any practical benefit in our scenario and what future improvements are needed for QSVM to outperform classical methods, if any.

BACKGROUND

2.1 Classical SVM Fundamentals

The Support Vector Machine (SVM) is a supervised learning algorithm widely used for classification tasks. At its core, a binary SVM attempts to find an optimal separating hyperplane between two classes of data in a feature space. For linearly separable data, the SVM determines the maximum-margin hyperplane that correctly partitions the examples by class, defined by a set of support vectors (critical training points that lie closest to the decision boundary). An example of a linear SVM separating two classes is shown in Figure 2.1, where the goal is to separate them with a line (in higher dimensions, a hyperplane) that maximizes the margin on either side. In cases where data are not linearly separable in the original input space, SVMs employ the *kernel trick*: data are implicitly mapped to a higher-dimensional feature space where a linear separator can be found. Common kernels include the polynomial, Gaussian radial basis function (RBF), and sigmoid kernels, which correspond to different nonlinear mappings. By using a kernel function

$$K(x_i, x_j) \;=\; \langle \Phi(x_i), \Phi(x_j) \rangle$$

that computes inner products in an (often high-dimensional) feature space Φ, the SVM can fit complex decision boundaries without explicitly computing high-dimensional coordinates of $\Phi(x)$ for each data point.

In practice, SVMs have been very successful for digit recognition and other pattern recognition tasks. On the MNIST handwritten digit dataset (70,000 28×28 images of digits 0–9), a classical SVM with an appropriate kernel can achieve high accuracy (often $> 95\%$ for binary digit classification) by finding a decision boundary in pixel space or a transformed space. However, classical SVMs face challenges as the feature space or dataset grows large: kernel matrices become computationally expensive to compute and store (scaling with the square of the number of samples), and certain complex mappings might not yield improved performance due to the dimensionality problem. If we have extremely high-dimensional data (for example, each 28×28 image can be seen as a 784-dimensional vector), a linear SVM might underperform, but a highly nonlinear kernel could be expensive or lead to overfitting. Thus, classical SVMs strike a balance by choosing kernel functions that are expressive yet tractable (RBF is a popular choice for image classification as it can model nonlinear boundaries).

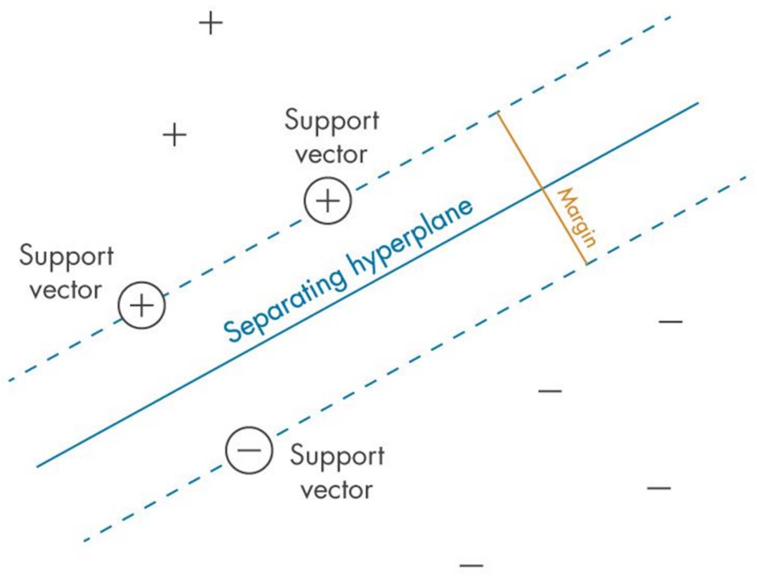


Figure 2.1: Illustration of a classical SVM separating two classes in a 2D feature space.

Mathematical Formulation. Let $\{(x_i, y_i)\}_{i=1}^N$ be a training set with $x_i \in R^d$ and labels $y_i \in \{-1, +1\}$. A support vector machine seeks a hyperplane

$$w^\top x + b = 0,$$

with parameters $w \in R^d$ and $b \in R$, that maximizes the margin between the two classes.

Primal Problem: The soft-margin SVM solves

$$\begin{aligned} \min_{w,b,\{\xi_i\}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^N \xi_i, \\ \text{subject to} \quad & y_i\big(w^\top x_i + b\big) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, N, \end{aligned}$$

where ξ_i are slack variables and $C > 0$ controls the penalty on misclassifications.

Dual Problem: Introducing Lagrange multipliers $\{\alpha_i\}$ for the margin constraints and $\{\mu_i\}$ for $\xi_i \geq 0$, one obtains

$$\begin{aligned} \max_{\{\alpha_i\}} \quad & \sum_{i=1}^N \alpha_i - \tfrac{1}{2}\sum_{i,j=1}^N \alpha_i\alpha_j y_i y_j\, K(x_i, x_j), \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \quad \sum_{i=1}^N \alpha_i y_i = 0, \quad i = 1, \dots, N, \end{aligned}$$

where $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ is the chosen kernel function.

