The complexity of quantum support vector machines

Gian Gentinetta[1,2], Arne Thomsen[3,2], David Sutter[2], and Stefan Woerner[2]

[1]Institute of Physics, École Polytechnique Fédérale de Lausanne
[2]IBM Quantum, IBM Research Europe – Zurich
[3]Department of Physics, ETH Zurich

Quantum support vector machines employ quantum circuits to define the kernel function. It has been shown that this approach offers a provable exponential speedup compared to any known classical algorithm for certain data sets. The training of such models corresponds to solving a convex optimization problem either via its primal or dual formulation. Due to the probabilistic nature of quantum mechanics, the training algorithms are affected by statistical uncertainty, which has a major impact on their complexity. We show that the dual problem can be solved in $\mathcal{O}(M^{4.67}/\varepsilon^2)$ quantum circuit evaluations, where M denotes the size of the data set and ε the solution accuracy compared to the ideal result from exact expectation values, which is only obtainable in theory. We prove under an empirically motivated assumption that the kernelized primal problem can alternatively be solved in $\mathcal{O}(\min\{M^2/\varepsilon^6, 1/\varepsilon^{10}\})$ evaluations by employing a generalization of a known classical algorithm called Pegasos. Accompanying empirical results demonstrate these analytical complexities to be essentially tight. In addition, we investigate a variational approximation to quantum support vector machines and show that their heuristic training achieves considerably better scaling in our experiments.

1 Introduction

Finding practically relevant problems where quantum computation offers a speedup compared to the best known classical algorithms is one of the central challenges in the field. Quantifying a speedup requires a provable convergence rate of the quantum algorithms, which restricts us to studying algorithms that can be analyzed rigorously. The impressive recent progress on building quantum computers gives us a new possibility: We can use heuristic quantum algorithms that can be run on current devices to demonstrate the speedup empirically. This however requires a hardware friendly implementation, i.e., a moderate number of qubits and shallow circuits.

In recent years, more and more evidence has been found supporting machine learning tasks as good candidates for demonstrating quantum advantage [1–4]. In particular, the so-called *supervised learning* setting, where in the simplest case the goal is to learn a binary classifier of classical data, received much attention. The reasons are manifold:

(i) The algorithms only require *classical access to data*. This avoids the common assumption that data is provided in the amplitudes of a quantum state which is hard

David Sutter: dsu@zurich.ibm.com; Gian Gentinetta and Arne Thomsen contributed equally and are listed in alphabetical order.

to justify [5]. It has been observed that if classical algorithms are provided with analogous sampling access to data, claimed quantum speedups may disappear [6, 7].

(ii) Kernelized support vector machines offer a framework that can be analyzed rigorously. In addition, this approach can be immediately lifted to the quantum setting by using a quantum circuit parametrized by the classical data as the feature map. This then defines a *quantum support vector machine* (QSVM) [2, 4].

(iii) It is known that, for certain (artificially constructed) data sets, QSVMs can offer a provable exponential speedup compared to any known classical algorithm [4]. Note, however, that according to current knowledge such QSVMs require deep circuits and, hence, rely on a fault-tolerant quantum computer.

Despite the growing interest in QSVMs, a rigorous complexity-theoretic treatment comparing the dual and primal approaches to training models employing arbitrary quantum kernels is missing (the analysis presented in [8] considers kernel functions defined by quantum states, but these are limited to classical polynomial- and RBF-kernels). In contrast to the classical case where the kernel function can be computed exactly, the runtime analysis of QSVMs is complicated by the fact that all kernel evaluations are unavoidably subject to statistical uncertainty stemming from finite sampling, even on a fully error corrected quantum computer. This is the case since the quantum kernel function is defined as an expectation value, which in practice can only be approximated as a sample mean over a finite number of measurement shots. Furthermore, it has recently been shown that quantum kernels can suffer from exponential concentration, if one is not careful with the construction of the feature map [9]. Then, an exponential number of measurements is necessary to distinguish the kernel evaluations. However, in this work, we focus on the training of the QSVM and, hence, assume that the feature map which embeds the data through a quantum circuit has been chosen reasonably (see assumption on noisy halfspace learning in [4, Lemma 14, Definition 15]). Being able to successfully train these models is a necessary prerequisite for the ultimate goals of quantum advantage and good generalization, which is why we think the training stage of the algorithms warrants a detailed investigation of its own.

The computational complexity of QSVMs is then defined as the total number of quantum circuit evaluations[1] necessary to find a decision function which, when evaluated on the training set, is ε-close to the solution employing an infinite number of measurement shots. Note that this definition solely in terms of training is legitimate since fitting the models is computationally dominant compared to evaluating the final classifier.

The beauty of QSVMs is that they allow the classification task to be characterized by an efficiently solvable convex optimization problem. Classically, the problem is especially simple in its dual form, where the kernel trick can be used to reduce the optimization to solving a quadratic program. The main drawback of the dual method is that the entire $M \times M$ kernel matrix needs to be calculated. When the kernel entries are affected by finite sampling noise, we find that the scaling is even steeper. In order to find an ε-accurate classifier $\mathcal{O}(M^{4.67}/\varepsilon^2)$ quantum circuit evaluations are found to be necessary in the setting of *noisy halfspace learning* [4, Lemma 14, Definition 15].[2] Here and throughout the manuscript, ϵ denotes an upper bound on the absolute difference between two fully

[1] Given a fixed feature map, the number of circuit evaluations is computationally dominant and therefore this definition of complexity is equivalent to the total runtime of finding such a solution.

[2] This is an polynomial improvement compared to the finding in [4, Lemma 19] which requires $\mathcal{O}(M^6/\varepsilon^2)$ circuit evaluations.

trained classifiers, where one is idealized and built using exact expectation values, and the other is constructed with a finite number of quantum circuit evaluations, which result in sample means that are subject to noise. The dependence on M poses a major challenge for problems with large data sets.

There are, however, two attempts to circumvent this potentially prohibitive scaling. Instead of solving the dual, we can make use of the probabilistic PEGASOS algorithm [10], which minimizes the primal formulation of the problem using stochastic gradient descent. Unlike standard solvers of the primal, the algorithm can be kernelized. Its main advantage is that instead of requiring the whole kernel matrix, PEGASOS is an iterative algorithm that only evaluates the kernel entries needed to compute the gradient in every step. We show that this results in $\mathcal{O}(\min\{M^2/\varepsilon^6, 1/\varepsilon^{10}\})$ measurement shots required to achieve an ε-accurate classification. Note that this runtime can be independent of M. However, compared to the dual, this comes at the cost of worse scaling in ε.

A second approach is to employ a variational quantum circuit containing d parameters that are trained to minimize a loss function evaluated on the training set. It has been shown that just like QSVMs, such models implement a linear decision boundary in feature space and can therefore be viewed as approximate QSVMs [2, 11]. For an appropriately chosen ansatz and a suitable training algorithm, the approximation quality gets better with increasing d. For this heuristic approach to training, we find an empirical scaling of $\mathcal{O}(1/\varepsilon^{2.9})$, which is again independent of M due to the use of stochastic gradient descent and marks a significant improvement over PEGASOS with respect to ε. The price to pay for this reduction in complexity is that the underlying optimization problem is non-convex and its solution therefore loses the guarantee of global optimality.

Results: Our findings are summarized in Table 1, where additional empirical scalings are included which show that the analytically bounds are essentially tight for practical data sets.

	dual	primal (PEGASOS)	approx. QSVM
classical equivalent	$\mathcal{O}(M^2)$	$\mathcal{O}(\min\{M/\varepsilon^2, 1/\varepsilon^4\})$	✗
analytical complexity	$\mathcal{O}(M^{4.67}/\varepsilon^2)$	$\mathcal{O}(\min\{M^2/\varepsilon^6, 1/\varepsilon^{10}\})$	✗
empirical complexity	$\mathcal{O}(M^{4.8\pm0.4}/\varepsilon^2)$	$\mathcal{O}(1/\varepsilon^{9.5\pm1.0})$	$\mathcal{O}(1/\varepsilon^{2.9\pm0.3})$

Table 1: **Complexity of QSVM training,** i.e. the asymptotic number of quantum circuit evaluations required to achieve an ε-accurate decision function on a training set of size M.

In Section 2 we give a brief overview of QSVMs, the optimization problems underlying their training, and the heuristic model designated approximate QSVM. The derivation of the analytical complexity statements for the dual and primal methods can be found in Sections 3.1 and 3.2, respectively. In Section 4, we present the numerical experiments justifying the empirical complexity statements.

2 Support vector machines

The machine learning task considered in this paper is the supervised training of binary classifiers on classical data using support vector machines (SVMs) [12–14]. Given an unknown probability distribution $P(\mathbf{x}, y)$ for data vectors $\mathbf{x} \in \mathbb{R}^r$ and class membership labels $y \in \{-1, 1\}$, we draw a set of training data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ with corresponding

labels $y = \{y_1, \ldots, y_M\}$. Using this set, the SVM defines a classification function $c : \mathbb{R}^r \to \{-1, 1\}$ implementing the trade-off between accurately predicting the true labels and maximizing the orthogonal distance (called *margin*) between the two classes. Once trained, the classification function can be applied to classify previously unseen data drawn from the same probability distribution. While this *generalization performance* of models is crucial when solving a concrete machine learning task, the sole focus of this work lies elsewhere, namely in the training of the models. Therefore, throughout this work only a training and not a test is considered.

We start by introducing classical kernelized support vector machines. The quantum case then constitutes a straightforward extension where the feature maps are defined by quantum circuits. To conclude the section, we present variational quantum circuits and interpret them as approximate quantum support vector machines.

2.1 Kernelized support vector machines

Support vector machines can implement nonlinear decisions in the original data space by transforming the input vectors with a nonlinear feature map. Here, we consider the finite dimensional case where the feature map is given by $\varphi : \mathbb{R}^r \to \mathbb{R}^s$. The decision boundary implemented by some $\mathbf{w} \in \mathbb{R}^s$ is then given by

$$c_{\mathrm{SVM}}(\mathbf{x}) = \mathrm{sign}\left[\mathbf{w}^\top \varphi(\mathbf{x})\right],$$

which is linear in feature space. We define the hyperplane $\mathbf{w}^\star$ as the solution to the primal optimization problem posed by support vector machines, which is given by

$$\text{(primal problem)} \qquad \min_{\mathbf{w}\in\mathbb{R}^s} \left\{ \frac{\lambda}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{M} \max\left\{0, 1 - y_i(\mathbf{w}^\top \varphi(\mathbf{x}_i))\right\} \right\}, \tag{1}$$

where the term containing $\lambda > 0$ provides regularization and the second term is a sum over the hinge losses of the data points contained in the training set.

Instead of solving the primal optimization problem, modern implementations [15] usually optimize the dual[3] of (1)

$$\text{(dual problem)} \qquad \begin{cases} \max\limits_{\alpha_i \in \mathbb{R}} & \sum_{i=1}^{M} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j \, k(\mathbf{x}_i, \mathbf{x}_j) - \frac{\lambda}{2}\sum_{i=1}^{M} \alpha_i^2 \\ \text{s.t.} & 0 \leq \alpha_i \quad \forall i = 1, \ldots, M \end{cases}. \tag{2}$$

The dual problem is typically favored because the *kernel trick* can be straightforwardly applied by defining the symmetric, positive semidefinite kernel function

$$k(\mathbf{x}, \mathbf{y}) := \varphi(\mathbf{x})^\top \varphi(\mathbf{y}), \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^r \tag{3}$$

as the inner product between feature vectors.

The solutions of the primal and dual optimization problems are connected via the Karush-Kuhn-Tucker condition as $\mathbf{w} = \sum_{i=1}^{M} \alpha_i y_i \varphi(\mathbf{x}_i)$ [4] and hence the classification function can be rewritten as

$$c_{\mathrm{SVM}}(\mathbf{x}) = \mathrm{sign}\left[\sum_{i=1}^{M} \alpha_i y_i \varphi(\mathbf{x}_i)^\top \varphi(\mathbf{x})\right] = \mathrm{sign}\left[\sum_{i=1}^{M} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x})\right], \tag{4}$$

[3] For technical reasons in the complexity analysis, we are not considering the exact dual of (1) here. Instead of using the ℓ^1 norm on the slack variables (one can show that this is equivalent to using the hinge loss like in (1)) , the dual in (2) corresponds to the use of the ℓ^2 norm in the primal. See [4, Equation C6] for a derivation.

from which it is apparent that both the dual optimization problem and the classification function only depend on the kernel function and feature vectors need not be explicitly computed at any point.

From (2), it follows that solving the dual requires the evaluation of the full kernel matrix $K \in \mathbb{R}^{M\times M}$ with entries

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) \quad \text{for} \quad i,j = 1,\ldots,M\,. \tag{5}$$

Note that given K, the dual optimization can be restated as a convex quadratic program (see (29) in Appendix A.2) and hence solved in polynomial time [16].

2.2 Primal Estimated sub-GrAdient SOlver (Pegasos)

Alternatively, we can solve the primal problem with an algorithm called Pegasos [17][4], which arises from the application of stochastic sub-gradient descent on the objective function $f(\mathbf{w})$ in the primal optimization problem (1). Starting with initial weights $\mathbf{w}^0 = \mathbf{0}$, we iteratively optimize $\mathbf{w}^t$ for $t \in \{1,\ldots,T\}$. In the following, we analyse how the weights are updated in every step.

