Supervised learning with quantum enhanced feature spaces

Vojtech Havlicek[1],[*] Antonio D. Córcoles[1], Kristan Temme[1], Aram W. Harrow[2],
Abhinav Kandala[1], Jerry M. Chow[1], and Jay M. Gambetta[1]

[1]*IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA and*
[2]*Center for Theoretical Physics, Massachusetts Institute of Technology, USA*
(Dated: June 7, 2018)

Machine learning and quantum computing are two technologies each with the potential for altering how computation is performed to address previously untenable problems. Kernel methods for machine learning are ubiquitous for pattern recognition, with support vector machines (SVMs) being the most well-known method for classification problems. However, there are limitations to the successful solution to such problems when the feature space becomes large, and the kernel functions become computationally expensive to estimate. A core element to computational speed-ups afforded by quantum algorithms is the exploitation of an exponentially large quantum state space through controllable entanglement and interference.

Here, we propose and experimentally implement two novel methods on a superconducting processor. Both methods represent the feature space of a classification problem by a quantum state, taking advantage of the large dimensionality of quantum Hilbert space to obtain an enhanced solution. One method, the quantum variational classifier builds on [1, 2] and operates through using a variational quantum circuit to classify a training set in direct analogy to conventional SVMs. In the second, a quantum kernel estimator, we estimate the kernel function and optimize the classifier directly. The two methods present a new class of tools for exploring the applications of noisy intermediate scale quantum computers [3] to machine learning.

The intersection between machine learning and quantum computing has been dubbed quantum machine learning, and has attracted considerable attention in recent years [4–6]. This has led to a number of recently proposed quantum algorithms [1, 2, 7–9]. Here, we present a quantum algorithm that has the potential to run on near-term quantum devices. A natural class of algorithms for such noisy devices are short-depth circuits, which are amenable to error-mitigation techniques that reduce the effect of decoherence [10, 11]. There are convincing arguments that indicate that even very sim-

ple circuits are hard to simulate by classical computational means [12, 13]. The algorithm we propose takes on the original problem of supervised learning: the construction of a classifier. For this problem, we are given data from a training set T and a test set S of a subset $\Omega \subset \mathbb{R}^d$. Both are assumed to be labeled by a map $m : T \cup S \to \{+1, -1\}$ unknown to the algorithm. The training algorithm only receives the labels of the training data T. The goal is to infer an approximate map on the test set $\tilde{m} : S \to \{+1, -1\}$ such that it agrees with high probability with the true map $m(\vec{s}) = \tilde{m}(\vec{s})$ on the test data $\vec{s} \in S$. For such a learning task to be meaningful it is assumed that there is a correlation between the labels given for training and the true map. A classical approach to constructing an approximate labeling function uses so-called support vector machines (SVMs) [14]. The data gets mapped non-linearly to a high dimensional space, *the feature space*, where a hyperplane is constructed to separate the labeled samples. A quantum version of this approach has already been proposed in [15], where an exponential improvement can be achieved if data is provided in a coherent superposition. However, when data is provided in the conventional way, i.e. from a classical computer, then the methods of [15] cannot be applied.

Here, we propose two SVM type classifiers that process data provided purely classically and use the quantum state space as the feature space to still obtain a quantum advantage. This is done by mapping the data non-linearly to a quantum state $\Phi : \vec{x} \in \Omega \to |\Phi(\vec{x})\rangle\langle\Phi(\vec{x})|$, c.f. Fig 1(a). We implement both classifiers on a superconducting quantum processor. In the first approach we use a variational circuit as given in [1, 2, 16, 17] that generates a separating hyperplane in the quantum feature space. In the second approach we use the quantum computer to estimate the kernel function of the quantum feature space directly and implement a conventional SVM. A necessary condition to obtain a quantum advantage, in either of the two approaches, is that the kernel cannot be estimated classically. This is true, even when complex variational quantum circuits are used as classifiers. In the experiment, we want to disentangle the question of whether the classifier can be implemented in hardware, from the problem of choosing a suitable feature map for a practical data set. The data that is classified here is chosen so that it can be classified with 100% success to verify the method. We demonstrate that this success ratio is subsequently achieved in experiment.

Our experimental device consists of five coupled superconducting transmons, only two of which are used in this
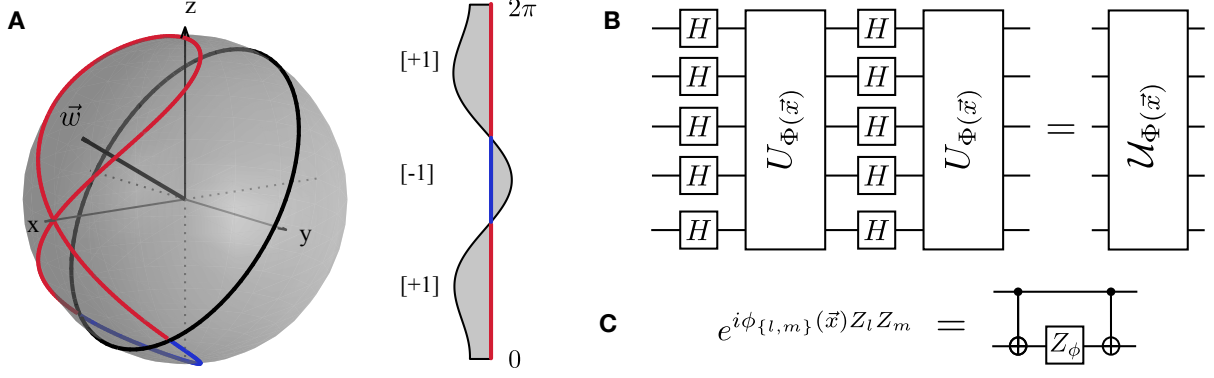
FIG. 1. **Quantum Kernel Functions**: (a) Feature map representation for a single qubit. A classical dataset in the interval $\Omega = (0, 2\pi]$ with binary labels (a, right) can be mapped onto the Bloch sphere (red / blue - lines) by using the non-linear feature map described in (b). For a single qubit $U_{\Phi(x)} = Z_x$ is a phase-gate of angle $x \in \Omega$. The mapped data can be separated by the hyperplane given by normal $\vec{w}$. States with a positive expectation value of $\vec{w}$ receive a $[+1]$ (red) label, while negative values are labeld $[-1]$(blue). (b) For the general circuit $U_{\Phi(\vec{x})}$ is formed by products of single- and two-qubit unitaries that are diagonal in the computational basis. In our experiments, both the training and testing data is artificially generated to be perfectly classifiable with the aforementioned feature map. The circuit family depends non-linearly on the data through the coefficients $\phi_S(\vec{x})$ with $|S| \leq 2$. (c) Experimental implementation of the parameterized diagonal two-qubit operations using CNOTs and $Z-$gates.

work, as shown in Fig. 2(a). Two co-planar waveguide (CPW) resonators, acting as quantum buses, provide the device connectivity. Each qubit has one additional CPW resonator for control and readout. Entanglement in our system is achieved via CNOT gates, which use cross-resonance [18] as well as single qubit gates as primitives. The quantum processor is thermally anchored to the mixing chamber plate of a dilution refrigerator.

Quantum feature map: Before discussing the two methods of classification, we discuss the feature map. Training and classification with conventional support vector machines is efficient when inner products between feature vectors can be evaluated efficiently [14, 19, 20]. We will see that classifiers based on quantum circuits, such as the one presented in Fig 2(c) cannot provide a quantum advantage over a conventional support vector machine if the feature vector kernel $K(\vec{x}, \vec{z}) = |\langle \Phi(\vec{x}) | \Phi(\vec{z}) \rangle|^2$ is too simple. For example, a classifier that uses a feature map that only generates product states can immediately be implement classically. To obtain an advantage over classical approaches we need to implement a map based on circuits that are hard to simulate classically. Since quantum computers are not expected to be classically simulable, there exists a long list of (universal) circuit families one can choose from. Here, we propose to use a circuit that works well in our experiments and is not too deep. We define a feature map on n-qubits generated by the unitary $\mathcal{U}_{\Phi}(\vec{x}) = U_{\Phi(\vec{x})} H^{\otimes n} U_{\Phi(\vec{x})} H^{\otimes n}$, where H denotes the conventional Hadamard gate and

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

is a diagonal gate in the Pauli Z - basis, c.f. Fig 1 (b). This circuit will act on $|0\rangle^n$ as initial state. We use the coefficients $\phi_S(\vec{x}) \in \mathbb{R}$, to encode the data $\vec{x} \in \Omega$. In general any diagonal unitary $U_{\Phi(\vec{x})}$ can be used if it can be implemented efficiently. This is for instance the case when only weight $|S| \leq 2$ interactions are considered. The exact evaluation of the inner-product between two states generated from a similar circuit with only a single diagonal layer $U_{\Phi(\vec{x})}$ is $\#P$ - hard [21]. Nonetheless, in the experimentally relevant context of additive error approximation, simulation of a single layer preparation circuit can be achieved efficiently classically by uniform sampling [22]. We conjecture that the evaluation of inner products generated from circuits with two basis changes and diagonal gates up to additive error to be hard, c.f. supplementary material for a discussion.