Decision Function: The learned classifier predicts

$$f(x) \ = \ \text{sign}\Big(\sum_{i=1}^N \alpha_i^* y_i\, K(x_i, x) + b^*\Big),$$

where $\{\alpha_i^*\}, b^*$ are the optimal dual variables.

Common Kernel Functions.

Linear:	$K(x, x') = x^\top x'$,
Polynomial:	$K(x, x') = (\gamma\, x^\top x' + r)^p$,
RBF (Gaussian):	$K(x, x') = \exp\bigl(-\gamma \lVert x - x' \rVert^2\bigr)$,
Sigmoid:	$K(x, x') = \tanh(\gamma\, x^\top x' + r)$.

Multi-class Extensions. To handle $M > 2$ classes, one commonly uses:

- *One-vs-Rest (OvR):* Train M binary SVMs, each separating one class from the rest; assign a test point to the class with the largest decision value.
- *One-vs-One (OvO):* Train $\binom{M}{2}$ binary SVMs for every pair of classes; assign by majority vote among all pairwise classifiers.

Overall, classical SVMs provide a robust and theoretically grounded approach for both binary and multi-class classification, serving as a strong baseline for structured data such as images. In our experiments, we employ an RBF kernel for its balance of expressivity and tractability in small-sample, moderate-dimensional settings.

2.2 Quantum Feature Maps and Kernels

A Quantum Support Vector Machine (QSVM) extends the classical SVM framework by embedding classical inputs into a quantum Hilbert space and defining a kernel via the overlap of quantum states. Formally, let $x \in R^d$ be a data vector. We choose a *quantum feature map*

$$U_\phi(x) \;:\; 0^{\otimes n} \;\longmapsto\; \phi(x) \;\in\; \mathcal{H}_{2^n}\,,$$

where n qubits span a 2^n-dimensional feature space. The induced *quantum kernel* is the fidelity between two mapped states,

$$K(x_i, x_j) \;=\; \bigl|\langle \phi(x_i) \mid \phi(x_j) \rangle\bigr|^2 .$$

By computing $K(x_i, x_j)$ on a quantum processor and assembling the Gram matrix $\bigl[K(x_i, x_j)\bigr]_{i,j=1}^{N}$, one obtains a precomputed kernel that can be supplied directly to any classical SVM solver (e.g., via scikit-learn's 'kernel='precomputed'' option).

Quantum Feature Map Design Designing $U_\phi(x)$ is the central challenge in QSVM. A good feature map must be both expressive—capable of encoding nonlinear combinations of input components—and implementable on near-term hardware. Common constructions include:

- **Angle encoding with entanglement:**

$$U_\phi(x) = \Bigl(\bigotimes_{j=1}^{n} R_Y(\alpha\, x_j)\Bigr) \;\times\; \Bigl(\prod_{\langle k,\ell \rangle} \mathrm{CNOT}_{k\to\ell}\Bigr),$$

  where each R_Y rotation encodes one feature and a chain (or ring) of CNOT gates entangles qubits.

- **Second-order ZZ feature map**: Alternating layers of Hadamards, data-driven $R_Z(x_j)$ rotations, and controlled-Z (or R_{ZZ}) gates between every pair (j, k). Repeated for multiple depth layers, this map encodes both first- and second-order feature products via quantum interference.

These circuits exploit superposition and entanglement to implicitly realize extremely high-dimensional feature spaces—formally, a space of dimension 2^n.

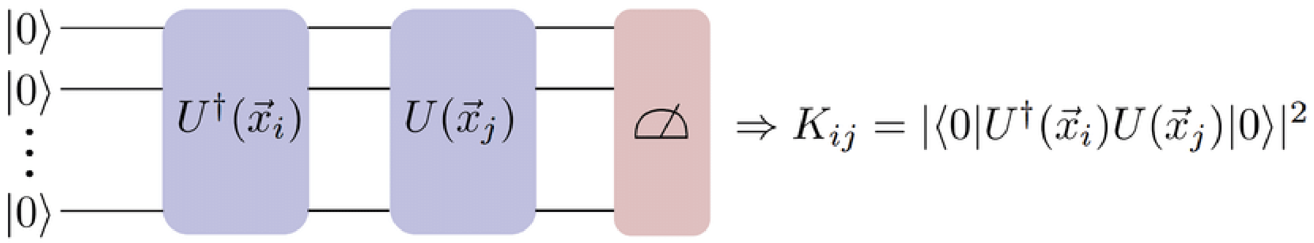

$|0\rangle$
$|0\rangle$
$|0\rangle$
$U^\dagger(\vec{x}_i)$
$U(\vec{x}_j)$
$\Rightarrow K_{ij} = |\langle 0|U^\dagger(\vec{x}_i)U(\vec{x}_j)|0\rangle|^2$

Figure 2.2: Quantum circuit for estimating the kernel $K(x_i, x_j) = |\langle \phi(x_i) \mid \phi(x_j)\rangle|^2$. Applying $U_\phi(x)^\dagger$ and measuring all qubits yields the all-zero probability equal to the fidelity.