As is standard in stochastic gradient descent, we start the optimization step t by uniformly sampling a random index $i_t \in \{1,\ldots,M\}$. We then define the partial objective f^t as the hinge-loss for the chosen datum $\mathbf{x}_{i_t} \in X \subset \mathbb{R}^r$ added to the regularization term

$$f^t(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \max\left[0, 1 - y_{i_t}\mathbf{w}^T\varphi(\mathbf{x}_{i_t})\right],$$

where $\varphi : \mathbb{R}^r \to \mathbb{R}^s$ denotes the feature map and λ is the regularization parameter. In order to find the steepest descent in the loss landscape defined by the partial objective f^t, we calculate the gradient of f^t with respect to $\mathbf{w}$ as

$$\frac{\partial f^t}{\partial \mathbf{w}} = \begin{cases} \lambda\mathbf{w}, & \text{if } y_{i_t}\mathbf{w}^T\varphi(\mathbf{x}_{i_t}) > 1 \\ \lambda\mathbf{w} - y_{i_t}\varphi(\mathbf{x}_{i_t}), & \text{otherwise.} \end{cases}$$

Next, we update the weights $\mathbf{w}^t = \mathbf{w}^{t-1} - \eta^t \left.\frac{\partial f^t}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}^{t-1}}$, where $\eta^t > 0$ is the learning rate. To further facilitate the calculation, we expand $\mathbf{w}^{t-1} = \frac{1}{\lambda(t-1)}\sum_{j=1}^{M}\alpha_j^{t-1}y_j\varphi(\mathbf{x}_j)$, leading to

$$\mathbf{w}^t = \begin{cases} \frac{1-\lambda\eta^t}{\lambda(t-1)}\sum_{j=1}^{M}\alpha_j^{t-1}y_j\varphi(\mathbf{x}_j), & \text{if } \frac{y_{i_t}}{t\lambda}\sum_{j=1}^{M}\alpha_j^{t-1}y_j k(\mathbf{x}_{i_t},\mathbf{x}_j) > 1 \\ \frac{1-\lambda\eta^t}{\lambda(t-1)}\sum_{j=1}^{M}\alpha_j^{t-1}y_j\varphi(\mathbf{x}_j) + \eta^t y_{i_t}\varphi(\mathbf{x}_{i_t}), & \text{otherwise.} \end{cases}$$

We can now fix the learning rate $\eta^t = 1/\lambda t$ to simplify the expression as

$$\mathbf{w}^t = \frac{1}{t\lambda}\sum_{j=1}^{M}\alpha_j^t y_j\varphi(\mathbf{x}_j),$$

where the coefficients $\boldsymbol{\alpha}^t$ are defined as $\alpha_j^t = \alpha_j^{t-1}$ for $j \neq i_t$ and

$$\alpha_{i_t}^t = \begin{cases} \alpha_{i_t}^{t-1}, & \text{if } \frac{y_{i_t}}{t\lambda}\sum_{j=1}^{M}\alpha_j^{t-1}y_j k(\mathbf{x}_{i_t},\mathbf{x}_j) > 1 \\ \alpha_{i_t}^{t-1} + 1, & \text{otherwise.} \end{cases}$$

[4] An implementation recently added to Qiskit [18] can be found here.

Crucially, the feature map is only accessed via the kernel entries, i.e. the PEGASOS algorithm allows for a kernelization of the primal optimization problem. While in theory we calculate the gradient of f^t with respect to $\mathbf{w}$, in practice $\mathbf{w}$ is never explicitly calculated. Instead, it suffices to calculate and store the integer coefficients $\boldsymbol{\alpha}^t$ both for training and prediction.

Algorithm 1 Kernelized PEGASOS [17, Figure 3]

Inputs:
training data $T = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M\}$
labels $L = \{y_1, y_2, ..., y_M\}$
regularization parameter $\lambda \in \mathbb{R}^+$
number of steps $T \in \mathbb{N}$

Initialize: $\boldsymbol{\alpha}_1 \leftarrow \mathbf{0} \in \mathbb{N}^M$

for $t = 1, 2, ..., T$ **do**
  Choose $i_t \in \{0, ..., M\}$ uniformly at random.
  for all $j \neq i_t$ **do**
    $\boldsymbol{\alpha}_{t+1}[j] \leftarrow \boldsymbol{\alpha}_t[j]$
  end for
  if $y_{i_t} \frac{1}{\lambda t} \sum_{j=1}^{M} \boldsymbol{\alpha}_t[j] y_j \, k(\mathbf{x}_{i_t}, \mathbf{x}_j) < 1$ **then**
    $\boldsymbol{\alpha}_{t+1}[i_t] \leftarrow \boldsymbol{\alpha}_t[i_t] + 1$
  else
    $\boldsymbol{\alpha}_{t+1}[i_t] \leftarrow \boldsymbol{\alpha}_t[i_t]$
  end if
end for

Output: $\boldsymbol{\alpha}_{T+1}$

2.3 Quantum support vector machines

The classical SVM formulation can be straightforwardly adapted to the quantum case by choosing a feature map

$$\begin{aligned} \psi \colon \mathbb{R}^r &\to \mathcal{S}(2^q) \\ \mathbf{x} &\mapsto |\psi(\mathbf{x})\rangle\langle\psi(\mathbf{x})| \,, \end{aligned}$$

where $\mathcal{S}(2^q)$ denotes the space of density matrices on q qubits [2]. The kernel function is then given by the Hilbert-Schmidt inner product

$$k(\mathbf{x}, \mathbf{y}) = \mathrm{tr}\big[\, |\psi(\mathbf{y})\rangle\langle\psi(\mathbf{y})| \; |\psi(\mathbf{x})\rangle\langle\psi(\mathbf{x})| \,\big] = |\langle\psi(\mathbf{x})|\psi(\mathbf{y})\rangle|^2 \,, \tag{6}$$

and can be estimated with the help of a quantum computer as illustrated in Figure 1 [4,11]. QSVMs are an active research topic [19–21].

The major difference between the classically computed kernel in (3) and the quantum one in (6) is that the latter expression can only be evaluated approximately with a finite number of measurement shots, due to the probabilistic nature of quantum mechanics. Hence, to prove a complexity statement for QSVMs, it is crucial to understand the robustness of the primal and dual optimization problems with respect to noisy kernel evaluations. A detailed analysis is included in Section 3.

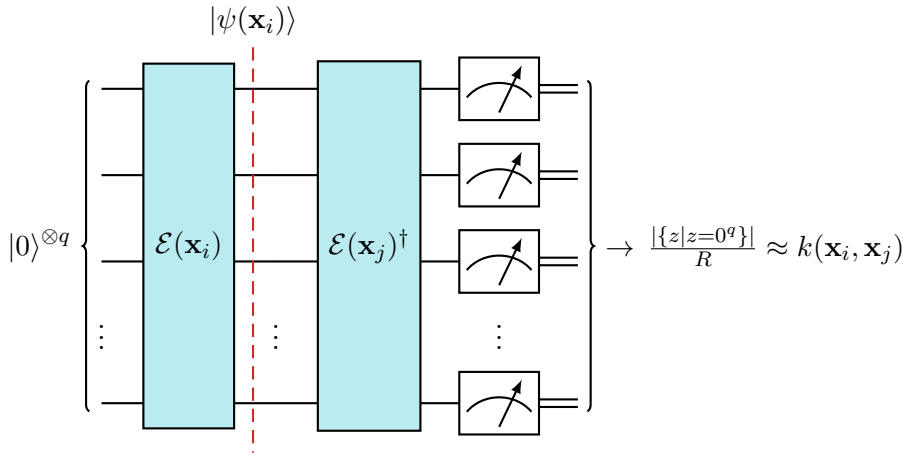


Figure 1: **Quantum kernel estimation** [4,11]**:** Let $\mathcal{E}(\mathbf{x}_i)$ denote a parametrized unitary fixed by the datum $\mathbf{x}_i$, which defines the feature map $|\psi(\mathbf{x}_i)\rangle = \mathcal{E}(\mathbf{x}_i)\,|0\rangle^{\otimes q}$. By preparing the state $\mathcal{E}(\mathbf{x}_j)^\dagger\mathcal{E}(\mathbf{x}_i)\,|0\rangle^{\otimes q}$ and then measuring all of the qubits in the computational basis, a bit string $z \in \{0,1\}^q$ is determined. When this process is repeated R-times, the frequency of the all zero outcome approximates the kernel value $k(\mathbf{x}_i, \mathbf{x}_j)$ in (6).

Note that while such noisy kernel evaluations are necessary for both training and prediction, the complexity of the prediction step is $\mathcal{O}(S^2/\varepsilon^2)$ [11] and subdominant compared to training (see table 1) as $S \leq M$ and often even $S \ll M$ for the number of support vectors S [22]. Therefore, our present analysis is restricted to fitting the support vector machines to a training set.

2.4 Approximate quantum support vector machines

Compared to the analytically tractable QSVM formulations introduced in Section 2.3, the heuristic model we denote *approximate QSVM* defined in the following, potentially offers favorable computational complexity. The main idea is that, in addition to the feature map encoding the data $|\psi(\mathbf{x})\rangle = \mathcal{E}(\mathbf{x})\,|0\rangle^{\otimes q}$ analogously to the QSVM, we implement a variational unitary $\mathcal{W}(\theta)$, whose parameters $\theta \in \mathbb{R}^d$ are trained to solve the classification problem. Figure 2 shows a quantum circuit implementing this architecture.

We define the decision function as the expectation value of the q-fold Z Pauli-operator [2]

$$h_\theta(\mathbf{x}) \coloneqq \langle\psi(\mathbf{x})|\,\mathcal{W}(\theta)^\dagger\, Z^{\otimes q}\,\mathcal{W}(\theta)\,|\psi(\mathbf{x})\rangle \in [-1,+1]\,, \tag{7}$$

which together with the introduction of a learnable bias $b \in \mathbb{R}$ is used to classify the data according to

$$c_\theta(\mathbf{x}) \coloneqq \operatorname{sign}\left[h_\theta(\mathbf{x}) + b\right]. \tag{8}$$

Given a loss function $\mathcal{L}$, we use a classical optimization algorithm to minimize the empirical risk on the training data X

$$\hat{\mathcal{R}}(c_\theta, X) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}(y_i, c_\theta(\mathbf{x}_i))\,.$$

The resulting model implements a linear decision boundary in the feature space accessed through the feature map circuit $\mathcal{E}$ and in this sense approximates a QSVM employing the same feature map (see [2] and [11, Section 2.3.1]). Note that the model above is just one possible definition to perform binary classification with the circuit depicted in Figure 2.

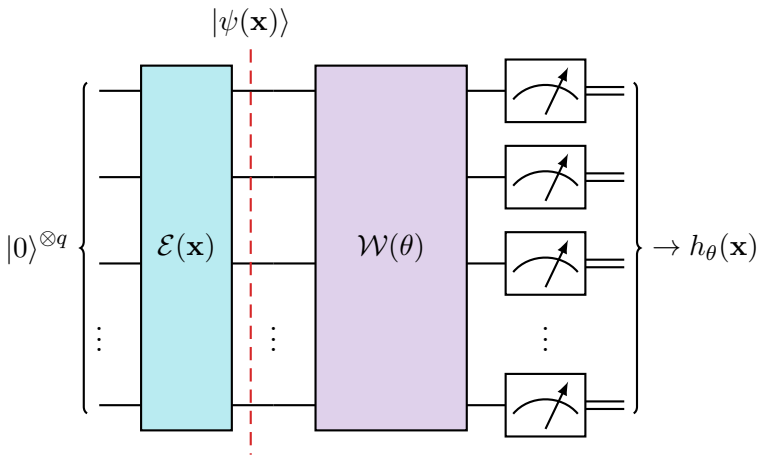


Figure 2: **Approximate QSVM** [2, 11]**:** The classical datum $\mathbf{x}$ is encoded with a feature map circuit $\mathcal{E}(\mathbf{x})$ analogous to the one in Figure 1. However, the resulting state is then acted upon by a variational circuit $\mathcal{W}(\theta)$, after which measurement in the computational basis is performed to determine the expectation value in (7).

One promising alternative is considering a local instead of a global observable. For both cases the model has to be chosen carefully to avoid unfortunate properties such as Barren plateaus [23].

The approximate QSVM model introduced above are also referred to as *quantum neural networks* [3, 24] or as *variational quantum circuits* [2, 25] in the literature.

3 Analytical complexity

Having introduced the models and associated optimization problems considered in this work, we now turn to analyzing their complexity. The goal of this section is to justify the analytical part of Table 1. For the scaling of the dual approach we are able to provide a mathematical proof. The scaling of the primal approach using Pegasos can also be analytically derived as long as an additional assumption on the data and feature map is fulfilled. While this assumption is not mathematically proven, we provide empirical evidence to support it.