The data: To test our two methods, we generate artificial data that can be fully separated by our feature map. We use the map for $n = d = 2$ - qubits in Fig. 1(b) with $\phi_{\{i\}}(\vec{x}) = x_i$ and $\phi_{\{1,2\}}(\vec{x}) = (\pi - x_1)(\pi - x_2)$. We generate the labels for data vectors $\vec{x} \in T \cup S \subset (0, 2\pi]^2$, by first choosing $\mathbf{f} = Z_1 Z_2$ as the parity function and a random unitary $V \in SU(4)$. We assign $m(\vec{x}) = +1$, when $\langle \Phi(\vec{x}) | V^\dagger \mathbf{f} V | \Phi(\vec{x}) \rangle \geq \Delta$ and $m(\vec{x}) = -1$ when $\langle \Phi(\vec{x}) | V^\dagger \mathbf{f} V | \Phi(\vec{x}) \rangle \leq -\Delta$, c.f. Fig 3(b). The data has been separated by a gap of $\Delta = 0.3$. Both the training sets and the classification sets consist of 20 data points per label. We show one of such classification sets as circle symbols in Fig. 3(b).

Quantum variational classification: The first classification protocol follows four steps. First, the data

$\vec{x} \in \Omega$ is mapped to a quantum state by applying the feature map circuit $\mathcal{U}_{\Phi(\vec{x})}$ in Fig. 1(b) to a reference state $|0\rangle^n$. Second, a short depth quantum circuit $W(\vec{\theta})$, described in Fig 2(b) is applied to the feature state. A circuit with l - layers is parametrized by $\vec{\theta} \in \mathbb{R}^{2n(l+1)}$ that will be optimized during training. Third, for a two label classification $y \in \{+1, -1\}$ problem, a binary measurement $\{M_y\}$ is applied to the state $W(\vec{\theta})\mathcal{U}_{\Phi(\vec{x})}|0\rangle^n$. This measurement is implemented by measurements in the Z - basis and feeding the output bit-string $z \in \{0,1\}^n$ to a chosen boolean function $f : \{0,1\}^n \to \{+1, -1\}$. The measurement operator is given by $M_y = 2^{-1}(\mathbb{1} + y\mathbf{f})$, where we have defined $\mathbf{f} = \sum_{z \in \{0,1\}^n} f(z)|z\rangle\langle z|$. The probability of obtaining outcome y is $p_y(\vec{x}) = \langle \Phi(\vec{x})|W^\dagger(\vec{\theta})M_yW(\vec{\theta})|\Phi(\vec{x})\rangle$. Fourth, for the decision rule we perform R repeated measurement shots to obtain the empirical distribution $\hat{p}_y(\vec{x})$. We assign the label $\tilde{m}(\vec{x}) = y$, whenever $\hat{p}_y(\vec{x}) > \hat{p}_{-y}(\vec{x}) - yb$, where we have introduced an additional bias parameter $b \in [-1, 1]$ that can be optimized during training.

The feature map circuit $\mathcal{U}_{\Phi(\vec{x})}$ as well as the boolean function f are fixed choices. During the training of the classifier we optimize the parameters $(\vec{\theta}, b)$. For the optimization, we need to define a cost-function. We define the empirical risk $R_{\text{emp}}(\vec{\theta})$ given by the error probability $\Pr(\tilde{m}(\vec{x}) \neq m(\vec{x}))$ of assigning the incorrect label averaged over the samples in the training set T,

$$R_{\text{emp}}(\vec{\theta}) = \frac{1}{|T|} \sum_{\vec{x} \in T} \Pr(\tilde{m}(\vec{x}) \neq m(\vec{x})) .$$

For the binary problem, the error probability of assigning the wrong label is given by the binomial cumulative density function (CDF) of the empirical distribution $\hat{p}_y(\vec{x})$, c.f. supplementary material for a derivation. The binomial CDF can be approximated for a large number of samples (shots) $R \gg 1$ by a sigmoid function $\text{sig}(x) = (1 + e^{-x})^{-1}$. The probability that the label $m(\vec{x}) = y$ is assigned incorrectly is approximated by

$$\Pr(\tilde{m}(\vec{x}) \neq m(\vec{x})) \approx \text{sig}\left(\frac{\sqrt{R}\left(\frac{1}{2} - \left(\hat{p}_y(\vec{x}) - \frac{yb}{2}\right)\right)}{\sqrt{2(1 - \hat{p}_y(\vec{x}))\hat{p}_y(\vec{x})}} \right) .$$

The experiment itself is split in to two phases; First, we train the classifier and optimize $(\vec{\theta}, b)$. We have found that Spall's SPSA [23, 24] stochastic gradient decent algorithm performs well in the noisy experimental setting. We can use the circuit as a classifier after the parameters have converged to $(\vec{\theta}^*, b^*)$. Second, in the classification phase, the classifier assigns labels to unlabeled data $\vec{s} \in S$ according to the decision rule $\tilde{m}(\vec{s})$.

We implement the quantum variational classifier $W(\vec{\theta})$ over 5 different depths ($l = 0$ through $l = 4$), c.f. Fig 2(b), in our superconducting quantum processor. We expect a higher classification success for increased
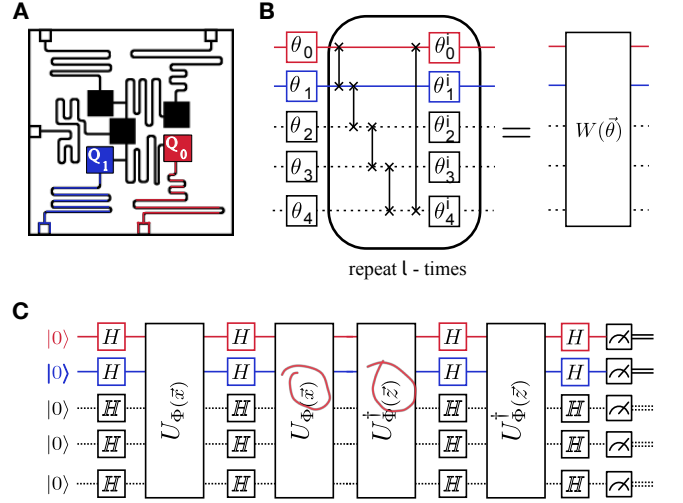

FIG. 2. **Experimental implementations** (a) Schematic of the 5-qubit quantum processor. The experiment was performed on qubits Q_0 and Q_1, highlighted in the image. (b) Variational circuit used for our optimization method. Here we choose a rather common Ansatz for the variational unitary $W(\vec{\theta}) = U_{\text{loc}}^{(l)}(\theta_l) \, U_{ent} \dots U_{\text{loc}}^{(2)}(\theta_2) \, U_{\text{ent}} \, U_{\text{loc}}^{(1)}(\theta_1)$ [16, 17]. We alternate layers of entangling gates $U_{ent} = \prod_{(i,j) \in E} \mathsf{CZ}(i,j)$ with full layers single qubit rotations $U_{loc}^{(t)}(\theta_t) = \otimes_{i=1}^n U(\theta_{i,t})$ with $U(\theta_{i,t}) \in \text{SU}(2)$. For the entangling step we use controlled phase gates $\mathsf{CZ}(i,j)$ along the edges $(i,j) \in E$ present in the connectivity of the superconducting chip. (c) Circuit to directly estimate the fidelity between a pair of feature vectors for data $\vec{x}$ and $\vec{y}$ as used for our second method.

depth. The binary measurement is obtained from the parity function $\mathbf{f} = Z_1 Z_2$. For each depth we train three different data sets, using training sets consisting of 20 data points per label. One of these data sets is shown in Fig. 3 (b), along with the training set used for this particular data set. Fig. 3(a) shows the optimization of the empirical risk $R_{\text{emp}}(\vec{\theta})$ for two different training sets and depths. In all experiments throughout this work we implemented an error mitigation technique which relies on zero-noise extrapolation to first order [10, 25]. To obtain a zero-noise estimate, a copy of the circuit was run on a time scale slowed down by a factor of 1.5, c.f. supplemental material. This technique is implemented at each trial step, and it is the mitigated cost function that is fed to the classical optimizer. We observe that the empirical risk in Fig. 3(a) converges to a lower value for depth $l = 4$ than for $l = 0$, albeit with more optimization steps. Whereas error mitigation does not appreciably improve the results for depth 0 - the noise in our system is not the limiting factor in that case-, it does help substantially for larger depths. Although $\Pr(\tilde{m}(\vec{x}) \neq m(\vec{x}))$ explicitly includes the number of experimental shots taken, we fixed $R = 200$ to avoid gradient problems, even though we took 2000 shots in the actual experiment.

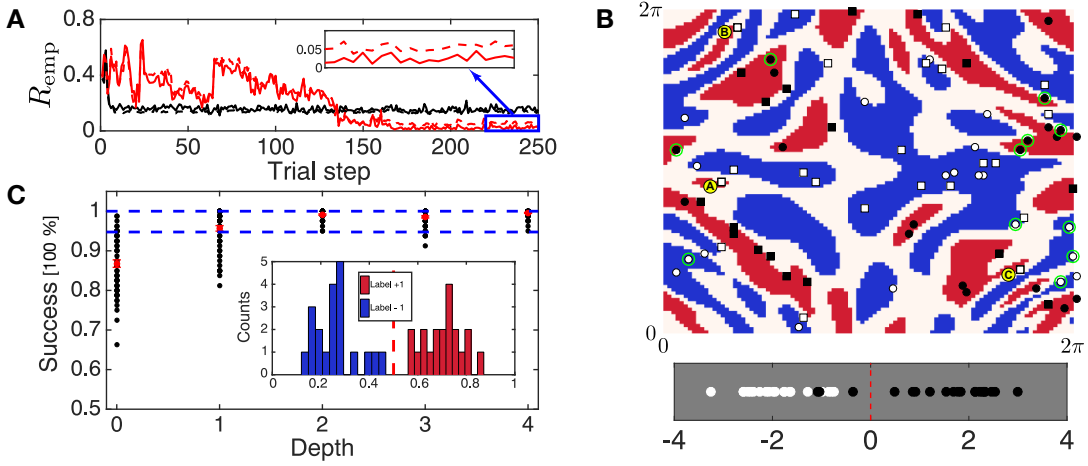After each training is completed, we use the trained