Kernel Estimation via Quantum Circuits To evaluate $K(x_i, x_j)$, one uses the so-called *overlap test* circuit (Figure 2.2). On the same set of n qubits, apply

$$U_\phi(x_i)\, U_\phi(x_j)^\dagger \;=\; \phi(x_i)\phi(x_j)\,.$$

Measuring all qubits in the computational basis yields the probability $\Pr(0^{\otimes n}) = K(x_i, x_j)$. Repeating for all (i, j) pairs constructs the full $N \times N$ kernel matrix, and similarly for test–train overlaps.

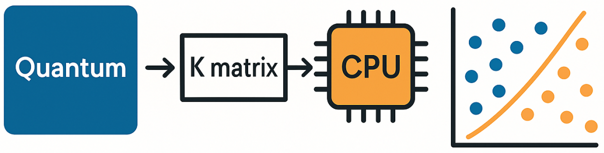
Theoretical and Practical Limitations

- *Kernel concentration and generalization:* Very expressive quantum feature maps may map most inputs to nearly orthogonal states, causing K to resemble the identity and harming generalization.
- *Data-encoding bottleneck:* Real images (e.g., 784-dimensional MNIST pixels) must be compressed (e.g., PCA) into $d \leq n$ components, potentially discarding discriminative information.
- *NISQ hardware constraints:* Current devices offer only tens of noisy qubits and limited gate depth. Deep circuits amplify decoherence and gate errors.
- *Simulation cost:* Statevector simulation scales as $O(2^n)$, restricting practical experiments to $n20$.

Despite these challenges, QSVM remains a compelling proof-of-concept for hybrid quantum–classical classification. Under current simulation, it enables exploration of novel kernels in exponentially large feature spaces, guiding future algorithm design as quantum hardware matures.

2.3 QSVM Workflow Overview

The QSVM combines quantum state preparation for kernel estimation with classical convex optimization for model training. We divide the workflow into three stages: (1) quantum kernel estimation, (2) classical SVM training, and (3) prediction (including multi-class extension).



2.3.1 Quantum Kernel Estimation

1. **Prepare feature-map states.** For each data point x_i, apply the chosen quantum feature map

$$0^{\otimes n} \xrightarrow{U_\phi(x_i)} \phi(x_i)$$

on an n-qubit register. Here $U_\phi(x)$ may be, for example, an angle-encoding circuit with single-qubit rotations and entangling gates or a deeper ZZ-feature map.

2. **Compute state-overlap.** To estimate $K(x_i, x_j) = |\langle \phi(x_i) \mid \phi(x_j) \rangle|^2$, construct the overlap circuit

$$U_\phi(x_i) \left[U_\phi(x_j) \right]^\dagger \; : \; 0^{\otimes n} \longmapsto U_\phi(x_i)\, U_\phi(x_j)^\dagger 0^{\otimes n}.$$

Measure all qubits in the computational basis; the frequency of the $0^{\otimes n}$ outcome over M shots yields an empirical estimate $\widehat{K}(x_i, x_j)$.

3. **Assemble Gram matrix.** Populate the $N \times N$ kernel matrix

$$K_{ij} \;=\; \widehat{K}(x_i, x_j), \quad i, j = 1, \ldots, N,$$

exploiting symmetry $K_{ij} = K_{ji}$ to halve the number of required circuit evaluations.

2.3.2 Classical SVM Training

With the quantum kernel matrix K in hand, model training proceeds exactly as in a classical SVM with a precomputed kernel:

1. **Solve the dual optimization.**

$$\max_{\{\alpha_i\}} \quad \sum_{i=1}^{N} \alpha_i \; - \; \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j \, y_i y_j \, K_{ij},$$
$$\text{subject to} \quad 0 \;\le\; \alpha_i \;\le\; C, \quad \sum_{i=1}^{N} \alpha_i \, y_i \;=\; 0,$$

   where $y_i \in \{-1, +1\}$ and $C > 0$ is the regularization parameter. A standard solver (e.g. SMO) yields $\{\alpha_i^*\}$ and bias b^*.

2. **Identify support vectors.** Points with nonzero α_i^* lie on the margin and define the decision boundary in the quantum feature space.

2.3.3 Prediction and Multi-class Extension

Binary classification. Given a new sample x, estimate its kernel vector $\big[K(x_i, x)\big]_{i=1}^{N}$ via the same quantum overlap test, then compute the decision function

$$f(x) \;=\; \operatorname{sign}\Big(\sum_{i=1}^{N} \alpha_i^* \, y_i \, K(x_i, x) \;+\; b^* \Big).$$

One-vs-Rest (OvR). For M classes, train M binary QSVMs with labels

$$y_i^{(m)} = \begin{cases} +1, & \text{if } x_i \text{ in class } m, \\ -1, & \text{otherwise,} \end{cases} \qquad m = 1, \ldots, M.$$

Predict by selecting the class m whose QSVM yields the largest decision value.

One-vs-One (OvO). Alternatively, train $\binom{M}{2}$ binary QSVMs for each class pair (p, q). At inference, each pairwise classifier votes for p or q, and the final label is obtained by majority vote.

2.3.4 Computational Considerations

- **Circuit count:** Kernel estimation naively requires $O(N^2)$ quantum circuits, each run for M shots. Exploiting symmetry and low-rank approximations can reduce this overhead.