3.1 Dual optimization

To solve the dual problem (2), the full kernel matrix K, whose elements are given as expectation values which are calculated on a quantum computer (see Figure 1), needs to be evaluated. As we can only perform a finite number of measurement shots in practice, the estimated kernel entries will always be affected by statistical errors, even when a fault tolerant quantum computer is employed. More precisely, we approximate each kernel entry by taking the sample mean

$$k_R(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{R} \sum_{l=1}^{R} \hat{k}_{ij}^{(l)}, \tag{9}$$

over R i.i.d. realizations of the random variable $\hat{k}_{ij}^{(l)}$, which is defined to be equal to 1 if the original all zero state $|0\rangle^{\otimes q}$ is recovered in the measurement, and equal to 0 for all other measurement outcomes. These entries define an approximate kernel matrix K_R, which in

the limit $R \to \infty$ converges to the true kernel matrix K by the law of large numbers. To prove a complexity statement for the dual approach, we need to quantify the speed of this convergence.

A detailed analysis which is shifted to Appendix A.1 for improved readability shows that the expected error on the kernel when using R samples per entry scales as

$$\mathbb{E}\left[\|K_R - K\|_2\right] = \mathcal{O}\left(\sqrt{\frac{M}{R}}\right). \tag{10}$$

Note that throughout this work, the notation $\|\cdot\|$ denotes the Euclidean vector norm and $\|\cdot\|_2$ the operator norm it induces.

In a next step, we analyze how this error propagates through the optimization problem (2) to the decision function (4). To make this step mathematically precise we need to employ an assumption on our kernel and data set.

Assumption 1. Our kernel and data set satisfy the noisy halfspace learning assumption defined in [4, Lemma 14].

Note that Assumption 1 is necessary to rigorously analyze the error propagation as done in Appendix A.2. Furthermore, as shown in [4] the assumption is reasonable for some kernels.

Denote by $\alpha_i^\star$ and $\alpha_{R,i}^\star$ the solution to the optimization problem utilizing an exact and noisy kernel matrix respectively. These solutions then lead to the terms

$$h(\hat{\mathbf{x}}) \coloneqq \sum_{i=1}^{M} \alpha_i^\star y_i \, k(\hat{\mathbf{x}}, \mathbf{x}_i) \qquad \text{and} \qquad h_R(\hat{\mathbf{x}}) \coloneqq \sum_{i=1}^{M} \alpha_{R,i}^\star y_i \, k_R(\hat{\mathbf{x}}, \mathbf{x}_i)\,,$$

inside the sign operator in the classification function (4), where $\hat{\mathbf{x}}$ is an arbitrary feature vector. We also refer to $h(\mathbf{x})$ as the decision function. We find that

$$|h(\hat{\mathbf{x}}) - h_R(\hat{\mathbf{x}})| = \mathcal{O}\left(\frac{M^{4/3}}{\sqrt{R}}\right).$$

Thus, in order to achieve an error of $|h(\hat{\mathbf{x}}) - h_R(\hat{\mathbf{x}})| \leq \varepsilon$, we need a total of

$$R_{\text{tot}} = \mathcal{O}\left(\frac{M^{4.67}}{\varepsilon^2}\right) \tag{11}$$

measurement shots, as the full symmetric kernel matrix consists of $\mathcal{O}(M^2)$ independent kernel entries. Once the approximated kernel matrix K_R has been calculated, the corresponding quadratic problem can be solved on a classical computer in time $\mathcal{O}\left(M^{7/2}\log(1/\varepsilon)\right)$[5]. Hence, estimating the kernel matrix is the computationally dominant part of the algorithm and the result in (11) defines the overall complexity for the dual approach.

3.2 Primal optimization via Pegasos

In order to analyze the computational complexity of solving the primal optimization problem (1), let $f(\mathbf{w})$ denote its objective function. Classically, where the kernels are known exactly, the runtime of the kernelized Pegasos-algorithm scales as $\mathcal{O}(M/\delta)$, where

$$\delta \coloneqq \left| f(\mathbf{w}^\star) - f(\mathbf{w}^P) \right| \tag{12}$$

[5]Quadratic programs are a special case of second order cone problems which can be solved in $\mathcal{O}\left(M^{7/2}\log(1/\varepsilon)\right)$ using interior point methods [16].

is the difference in the objective between the true optimizer $\mathbf{w}^\star$ and the result of Pegasos $\mathbf{w}^P$ [17]. When using quantum kernels, however, we have to consider the inherent finite sampling noise afflicting their evaluation. In order to carry out the complexity analysis for such noisy kernels, we next impose an assumption on Algorithm 1 for which we have empirical evidence as provided in Appendix B.1.

Assumption 2. The convergence of Algorithm 1 is unaffected if the sum in line 14 of Algorithm 1 is only δ-accurate for a sufficiently small $\delta > 0$.

In other words, this means that above a certain approximation accuracy, Algorithm 1 is robust to the statistical uncertainty inherent to quantum kernels which have been evaluated using a finite number of measurement shots.

Thus, we now analyze how many measurement shots are required for the sum in line 14 to be δ-accurate in terms of the standard deviation. First, note that initially $\boldsymbol{\alpha}_1[j] = 0$ for all j and in every iteration at most one component of $\boldsymbol{\alpha}$ becomes non-zero (in line 15). At the t-th iteration, the sum therefore contains at most t-terms. In order for the sum to be δ-accurate, every individual term should thus be $(\delta/\sqrt{t})$-accurate, so we need t/δ^2 measurement shots per term and t^2/δ^2 shots to evaluate the whole sum. Bearing in mind that the total number of iterations T is bounded by $T = \mathcal{O}(1/\delta)$ [17], this results in a total of $\mathcal{O}(T^3/\delta^2) = \mathcal{O}(1/\delta^5)$ quantum circuit evaluations to train a QSVM with Pegasos. Alternatively, the number of non-zero terms in the sum conditioned on in line 14 can be bounded by M, leading to a total of $\mathcal{O}(TM^2/\delta^2) = \mathcal{O}(M^2/\delta^3)$ measurement shots. Combining these results implies $R_{\text{tot}} = \mathcal{O}(\min\{M^2/\delta^3, 1/\delta^5\})$.

To directly compare this scaling with (11), we need to clarify the relation between the error

$$\varepsilon \coloneqq \max_{\hat{\mathbf{x}} \in X} |h(\hat{\mathbf{x}}) - h_P(\hat{\mathbf{x}})| \tag{13}$$

and δ defined in (12), where h is the ideal decision function and h_P the decision function resulting from the noisy Pegasos algorithm. As the SVM optimization problem is λ-strongly convex for λ the regularization constant, this connection can be derived as

$$\varepsilon \leq \sqrt{\frac{2\delta}{\lambda}}, \tag{14}$$

see Appendix B.2.

Combining these results, the number of measurement shots required to achieve ε-accurate decision functions is bounded by

$$R_{\text{tot}} = \mathcal{O}\left(\min\left\{\frac{M^2}{\lambda^3\varepsilon^6}, \frac{1}{\lambda^5\varepsilon^{10}}\right\}\right).$$

4 Empirical complexity

In this section we provide experiments justifying the empirical part of Table 1. In Section 4.1, we define the training data used in the following experiments. In Sections 4.2 and 4.3, we respectively show that the analytical complexity for the dual and primal methods are confirmed in our experiments as close to being tight. Additionally, we provide an empirical scaling for the approximate QSVMs in Section 4.4. Table 2 includes the results of the experiments performed in this section for the two settings described in Section 4.1.

	dual	primal (PEGASOS)	approximate QSVM
separable data	$\mathcal{O}(M^{4.8\pm0.4}/\varepsilon^2)$	$\mathcal{O}(1/\varepsilon^{8.3\pm1.6})$	$\mathcal{O}(1/\varepsilon^{2.9\pm0.3})$
overlapping data	$\mathcal{O}(M^{4.5\pm0.3}/\varepsilon^2)$	$\mathcal{O}(1/\varepsilon^{9.5\pm1.0})$	$\mathcal{O}(1/\varepsilon^{2.8\pm0.3})$

Table 2: **Empirical complexities of QSVMs**, i.e. the total number of quantum circuit evaluations required to achieve an ε-accurate solution for the decision function resulting from the dual, primal, and approximate QSVM approach. We consider a data set of size M.

4.1 Training data

The training data used throughout the following experiments is artificially generated according to Algorithm 2 in the Appendix with respect to a fixed feature map of the architecture illustrated in Figure 3. Unless declared otherwise, the experiments are run on 8 qubits and the data thus consists of 8 features. In particular, we consider two settings: data sets that are linearly separable in feature space and ones where there is an overlap between the classes. One realization per setting is illustrated in Figure 4.

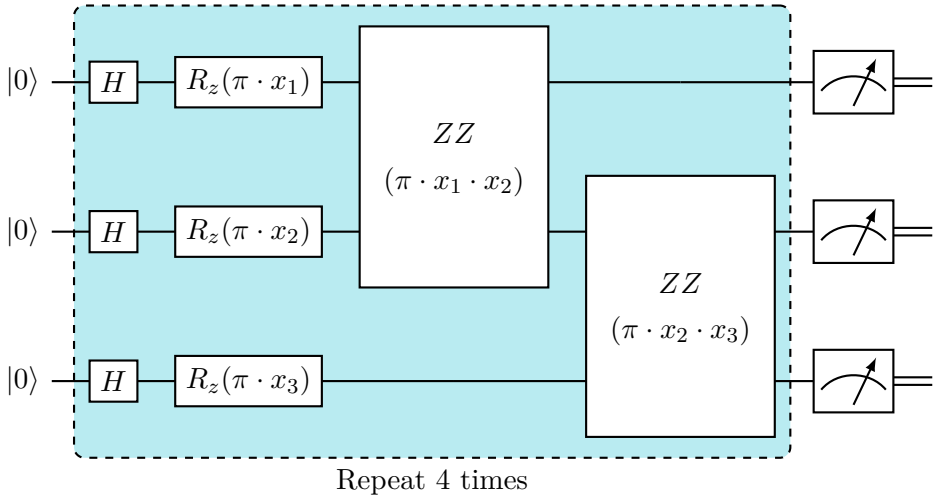
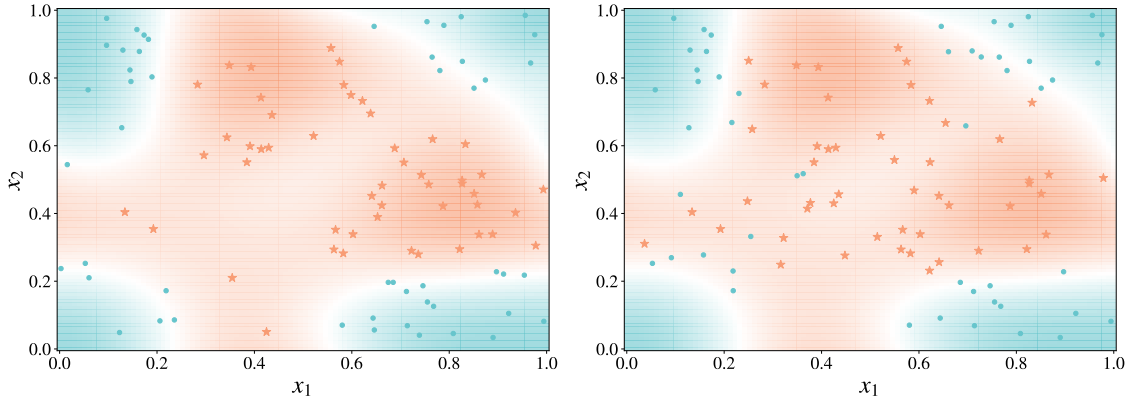


Figure 3: **Feature map circuit:** This quantum circuit (scaled to 8 qubits with nearest-neighbour entanglement) is employed as the feature map in the QSVMs throughout Section 4. H denotes a Hadamard-gate, $R_z(\theta)$ a single-qubit rotation about the z-axis, and $ZZ(\theta)$ a parametric 2-qubit $Z \otimes Z$ interaction (maximally entangled for $\theta = \pi/2$). The feature components x_1 and x_2 (where $\mathbf{x} = (x_1, x_2)$ is a classical input datum) provide the angles for the rotations. This feature map was chosen due to its previous use in the literature [2, 3].



(a) Linearly separable data

(b) Overlapping data

Figure 4: **Artificial data:** The training data is generated using Algorithm 2 with the feature map from Figure 3 and a sample size $M = 100$. The linearly separable data is achieved by setting the margin positive ($\mu = 0.1$ in this case), while the margin for the overlapping data is negative ($\mu = -0.1$). The colouring in the background corresponds to the fixed classifier used to generate the data (note that this is not necessarily the ideal classifier for the generated data points). To allow visualisation, the data displayed here is generated for 2 qubits according to the same algorithm.

4.2 Dual optimization

In addition to the complexity-theoretic scaling derived in Section 3.1, we provide an empirical scaling for the dual optimization problem based on numerical experiments. Using shot based noisy kernels, we first determine the ε-dependence of the runtime. In a separate experiment, we analyze the M-dependence, which the theory indicates to pose the main limitation of the method.