FIG. 3. **Convergence of the method and classification results**: (a) Convergence of the cost function $R_{\mathrm{emp}}(\vec{\theta})$ after 250 iterations of Spall's SPSA algorithm. Red (black) curves correspond to $l = 4$ ($l = 0$). The value of the cost function, obtained from estimates of $\hat{p}_k$ after zero-noise extrapolation (solid lines), is compared with values obtained from experimental measurements of $\hat{p}_k$ with standard single and two-qubit gate times (dashed). We train three sets of data per depth and perform 20 classifications per trained set. The results of these classifications are shown in (c) as black dots (amounting to 60 per depth), with mean values at each depth represented by red dots. The error bar is the standard error of the mean. The inset shows histograms as a function of the probability of measuring label $+1$ for a test set of 20 points per label obtained with a $l = 4$ classifier circuit, depicting classification with 100% success.. The dashed blue lines show the results of our direct kernel estimation method for comparison, with Sets I and II yielding 100% success and Set III yielding 94.75% success. (b) Example data set used for both methods in this work. The data labels (red for $+1$ label and blue for -1 label) are generated with a separation gap of magnitude 0.3 between them (white areas). The 20 points per label training set is shown as white and black circles. For our quantum kernel estimation method we show the obtained support vectors (green circles) and a classified test set (white and black squares). Three of the test sets points are misclasified, labeled as A, B, and C. For each of the test data points $\vec{x}_j$ we plot at the bottom of (b) the amount $\sum_i y_i \alpha_i^* K(\vec{x}_i, \vec{x}_j)$ over all support vectors $\vec{x}_i$, where $y_i \in \{+1, -1\}$ are the labels of the support vectors. Points A, B, and C, all belonging to label $+1$, give $\sum_i y_i \alpha_i^* K(\vec{x}_i, \vec{x}_j) =$ -1.033, -0.367 and -1.082, respectively.

set of parameters $(\vec{\theta}^*, b^* = 0)$ to classify 20 different test sets -randomly drawn each time- per data set. We run these classification experiments at 10,000 shots, versus the 2,000 used for training. The classification of each data point is error-mitigated and repeated twice, averaging the success ratios obtained in each of the two classifications. Fig. 3 (c) shows the classification results for our quantum variational approach. We clearly see an increase in classification success with increasing circuit depth, c.f. Fig. 3(c), reaching values very close to 100% for depths larger than 1. This classification success remarkably remains up to depth 4, despite the decoherence associated with 8 CNOTs in the training and classification circuits, for $l = 4$.

A path to quantum advantage: Such variational circuit classifiers are directly related to conventional SVMs [14, 19]. To see why a quantum advantage can only be obtained for feature maps with a classically hard to estimate kernel, we point out the following: The decision rule $p_y(\vec{x}) > p_{-y}(\vec{x}) - yb$ can be restated as $\tilde{m}(\vec{x}) = \mathrm{sign}(\langle \Phi(\vec{x}) | W^\dagger(\vec{\theta}) \mathbf{f} W(\vec{\theta}) | \Phi(\vec{x}) \rangle + b)$. The variational circuit W followed by a binary measurement can be understood as a separating hyperplane

in quantum state space. Choose an orthogonal, hermitian, matrix basis $\{P_\alpha\} \subset \mathbb{C}^{2^n \times 2^n}$, where $\alpha = 1, \ldots, 4^n$ with $\mathrm{tr}\left[P_\alpha^\dagger P_\beta\right] = 2^n \delta_{\alpha,\beta}$ such as the Pauli-group on n-qubits. Expand both the quantum state $|\Phi(\vec{x})\rangle\langle\Phi(\vec{x})|$ and the measurement $W^\dagger(\vec{\theta})\mathbf{f}W(\vec{\theta})$ in this matrix basis. Both the expectation value of the binary measurement and the decision rule can be expressed in terms of $w_\alpha(\vec{\theta}) = \mathrm{tr}\left[W^\dagger(\vec{\theta})\mathbf{f}W(\vec{\theta})P_\alpha\right]$ and $\Phi_\alpha(\vec{x}) = \langle\Phi(\vec{x})|P_\alpha|\Phi(\vec{x})\rangle$. For any variational unitary the classification rule can be restated in the familiar SVM form $\tilde{m}(x) = \mathrm{sign}\left(2^{-n}\sum_\alpha w_\alpha(\vec{\theta})\Phi_\alpha(\vec{x}) + b\right)$. The classifier can only be improved when the constraint is lifted that the w_α come from a variational circuit. The optimal w_α can alternatively be found by employing kernel methods and considering the standard Wolfe - dual of the SVM [14]. Moreover, this decomposition indicates that one should think of the feature space as the quantum state space with feature vectors $|\Phi(\vec{x})\rangle\langle\Phi(\vec{x})|$ and inner products $K(\vec{x}, \vec{z}) = |\langle\Phi(\vec{x})|\Phi(\vec{z})\rangle|^2$. Indeed, the direct use of the Hilbert space $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$ as a feature space would lead to a conceptual problem, since a vector $|\Phi(\vec{x})\rangle \in \mathcal{H}$ is only physically defined up to a global phase.

Quantum kernel estimation: The second classification protocol uses this connection to implement the SVM directly. Rather than using a variational quantum circuit to generate the separating hyperplane, we use a classical SVM for classification. The quantum computer is used twice in this protocol. First, the kernel $K(\vec{x}_i, \vec{x}_j)$ is estimated on a quantum computer for all pairs of training data $\vec{x}_i, \vec{x}_j \in T$, c.f. Fig. 2(c). Here it will be convenient to write $T = \{\vec{x}_1, \ldots, \vec{x}_t\}$ with $t = |T|$; also let $y_i = m(\vec{x}_i)$ be the corresponding label. The optimization problem for the optimal SVM can be formulated in terms of a dual quadratic program that only uses access to the kernel. We maximize

$$L_D(\alpha) = \sum_{i=1}^{t} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{t} y_i y_j \alpha_i \alpha_j K(\vec{x}_i, \vec{x}_j),$$

subject to $\sum_{i=1}^{t} \alpha_i y_i = 0$ and $\alpha_i \geq 0$ for each i. This problem is concave whenever $K(\vec{x}_i, \vec{x}_j)$ is a positive definite matrix. The solution to this problem will be given by a nonnegative vector $\vec{\alpha} = (\alpha_1, \ldots, \alpha_t)$. The quantum computer is used a second time to estimate the kernel for a new datum $\vec{s} \in S$ with all the support vectors. The optimal solution $\vec{\alpha}^*$ is used to construct the classifier

$$\tilde{m}(\vec{s}) = \text{sign}\left(\sum_{i=1}^{t} y_i \alpha_i^* K(\vec{x}_i, \vec{s}) + b\right).$$

Due to complementary slackness, we expect that many of the α_i will be zero. This can make the evaluation of $\tilde{m}(\vec{s})$ cheaper, since $K(\vec{x}_i, \vec{s})$ only needs to be estimated when $\alpha_i^* > 0$. The bias b in $\tilde{m}(\vec{s})$ can calculated from the weights α_i^* by choosing any i with $\alpha_i^* > 0$ and solving $\sum_j y_j \alpha_j^* K(\vec{x}_j, \vec{x}_i) + b = y_i$ for b.

Let us discuss how the quantum computer is used to estimate the kernel. The kernel entries are the fidelities between different feature vectors. Various methods [26, 27] exist, such as the swap test, to estimate the fidelity between general quantum states. However, since the states in the feature space are not arbitrary, the overlap can be estimated directly from the transition amplitude $|\langle \Phi(\vec{x}) | \Phi(\vec{z}) \rangle|^2 = |\langle 0^n | \mathcal{U}_{\Phi(\vec{x})}^\dagger \mathcal{U}_{\Phi(\vec{z})} | 0^n \rangle|^2$. First, we apply the circuit Fig. 2(c), a composition of two consecutive feature map circuits, to the initial reference state $|0^n\rangle$. Second, we measure the final state in the Z-basis R - times and record the number of all zero strings 0^n. The frequency of this string is the estimate of the transition probability. The kernel entry is obtained to an additive sampling error of $\tilde{\epsilon}$ when $\mathcal{O}(\tilde{\epsilon}^{-2})$ shots are used. In the training phase a total of $\mathcal{O}(|T|^2)$ amplitudes have to be estimated. An estimator $\hat{K}$ for the kernel matrix that deviates with high probability in operator norm from the exact kernel K by at most $\|K - \hat{K}\| \leq \epsilon$ can be obtained with a total of $R = \mathcal{O}(\epsilon^{-2}|T|^4)$ shots. The sampling error can compromise the positive semi-definiteness of the kernel. Although not applied in this work, this can
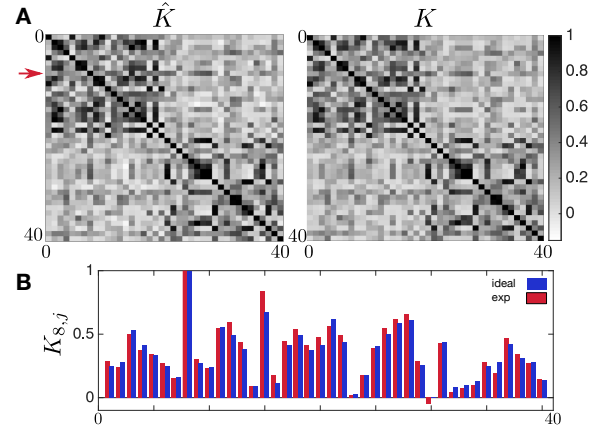

FIG. 4. **Kernels for Set III**. (a) Experimental (left) and ideal (right) kernel matrices containing the inner products of all data points used for training Set III (round symbols in Fig. 3 (b)). The maximal deviation from the ideal kernel $|K - \hat{K}|$ occurs at element $K_{8,15}$. A cut through row 8 (indicated by red arrow in (a)) is shown in (b), where the experimental (ideal) results are shown as red (blue) bars. Note that entries that are close zero in the kernel can become negative (c.f. $K_{8,30}$ (b)) when the error-mitigation technique is applied.

be remedied by employing an adaption of the scheme presented in [28]. The direct connection to conventional SVMs enables us to use the conventional bounds on the VC-dimension that ensure convergence and guide the structural risk minimization.