- **Shot noise and error mitigation:** Finite-shot sampling and hardware noise introduce uncertainty in $\widehat{K}$. Calibration, zero-noise extrapolation, or randomized compiling may improve fidelity estimates.

- **Scalability trade-offs:** Increasing qubit count n enhances expressivity (2^n-dimensional feature space) but escalates simulation cost ($O(2^n)$) and hardware error rates. Dimensionality reduction (e.g. PCA) and shallow feature maps seek a balance between expressivity and implementability.

METHODOLOGY

3.1 Dataset and Feature Extraction

Dataset

We used a subset of the MNIST handwritten digit dataset, focusing on specific classes for targeted experiments. From the full MNIST dataset (70,000 grayscale images of digits 0–9, each of size 28×28 pixels), we extracted samples corresponding to the digits **6, 9, 1, 7, 2, 3**, and **4** as needed for each classification task.

Given the computational limits of our QSVM simulation, we dramatically reduced the dataset size. For each *binary classification* task, we randomly selected 80 images for training and 20 images for testing. For the *four-class experiment* (digits 1, 2, 3, 4), we similarly took a small balanced subset of 80 training images and 20 testing images.

While such small sample sizes would be insufficient to train a high-accuracy classical model in general, they are aligned with our goal of testing QSVM on a manageable scale. The use of equal class representation in both training and test sets ensures that classification accuracy is not biased by class imbalance.

Feature Extraction

Each MNIST image consists of $28 \times 28 = 784$ pixels, with intensity values ranging from 0 to 255. Directly using 784-dimensional feature vectors is not possible in any current quantum model.

To reduce dimensionality while preserving class-discriminative information, we applied the following preprocessing steps inspired by common approaches:

1. **Binarization:** We converted grayscale images to binary black-and-white by thresholding: any pixel with intensity > 230 was set to 1, and all others to 0. This step drastically reduces the size, converting each pixel to a 0-1 instead of the 0-255 that was before.

2. **Principal Component Analysis (PCA):** We performed PCA on the binarized data to project it into a lower-dimensional subspace. For each binary classification task (e.g., digits 6 vs 9), PCA was fit on the 80 training samples to extract the top components capturing the most variance. We retained the top 4 principal components, resulting in each image being represented by a 4-dimensional feature vector (z_1, z_2, z_3, z_4). This step reduces dimensionality from 784 to 4, where each component is a linear combination of pixel values capturing the most important variations.

3. **Feature Scaling:** After PCA, the feature values were normalized into the range $[-\pi, \pi]$, as these values are used as rotation angles (in radians) in the quantum circuit encoding. After scaling, the feature vectors typically have magnitudes on the order of 1–3.

The choice of retaining 4 features was dictated by the 4-qubit limitation of our quantum hardware simulation. As noted by Bindhani et al. (2020), dimensionality reduction via PCA is a necessary step before quantum encoding when limited qubits are available. To ensure fairness in performance comparison, we also used the same 4-dimensional feature vectors as input to classical SVM baselines. This ensures that any differences in performance arise from the model type (quantum vs. classical), rather than from differences in input representation.

3.2 Classical SVM Baselines

We trained classical Support Vector Machine (SVM) models on the reduced-feature data to serve as benchmarks. For the **binary classification tasks** (e.g., digits 6 vs 9 and 1 vs 7), we evaluated two kernel settings:

- **Linear SVM:** Attempts to separate the classes with a hyperplane in the 4-dimensional PCA feature space.
- **RBF-kernel SVM:** Capable of modeling non-linear decision boundaries, potentially capturing more complex class separations.

Given the low-dimensional feature space (4 dimensions) and the generally decent class separability observed (especially for the 6 vs 9 task), we expected the linear SVM to perform adequately. The RBF kernel was included to capture any subtle non-linear patterns, if present.

We used the `scikit-learn` implementation of `SVC` for all classical models, with default regularization parameter $C = 1$, and specified the kernel accordingly. Each SVM was trained on 80 training samples (40 per class) and evaluated on 20 test samples (10 per class).

For the **multi-class classification task** (digits 1, 2, 3, 4), we employed a single SVM model with an RBF kernel using the One-vs-One (OvO) strategy, which is the default behavior of `SVC` in `scikit-learn` for multi-class problems. The linear kernel was not considered for this task, as non-linear boundaries were more likely to be needed with 4 classes and only 4 features. This OvO strategy involves training one binary classifier for each pair of classes among $\{1, 2, 3, 4\}$.

Evaluation Metrics

The performance of each classical SVM model was evaluated using:

- **Overall test accuracy**
- **Confusion matrix analysis**

Given the small size of the test sets, accuracy was sensitive to individual errors:

- In binary tasks (20 test samples), each misclassification leads to a 10% drop in accuracy.
- In the 4-class task (20 test samples), each misclassification corresponds to a 5% drop.

We report classification accuracy and note differences between linear and RBF kernels. However, for brevity, much of our analysis focuses on the best-performing classical baseline in comparison to the QSVM results.