For the ε-dependence, we fix the data size to $M = 256$. First, the exact kernel is calculated with a statevector simulator [18]. The quadratic program (29) is solved with help of the `quadprog` python library [26] in order to find a reference decision function $h_\infty(\mathbf{x})$ corresponding to an infinite number of measurement shots. In the next step, we emulate the shot based kernel entries[6]. We know that for the exact kernel entries K_{ij}, the probability of measuring the all $|0\rangle$ outcome is equal to $|\langle\psi(\mathbf{x}_i)|\psi(\mathbf{x}_j)\rangle|^2 = K_{ij}$. We can thus emulate a sample mean of R measurement shots using the binomial distribution $B(R, K_{ij})$. This procedure is repeated for values of R ranging from 10^9 to 10^{19} [7] to find the noisy decision functions $h_R(\mathbf{x})$. The error is then defined as

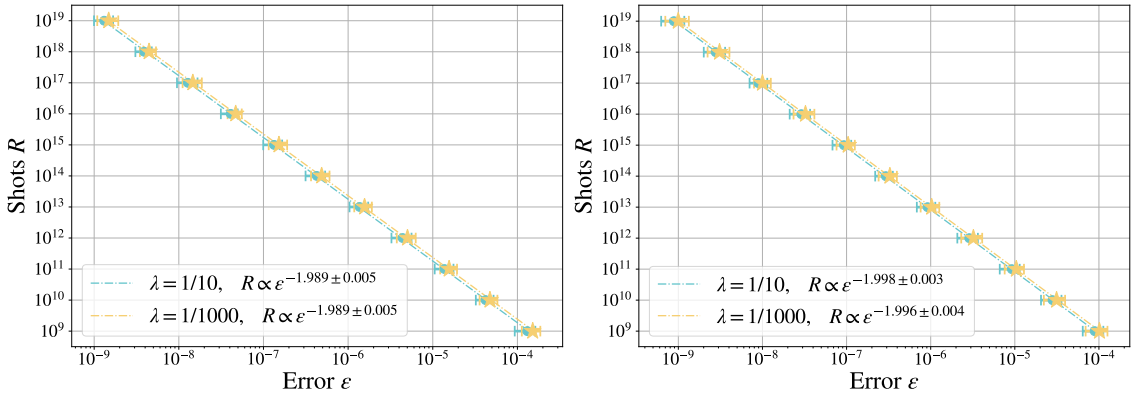
$$\varepsilon := \max_{\mathbf{x}_i \in X} |h_R(\mathbf{x}_i) - h_\infty(\mathbf{x}_i)| \ . \tag{15}$$

Analyzing the plots in Figure 5, we find that the bound $R = \mathcal{O}(1/\varepsilon^2)$ predicted in Section 3.1 is tight and in precise agreement with the experiments.

In the second experiment, we analyze how the runtime of the dual optimization problem scales with M. To this end, the dual problem is solved for different data sizes M and shots per kernel evaluation R to calculate the corresponding error $\varepsilon(M, R)$ according to (15). We then fix some $\varepsilon_0 > 0$ and define $R_{\varepsilon_0}(M)$ as the smallest R such that $\varepsilon(M, R) < \varepsilon_0$,

[6] Emulating the noisy kernel in this way is equivalent to using a QASM simulator but computationally more efficient.

[7] The numbers of shots are chosen so large in order for the condition $||K - K_R|| < \mu$ in Theorem 4 to be fulfilled. For smaller R, we would need to pre-process K_R to be positive definite in some way, further complicating the analysis.



(a) Linearly separable data

(b) Overlapping data

Figure 5: **ε-scaling of the dual:** Using noisy kernel evaluations drawn from a binomial distribution to simulate the number of shots R, the kernel matrix K_R is constructed and the dual optimization problem solved. The number of shots is plotted as a function of ε on a doubly logarithmic scale, where ε is calculated according to (15). A linear fit inside the log-log plot is then used to determine the empirical exponent. The experiment is repeated for the different regularization parameters $\lambda = 1/10$ and $\lambda = 1/1000$ and run on $n = 100$ different realizations of the training data (see Section 4.1). The markers shown are the means over the different runs and the horizontal error bars correspond to the interval between the 15.9 and 84.1 percentile.

i.e. $R_{\varepsilon_0}(M)$ is the minimal number of shots needed to achieve an ε_0-accurate solution. In Figure 6, a log-log plot of $R_{\text{tot}} = R_{\varepsilon_0}(M)\,[M(M+1)/2]$[8] as a function of M for different ε_0 is included. From a linear least squares fit in this plot the empirical scaling is determined as $R_{\text{tot}} = \mathcal{O}(M^{4.8\pm0.4})$ for separable data and $R_{\text{tot}} = \mathcal{O}(M^{4.5\pm0.3})$ for overlapping data.

4.3 Primal optimization via Pegasos

In order to determine the empirical scaling of the Pegasos algorithm, we observe how the error ε of the decision function changes when the number of shots per evaluation is varied. Again employing the feature map and training data described in Section 4.1 (with size $M = 100$), we first train a reference decision function $h_\infty(\mathbf{x})$ by running Pegasos for $T = 1000$ iterations with a statevector simulator. This number of steps is sufficient for convergence in our case as can be seen in Figure 8. In a next step, QASM-simulators [18] with fixed number of shots per kernel evaluation R are used to run Pegasos. After every iteration t, we calculate the hinge loss

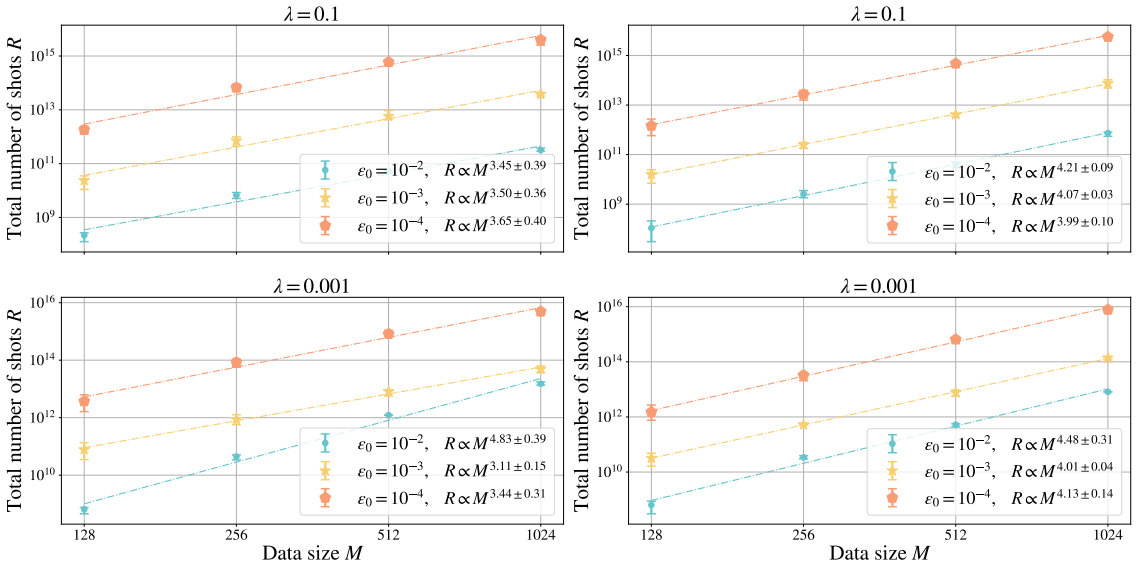
$$\mathcal{L}_R^t = \frac{1}{M}\sum_{i=1}^{M} \max\left\{0, 1 - y_i h_R^t(\mathbf{x}_i)\right\} ,$$

where $h_R^t(\mathbf{x})$ is the decision function after t iterations and y_i the true label. This procedure is repeated for T iterations until convergence, which is defined to be reached when

$$\left|\mathcal{L}_R^t - \mathcal{L}_R^{t-1}\right| < \tau \tag{16}$$

for some tolerance which is fixed to $\tau = 10^{-4}$ for this experiment. Finally, the decision function after convergence h_R^T is compared to the reference decision function, defining an

[8] The factor $[M(M+1)/2]$ comes from the number of independent entries in the symmetrix $M \times M$ kernel matrix.



(a) Linearly separable data

(b) Overlapping data

Figure 6: **M-scaling of the dual:** It is plotted how the minimal number of total shots necessary to achieve an ε_0-accurate decision functions depends on M. The experiment is repeated for the different regularization parameters $\lambda = 1/10$ and $\lambda = 1/1000$. The markers shown are the means and the error bars indicate the interval between the 15.9 and 84.1 percentile.

error

$$\varepsilon := \max_{\mathbf{x}_i \in X} \left| h_R^T(\mathbf{x}_i) - h_\infty(\mathbf{x}_i) \right|. \tag{17}$$

Figure 7 shows the relationship between ε and R. Figure 8 further suggests that the number of iterations needed until convergence is independent of the number of shots R per kernel evaluation for sufficiently large R. Thus, we can assume the total number of shots to scale as $R_{\text{tot}} = TR$ for some constant $T > 0$. Under this assumption, the experiments yield an empirical scaling of

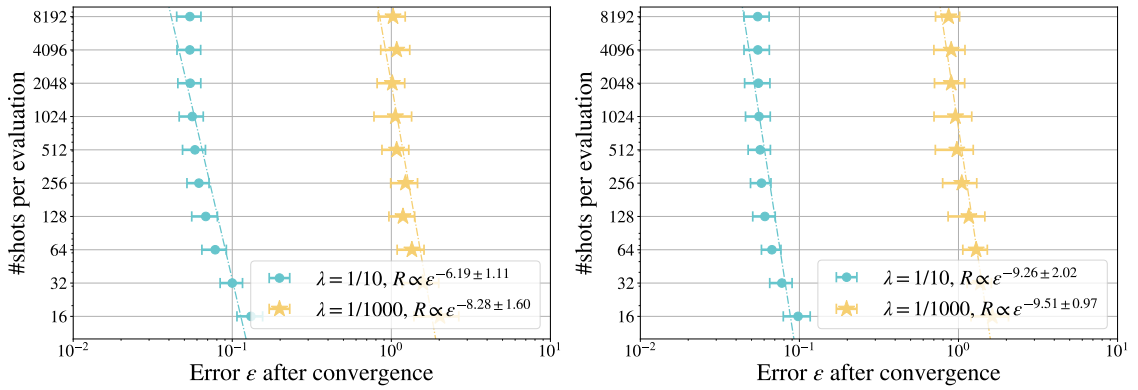
$$R_{\text{tot}} = \mathcal{O}(R) \approx \mathcal{O}\left(\frac{1}{\varepsilon^{9.5\pm1.0}}\right)$$

with respect to ε.

In addition to the ε-dependence of the complexity of Pegasos, we should also remark on the M-dependence. Figure 8 shows that for large enough R, the number of steps needed to converge is independent of R. Thus, it suffices to analyze how fast Pegasos converges for varying M when a statevector simulator corresponding to an infinite number of measurement shots is used. Figure 9 shows that there is no discernible correlation between the number of steps until convergence of Pegasos and the data size M as expected from the theoretical analysis in Section 3.2.

4.4 Heuristic training of approximate QSVMs

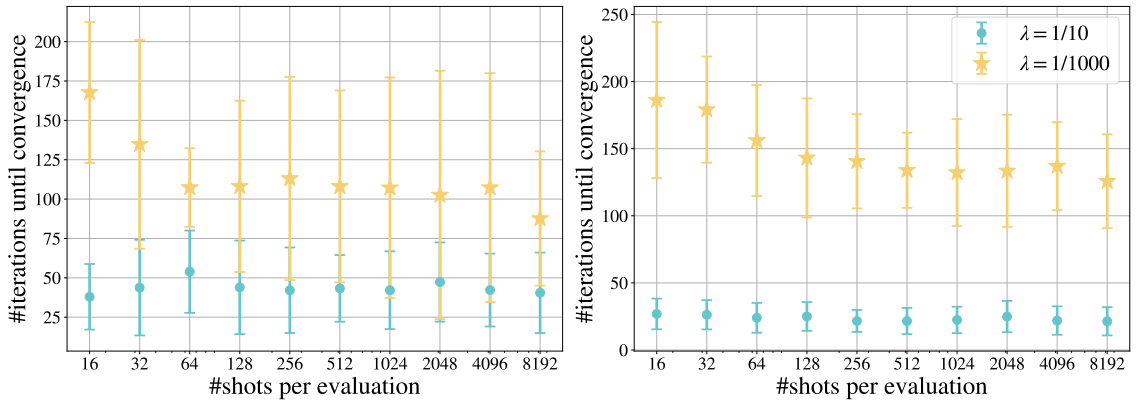
Before we tackle the empirical scaling of the non-convex optimization problem that the training of approximate QSVMs poses, we analyze how the expectation value of $h_\theta(\mathbf{x})$ defined in (7) is affected by finite sampling statistics. Let $Q \in \{-1, 1\}$ denote the random measurement outcome, then $\frac{1}{2}(1+Q) \sim \text{Bernoulli}(p)$ follows a Bernoulli distribution where p is the probability that $Q = 1$. For an i.i.d. sample of size R, the standard deviation



(a) Linearly separable data

(b) Overlapping data

Figure 7: **ε-scaling of Pegasos:** Using QASM-simulators with 16 to 8192 shots per evaluation, the Pegasos algorithm is run for 750 iterations to ensure convergence. The number of shots is plotted as a function of ε on a doubly logarithmic scale, where ε is calculated as defined in (17). A linear fit is used to determine the empirical exponent. The experiment is repeated for the different regularization parameters $\lambda = 1/10$ and $\lambda = 1/1000$. Every data point corresponds to the mean over 50 random runs of the Pegasos-algorithm and the error bars indicate the standard deviation.