For the experimental implementation of estimating the kernel matrix K, c.f. circuit Fig. 2(c), we again apply the error-mitigation protocol [10, 25] to first order. The kernel entries are obtained by running a time-stretched copy of the circuit and reporting the mitigated entry. We use $50,000$ shots per matrix entry. Using this protocol, we obtain support vectors α_i that are very similar to the noise-free case. We run the training stage on three different data sets, which we will label as Set I, Set II and Set III. Set III is shown in Fig. 3(b). Note that the training data used to obtain the kernel and the support vectors is the same data used in training of our variational classifier. The support vectors (green circles in (b)) are then used to classify 10 different test sets randomly drawn from each entire set. Set I and Set II yield 100% success over the classification of all 10 different test sets each, whereas Set III averages a success of 94.75%. For more details see the supplementary information. These classification results are given in Fig. 3(c) as dashed blue lines to compare with the results of our variational method. In Fig. 4(a) we show the ideal and the experimentally obtained kernel matrices, K and $\hat{K}$, for Set III. The maximum difference across the matrices between K and $\hat{K}$ is found at row (or column) 8. This is shown in Fig. 4(b). All support vectors for the three sets and

equivalent plots are given in the supplementary material.

Conclusions: We have experimentally demonstrated a classifier that exploits a quantum feature space. The kernel of this feature space has been conjectured to be hard to estimate classically. In the experiment we find that even in the presence of noise, we are capable of achieving success rates up to 100%. In the future it becomes intriguing to find suitable feature maps for this technique with provable quantum advantages while providing significant improvement on real world data sets. With the ubiquity of kernel methods in machine learning, we are optimistic that our technique will find application beyond binary classification.

During composition of this manuscript we became aware of the independent theoretical work by Schuld et al. [29, 30].

Author contributions The work on the classifier theory was led by V.H. and K.T. The experiment was designed by A.D.C, J.M.G and K.T. and implemented by A.D.C. All authors contributed to the manuscript.

Author information The authors declare no competing financial interests. Correspondence and requests for materials should be addressed to A.D.C. and K.T.

[1] Mitarai, K., Negoro, M., Kitagawa, M. & Fujii, K. Quantum circuit learning. *arXiv preprint arXiv:1803.00745* (2018).

[2] Farhi, E. & Neven, H. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002* (2018).

[3] Preskill, J. Quantum computing in the nisq era and beyond. *arXiv preprint arXiv:1801.00862* (2018).

[4] Arunachalam, S. & de Wolf, R. Guest column: a survey of quantum learning theory. *j-SIGACT* **48**, 41–67 (2017).

[5] Ciliberto, C. *et al.* Quantum machine learning: a classical perspective. *Proc. R. Soc. A* **474**, 20170551 (2018).

[6] Dunjko, V. & Briegel, H. J. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics* (2018).

[7] Biamonte, J. *et al.* Quantum machine learning. *Nature* **549**, 195 (2017).

[8] Romero, J., Olson, J. P. & Aspuru-Guzik, A. Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology* **2**, 045001 (2017).

[9] Wan, K. H., Dahlsten, O., Kristjánsson, H., Gardner, R. & Kim, M. Quantum generalisation of feedforward neural networks. *arXiv preprint arXiv:1612.01045* (2016).

[10] Temme, K., Bravyi, S. & Gambetta, J. M. Error mitigation for short-depth quantum circuits. *Physical review letters* **119**, 180509 (2017).

[11] Li, Y. & Benjamin, S. C. Efficient variational quantum simulator incorporating active error minimization. *Physical Review X* **7**, 021050 (2017).

[12] Terhal, B. M. & DiVincenzo, D. P. Adaptive quantum computation, constant depth quantum circuits and arthur-merlin games. *Quant. Inf. Comp.* **4**, 134–145 (2004).

[13] Bremner, M. J., Montanaro, A. & Shepherd, D. J. Achieving quantum supremacy with sparse and noisy commuting quantum computations. *Quantum* **1**, 8 (2017). URL https://doi.org/10.22331/q-2017-04-25-8.

[14] Vapnik, V. *The nature of statistical learning theory* (Springer science & business media, 2013).

[15] Rebentrost, P., Mohseni, M. & Lloyd, S. Quantum support vector machine for big data classification. *Physical review letters* **113**, 130503 (2014).

[16] Kandala, A. *et al.* Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature* **549**, 242 (2017).

[17] Farhi, E., Goldstone, J., Gutmann, S. & Neven, H. Quantum algorithms for fixed qubit architectures. *arXiv preprint arXiv:1703.06199* (2017).

[18] Rigetti, C. & Devoret, M. Fully microwave-tunable universal gates in superconducting qubits with linear couplings and fixed transition frequencies. *Phys. Rev. B* **81**, 134507 (2010). URL https://link.aps.org/doi/10.1103/PhysRevB.81.134507.

[19] Burges, C. J. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery* **2**, 121–167 (1998).

[20] Boser, B. E., Guyon, I. M. & Vapnik, V. N. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, 144–152 (ACM, 1992).

[21] Goldberg, L. A. & Guo, H. The complexity of approximating complex-valued ising and tutte partition functions. *computational complexity* **26**, 765–833 (2017).

[22] Demarie, T. F., Ouyang, Y. & Fitzsimons, J. F. Classical verification of quantum circuits containing few basis changes. *Physical Review A* **97**, 042319 (2018).

[23] Spall, J. C. A one-measurement form of simultaneous perturbation stochastic approximation. *Automatica* **33**, 109 (1997).

[24] Spall, J. C. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE Transaction on Automatic Control* **45**, 1839 (2000).

[25] Kandala, A. *et al.* Extending the computational reach of a noisy superconducting quantum processor. *arXiv preprint arXiv:1805.04492* (2018).

[26] Buhrman, H., Cleve, R., Watrous, J. & De Wolf, R. Quantum fingerprinting. *Physical Review Letters* **87**,

167902 (2001).

[27] Cincio, L., Subaşı, Y., Sornborger, A. T. & Coles, P. J. Learning the quantum algorithm for state overlap. *arXiv preprint arXiv:1803.04114* (2018).

[28] Smolin, J. A., Gambetta, J. M. & Smith, G. Efficient method for computing the maximum-likelihood quantum state from measurements with additive gaussian noise. *Physical review letters* **108**, 070502 (2012).

[29] Schuld, M. & Killoran, N. Quantum machine learning in feature hilbert spaces. *arXiv preprint arXiv:1803.07128* (2018).

[30] Schuld, M., Bocharov, A., Svore, K. & Wiebe, N. Circuit-centric quantum classifiers. *arXiv preprint arXiv:1804.00633* (2018).

[31] Stoudenmire, E. & Schwab, D. J. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems*, 4799–4807 (2016).

[32] Van Dam, W., Hallgren, S. & Ip, L. Quantum algorithms for some hidden shift problems. *SIAM Journal on Computing* **36**, 763–778 (2006).

[33] Rötteler, M. Quantum algorithms for highly non-linear boolean functions. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms*, 448–457 (Society for Industrial and Applied Mathematics, 2010).

[34] Bremner, M. J., Montanaro, A. & Shepherd, D. J. Average-case complexity versus approximate simulation of commuting quantum computations. *Physical review letters* **117**, 080501 (2016).

[35] d'Alessandro, D. *Introduction to quantum control and dynamics* (CRC press, 2007).

[36] Tropp, J. A. *et al.* An introduction to matrix concentration inequalities. *Foundations and Trends ® in Machine Learning* **8**, 1–230 (2015).

[37] Bergeal, N. *et al.* Analog information processing at the quantum limit with a josephson ring modulator. *Nature Physics* **6**, 296 EP – (2010). URL http://dx.doi.org/10.1038/nphys1516. Article.

[38] McKay, D. C., Wood, C. J., Sheldon, S., Chow, J. M. & Gambetta, J. M. Efficient z gates for quantum computing. *Physical Review A* **96**, 022330 (2017).

[39] Gambetta, J. M. *et al.* Characterization of addressability by simultaneous randomized benchmarking. *Phys. Rev. Lett.* **109**, 240504 (2012). URL https://link.aps.org/doi/10.1103/PhysRevLett.109.240504.

[40] Córcoles, A. D. *et al.* Process verification of two-qubit quantum gates by randomized benchmarking. *Phys. Rev. A* **87**, 030301 (2013). URL https://link.aps.org/doi/10.1103/PhysRevA.87.030301.

[41] Sheldon, S., Magesan, E., Chow, J. M. & Gambetta, J. M. Procedure for systematically tuning up cross-talk in the cross-resonance gate. *Phys. Rev. A* **93**, 060302 (2016).