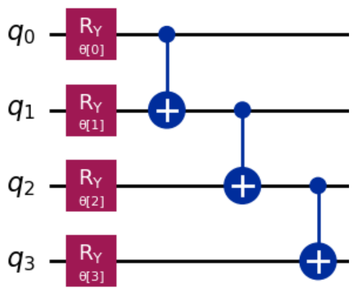
In all cases, classical SVM training and inference times were negligible due to the very small dataset sizes, and were significantly faster than the QSVM simulation times.

3.3 Quantum SVM Implementation

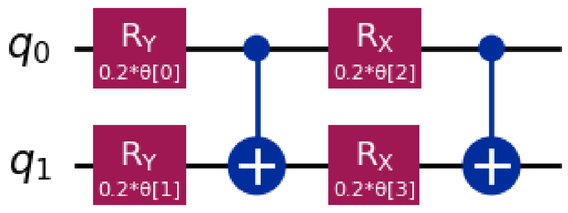
The core of our project was implementing the Quantum Support Vector Machine (QSVM) and experimenting with different quantum feature maps (kernels). We used IBM's `Qiskit` framework for constructing and simulating quantum circuits. All QSVM computations were run on the `Qiskit AerSimulator`, since we did not have access to a real quantum backend for extended jobs (attempts to use IBM Quantum cloud resulted in queue limits, as noted in our conclusions). The workflow for QSVM was as follows:

1. **Feature Map Circuit Design:** We created parameterized quantum circuits $U_\phi(x)$ that take four input features and outputs a quantum state $|\phi(x)\rangle$. We implemented three main feature maps:

   - **Feature Map A (Simple):** A relatively shallow circuit with one layer of data-encoded rotations and entanglement. Specifically, each of the four qubits is first prepared in state $|0\rangle$ and has a single-qubit rotation
     $$R_Y(\theta_j)$$
     applied, where θ_j is proportional to feature z_j. We then include a ring of CNOT entanglements connecting qubits in the order
     $$q_0 \to q_1,\ q_1 \to q_2,\ q_2 \to q_3,\ q_3 \to q_0.$$
     This map has four parameters (one per qubit). The resulting quantum state $|\phi_A(x)\rangle$ encodes the four features in the amplitudes and entanglements of the 4-qubit state.



   - **Feature Map B (Layered Rotations with Entanglement):** A compact 2-qubit circuit featuring two sequential layers of data-driven single-qubit rotations interleaved with entangling gates. We start by initializing both qubits in $|0\rangle$. In the first layer, each qubit $j \in \{0, 1\}$ undergoes a rotation about the Y-axis,
     $$R_Y\big(\alpha\,\theta_j\big),$$
     where θ_j is proportional to feature z_j and $\alpha = 0.2$ sets the encoding strength. We then apply a CNOT from qubit 0 to qubit 1 to create entanglement. In the second layer, we apply X-axis rotations,
     $$R_X\big(\alpha\,\theta_{j+2}\big)$$
     on qubits 0 and 1 (using parameters θ_2 and θ_3), capturing additional feature interactions, followed by another CNOT from qubit 0 to qubit 1. Overall, this map uses four parameters $\{\theta_0, \theta_1, \theta_2, \theta_3\}$, encodes two features across two rotation layers, and leverages entanglement to allow joint feature interactions. The resulting state $|\phi_B(x)\rangle$ embeds $x = (z_0, z_1)$ in both amplitudes and correlations of the 2-qubit system.

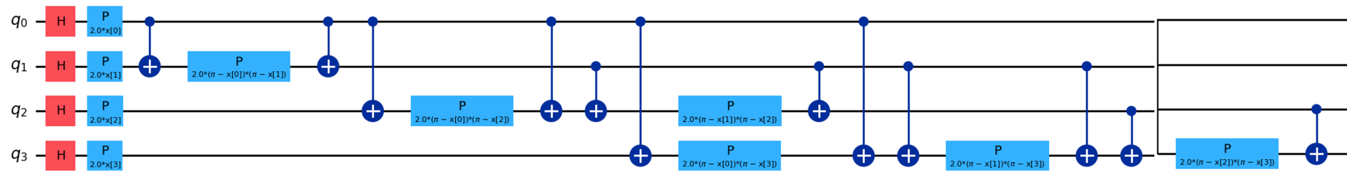


- **Feature Map C (Advanced):** A deeper circuit inspired by second-order feature maps, including pairwise feature interactions. We utilized Qiskit's built-in `ZZFeatureMap` with two repetitions (layers). This feature map applies:
  (a) A layer of Hadamard gates on each qubit.
  (b) Data-driven R_Z rotations on each qubit, encoding each feature z_j.
  (c) Entangling ZZ rotations (or controlled-Z) gates between every pair of qubits, encoding products $z_i z_j$.
  (d) Repeat the above three steps for a second layer.