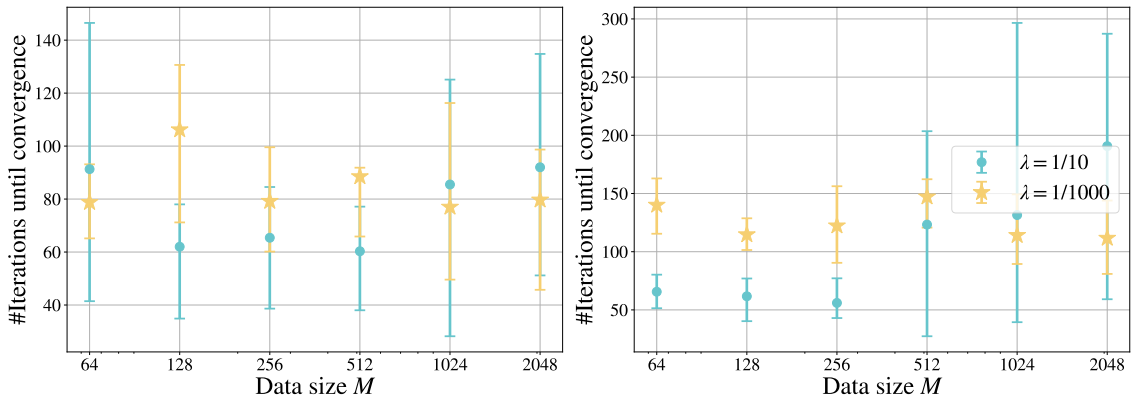
(a) Linearly separable data

(b) Overlapping data

Figure 8: **Convergence of Pegasos:** Like in Figure 7, QASM-simulators ranging from 16 to 8192 shots per evaluation are employed to run Pegasos until convergence as defined in (16). The number of iterations until convergence is plotted as a function of the number of shots per kernel evaluation. The experiment is repeated for the different regularization parameters $\lambda = 1/10$ and $\lambda = 1/1000$. For every plotted data point, 50 runs are performed using different random seeds for the choice of indices in the Pegasos algorithm. The markers shown are the means of the respective steps until convergence and the error bars mark the standard deviation.

of the sample mean $Q_R \coloneqq \frac{1}{R}\sum_{i=1}^{R} Q_i$ is given as $\sigma_{Q_R} = \sigma_Q/\sqrt{R}$, where σ_Q denotes the standard deviation of the distribution from which the Q_i are sampled. With $\sigma^2_{\text{Bernoulli}} = p(1-p) \leq 1/4$ we find $\sigma_Q^2 = 4\sigma^2_{\text{Bernoulli}} \leq 1$ and, thus, if we require the standard deviation to be bounded by $\sigma_{Q_R} = \mathcal{O}(\varepsilon)$ in order to get a confidence interval of width $\mathcal{O}(\varepsilon)$, we need to perform $R = \mathcal{O}(1/\varepsilon^2)$ measurement shots per expectation value. Assuming that the training converges in $\mathcal{O}(1/\varepsilon)$ steps[9], the expected total number of shots required to fit the

[9]Although the theory is not applicable here, we assume a convergence rate of $\mathcal{O}(1/\varepsilon)$ like first order methods for convex problems [27].



(a) Linearly separable data

(b) Overlapping data

Figure 9: **M-scaling of Pegasos:** Pegasos is applied on the artificial data introduced in Section 4.1 for different training set sizes M. The number of iterations until convergence using a statevector simulator is plotted as a function of M. The experiment is repeated for the two regularization parameters $\lambda = 1/10$ and $\lambda = 1/1000$. Every data point is the mean over 10 different runs of the probabilistic Pegasos-algorithm and the error bars mark the interval between the 15.9 and 84.1 percentile.

model to the training set then scales as

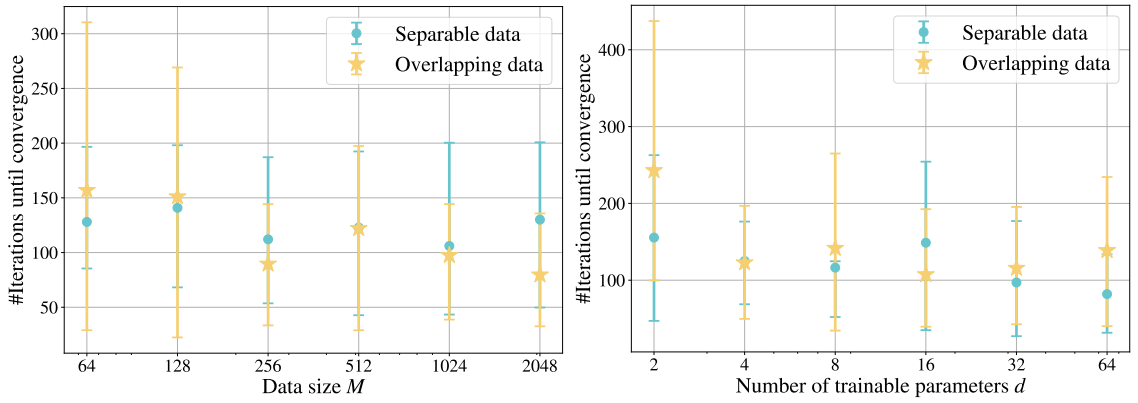
$$R_{\text{tot}} = \mathcal{O}\left(\frac{1}{\varepsilon^3}\right). \tag{18}$$

Because the optimization problem is non-convex, a rigorous proof of the conjectured scaling in (18) remains an unsolved problem. Instead, we perform numerical experiments to determine an empirical scaling. The circuit employed to implement the approximate QSVM is of the form shown in Figure 2, where the unitaries $\mathcal{E}(\mathbf{x})$ and $\mathcal{W}(\theta)$ are `ZZFeatureMap` and `RealAmplitudes` circuits, respectively, as provided by the Qiskit Circuit Library [18]. The experiment is performed on both separable and overlapping data generated as outlined in Section 4.1 and a size of $M = 100$. To train the parameters θ, we minimize the `cross-entropy loss` using a gradient based method. Calculating the full gradient scales linearly with both the training set size M and the number of parameters d. To avoid this, we use SPSA [28], which ensures that the scaling is independent of d. In addition, we use stochastic gradient descent to approximate the gradient on a batch of 5 data points instead of the whole training set, removing the M-dependence. Figure 10 includes empirical evidence in support of this claim.

To analyze the effect of finite sampling noise, the training is performed with Qiskit's shot based QASM-simulator [18] using R shots per expectation value. The optimization of the models is first performed for 1000 training steps, resulting in the noisy decision function $h_{\theta_R}(\mathbf{x})$. To enable a direct comparison to the ideal case, we minimize the loss using full gradient descent (i.e. all parameters and all data in the training set are considered in the gradient evaluations) and a statevector simulator in an additional experiment. In this way, the noiseless decision function $h_{\theta_\infty}(\mathbf{x})$ is determined. Here, θ_∞ are the parameters that the statevector optimization ($R \to \infty$) converged to given the initial parameters θ_R. This procedure is repeated multiple times with different random initializations of the trained weights and varying R. For an error defined as

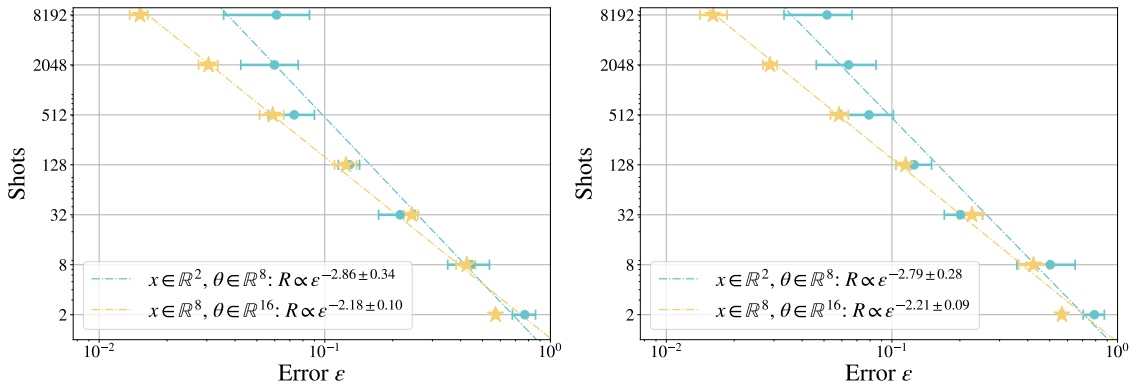
$$\varepsilon := \max_{\mathbf{x} \in X} \left| h_{\theta_R}(\mathbf{x}) - h_{\theta_\infty}(\mathbf{x}) \right|, \tag{19}$$

a least squares fit with respect to the number of shots is performed to conclude the experiment. The result is presented in Figure 11. From that experiment, the empirical



(a) M-dependence (b) d-dependence

Figure 10: **M- and d-scaling of the approximate QSVM model:** The approximate QSVM is trained using a combination of the SPSA and SGD optimization algorithms and a statevector simulator. Convergence is defined as the iteration T where the parameters fulfill $||\theta^T - \theta^{T-1}||/d < 10^{-4}$. The experiment is repeated for different (a) sizes M of the 2-dimensional data set (with $d = 8$ fixed) and (b) number of trainable parameters d with ($M = 256$ fixed). The total number of circuit evaluations for SPSA with a batch size of 5 is then given as $R_{tot} = 5 \cdot 2 \cdot T$. Every experiment is repeated $n = 10$ times, such that the markers correspond to the means and the error bars to the interval between the 15.9 and 84.1 percentile over these runs.



(a) Linearly separable data (b) Overlapping data

Figure 11: **ε-scaling of the approximate QSVM model:** QASM-simulators with shots per expectation value ranging from 2 to 8192 are employed. The number of shots R is plotted as a function of ε as defined in (19) for the training procedure detailed in the main text. A linear fit inside the log-log plot is used to determine the empirical exponent realized in our experimental setup. We analyze the case of 2-dimensional data with 8 trainable parameters (blue) and of 8-dimensional data with 16 trainable parameters (orange) both for separable and overlapping data. Every experiment has been repeated $n = 10$ times and the markers shown are the means of the resulting errors, while the error bars mark the interval between the 15.9 and 84.1 percentile.

complexity of the measurement shots needed for an ε-accurate solution is found to be approximately given by

$$R_{\text{tot}} = \mathcal{O}\left(\frac{1}{\varepsilon^{2.9 \pm 0.3}}\right).$$

5 Conclusion

The results summarized in Table 1 show that QSVMs can be trained in polynomial time with multiple training algorithms that differ in their computational complexity with respect to the size of the data set M and the accuracy ε of the decision function. Both the dual- and primal-QSVM solvers benefit from the convexity of the optimization problem that guarantees convergence to the global optimum. For the dual optimization we find that the complexity scales as $\mathcal{O}(M^{4.67}/\varepsilon^2)$, which entails a significant overhead over training a classical SVM due to shot noise that is inherent in any quantum measurements. Conversely, the Pegasos algorithm provides a scaling that is independent of M by solving the primal optimization problem using stochastic gradient descent. While this provides an advantage for training the classifier on large data sets, this algorithm scales worse with respect to the ε. The cost of training the algorithm with quantum kernels increases to $\mathcal{O}(1/\varepsilon^{10})$ from $\mathcal{O}(1/\varepsilon^4)$ when using classical kernels. While this scaling looks daunting, it is important to remark that ε denotes the error between the noisy and noiseless decision function evaluated on the training set. However, when solving a classification task in practice, typically the generalization performance of the models on some test set is the quantity of interest. The relationship between ε and the generalization error is an open question and promising direction for future research.

The best empirical scaling for training the QSVM optimization problem is found to be given by the approximate QSVM as the cost is also independent of M and the dependence on ε is reduced to $\mathcal{O}(1/\varepsilon^3)$. However, by choosing the approximate QSVM, we lose any guarantees on finding the global optimum of the optimization problem as the loss landscape now becomes non-convex. In addition to ensuring that the quantum kernel does not suffer from exponential concentration, we now also need to make sure that the loss function and initial choice of parameters does not lead to trainability problems such as barren plateaus. Nevertheless, the efficient scaling which might render the approximate QSVM the only feasible variant in practical settings.

Finally, we note that in this work we set the focus on the complexity of training QSVMs. We show that the optimization problem can be solved efficiently but the effect of statistical noise inherent to the quantum kernel introduces a polynomial overhead over the classical equivalent. It remains an open question whether quantum feature maps that provide a practical advantage over classical machine learning methods for real life classification tasks exist.

Author contributions AT and GG contributed equally to this work. The theoretical analysis was mainly conducted by AT while GG performed most of the numerical experiments. The work was supervised by DS and SW. All authors contributed to the write-up of this manuscript.

Competing interests The authors declare that there are no competing interests.

Acknowledgements We thank Amira Abbas for fruitful discussions about QSVMs, Julien Gacon for help with the numerical experiments and Supanut Thanasilp for insightful discussions about exponential concentration of quantum kernels. GG acknowledges support by the NCCR MARVEL, a National Centre of Competence in Research, funded by the Swiss National Science Foundation (grant number 205602).

Data availability The code and data to reproduce all numerical experiments and plots in this paper is available at [29].