SUPPLEMENTARY INFORMATION:

SUPERVISED LEARNING WITH QUANTUM ENHANCED FEATURE SPACES

Classification problems

Consider a classification task on a set $C = \{1, 2 \ldots c\}$ of c classes (labels) in a supervised learning scenario. In such settings, we are given a training set T and a test set S, both are assumed to be labeled by a map $m : T \cup S \to C$ unknown to the programmer. Both sets S and T are provided to the programmer, but the programmer only receives the labels of the training set. So, formally, the programmer has only access to a restriction $m_{|T}$ of the indexing map m:

$$m_{|T} : T \to C, \text{ s.t.: } m_{|T}(\vec{t}) = m(\vec{t}), \ \forall \vec{t} \in T.$$

It is the programmer's goal to use the knowledge of $m_{|T}$ to infer an indexing map $\tilde{m} : S \to C$ over the set S, such that $m(\vec{s}) = \tilde{m}(\vec{s})$ with high probability for any $\vec{s} \in S$. The accuracy of the approximation to the map is quantified by a classification success rate, proportional to the number of collisions of m and $\tilde{m}$:

$$\nu_{succ.} = \frac{|\{\vec{s} \in S | m(\vec{s}) = \tilde{m}(\vec{s})\}|}{|S|}.$$

For such a learning task to be meaningful it is assumed that there is a correlation in output of the indexing map m over the sets S and T. For that reason, we assume that both sets could in principle be constructed by drawing the S and T sample sets from a family of d-dimensional distributions $\left\{p_c : \Omega \subset \mathbb{R}^d \to \mathbb{R}\right\}_{c \in C}$ and labeling the outputs according to the distribution. It is assumed that the hypothetical classification function m to be learned is constructed this way. The programmer, however, does not have access to these distributions of the labeling function directly. She is only provided with a large, but finite number of samples and the matching labels.

The conventional approach to this problem is to construct a family of classically computable function $\tilde{m} : \langle \vec{\theta}, S \rangle \to C$, indexed by a set of parameters $\vec{\theta}$. These weights are then inferred from $m_{|T}$ by a optimization procedure on a classical cost function. We consider a scenario where the whole, or parts of the classification protocol m, are generated on a quantum computer.

Description of the Algorithm

We consider two different learning schemes. The first is referred to as "Quantum variational classification", the second is referred to "Quantum kernel estimation". Both schemes construct a separating hyperplane in the state space of n qubits. The classical data is mapped to this space with dim $= 4^n$ using a unitary circuit family starting from the reference state $|0\rangle\langle 0|^n$.

Quantum variational classification

For our first classification approach we design a variational algorithm which exploits the large dimensional Hilbert space of our quantum processor to find an optimal cutting hyperplane in a similar vein as Support Vector Machines (SVM) do. The algorithm consists of two main parts: a training stage and a classification stage. For the training stage, a set of labeled data points are provided, on which the algorithm is performed. For the classification stage, we take a different set of data points and run the optimized classifying circuit on them without any label input. Then we compare the label of each data point to the output of the classifier to obtain a success ratio for the data set. For both the training and the classification stages, the quantum circuit that implements the algorithm comprises three main parts: the encoding of the feature map, the variational optimization and the measurement, Fig S1. The training phase consists of these steps.
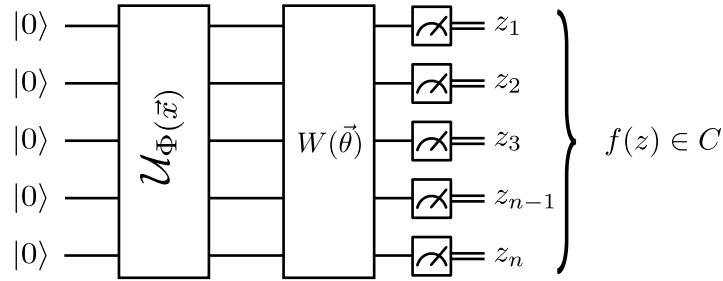

FIG. S1. Quantum variational classification: The circuit takes a references state, $|0\rangle^n$, applies the unitary $\mathcal{U}_{\Phi(vecx)}$ followed by the variational unitary $W(\vec{\theta})$ and applies a measurement in the Z-basis. The resulting bit string $z \in \{0,1\}^n$ is then mapped to a label in C. This circuit is re - run multiple times and sampled to estimate the expectation value $p_y = \langle\Phi(\vec{x})|W^\dagger(\vec{\theta})M_y W(\vec{\theta})|\Phi(\vec{x})\rangle$ for the labels $y \in C$. In the experiment we consider $C = \{+1, -1\}$.

Algorithm 1 Quantum variational classification: the training phase

---

1: **Input** Labeled training samples $T = \{\vec{x} \in \Omega \subset \mathbb{R}^n\} \times \{y \in C\}$, Optimization routine,
2: **Parameters** Number of measurement shots R, and initial parameter $\vec{\theta}_0$.
3: Calibrate the quantum Hardware to generate short depth trial circuits.
4: Set initial values of the variational parameters $\vec{\theta} = \vec{\theta}_0$ for the short-depth circuit $W(\vec{\theta})$
5: **while** Optimization (e.g. SPSA) of $R_{\mathrm{emp}}(\vec{\theta})$ has not converged **do**
6:     **for** $i = 1$ **to** $|T|$ **do**
7:         Set the counter $r_y = 0$ for every $y \in C$.
8:         **for** $shot = 1$ **to** R **do**
9:             Use $\mathcal{U}_{\Phi(\vec{x}_i)}$ to prepare initial feature-map state $|\Phi(\vec{x}_i)\rangle\langle\Phi(\vec{x}_i)|$
10:             Apply discriminator circuit $W(\vec{\theta})$ to the initial feature-map state .
11:             Apply $|C|$ - outcome measurement $\{M_y\}_{y\in C}$
12:             Record measurement outcome label y by setting $r_y \to r_y + 1$
13:         **end for**
14:         Construct empirical distribution $\hat{p}_y(\vec{x}_i) = r_y R^{-1}$.
15:         Evaluate $\Pr(\tilde{m}(\vec{x}_i) \neq y_i | m(\vec{x}) = y_i)$ with $\hat{p}_y(\vec{x}_i)$ and y_i
16:         Add contribution $\Pr(\tilde{m}(\vec{x}_i) \neq y_i | m(\vec{x}) = y_i)$ to cost function $R_{\mathrm{emp}}(\vec{\theta})$.
17:     **end for**
18:     Use optimization routine to propose new $\vec{\theta}$ with information from $R_{\mathrm{emp}}(\vec{\theta})$
19: **end while**
20: **return** the final parameter $\vec{\theta}^*$ and value of the cost function $R_{\mathrm{emp}}(\theta^*)$

The classification can be applied when the training phase is complete. The optimal parameters are used to decide the correct label for new input data. Again, the same circuit is applied as in Fig S1, however, this time the parameters are fixed and and the outcomes are combined to determine the label which is reported as output of the classifier.

Algorithm 2 Quantum variational classification: the classification phase

---

1: **Input** An unlabeled sample from the test set $\vec{s} \in S$, optimal parameters $\vec{\theta}^*$ for the discriminator circuit.
2: **Parameters** Number of measurement shots R
3: Calibrate the quantum Hardware to generate short depth trial circuits.
4: Set the counter $r_y = 0$ for every $y \in C$.
5: **for** $shot = 1$ **to** R **do**
6:     Use $\mathcal{U}_{\Phi(\vec{s})}$ to prepare initial feature-map state $|\Phi(\vec{s})\rangle\langle\Phi(\vec{s})|$
7:     Apply optimal discriminator circuit $W(\vec{\theta}^*)$ to the initial feature-map state .
8:     Apply $|C|$ - outcome measurement $\{M_y\}_{y \in C}$
9:     Record measurement outcome label y by setting $r_y \to r_y + 1$
10: **end for**
11: Construct empirical distribution $\hat{p}_y(\vec{s}) = r_y R^{-1}$.
12: Set label $= \mathrm{argmax}_y\{\hat{p}_y(\vec{s})\}$
13: **return** label

Quantum kernel estimation

For the second classification protocol, we restrict ourselves to the binary label case, with $C = \{+1, -1\}$. In this protocol we only use the quantum computer to estimate the $|T| \times |T|$ kernel matrix $K(\vec{x}_i, \vec{x}_j) = |\langle\Phi(\vec{x}_i)|\Phi(\vec{x}_j)\rangle|^2$. For all pairs of points $\vec{x}_i, \vec{x}_j \in T$ in the the training data, we sample the overlap to obtain the matrix entry in the kernel. This output probability can be estimated from the circuit depicted in Fig. S5.b. by sampling the output distribution with R shots and only taking the 0^n count. After the kernel matrix for the full training data has been constructed we use the conventional (classical) support vector machine classifier. The optimal hyperplane is constructed by solving the dual problem L_D in eqn. (6), which is completely specified after we have been given the labels y_i and have estimated the kernel $K(\vec{x}_i, \vec{x}_j)$. The solution of the optimization problem is given in terms of the support vectors N_S for which $\alpha_i > 0$.

In the classification phase, we want to assign a label to a new datum $\vec{s} \in S$ of the test set. For this, the inner product $K(\vec{x}_i, \vec{s})$ between all support vectors $\vec{x}_i \in T$ with $i \in N_S$ and the new datum $\vec{s}$ has to be estimated on the quantum computer. The new label $\tilde{m}(\vec{s})$ for the datum is assigned according to eqn. (14). Since all support vectors are known from the training phase and we have obtained access to the kernel $K(\vec{x}_i, \vec{s})$ from the quantum hardware, the label can be directly computed.