  Concretely, for four qubits and depth 2, the circuit implements:

$$\left(\mathrm{H}^{\otimes 4}\right) R_Z(\mathbf{z})\, \mathrm{CZ}_{\text{pairs}} \left(\mathrm{H}^{\otimes 4}\right) R_Z(\mathbf{z})\, \mathrm{CZ}_{\text{pairs}}.$$

  The result is a state $|\phi_C(x)\rangle$ in a 4-qubit space containing information up to second-order feature combinations. This construction follows the second-order feature map proposed in Havlíček *et al.* (Nature), aiming to achieve quantum advantage. We set the parameters so that it uses our four PCA features as inputs, with Qiskit automatically mapping each feature to the appropriate gate angles.


q_0
q_1
q_2
q_3
H
P 2.0*x[0]
P 2.0*x[1]
P 2.0*x[2]
P 2.0*x[3]
P 2.0*(π − x[0])*(π − x[1])
P 2.0*(π − x[0])*(π − x[2])
P 2.0*(π − x[1])*(π − x[2])
P 2.0*(π − x[0])*(π − x[3])
P 2.0*(π − x[1])*(π − x[3])
P 2.0*(π − x[2])*(π − x[3])

2. **Kernel Matrix Computation:** For each feature map, we implemented the quantum kernel evaluation using the "quantum kernel estimator" approach. We treated the QSVM as a black-box kernel machine where the kernel is given by the fidelity between states. Concretely, for every pair of training examples (x_i, x_j), we:

   (a) Prepare the state $|\phi(x_i)\rangle$ on four qubits using the feature-map circuit.

   (b) Append the inverse circuit $U_\phi(x_j)^\dagger$ to uncompute $|\phi(x_j)\rangle$.

   (c) Measure all qubits in the computational basis and estimate the probability

   $$p_{00\ldots0} = \left|\langle 00\ldots0|\, U_\phi(x_j)^\dagger\, U_\phi(x_i)\, |00\ldots0\rangle\right|^2.$$

   (d) Interpret this probability as the kernel value

   $$K(x_i, x_j) = p_{00\ldots0}\,.$$

   Repeating this pairwise procedure for all N training points populates the Gram matrix $K \in R^{N\times N}$. Likewise, for each test point x_{test}, we compute the kernel vector

   $$K(x_{\text{test}}, X_{\text{train}}) = \big[\, K(x_{\text{test}}, x_1), \ldots, K(x_{\text{test}}, x_N)\big].$$

3. **QSVM Training:** Given the quantum kernel matrix and the corresponding training labels, we solved the SVM dual optimization problem using a classical solver. Specifically, we employed the `scikit-learn` `SVC` class with `kernel='precomputed'` mode, supplying it with our quantum-computed kernel matrix.

   This method computes the optimal support vector coefficients α_i and the bias term b, which together define the decision boundary in the quantum feature space. The regularization parameter was set to $C = 1.0$, consistent with our classical SVM baseline.

   The QSVM training phase is computationally efficient once the kernel matrix is available. In the binary classification case, the optimization involves solving a quadratic program with 80 variables (one per training sample). In the multi-class One-vs-One (OvO) setting, the optimization scales with the number of pairwise classifiers, but remains fast. The dominant computational cost in the QSVM pipeline is the kernel matrix computation itself.

4. **Prediction:** After training, we computed the decision function for the test samples using the precomputed kernel. For a binary classifier, the SVM decision function is given by:

   $$\text{sign}\left(\sum_{i\in SV} \alpha_i y_i K(x_i, x_{\text{test}}) + b\right),$$

   where the sum runs over the support vectors x_i, with labels y_i, weights α_i, and the quantum kernel function $K(\cdot,\cdot)$.

   We computed the kernel matrix between test and training samples and passed it to the trained `SVC` model to obtain predictions. The predicted labels were compared to the true test labels to compute classification accuracy. Additionally, we constructed confusion matrices to identify which digit pairs were misclassified and to analyze the model's performance in greater detail.

We repeated the above QSVM process separately for each experiment (6 vs 9, 1 vs 7, and 1/2/3/4). Each experiment was done for two different feature maps, allowing us to observe the impact of the kernel choice.
To summarize, our QSVM methodology closely mirrors the theoretical QSVM procedure: encode data into quantum states via a chosen feature map, use the quantum (simulated) processor to compute kernel values, and then apply a classical SVM solver. By varying the feature map, we effectively test different quantum kernels. The different feature maps will be compared to see if a more expressive quantum kernel leads to better classification or if it overfits or gives a poorer generalization on the test set. All other factors (data, SVM C, etc.) are kept constant to ensure a controlled comparison.

RESULTS

After implementing both classical SVMs and the QSVM with two feature maps, we evaluated their performance on the three classification tasks. Table 4.1 below summarizes the accuracy results for each scenario. We will discuss each in detail, including observations from confusion matrices and decision boundaries.

Task	**Classical SVM**	**F. Map A**	**F. Map B**	**F. Map C**
6 vs 9 (binary)	95% (RBF kernel)	90%	–	–
1 vs 7 (binary)	95% (linear kernel)	–	90%	100%
1,2,3,4 (4-class)	90% (RBF kernel)	–	70%	50%

Table 4.1: Test set accuracy for each classification experiment. Classical SVM results reflect the best-performing kernel (linear or RBF) for that task. QSVM Feature Map A is the simpler quantum kernel, and Feature Map B the more complex one. (For 6 vs 9, only one quantum feature map was used in practice.)