References

[1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017. DOI: 10.1038/nature23474.

[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*, 567(7747):209–212, 2019. DOI: 10.1038/s41586-019-0980-2.

[3] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner. The power of quantum neural networks. *Nature Computational Science*, 1(June), 2020. DOI: 10.1038/s43588-021-00084-1.

[4] Y. Liu, S. Arunachalam, and K. Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 2021. DOI: 10.1038/s41567-021-01287-z.

[5] S. Aaronson. Read the fine print. *Nature Physics*, 11(4):291–293, 2015. DOI: 10.1038/nphys3272.

[6] E. Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 217–228, New York, NY, USA, 2019. Association for Computing Machinery. DOI: 10.1145/3313276.3316310.

[7] N.-H. Chia, A. Gilyén, T. Li, H.-H. Lin, E. Tang, and C. Wang. *Sampling-Based Sublinear Low-Rank Matrix Arithmetic Framework for Dequantizing Quantum Machine Learning*, page 387–400. Association for Computing Machinery, New York, NY, USA, 2020. Available online: https://doi.org/10.1145/3357713.3384314.

[8] T. Li, S. Chakrabarti, and X. Wu. Sublinear quantum algorithms for training linear and kernel-based classifiers. In *International Conference on Machine Learning*, pages 3815–3824. PMLR, 2019.

[9] S. Thanasilp, S. Wang, M. Cerezo, and Z. Holmes. Exponential concentration and untrainability in quantum kernel methods, 2022. DOI: 10.48550/ARXIV.2208.11060.

[10] S. Shalev-Shwartz and N. Srebro. SVM optimization: Inverse dependence on training set size. *Proceedings of the 25th International Conference on Machine Learning*, pages 928–935, 2008.

[11] A. Thomsen. Comparing quantum neural networks and quantum support vector machines. Master's thesis, ETH Zurich, 2021-09-06. DOI: 20.500.11850/527559.

[12] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. Association for Computing Machinery. DOI: 10.1145/130385.130401.

[13] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[14] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Science+Business Media, LLC, 2000.

[15] F. Pedregosa et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. Available online: http://jmlr.org/papers/v12/pedregosa11a.html.

[16] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[17] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical Programming*, 127(1):3–30, 2011. DOI: 10.1007/s10107-010-0420-4.

[18] M. D. S. Anis et al. Qiskit: An Open-source Framework for Quantum Computing, 2021. DOI: 10.5281/zenodo.2573505.

[19] P. Rebentrost, M. Mohseni, and S. Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(3):1–5, 2014. DOI: 10.1103/PhysRevLett.113.130503.

[20] J. Kübler, S. Buchholz, and B. Schölkopf. The inductive bias of quantum kernels. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12661–12673. Curran Associates, Inc., 2021. Available online: https://proceedings.neurips.cc/paper_files/paper/2021/file/69adc1e107f7f7d035d7baf04342e1ca-Paper.pdf.

[21] V. Heyraud, Z. Li, Z. Denis, A. Le Boité, and C. Ciuti. Noisy quantum kernel machines. *Phys. Rev. A*, 106:052421, 2022. DOI: 10.1103/PhysRevA.106.052421.

[22] C. J. C. Burges and C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998. Available online: https://www.microsoft.com/en-us/research/publication/a-tutorial-on-support-vector-machines-for-pattern-recognition/.

[23] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12(1):1791, 2021. DOI: 10.1038/s41467-021-21728-w.

[24] Belis, Vasilis, González-Castillo, Samuel, Reissel, Christina, Vallecorsa, Sofia, Combarro, Elías F., Dissertori, Günther, and Reiter, Florentin. Higgs analysis with quantum classifiers. *EPJ Web Conf.*, 251:03070, 2021. DOI: 10.1051/epjconf/202125103070.

[25] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021. DOI: 10.1038/s42254-021-00348-9.

[26] R. McGibborn et al. quadprog: Quadratic programming solver (python). https://github.com/quadprog/quadprog, 2022.

[27] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Applied Optimization. Springer, 2004. DOI: 10.1007/978-1-4419-8853-9.

[28] J. Spall. An Overview of the Simultaneous Perturbation Method for Efficient Optimization. *John Hopkins APL Technical Digest, 19(4)*, pages 482–492, 1998.

[29] G. Gentinetta, A. Thomsen, D. Sutter, and S. Woerner. Code for manuscript "The complexity of quantum support vector machines". 2022. DOI: https://doi.org/10.5281/zenodo.6303725.

[30] T. Hubregtsen, D. Wierichs, E. Gil-Fuster, P.-J. H. S. Derks, P. K. Faehrmann, and J. J. Meyer. Training quantum embedding kernels on near-term quantum computers. *Phys. Rev. A*, 106:042431, 2022. DOI: 10.1103/PhysRevA.106.042431.

[31] R. Latała. Some estimates of norms of random matrices. *Proceedings of the American Mathematical Society*, 133(5):1273–1282, 2005. DOI: 10.1090/s0002-9939-04-07800-1.

[32] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *Compressed Sensing: Theory and Applications*, pages 210–268, 2009. DOI: 10.1017/CBO9780511794308.006.

[33] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *Annals of Statistics*, 36(3):1171–1220, 2008. DOI: 10.1214/009053607000000677.

[34] J. W. Daniel. Stability of the solution of definite quadratic programs. *Mathematical Programming*, 5(1):41–53, 1973. DOI: 10.1007/BF01580110.

A Rigorous analysis of the dual approach

In this appendix, we provide a rigorous mathematical treatment of the steps outlined in Section 3.1. In the first section, we derive how the error of the kernel matrix scales when its entries are subject to finite sampling noise. We then analyze how this error affects the result of the QSVM optimization problem in a second section.

A.1 Justification of (10)

In a first step, define the kernel or Gram matrix $K \in \mathbb{R}^{M\times M}$ with entries

$$(K)_{ij} = k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) \quad \forall\, \mathbf{x}_i, \mathbf{x}_j \in X,$$

given by the kernel function evaluated for all combinations of data pairs in the training set. Writing $|\psi(\mathbf{x})\rangle = \mathcal{E}(\mathbf{x})\,|0\rangle$ for some unitary $\mathcal{E}(\mathbf{x})$, we can approximate the expectation value $|\,\langle\psi(\mathbf{x})|\psi(\mathbf{x}')\rangle\,|^2$ by preparing and measuring the state $\mathcal{E}(\mathbf{x}')^\dagger\mathcal{E}(\mathbf{x})\,|0\rangle$ in the computational basis a total of R times.

A more formal treatment of this method introduces the Bernoulli distributed random variable $\hat{k}_{ij}$ taking on the values

$$\hat{k}_{ij} = \begin{cases} 1 & \text{if the zero state } |0\rangle \text{ is recovered in the measurement} \\ 0 & \text{otherwise} \end{cases}.$$

The kernel is then equal to the true mean

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbb{E}\left[\hat{k}_{ij}\right], \tag{20}$$

because the expectation value of the Bernoulli distribution is equal to the probability of $\hat{k}_{ij} = 1$. Fundamentally, we can only approximate this expectation value by the sample mean

$$k_R(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{R}\sum_{l=1}^{R} \hat{k}_{ij}^{(l)},$$

where the $\hat{k}_{ij}^{(l)}$ are the i.i.d. outcomes of R measurement shots performed for every single entry of K. Denote by K_R the matrix with entries $k_R(\mathbf{x}_i, \mathbf{x}_j)$. Closely following [30, Section

V. C.], the goal is now to bound the operator distance between the ideal K and obtainable K_R.

To do so, define the error

$$E_R \coloneqq K_R - K, \tag{21}$$

and observe that

$$\mathbb{E}\left[(E_R)_{ij}\right] = \mathbb{E}\left[k_R(\mathbf{x}_i, \mathbf{x}_j) - k(\mathbf{x}_i, \mathbf{x}_j)\right] = \frac{1}{R}\sum_{l=1}^{R} \mathbb{E}\left[\hat{k}_{ij}^{(l)}\right] - \mathbb{E}\left[k(\mathbf{x}_i, \mathbf{x}_j)\right] = 0\,, \tag{22}$$

where the linearity of the expectation value and (20) have been used. The second moment of E_R can be calculated as

$$\begin{aligned}
\mathbb{E}\left[|(E_R)_{ij}|^2\right] &= \mathbb{E}\left[\left(\frac{1}{R}\sum_{\ell=1}^{R}\hat{k}_{ij}^{(\ell)} - k_{ij}\right)^2\right] \\
&= \frac{1}{R^2}\,\mathbb{E}\left[\left(\sum_{l=1}^{R}\hat{k}_{ij}^{(l)}\right)^2\right] \underbrace{-\frac{2k_{ij}}{R}\sum_{l=1}^{R}\underbrace{\mathbb{E}\left[\hat{k}_{ij}^{(l)}\right]}_{=k_{ij}} + k_{ij}^2}_{=-k_{ij}^2} \\
&= \frac{1}{R^2}\left(R\,\underbrace{\mathbb{E}\left[\left(\hat{k}_{ij}^{(l)}\right)^2\right]}_{(\star)} + 2\binom{R}{2}\underbrace{\mathbb{E}\left[\hat{k}_{ij}^{(l)}\hat{k}_{ij}^{(m)}\right]}_{(\dagger)}\right) - k_{ij}^2,
\end{aligned} \tag{23}$$

where in the step from the second to the third line, the sum inside the square is expanded into two terms collecting the two cases when both samples are identical and when they are different with $l \neq m$. The expression can be further simplified knowing

$$(\star) = \mathrm{Var}\left[\hat{k}_{ij}\right] - \left(\mathbb{E}\left[\hat{k}_{ij}\right]\right)^2 = k_{ij}(1 - k_{ij}) + k_{ij}^2 = k_{ij},$$

because $\hat{k}_{ij}$ is Bernoulli distributed and

$$(\dagger) = \mathbb{E}\left[\hat{k}_{ij}^{(l)}\right]\mathbb{E}\left[\hat{k}_{ij}^{(m)}\right] = k_{ij}^2,$$

because the $\hat{k}_{ij}^{(\cdot)}$ are independently and identically distributed. Plugging in these results back into (23) and making use of

$$\binom{R}{2} = \frac{R!}{2!(R-2)!} = \frac{R(R-1)}{2} \leq \frac{R^2}{2}$$

finally yields

$$\mathbb{E}\left[|(E_R)_{ij}|^2\right] = \frac{1}{R^2}\left[R\,k_{ij} + 2\binom{R}{2}k_{ij}^2\right] - k_{ij}^2 \leq \frac{k_{ij}}{R} = \mathcal{O}\left(\frac{1}{R}\right). \tag{24}$$

Analogously, it can be shown that the fourth moment scales as

$$\mathbb{E}\left[|(E_R)_{ij}|^4\right] = \mathcal{O}\left(\frac{1}{R^2}\right). \tag{25}$$

Knowing this, the following result from random matrix theory can be harnessed:

Theorem 3 (Latala's theorem [31, Theorem 2] and [32, Theorem 5.37])**.** *For any matrix $X \in \mathbb{R}^{n\times n}$ whose entries are independent random variables x_{ij} of zero mean, there exists a constant $c > 0$ such that*

$$\mathbb{E}\left[\|X\|_2\right] \leq c\left[\max_{i\in[n]}\left(\sum_{j=1}^{n}\mathbb{E}\left[x_{ij}^2\right]\right)^{\frac{1}{2}} + \max_{j\in[n]}\left(\sum_{i=1}^{n}\mathbb{E}\left[x_{ij}^2\right]\right)^{\frac{1}{2}} + \left(\sum_{ij=1}^{n}\mathbb{E}\left[x_{ij}^4\right]\right)^{\frac{1}{4}}\right], \tag{26}$$

where $\|\cdot\|_2 = \sigma_{\max}(\cdot)$ denotes the operator norm induced by the Euclidean vector norm.

The matrix E_R satisfies all of the assumptions in Theorem 3 and by (22), (24) and (25), yields

$$\mathbb{E}\left[\|E_R\|_2\right] = \mathbb{E}\left[\|K_R - K\|_2\right] = \mathcal{O}\left(\frac{\sqrt{M}}{\sqrt{R}}\right). \tag{27}$$

With this, a bound on the accuracy of the kernel matrix is established.

A.2 Justification of (11)

The next question is how the solution to the SVM optimization problem is affected by perturbations to the ideal K. The following treatment is based on [4, Appendix C]. We show that we need $\mathcal{O}(M^{8/3}/\varepsilon^2)$ measurement shots per kernel entry to ensure that $|h(\hat{\mathbf{x}}) - h_R(\hat{\mathbf{x}})| \leq \varepsilon$. Given that K is symmetric a total of $M(M+1)/2 = \mathcal{O}(M^2)$ unique entries need to be calculated, implying (11).

As a reminder, the goal is to solve the dual optimization problem (2), which can be expressed in terms of matrices. Define Q as the matrix with entries $(Q)_{ij} \coloneqq y_i y_j\, k(\mathbf{x}_i, \mathbf{x}_j)$. The matrix Q is positive semidefinite if K is, since

$$Q = \operatorname{diag}(\mathbf{y})\, K \operatorname{diag}(\mathbf{y})\,, \tag{28}$$

for the vector of training labels $\mathbf{y} \in \mathbb{R}^M$, and $\operatorname{diag}(\mathbf{y})$ can be absorbed into the $\mathbf{v}$ in the definition of positive semidefiniteness of a matrix $\mathbf{v}^\top Q\,\mathbf{v} \geq 0 \quad \forall \mathbf{v} \in \mathbb{R}^M \setminus \{\mathbf{0}\}$. The kernel matrix K is always positive semidefinite [33, Section 2.2], implying that Q must be too.