The Relationship of variational quantum classifiers to support vector machines

The references [14, 19] provide a detailed introduction to the construction of support vector machines for pattern recognition. Support vector machines are an important tool to construct classifiers for tasks in supervised learning. We will show that the variational circuit classifier bears many similarities to a classical non-linear support vector machine.

Support vector machines (SVM):

First, let us briefly review the training task of classical, linear support vector machines for data where $C = \{+1, -1\}$, so that $(\vec{x}_i, y_i)_{i=1,...,t}$ with $\vec{x}_i \in T \subset \mathbb{R}^d$, $y_i \in \{+1, -1\}$ that is linearly separable. Linear separability asks that the set of points can be split in two regions by a hyperplane $(\mathbf{w}, b)$, parametrized by a normal vector $\mathbf{w} \in \mathbb{R}^d$ and a bias $b \in \mathbb{R}$. The points $\vec{x} \in \mathbb{R}^d$ that lie directly on the hyperplane satisfy the equation

$$\mathbf{w} \circ \vec{x} + b = 0 \tag{1}$$

expressed in terms of the inner product $\circ$ for vectors in $\mathbb{R}^d$. The perpendicular distance of the hyperplane to the origin in $\mathbb{R}^n$ is given by $b\|\mathbf{w}\|^{-1}$. The data set $\{\mathbf{x_i}, y_i\}$ is linearly separable by margin $2\|\mathbf{w}\|^{-1}$ in $\mathbb{R}^d$ if there exists a

vector $\mathbf{w}$ and a b, such that:

$$y_i \left(\mathbf{w} \circ \vec{x}_i + b\right) \geq 1. \quad \forall i = 1, \ldots, t \tag{2}$$

The classification function $\tilde{m}(\vec{x}, (\mathbf{w}, b))$ that is constructed from such a hyperplane for any new data point $\vec{x} \in \mathbb{R}^n$ assigns the label according to which side of the hyperplane the new data-point lies by setting

$$\tilde{m}(\vec{x}, (\mathbf{w}, b)) = \text{sign}\left(\mathbf{w} \circ \vec{x} + b\right). \tag{3}$$

The task in constructing a linear support vector machine (SVM) in this scenario is the following. One is looking for a hyperplane that separates the data, with the largest possible distance between the two separated sets. The perpendicular distance between the plane and two points with different labels is called a margin and such points are referred to as 'support vectors'. This means that we want to maximize the margin by minimizing $||\mathbf{w}||$, or equivalently $||\mathbf{w}||^2$ subject to the constraints as given in eqn. (2), for all data points in the training set T. The corresponding cost function can be written as:

$$L_P = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{t} \alpha_i y_i (\mathbf{w} \circ \vec{x}_i + b) + \sum_{i=1}^{t} \alpha_i, \tag{4}$$

where $\alpha_i \geq 0$ are Lagrange multipliers chosen to ensure the constraints are satisfied.

For non-separable datasets, it is possible to introduce non-negative slack variables $\{\xi_i\}_{i=1,\ldots,t} \in \mathbb{R}_0^+$ which can be used to soften the constraints for linear separability of $(\vec{x}_i, y_i)$ to

$$y_i \left(\mathbf{w} \circ \vec{x}_i + b\right) \geq (1 - \xi_i),$$
$$\xi_i \geq 0. \tag{5}$$

These slack variables are then used to modify the objective function by $1/2||\mathbf{w}||^2 \rightarrow 1/2||\mathbf{w}||^2 + C(\sum_i \xi_i)^r + \sum_i \mu_i \xi$. When we choose $r \geq 1$ the optimization problem remains convex and a dual can be constructed. In particular, for $r = 1$, neither the ξ_i or their Lagrange multipliers μ_i appear in the dual Lagrangian.

It is very helpful to consider the dual of the original primal problem L_P in eqn. (4). The primal problem is a convex, quadratic programming problem, for which the Wolfe dual cost function L_D for the Lagrange multipliers can be readily derived by variation with respect to $\mathbf{w}$ and b. The dual optimization problem is

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \vec{x}_i \circ \vec{x}_j, \tag{6}$$

subject to constraints:

$$0 \leq \alpha \leq C, \qquad \qquad \sum_i \alpha_i y_i = 0.$$

The variables of the primal are given in terms of the dual variables by

$$\sum_i \alpha_i y_i \vec{x}_i = \mathbf{w} \tag{7}$$

and the bias b can be computed from the Karush-Kuhn-Tucker (KKT) conditions when the corresponding Lagrange multiplier does not vanish. The optimal variables satisfy the KKT conditions and play an important role in the understanding of the SVM. They are given for primal as

$$\partial_{\mathbf{w}_\beta} L_P = \mathbf{w}_\beta - \sum_i \alpha_i y_i \vec{x}_{i\beta} = 0 \quad \text{for} \quad \beta = 1, \ldots, d. \tag{8}$$

$$\partial_b L_P = -\sum_i \alpha_i y_i = 0 \tag{9}$$

$$\alpha_i \geq 0 \tag{10}$$

$$y_i \left(\vec{x} \circ \mathbf{w} + b\right) - 1 \geq 0 \tag{11}$$

$$\alpha_i \left(y_i(\mathbf{w} \circ \vec{x}_i + b) - 1\right) = 0. \tag{12}$$

Note that the condition eqn. (12) ensures that either the optimal $\alpha_i = 0$ or the corresponding constraint eqn. (12) is tight. This is a property referred to as complementary slackness, and indicates that only the vectors for which the constraint is tight give rise to non-zero $\alpha_i > 0$. These vectors are referred to as the support vectors and we will write N_S for their index set. The classifier in the dual picture is given by substituting $\mathbf{w}$ from eqn. (7) and b into the classifier eqn. (3). The bias b is obtained for any $i \in N_S$ from the equality in eqn. (11).

The method can be generalized to the case when the decision function does depend non-linearly on the data by using a trick from [20] and introducing a high-dimensional, non-linear feature map. The data is mapped via

$$\Phi : \mathbb{R}^d \to \mathcal{H} \tag{13}$$

from a low dimensional space non-linearly in to a high dimensional Hilbert-space $\mathcal{H}$. This space is commonly referred to as the feature space. If a suitable feature map has been chosen, it is then possible to apply the SVM classifier for the mapped data in $\mathcal{H}$, rather than in $\mathbb{R}^d$.

It is important to note that it is in fact not necessary to construct the mapped data $\Phi(\vec{x}_i)$ in $\mathcal{H}$ explicitly. Both the training data, as well as the new data to be classified enters only through inner products, in both the optimization problem for training, c.f. eqn. (6), as well as in the classifier, eqn. (3). Hence, we can construct the SVM for arbitrarily high dimensional feature maps $(\Phi, \mathcal{H})$, if we can efficiently evaluate the inner products $\Phi(\vec{x}_i) \circ \Phi(\vec{x}_j)$ and $\Phi(\vec{x}_i) \circ \Phi(\vec{s})$, for $\vec{x}_i \in T$ and $\vec{s} \in S$. In particular, if we can find a kernel $K(\vec{x}, \vec{y}) = \Phi(\vec{x}) \circ \Phi(\vec{y})$ that satisfies Mercer's condition (which ensures that the kernel is positive semi-definite and can be interpreted as matrix of inner products) [14, 20], we can construct a classifier by setting

$$\tilde{m}(\vec{s}) = \text{sign}\left(\sum_{i \in N_S} \alpha_i y_i K(\vec{x}_i, \vec{s}) + b \right). \tag{14}$$

Here we only need to sum over all support vectors $i \in N_S$ for which $\alpha_i > 0$. Moreover, one can replace the inner product in the optimization problem eqn. (6) by the kernel. Examples of such kernels that are frequently considered in the classical literature are for instance the polynomial kernel $K(\vec{x}, \vec{y}) = (\vec{x} \circ \vec{y} + 1)^d$ or even the infinite dimensional Gaussian kernel $K(\vec{x}, \vec{y}) = \exp(-1/2\|\vec{x} - \vec{y}\|^2)$. If the feature map is sufficiently powerful, increasingly complex distributions can be classified. In this paper, the feature map is a classical to quantum mapping by a tunable quantum circuit family, that maps $\Phi : \mathbb{R}^d \to \mathcal{S}(\mathcal{H}_2^{\otimes n})$ in to the state space, or space of density matrices, of n qubits with $\dim\left(\mathcal{S}(\mathcal{H}_2^{\otimes n})\right) = 4^n$. The example of the Gaussian kernel indicates, that the sheer dimension of the Hilbert space on a quantum computer by itself does not provide an advantage, since classically even infinite dimensional spaces are available by for instance using the Gaussian kernel. However, this hints towards a potential source of quantum advantage as we may construct states in feature space with hard-to-estimate overlaps.

Variational circuit classifiers:

Let us now turn to the case of binary classification based on variational quantum circuits. Recall that in our setting, we first take the data $\vec{x} \in \mathbb{R}^d$ and map it to a quantum state $|\Phi(\vec{x})\rangle\langle\Phi(\vec{x})| \in \mathcal{S}(\mathcal{H}_2^{\otimes n})$ on n-qubits, c.f. eqn. (22). Then we apply a variational circuit $W(\vec{\theta})$ to the initial state that depends on some variational parameters $\vec{\theta}$, c.f. eqn. (31). Lastly, for a binary classification task, we measure the resulting state in the canonical Z-basis and assign the resulting bit-string $z \in \{0,1\}^n$ to a label based on a predetermined boolean function $f : \{0,1\}^n \to \{+1, -1\}$. Hence the probability of measuring either label $y \in \{+1, -1\}$ is given by:

$$p_y = \frac{1}{2}\left(1 + y\langle\Phi(\vec{x})|W^\dagger(\theta)\,\mathbf{f}\,W(\theta)|\Phi(\vec{x})\rangle\right), \tag{15}$$

where we have defined the diagonal operator

$$\mathbf{f} = \sum_{z \in \{0,1\}^n} f(z)|z\rangle\langle z|. \tag{16}$$

In classification tasks we assign c.f. eqn. (34), the label with the highest empirical weight of the distribution p_y. We ask whether the outcome $+1$ is more likely than -1, or vice versa. That is, we ask, whether $p_{+1} > p_{-1} - b$ or whether the converse is true. This of course depends on the sign of the expectation value $\langle\Phi(\vec{x})|W^\dagger(\theta, \varphi)\,\mathbf{f}\,W(\theta, \varphi)|\Phi(\vec{x})\rangle$ for

the data point $\vec{x}$.