4.1 6 vs 9 Classification Results

For the task of distinguishing digit 6 from digit 9, the classical SVM baseline already performed exceedingly well. Both a linear SVM and an RBF SVM achieved 95% accuracy on the test set (19/20 correct), misclassifying just one sample. This indicates the classes are almost linearly separable in the 4D PCA. The confusion matrix for the RBF SVM showed one false positive (a 6 mis-labeled as 9) and no false negatives. Given this strong performance, any quantum model would have to be nearly perfect to outperform it.v

Our QSVM with Feature Map A reached 90% accuracy (18/20 correct) on the same 6 vs 9 test set. It misclassified two of the digit 6 images as 9, while correctly classifying all digit 9 images. So, its confusion matrix had two errors in the row corresponding to actual 6's. This result – while slightly worse than the classical SVM – is still quite good, confirming that the QSVM did learn to differentiate 6 vs 9 to a large extent. The fact that QSVM lagged by 5% suggests that its decision boundary was not as tight as the classical SVM's.

It's worth noting that we did not test Feature Map B on 6 vs 9, because Feature Map A was already performing well and our initial aim was to use the simpler map for the first iteration.

In summary, for 6 vs 9, classical SVM slightly outperformed QSVM. The QSVM achieved comparable results (90% vs 95%), demonstrating that it can indeed learn the classification, but it did not exceed the classical approach. This is not too surprising – as the data was almost linearly separable, the classical model was already near-optimal. Thus, no quantum advantage was observed in this scenario; rather, it served as a baseline check that QSVM is at least performing reasonably.

4.2 Binary Classification: 1 vs 7

The second binary task, distinguishing digit 1 from digit 7, proved more challenging for both classical and quantum models. The classical SVM baseline achieved 95% accuracy (19/20 correct). This result suggests that even though 1 vs 7 would seem harder than 6 vs 9 (because of the similarity of the digits), our PCA preprocessing made it separable. At least one misclassification indicates some overlap remained, but the SVM found a decision boundary that handled most cases.

For QSVM, we ran experiments with both feature maps on 1 vs 7. Using the simpler Feature Map A, the QSVM reached 90% accuracy (18/20 correct), mirroring the pattern from 6 vs 9. QSVM (Map A) made two errors versus the classical SVM's one error. Essentially, with Feature Map A, the QSVM again worked well but did not outperform the classical baseline – it was slightly worse. Given the increased difficulty of 1 vs 7, we were particularly interested to see if a more expressive quantum kernel could close the gap or even surpass classical.

Indeed, with Feature Map B (advanced), the QSVM achieved a perfect 100% accuracy on 1 vs 7, classifying all 20 test images correctly. This was a significant result: the quantum kernel managed to carve out a decision boundary that separated the two classes with no errors on our test set, whereas the best classical model had one error. While we must be cautious (20 samples is tiny and 100% could be a bit of luck), this outcome suggests that Feature Map B provided additional flexibility that helped the QSVM fit the training data more accurately and generalize at least to our test points. With such a small dataset, overfitting can sometimes coincide with perfect generalization on test if the test happens to be simple, but it might not hold on larger tests. Nonetheless, it demonstrates that a carefully chosen quantum feature map can slightly outperform a classical SVM in this scenario. In percentage terms, QSVM (Map B) beat classical by 5% (100 vs 95), which is a modest gain, but notable given how tuned classical methods are for this task.

Overall for 1 vs 7, we see a quantum feature map can make a difference. Feature Map A was insufficient to beat classical, but Feature Map B succeeded in this limited test, achieving the only instance in our study where QSVM surpassed the classical baseline. This validates, in principle, the idea that a more expressive kernel can improve accuracy on a problem where the classes are not trivially separable. However, one must balance this with the risk of overfitting and the added computational cost. In a sense, Feature Map B "overpowered" the small dataset, fitting it perfectly. We would need more data to ensure this complex kernel is truly beneficial.

4.3 Multi-class Classification: 1 vs 2 vs 3 vs 4

The most challenging experiment was the 4-class classification with digits 1, 2, 3, and 4. Here the classical SVM with RBF kernel obtained 90% accuracy on the 20 test samples, making 2 mistakes. This result is quite impressive given only 20 training samples per class – evidently, the RBF SVM managed to capture some separation among these four digits in the PCA space.

Turning to QSVM, the results were mixed and illustrate the limitations of our approach. With Feature Map B, the QSVM's one-vs-rest scheme achieved 70% accuracy (14/20 correct). The confusion breakdown was as follows: all 5 instances of digit '1' were correctly identified, but other classes saw significant errors. Digit '2' was especially problematic – out of 5 '2' images, the QSVM only correctly classified 1. Most of the '2's were misclassified as '3' or '4', indicating the QSVM struggled to differentiate 2 from those. Digit '3' had about 71% recall (2 out of 7 total actual 3's were missed), and digit '4' had 75% recall (1 out of 4 missed). These misclassifications suggest the QSVM with the simple kernel could not carve out clear regions for each class in the feature space. The quantum kernel was not expressive enough to properly separate all four classes simultaneously. It perhaps separated "1" vs "not 1" well, but struggled with the fine-grained differences among 2, 3, and 4.