The dual problem in (2) can be written as the quadratic program [4, Eq. (D19)]

$$\begin{cases} \min\limits_{\boldsymbol{\alpha}\in\mathbb{R}^M} & \frac{1}{2}\boldsymbol{\alpha}^\top\left(Q + \lambda \mathrm{id}\right)\boldsymbol{\alpha} - \mathbf{1}^\top\boldsymbol{\alpha} \\ \text{s.t.} & \boldsymbol{\alpha} \geq 0\,, \end{cases} \tag{29}$$

where $\boldsymbol{\alpha}$ denotes the vector with components α_i, $\mathbf{1} \in \mathbb{R}^M$ the all one vector and $\mathrm{id} \in \mathbb{R}^{M\times M}$ the identity matrix. Because the identity commutes with all matrices, Q and $\lambda\mathrm{id}$ can be simultaneously diagonalized. Since Q is positive semidefinite, its eigenvalues are all greater than or equal to zero. So when Q and $\lambda\mathrm{id}$ are expressed in a basis where both matrices are diagonal, adding the two yields a diagonal matrix with entries greater than or equal to λ, implying that

$$\mu \geq \lambda \tag{30}$$

for the the smallest eigenvalue μ of the matrix $(Q + \lambda\mathrm{id})$.

With this, the following theorem can be invoked to prove the robustness of the dual QSVM optimization when the kernel function is only evaluated up to some finite precision.

Theorem 4 (Daniel's Theorem [34, Theorem 2.1] [4, Lemma 16])**.** *Let x_0 be the solution to the quadratic program*

$$\begin{cases} \min\limits_{x} & \frac{1}{2}x^\top K x - c^\top x \\ \text{s.t.} & Gx \leq g \\ & Dx = d\,, \end{cases} \tag{31}$$

where K is positive definite with smallest eigenvalue $\mu > 0$ and the dimensions of the vectors c, g, d and matrices G, D are such that all operations are well-defined. Let K' be a symmetric matrix such that $\|K' - K\|_2 \leq \varepsilon < \mu$.[10] *Let x_0' be the solution to (31) with K replaced by K'. Then*

$$\|x_0' - x_0\| \leq \frac{\varepsilon}{\mu - \varepsilon}\|x_0\|. \tag{32}$$

From the ground up, classification with QSVMs can be divided into two subsequent steps, training and prediction. For training, the quantum kernel matrix K defined in (5) is initially evaluated on a quantum computer. Then, the QSVM is fit by running the dual program in (29) on a classical computer, yielding a solution vector $\boldsymbol{\alpha}^\star$. For prediction, a new datum $\hat{\mathbf{x}} \in \mathbb{R}^d$ is assigned a class membership $\hat{y} \in \{-1, +1\}$ via the classification function $c_{\text{SVM}}(\hat{\mathbf{x}})$ in (4). To this end, the quantum computer has to be employed again to determine the quantum kernel value $k(\hat{\mathbf{x}}, \mathbf{x}_i)$ for all $\mathbf{x}_i$ in the training set X.

From these two steps, it is clear that there are two separate instances in the QSVM algorithm where quantum kernels are evaluated and the statistical uncertainty inherent to quantum expectation values unavoidably enters. Because even on a fault tolerant quantum computer, fundamentally, only an approximate kernel function $k_R(\hat{\mathbf{x}}, \mathbf{x}_i)$ as defined in (9) is feasible. Consequently, for training, K has to be replaced by its approximation K_R, in turn leading to a noisy Q_R analogous to (28). The solution to the dual (29) when Q is replaced by Q_R is then denoted as $\boldsymbol{\alpha}_R^\star$. In the same way, the kernel evaluations in the decision function are replaced by evaluations of $k_R(\hat{\mathbf{x}}, \mathbf{x}_i)$ in the prediction step. Putting all of this together, a statement on the robustness of the overall QSVM with respect to measurement uncertainty can be made on the level of

$$h(\hat{\mathbf{x}}) \coloneqq \sum_{i=1}^{M} \alpha_i^\star y_i\, k(\hat{\mathbf{x}}, \mathbf{x}_i) \qquad \text{and} \qquad h_R(\hat{\mathbf{x}}) \coloneqq \sum_{i=1}^{M} \alpha_{R,i}^\star y_i\, k_R(\hat{\mathbf{x}}\,, \mathbf{x}_i),$$

the ideal and noisy version of the term inside the sign function in the classification function (4). The goal is to show the robustness of QSVMs with respect to finite sampling noise by bounding the difference between these terms with high probability.

The following lemma is adapted from [4, Lemma 19].

Lemma 5. *Let $\varepsilon > 0$ and suppose $R = \mathcal{O}(M^{8/3}/\varepsilon^2)$ measurement shots are taken for each quantum kernel estimation circuit. Furthermore, assume the setting of noisy halfspace learning (see Assumption 1). Then, with fixed probability $p > \frac{1}{2}$, where the probability is stemming from the choice of random training samples and uncertainty coming from the finite sampling statistics, for every $\mathbf{x} \in \mathbb{R}^d$ we have*

$$|h_R(\mathbf{x}) - h(\mathbf{x})| \leq \varepsilon\,.$$

[10] Positive definiteness of K' is then implied.

Proof. Start by considering the operator distance between the noisy matrix Q_R resulting from quantum kernel evaluations with a finite number of measurement shots R per entry and the ideal matrix Q. Making use of the connection between Q and K in (28), we find

$$\begin{aligned}
\|Q_R - Q\|_2 &= \sup_{\mathbf{v}\neq 0} \frac{\|(Q_R - Q)\mathbf{v}\|}{\|\mathbf{v}\|} \\
&= \sup_{\mathbf{v}\neq 0} \frac{\|\mathrm{diag}(\mathbf{y})\,(K_R - K)\,\mathrm{diag}(\mathbf{y})\mathbf{v}\|}{\|\mathbf{v}\|} \\
&= \sup_{\mathbf{v}\neq 0} \frac{\sqrt{\sum_{i=1}^{M}\left|\sum_{j=1}^{M} y_i y_j (k_R(\mathbf{x}_i,\mathbf{x}_j) - k(\mathbf{x}_i,\mathbf{x}_j))v_j\right|^2}}{\|\mathbf{v}\|} \\
&\leq \sup_{\mathbf{v}\neq 0} \frac{\sqrt{\sum_{i=1}^{M}\left|\sum_{j=1}^{M} (k_R(\mathbf{x}_i,\mathbf{x}_j) - k(\mathbf{x}_i,\mathbf{x}_j))v_j\right|^2}}{\|\mathbf{v}\|} \\
&= \|K_R - K\|_2\,,
\end{aligned}$$

where in the inequality step uses $|y_i| = 1$ and the triangle inequality. Then, (10) implies

$$\mathbb{E}\left[\|Q_R - Q\|_2\right] = \mathcal{O}\left(\frac{\sqrt{M}}{\sqrt{R}}\right). \tag{33}$$

For the setting of noisy halfspace learning as defined in [4, Lemma 14, Definition 15], it can be shown [4, Remark 2] that

$$\mathbb{E}\left[\|\boldsymbol{\alpha}^\star\|\right] = \mathcal{O}\left(M^{1/3}\right), \tag{34}$$

where $\boldsymbol{\alpha}^\star$ is the solution to (29). With Markov's inequality

$$\Pr\left[X \leq a\,\mathbb{E}[X]\right] \geq 1 - \frac{1}{a} \overset{!}{=} p'\,,$$

the statements (33) and (34) in expectation can be transformed into statements in probability. To this end, choose a such that $p' > 1 - \frac{1}{a} > \frac{1}{\sqrt[3]{2}}$ is fulfilled, yielding

$$\eta \coloneqq \|Q_R - Q\|_2 = \mathcal{O}\left(\frac{\sqrt{M}}{\sqrt{R}}\right) \qquad \text{and} \qquad \|\boldsymbol{\alpha}^\star\| = \mathcal{O}\left(M^{1/3}\right), \tag{35}$$

which hold with a probability greater than or equal to p' respectively.

Now, Theorem 4 can be leveraged as Q is positive-definite because there is a lower bound on its smallest eigenvalue μ in (30). Then, for $\delta_i \coloneqq \alpha^\star_{R,i} - \alpha^\star_i$ and with probability at least p'^2, (32) yields

$$\begin{aligned}
\|\delta\| &= \|\boldsymbol{\alpha}^\star_R - \boldsymbol{\alpha}^\star\| \\
&\leq \frac{\eta}{\mu - \eta}\|\boldsymbol{\alpha}^\star\| \\
&= \left(\frac{\eta}{\mu} + \mathcal{O}\left(\frac{\eta^2}{\mu^2}\right)\right)\|\boldsymbol{\alpha}^\star\| \\
&= \mathcal{O}\left(\frac{\sqrt{M}}{\sqrt{R}}\right)\mathcal{O}\left(M^{1/3}\right)
\end{aligned}$$

$$= \mathcal{O}\left(\frac{M^{5/6}}{\sqrt{R}}\right),$$

where R has to be chosen such that $\eta < \mu$, which is always feasible due to the lower bound on μ.

In a next step, denote by $\nu_i := k_R(\hat{\mathbf{x}}, \mathbf{x}_i) - k(\hat{\mathbf{x}}, \mathbf{x}_i)$ for $i = 1, 2, ..., M$ the difference between the obtainable and ideal kernel function evaluated for pairs of the datum to be classified $\hat{\mathbf{x}}$ and the elements in the training set $\mathbf{x}_i$. As ν_i is a special case of the components in E_R in (21), $\mathbb{E}[\nu_i] = 0$ is implied by (22) and $\mathbb{E}[\nu_i^2] = \mathcal{O}\left(\frac{1}{R}\right)$ by (24). By making use of Chebyshev's inequality

$$\Pr\left(|X - \mathbb{E}[X]| \leq a\sqrt{\mathrm{Var}[X]}\right) \geq 1 - \frac{1}{a^2} \stackrel{!}{=} p',$$

and $\mathrm{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$, the above expectation values can be converted into the statement that

$$\|\nu\| = \mathcal{O}\left(\frac{\sqrt{M}}{\sqrt{R}}\right) \tag{36}$$

holds with probability at least $p' > 1/\sqrt[3]{2}$, where the $\sqrt{M}$ scaling comes from the Euclidean norm and the fact that the vector ν has M components ν_i.

Returning to the ideal and noisy decision functions, the bounds (35) and (36) can be combined to give

$$\begin{aligned}
|h_R(\hat{\mathbf{x}}) - h(\hat{\mathbf{x}})| &= \left|\sum_{i=1}^{M} \alpha_{R,i}^{\star} y_i \, k_R(\hat{\mathbf{x}}, \mathbf{x}_i) - \sum_{i=1}^{M} \alpha_i^{\star} y_i \, k(\hat{\mathbf{x}}, \mathbf{x}_i)\right| \\
&= \left|\sum_{i=1}^{M} (\alpha_i^{\star} + \delta_i) y_i (k(\hat{\mathbf{x}}, \mathbf{x}_i) + \nu_i) - \sum_{i=1}^{M} \alpha_i^{\star} y_i \, k(\hat{\mathbf{x}}, \mathbf{x}_i)\right| \\
&= \left|\sum_{i=1}^{M} \alpha_i^{\star} \nu_i + \delta_i k(\hat{\mathbf{x}}, \mathbf{x}_i) + \delta_i \nu_i\right| \\
&\leq \left|\sum_{i=1}^{M} \alpha_i^{\star} \nu_i\right| + \left|\sum_{i=1}^{M} \delta_i k(\hat{\mathbf{x}}, \mathbf{x}_i)\right| + \left|\sum_{i=1}^{M} \delta_i \nu_i\right| \\
&\leq \|\alpha^{\star}\| \|\nu\| + \sqrt{M} \|\delta\| + \|\delta\| \|\nu\| \\
&= \mathcal{O}\left(\frac{M^{4/3}}{\sqrt{R}}\right),
\end{aligned}$$

which is true with probability at least $p := (p')^3 > \frac{1}{2}$ for any $\hat{\mathbf{x}} \in \mathbb{R}^d$. The penultimate step uses Cauchy-Schwarz and property $k(\hat{\mathbf{x}}, \mathbf{x}_i) \in [0, 1]$. Hence, the desired accuracy $|h_R(\hat{\mathbf{x}}) - h(\hat{\mathbf{x}})| \leq \varepsilon$ for the QSVM decision function is obtained with probability greater than or equal to p for

$$R = \mathcal{O}\left(\frac{M^{8/3}}{\varepsilon^2}\right) \tag{37}$$

measurement shots per quantum kernel evaluation. □

As remarked at the beginning of this appendix, this result justifies (11) in the main text.