To understand how this relates to the SVM in greater detail, we need to choose an orthonormal operator basis, such as for example the Pauli group

$$\mathcal{P}_n = \langle X_i, Y_i, Z_i \rangle_{i=1,\ldots,n}. \tag{17}$$

Note that when fixing the phase to $+1$ every element $P_\alpha \in \mathcal{P}_n$, with $\alpha = 1, \ldots, 4^n$ of the Pauli-group is an orthogonal reflection $P_\alpha^2 = \mathbb{1}$. Furthermore, Pauli matrices are mutually orthogonal in terms of the trace inner product

$$\mathrm{tr}\left[P_\alpha P_\beta\right] = \delta_{\alpha,\beta} 2^n. \tag{18}$$

This means that both the measurement operator $W^\dagger(\theta)\,\mathbf{f}\,W(\theta)$ in the W-rotated frame as well as the state $|\Phi(\vec{x})\rangle\langle\Phi(\vec{x})|$ can be expanded in terms of the operator basis with only real coefficients as

$$W^\dagger(\theta, \varphi)\,\mathbf{f}\,W(\theta, \varphi) = \frac{1}{2^n}\sum_\alpha w_\alpha(\theta, \varphi)P_\alpha \quad \text{with} \quad w_\alpha(\theta, \varphi) = \mathrm{tr}\left[W^\dagger(\theta, \varphi)\,\mathbf{f}\,W(\theta, \varphi)P_\alpha\right]$$

$$|\Phi(\vec{x})\rangle\langle\Phi(\vec{x})| = \frac{1}{2^n}\sum_\alpha \Phi_\alpha(\vec{x})P_\alpha \quad \text{with} \quad \Phi_\alpha(\vec{x}) = \mathrm{tr}\left[|\Phi(\vec{x})\rangle\langle\Phi(\vec{x})|P_\alpha\right]. \tag{19}$$

Note, that the values $w_\alpha(\theta, \varphi)$ as well as $\Phi_\alpha(\vec{x})$ are constrained due to the fact that they originate from a rotated projector and from a pure state. Since $\mathbf{f}^2 = \mathbb{1}$, we have that
$\mathrm{tr}\left[\left(W^\dagger(\theta, \varphi)\,\mathbf{f}\,W(\theta, \varphi)\right)^2\right] = 2^n$. Furthermore, the projector squares to itself so that
$\mathrm{tr}\left[|\Phi(\vec{x})\rangle\langle\Phi(\vec{x})|^2\right] = 1$. In particular, this means that the norms of both vectors satisfy
$\sum_\alpha \Phi_\alpha^2(\vec{x}) = 2^n$ as well as $\sum_\alpha w_\alpha^2(\theta, \varphi) = 4^n$. Since the expectation value of the measured observable is
$\langle\Phi(\vec{x})|W^\dagger(\theta, \varphi)\mathbf{f}W(\theta, \varphi)|\Phi(\vec{x})\rangle = \mathrm{tr}\left[|\Phi(\vec{x})\rangle\langle\Phi(\vec{x})|W^\dagger(\theta, \varphi)\,\mathbf{f}\,W(\theta, \varphi)\right]$, it can be expressed in terms of the inner product:

$$\langle\Phi(\vec{x})|W^\dagger(\theta)\,\mathbf{f}\,W(\theta)|\Phi(\vec{x})\rangle = \frac{1}{2^n}\sum_\alpha w_\alpha(\theta)\Phi_\alpha(\vec{x}). \tag{20}$$

Observe that $\mathbf{f}$ only has eigenvalues $+1, -1$, and we have that $\langle\Phi(\vec{x})|W^\dagger(\theta)\,\mathbf{f}\,W(\theta)|\Phi(\vec{x})\rangle \in [-1, +1]$. Let us now consider a decision rule, where we assign the label $y \in \{+1, -1\}$ over the label $-y$ with some fixed bias $b \in [-1, +1]$. In that case we demand that $p_y > p_{-y} - yb$. If we substitute eqn. (15) and use the expansion in eqn. (20) we have that the corresponding label is given by the decision function $y = \tilde{m}(\vec{x})$, where

$$\tilde{m}(\vec{x}) = \mathrm{sign}\left(\frac{1}{2^n}\sum_\alpha w_\alpha(\theta)\Phi_\alpha(\vec{x}) + b\right). \tag{21}$$

This expression is identical to the conventional SVM classifier, c.f. eqn. (3), after the feature map has been applied. However, in the experiment we only have access to the probabilities p_y through estimation. Furthermore, the $w_\alpha(\theta)$ are constrained to stem from the observable $\mathbf{f}$ measured in the rotated frame.

This means, that the correct feature space, where a linearly separating hyperplane is constructed is in fact the quantum state space of density matrices, and not the Hilbert space $\mathcal{H}_2^n$ itself. This is reasonable, since the physical states in $\mathcal{H}_2^n$ are only defined up to a global phase $|\psi\rangle \sim e^{i\eta}|\psi\rangle$. The equivalence of states up to a global phase would make it impossible to find a separating hyperplane, since both $|\psi\rangle$ and $-|\psi\rangle$ give rise to the same physical state but can lie on either side of a separating plane.

Encoding of the data using a suitable feature map

In the quantum setting, the feature map is an injective encoding of classical information $\vec{x} \in \mathbb{R}^d$ into a quantum state $|\Phi\rangle\langle\Phi|$ on an n-qubit register. Here $\mathcal{H}_2 = \mathbb{C}^2$ is a single qubit Hilbert space, and $\mathcal{S}\left(\mathcal{H}_2^{\otimes n}\right)$ denotes the cone of positive semidefinite density matrices $\rho \geq 0$ with unit trace $\mathrm{tr}\left[\rho\right] = 1$. This cone is a subset of the 4^n dimensional Hilbert space of $\mathcal{M}_{2^n \times 2^n}(\mathbb{C})$ of complex matrices when fitted with the inner product $\mathrm{tr}\left[A^\dagger B\right]$ for $A, B \in \mathcal{M}_{2^n \times 2^n}(\mathbb{C})$. The feature map acts as

$$\Phi : \Omega \subset \mathbb{R}^d \to \mathcal{S}\left(\mathcal{H}_2^{\otimes n}\right), \qquad\qquad \Phi : \vec{x} \mapsto |\Phi(\vec{x})\rangle\langle\Phi(\vec{x})|. \qquad (22)$$

The action of the map can be understood by a unitary circuit family denoted by $\mathcal{U}_{\Phi(\vec{x})}$ that is applied to some reference state, e.g. $|0\rangle^n$. The resulting state is given by $|\Phi(\vec{x})\rangle = \mathcal{U}_{\Phi(\vec{x})}|0\rangle^n$. The state in the feature space should depend non-linearly on the data. Let us discuss proposals for possible feature maps

Product state feature maps   There are many choices for the feature map Φ. Let us first discuss what would happen if we were to choose a feature map that corresponds to a product input state. We assume a feature map, comprised of single qubit rotations $U(\varphi) \in \mathrm{SU}(2)$ on every qubit on the quantum circuit. The angles for every qubit can be chosen as a non-linear function $\varphi : \vec{x} \to (0, 2\pi]^2 \times [0, \pi]$ into the space of Euler angles for the individual qubits, so that the full feature map can be implemented as:

$$\vec{x} \mapsto |\phi_i(\vec{x})\rangle = U(\varphi_i(\vec{x}))|0\rangle, \quad \text{for an individual qubit, so that} \qquad (23)$$

$$\Phi : \vec{x} \mapsto |\Phi(\vec{x})\rangle\langle\Phi(\vec{x})| = \bigotimes_{i=1}^{n} |\phi_i(x)\rangle\langle\phi_i(x)| \quad \text{for the full qubit state.} \qquad (24)$$

One example for such an implementation is the unitary implementation of the feature map used in the context of the classical classifiers by Stoudenmire and Schwab [31] based on tensor networks. There each qubit encodes a single component x_i of $\vec{x} \in [0,1]^n$ so that n qubits are used. The resulting state that that is prepared is then

$$\bigotimes_{i=1}^{n} |\phi_i(x)\rangle\langle\phi_i(x)| = \frac{1}{2^n} \bigotimes_{j=1}^{n} \left(\sum_{\alpha_j} \Phi_j^{\alpha_j}\left(\theta_j(\vec{x})\right) P_{\alpha_j} \right), \qquad (25)$$

when expanded in terms of the Pauli-matrix basis where $\Phi_i^{\alpha}(\theta_i(\vec{x})) = \langle\phi_i(x)|P_{\alpha_i}|\phi_i(x)\rangle$ for all $i = 1, \ldots n$. and $P_{\alpha_i} \in \{\mathbb{1}, X_i, Z_i, Y_i\}$. The corresponding decision function can be constructed as in eqn. (14), where the kernel $K(\vec{x}, \vec{y}) = \prod_{i=1}^{n} |\langle\phi_i(\vec{x})|\phi_i(\vec{y})\rangle|^2$ is replaced by the inner product between the resulting product states. These can be evaluated with resources scaling linearly in the number of qubits, so that no quantum advantage can be expected in this setting.