When we used Feature Map C (advanced) for the multi-class QSVM, we anticipated possibly better separation, as this map succeeded in the 1 vs 7 case. However, the result we got was actually worse: 50% accuracy (only 10/20 correct). This essentially amounts to random guessing for a 4-class classification. The confusion matrix in this case was quite poor: the QSVM with the complex kernel severely overfitted the training data and failed to generalize to test data. It might have effectively memorized a very convoluted decision boundary that does not align with the true class structure for unseen examples. Specifically, it misclassified all of the digit '2' and digit '4' test examples (precision and recall were 0 for one of the classes according to the classification report) and had low precision on others. The Feature Map C in a multi-class setting probably introduced so many high-order feature interactions that, with only 80 training points (20 per class), the QSVM decision surfaces became essentially nonsense for new points. It's analogous to using a very high-degree polynomial kernel on a small multi-class dataset – it will fit the training data's quirks (even noise) and lose predictive power.

Therefore, for the 4-class problem, the classical SVM decisively outperformed all QSVM variants. The best QSVM (Feature Map A) reached 70%, well below 90%, and the QSVM with Feature Map B was effectively unusable at 50%.

One factor to mention is that our QSVM multi-class was implemented as one-vs-rest. One-vs-rest with kernel SVMs can sometimes be inferior to one-vs-one. Our classical SVM was effectively one-vs-one (since scikit-learn's SVC default for multi-class is one-vs-one with voting), which is often considered more robust for multi-class SVM. It's possible that one-vs-rest QSVM is at a disadvantage, and a one-vs-one QSVM could have done a bit better (maybe approaching 70% or more). However, given the magnitude of difference, it's unlikely to bridge the gap to 90%. Also, the classical OvR SVM (if we forced it) likely would still be ¿70% in accuracy. So the multi-class task underscores that the QSVM provided no advantage in a more complex classification scenario – in fact, it was significantly worse than classical, especially when using an overly expressive feature map.

CONCLUSION

In this work, we have presented a proof-of-concept implementation of the Quantum Support Vector Machine (QSVM) for handwritten digit classification on a small MNIST subset. By encoding four PCA-reduced features into 4-qubit circuits and estimating the kernel via state-overlap measurements on the Qiskit AerSimulator, we demonstrated that QSVM can achieve accuracy comparable to that of a classical SVM: 90% vs. 95% on digits 6 vs. 9, and 100% vs. 95% on digits 1 vs. 7, while underperforming in the four-class task (70% vs. 90%). These results confirm that, under ideal (noise-free) simulation conditions, QSVM constitutes a viable hybrid quantum–classical classification pipeline, yet does not yet deliver a clear accuracy advantage over classical methods on such small-scale benchmarks.

Our experiments further highlight the critical role of quantum kernel design. A simple first-order feature map produced stable but unremarkable results, whereas a more expressive second-order map achieved perfect accuracy in one binary task yet suffered from severe overfitting in the multi-class scenario. This sensitivity mirrors the classical bias–variance trade-off and underscores the necessity of carefully balancing kernel expressiveness against generalization, especially in small-data regimes where heavy PCA preprocessing and $O(N^2)$ circuit simulations can negate potential quantum benefits. Moreover, simulating kernel estimation scales quadratically in the number of training points and offers no runtime advantage without access to real quantum hardware; queue limits and noise on current devices further constrain practical deployment.

Looking forward, realizing genuine quantum advantage will require (1) experimental validation on error-mitigated hardware to assess noise resilience, (2) scaling to larger datasets and qubit counts (e.g. via optimized simulators or distributed execution), and (3) the development of adaptive or hybrid kernel strategies—such as variational feature maps or classical–quantum kernel combinations—to tailor inductive bias and mitigate overfitting. As quantum processors mature and allow evaluation of classically intractable kernels, the insights from this study will inform the design of QSVMs capable of outperforming their classical counterparts on complex, high-dimensional tasks.

Bibliography

[1] Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747), 209–212.

[2] Li, K., Zhao, P., Dai, S., et al. (2024). Exploring the Impact of Quantum Computing on Machine Learning Performance. *Middle East Journal of Applied Science & Technology*, January 2024.

[3] Rana, A., Vaidya, P., & Gupta, G. (2021). A comparative study of quantum support vector machine algorithm for handwritten recognition with support vector machine algorithm. *Materials Today: Proceedings*, 56(1), in press.

[4] Kariya, A., & Behera, B. K. (2021). Investigation of Quantum Support Vector Machine for Classification in NISQ era. *arXiv preprint* arXiv:2112.06912.

[5] Bindhani, M., Behera, B. K., & Panigrahi, P. K. (2020). Hand Written Digit Recognition Using Quantum Support Vector Machine. Preprint. DOI:10.13140/RG.2.2.36253.74727.

[6] Carrascal de las Heras, G. (2025). *Quantum Support Vector Machine (QSVM).* Jupyter Notebook, 2025.