Remark 6. We note that this analysis is an improvement of the results derived in [4, Lemma 19], where they showed that $R_{\text{tot}} = \mathcal{O}(M^6/\varepsilon^2)$ shots are necessary under the same assumptions.[11]

B Extended analysis of the primal approach

In this appendix, we provide justifications for the steps discussed in Section 3.2. We first provide experiments supporting Assumption 2. Then, we present a rigorous mathematical derivation of the connection between ε and δ in (14) in the main text.

B.1 Empirical justification of Assumption 2

Assumption 2 claims that Pegasos converges independently of the sampling noise when the number of measurement shots is large enough. To provide numerical evidence for this statement, we compare the evolution of the algorithm for different numbers of shots per kernel evaluation R.

For the experiments presented in Figure 12, every optimization step of the Pegasos algorithm is performed with a noisy kernel value k_R (see (9)) in line 14 of the algorithm. The accuracy plotted is then computed according to (4) evaluated for exact kernel evaluations, which facilitates a direct comparison of the different optimization runs. We observe that for R sufficiently large, the algorithm converges to a solution which is close to the one obtained by performing an infinite number of quantum circuit evaluations. Furthermore, visually the rate of convergence is not significantly slower for smaller but sufficiently large R. Taken together, these observations suggests that Assumption 2 is fulfilled.

In Figure 13, we additionally present experiments probing how the coefficients behave under noisy kernel evaluations. The integer coefficients from Pegasos are scaled by a factor of λ/t, similar to the sum in line 14. Unsurprisingly, the error is quite large in the first few iterations as t in the denominator is small. However, after around $t = 40$ iterations the error stabilizes and in some cases even decreases. This provides further evidence that the convergence of Pegasos is not strongly affected by finite sampling noise.

B.2 Justification of (14)

In the following we denote by $f(\mathbf{w})$ the objective function of the primal optimization problem (1) where $\mathbf{w}^\star$ is the optimizer.

Lemma 7. *The objective function $f(\mathbf{w})$ is λ-strongly convex in $\mathbf{w}$.*

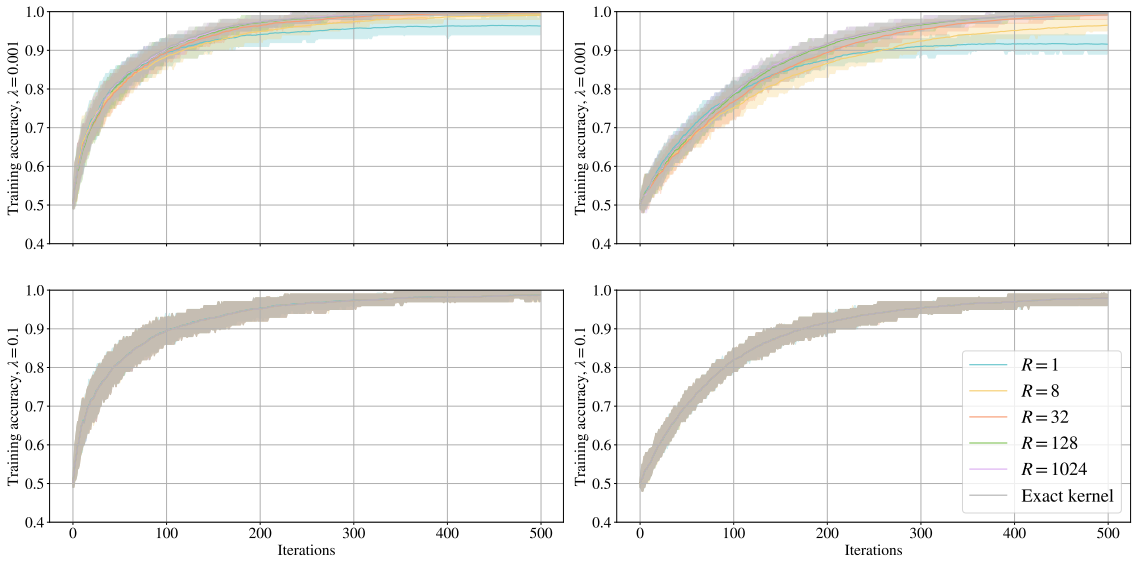
Proof. The hinge loss $\mathcal{L}_{\text{hinge}}(y_i, \mathbf{w}^\top \mathbf{x} + b) = \max\left[0, 1 - y_i(\mathbf{w}^\top \mathbf{x} + b)\right]$ is convex and thus the sum

$$g(\mathbf{w}) = \sum_{i=1}^{M} \mathcal{L}_{\text{hinge}}(y_i, \mathbf{w}^\top \mathbf{x}_i + b)$$

is convex as well. Now f is λ-strongly convex if and only if $f(\mathbf{w}) - \frac{\lambda}{2}\|\mathbf{w}\|$ is convex. But $f(\mathbf{w}) - \frac{\lambda}{2}\|\mathbf{w}\| = g(\mathbf{w})$ is convex and, thus, f is indeed strongly convex. □

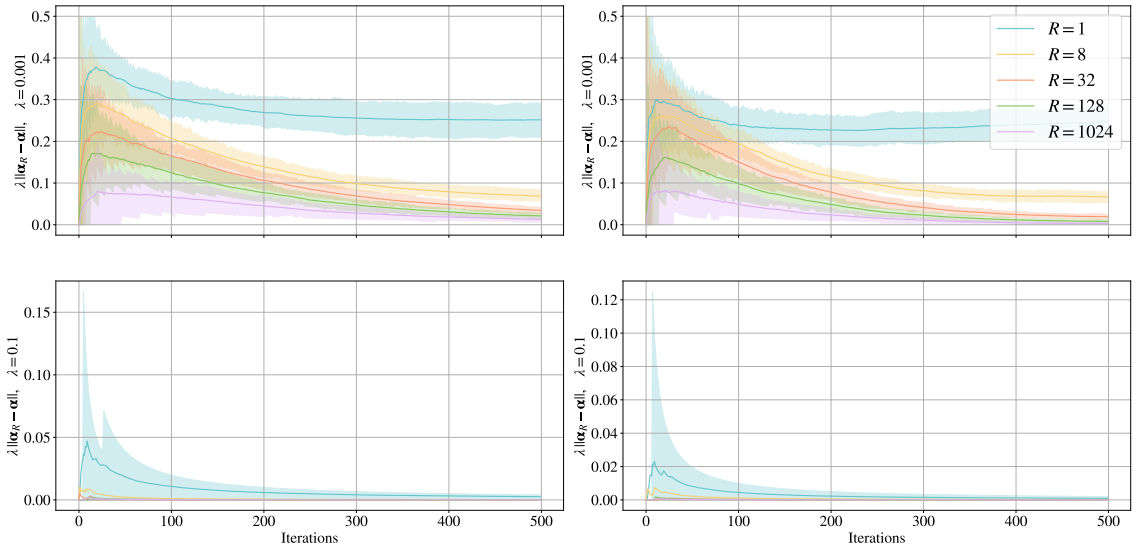
Corollary 8. *Let $\varepsilon > 0$ and $\mathbf{w} \in \mathbb{R}^s$ be such that $f(\mathbf{w}) \leq f(\mathbf{w}^\star) + \varepsilon$. Then, $\|\mathbf{w} - \mathbf{w}^\star\| \leq \sqrt{\frac{2\varepsilon}{\lambda}}$.*

[11] Note that the result [30, Section V.C.] claiming that $R_{\text{tot}} = \mathcal{O}(M^3/\varepsilon^2)$ suffice is not comparable, since there, the analysis is restricted to the kernel matrix and ε is defined differently as a result.



(a) Linearly separable data (b) Overlapping data

Figure 12: **Training accuracy of Pegasos:** The QSVM is trained with PEGASOS for different numbers of shots per kernel evaluation R. The class labels are then predicted for the whole training set using the exact kernels and compared to the ground truth. The resulting accuracy is plotted as a function of the iteration steps t. In the top plots, the regularization constant is $\lambda = 1/1000$, while the lower plots are regularized with $\lambda = 1/10$. Every experiment has been performed $n = 100$ times with different random seeds, such that the lines shown are the means over the runs and the shaded areas mark the interval between the 15.9 and 84.1 percentile.



(a) Linearly separable data (b) Overlapping data

Figure 13: **Error evolution of Pegasos:** We plot the error in the coefficients $\boldsymbol{\alpha}$ for different numbers of shots per kernel evaluation R as functions of the iteration steps t. The top plots were generated with a regularization constant $\lambda = 1/1000$ while the lower plots are regularized with $\lambda = 1/10$. Every experiment has been performed $n = 100$ times with different random seeds such that the lines shown are the means over these runs and the shaded areas mark the interval between the 15.9 and 84.1 percentile.

Proof. By definition of strong convexity and using Lemma 7 we have

$$f(\mathbf{w}) \geq f(\mathbf{w}^\star) + \nabla f(\mathbf{w}^\star)^T(\mathbf{w} - \mathbf{w}^\star) + \frac{\lambda}{2}\|\mathbf{w} - \mathbf{w}^\star\|^2 .$$

Noting that the gradient vanishes at the minimum we find

$$\|\mathbf{w} - \mathbf{w}^\star\|^2 \leq \frac{2}{\lambda}(f(\mathbf{w}) - f(\mathbf{w}^\star)) \leq \frac{2\varepsilon}{\lambda}\,.$$

□

We are finally ready to prove the formal statement of (14).

Lemma 9. *For $\varepsilon = |h^\star(x) - h^N(x)|$ the distance between in the decision function $h^N(x) = \sum_{j=1}^M \alpha_j^N y_j k(x, x_j)$ after N iterations of the* Pegasos *algorithm and the optimal decision function $h^\star$, we find*

$$\varepsilon \leq \sqrt{\frac{2\delta}{\lambda}}\,,$$

where $\delta = |f(\mathbf{w}^\star) - f(\mathbf{w}^N)|$ is the deviation from the optimum.

Proof. Using the Kernel trick we write in general $\mathbf{w} = \sum_{i=1}^M \alpha_i y_i \phi(x_i)$, allowing the following calculation:

$$\begin{aligned}
|h^\star(x) - h^N(x)| &= |\sum_{j=1}^M \alpha_j^\star y_j k(x, x_j) - \sum_{j=1}^M \alpha_j^N y_j k(x, x_j)| \\
&= |\sum_{j=1}^M \alpha_j^\star y_j \phi(x)^T \phi(x_i) - \sum_{j=1}^M \alpha_j^N y_j \phi(x)^T \phi(x_i)| \\
&= |\phi(x)^T \Big(\sum_{j=1}^M \alpha_j^\star y_j \phi(x_i) - \sum_{j=1}^M \alpha_j^N y_j \phi(x_i) \Big)| \\
&= |\phi(x)^T (\mathbf{w}^N - \mathbf{w}^\star)| \\
&\leq \|\phi\| \Big\| \mathbf{w}^N - \mathbf{w}^\star \Big\| \\
&\leq \sqrt{\frac{2\delta}{\lambda}}\,,
\end{aligned}$$

where the first inequality uses Cauchy-Schwarz and in the final follows from $\|\phi(x)\| \leq 1$ as well as Corollary 8. □

C Generating artificial data

In this appendix we provide the algorithm used to generate the training data for the numerical experiments in section Section 3. In addition to the data set size M and the margin μ separating the two classes, the algorithm requires a decision function h_θ as an input. For the approximate QSVM model, this decision function is given by (8). To generate data for the experiments in Sections 4.2 and 4.3 and appendix B.1, the variatonal form $\mathcal{W}(\theta)$ in (7) is chosen as the identity and thus trivial. In that case, the different realizations of the training set correspond to differing samples from the uniform distribution in Algorithm 2, while the underlying decision boundary remains fixed. In Section 4.4, the θ in $\mathcal{W}(\theta)$ are additionally randomly sampled.

Algorithm 2 Generating artificial data

Inputs:
$h_\theta(\mathbf{x})$ with fixed θ,
size of the data set $M \in \mathbb{N}$,
size of the margin $\mu \in \mathbb{R}$ (negative μ results in overlapping data).

$i = 0$
while $|y = -1| \leq \frac{M}{2} \wedge |y = 1| \leq \frac{M}{2}$ **do**
  Sample $\mathbf{x} \sim \mathcal{U}_{[0,1]} \times \mathcal{U}_{[0,1]}$ once
  $\tilde{y} = h_\theta(\mathbf{x})$
  if $\tilde{y} \leq -\frac{\mu}{2}$ **and** $\tilde{y} < +\frac{\mu}{2}$ **then**
    $\mathbf{x}_i \leftarrow \mathbf{x}$
    $y_i \leftarrow -1$
    $i \leftarrow i + 1$
  end if
  if $\tilde{y} \geq +\frac{\mu}{2}$ **and** $\tilde{y} > -\frac{\mu}{2}$ **then**
    $\mathbf{x}_i \leftarrow \mathbf{x}$
    $y_i \leftarrow +1$
    $i \leftarrow i + 1$
  end if
  if $\tilde{y} \geq +\frac{\mu}{2}$ **and** $\tilde{y} \leq -\frac{\mu}{2}$ **then**
    Sample $p \sim \mathcal{U}_{[0,1]}$
    if $p > \frac{1}{2}$ **then**
      $y_i \leftarrow +1$
    else
      $y_i \leftarrow -1$
    end if
    $i \leftarrow i + 1$
  end if
end while

Output: Balanced training data $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M\}$ with labels $y = \{y_1, y_2, ..., y_M\}$.