Non-trivial feature map with entanglement

There are many choices of feature maps, that do not suffer from the malaise of the aforementioned product state feature maps. To obtain an quantum advantage we would like these maps to give rise to a kernel $K(\vec{x}, \vec{y}) = |\langle\Phi(\vec{x})|\Phi(\vec{y})\rangle|^2$ that is computationally hard to estimate up to an additive polynomially small error by classical means. Otherwise the map is immediately amenable to classical analysis and we are guaranteed to have lost any conceivable quantum advantage.

Let us therefore turn to a family of feature maps, c.f. Fig S2 for which we conjecture that it is hard to estimate the overlap $|\langle\Phi(\vec{x})|\Phi(\vec{y})\rangle|^2$ on a classical computer. We define the family of feature map circuit as follows

$$|\Phi(\vec{x})\rangle = U_{\Phi(\vec{x})} H^{\otimes n} U_{\Phi(\vec{x})} H^{\otimes n}|0\rangle^{\otimes n} \quad \text{where,} \quad U_{\Phi(\vec{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{i \in S} Z_i \right). \qquad (26)$$

where now the 2^n possible coefficients $\phi_S(\vec{x}) \in \mathbb{R}$ are non-linear functions of the input data $\vec{x} \in \mathbb{R}^n$. It is convenient to use maps with low-degree expansions, i.e. $|S| \le d$. Any such map can be efficiently implemented. In the experiments reported in this paper we have restricted to $d = 2$. So we only consider Ising type interactions in the unitaries $U_{\Phi(\vec{x})}$. In particular we choose these interactions as the ones that are present in the actual connectivity graph of the superconducting chip $G = (E, V)$. This ensures that the feature map can be generated from a short depth circuit. The resulting unitary can then be generated from one- and two- qubit gates of the form

$$U_{\phi_{\{l,m\}}(\vec{x})} = \exp\left(i\phi_{\{k,l\}}(\vec{x})Z_k Z_l\right) \quad \text{and} \quad U_{\phi_{\{k\}}(\vec{x})} = \exp\left(i\phi_{\{k\}}(\vec{x})Z_k\right), \qquad (27)$$

which leaves $|V| + |E|$, real parameters to encode the data. In particular, we know that we have at least $|V| = n$ real numbers to encode the data. Furthermore, depending on the connectivity of the interactions, we have $|E| \le n(n-1)/2$ further parameters that can be used to encode more data or nonlinear relations of the initial data points.
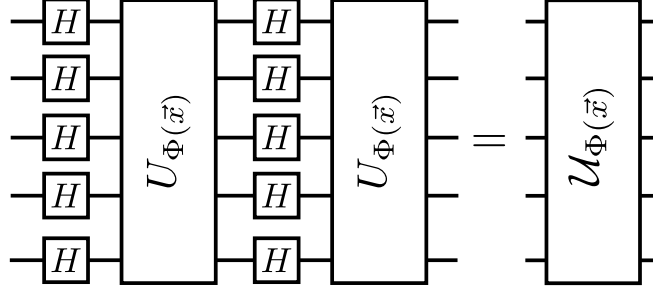
FIG. S2. A circuit representation of the feature map family we consider here. We first apply a series of Hadamard gates before applying the diagonal phase gate component. Then we apply a second layer of Hadamard gates, followed by the same diagonal phase gate. This encodes both the actual function value of the phase $\Phi_{\vec{x}}(z)$ as well as the value of the $\mathbb{Z}_2^N$ Fourier transform $\hat{\Phi}_{\vec{x}}(S)$ for every basis element.

This feature map encodes both the actual function $\Phi_{\vec{x}}(z)$ of the diagonal phases, as well as the corresponding Fourier-Walsh transform $\hat{\Phi}_{\vec{x}}(p)$ at $z, p \in \{0,1\}^n$

$$\Phi_{\vec{x}}(z) = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{i \in S} (-1)^{z_i}\right) \quad \text{and} \quad \hat{\Phi}_{\vec{x}}(p) = \frac{1}{2^n} \sum_{z \in \{0,1\}^n} \Phi_{\vec{x}}(z)(-1)^{p \circ z}, \tag{28}$$

for every basis element respectively. The resulting state, after the datum $\vec{x} \in \mathbb{R}^d$ has been mapped to the feature space, is given by

$$|\Phi(\vec{x})\rangle = \sum_{p \in \{0,1\}^N} \Phi_{\vec{x}}(p)\hat{\Phi}_{\vec{x}}(p)|p\rangle. \tag{29}$$

We conjecture that it is hard to estimate this kernel up to an additive polynomially small error by classical means. The intuition for the conjecture stems from a connection of the feature map to a particular circuit family for the hidden shift problem for boolean functions [32]. The feature map is similar to a quantum algorithm for estimating the hidden shift in boolean bent functions, c.f. Ref. [33], algorithm $\mathcal{A}_1$ in Theorem 6. The circuit $\mathcal{U}_\Phi$ we consider is indeed very similar to the one discussed in [33]. Let us make a minor modification, and ask the two diagonal layers in $\tilde{\mathcal{U}}_\Phi = U_{\Phi^2} H^{\otimes n} U_{\Phi^1} H^{\otimes n}$ to differ. We choose the phase gate layers so that U_{Φ^1} and U_{Φ^2} encode a shifted bent-function and the function's dual, then the circuit $\mathcal{A}_1$ is given by $H^{\otimes n} \tilde{\mathcal{U}}_\Phi$. If the interaction graph G is bi-partite, it is possible to encode Maiorana-McFarland bent-functions for $d = 2$ by choosing the corresponding $\phi_{\{k,l\}}(\vec{x}), \phi_k(\vec{x})$ to be either π or 0. In [33], the diagonal layer in $\mathcal{A}_1$, are queries to an oracle encoding the the shifted bent-function and it's dual. It can be shown that with respect to this oracle there is an exponential separation in query complexity over classical query algorithms. The final circuit we implement to estimate the overlap $|\langle \Phi(\vec{x})|\Phi(\vec{y})\rangle|^2$ is still larger. This circuit has four layers of Hadamard gates and 3 diagonal layers. If the conjecture could be proven, it would establish a valuable step in rigorously establishing a quantum advantage on near-term devices. If, however, it ultimately turns out that this family can also be evaluated classically, we would need to improve the complexity of this circuit family.

A good example of a circuit family that entails a hardness result, yet does not provide an advantage in our setting is the following: One could consider the case, where only a single layer of Hadamard gates and a single diagonal unitary is applied. Such a feature map is directly related to the circuit family introduced in [34]. Indeed the resulting kernel $K(\vec{x}, \vec{y}) = |\langle 0^n|H^{\otimes n} U_{\Phi(\vec{y})} U_{\Phi(\vec{x})} H^{\otimes n}|0^n\rangle|^2$ corresponds to the output probability of the considered IQP circuits. In fact it is known that if general real values $\phi_S(\vec{x})$ are allowed that $d = 2$ suffices to encode #P-hard problems in the output probability [21]. This hardness result, only applies to the case where the output probability can be approximated up to a multiplicative error. A noisy quantum computer, however, is also not able to provide a multiplicative error estimate to this quantity. Nevertheless, Bremner et al. show, contingent on additional complexity theoretic conjectures, that it is hard for a classical sampler to produce samples from the output distribution of the IQP circuit. Does this result imply some form of hardness for the feature map version of this circuit? The answer is unfortunately no. The kernel can be estimated up to an additive error of ϵ by drawing $R = \epsilon^{-2}$ samples from the uniform distribution over n

classical bits and averaging $\tilde{\Phi}_{\vec{x}-\vec{y}}(z) = \Phi_{\vec{x}}(z)\overline{\Phi}_{\vec{y}}(z)$, c.f. eqn. (28)

$$|\langle 0^n | H^{\otimes n} U_{\Phi(\vec{y})} U_{\Phi(\vec{x})} H^{\otimes n} | 0^n \rangle|^2 = \left| \frac{1}{2^n} \sum_{z \in \{0,1\}^n} \tilde{\Phi}_{\vec{x}-\vec{y}}(z) \right|^2 . \tag{30}$$

Since we have that the variance of the random variable is bounded by 1 since $|\Phi_{\vec{x}}(z)|^2 = 1$, we get an additive error that scales as $\mathcal{O}(\epsilon)$. This means for a single layer, the kernel can be estimated classically.

Quantum variational classification

Following the structure of the feature map circuit, we construct the classifier part of the variational algorithm by appending layers of single-qubit unitaries and entangling gates Fig. S3.a. Each subsequent layer, or depth, contains an additional set of entanglers across all the qubits used for the algorithm. We use a coherently controllable quantum mechanical system, such as for example the superconducting chip with n transmon qubits to prepare a short depth quantum circuit $W(\vec{\theta})$. In the experiment here, comprising $n = 2$ qubits, one controlled-phase gate is added per depth. The single-qubit unitaries used in the classifier are limited to Y and Z rotations to simplify the number of parameters to be handled by the classical optimizer. Our use of controlled-phase, rather than CNOT, gates for the entanglers is justified by our aim at increased generality in our software. Using controlled-phase gates does not require to particularize this part of the algorithm for different systems topologies. A specific entangling map for a given device can then be used by our compiler to translate each controlled-phase gate into the CNOTs available in our system.

The general circuit is comprised of the following sequence of single qubit and multi-qubit gates:

$$W(\vec{\theta}) = U_{\text{loc}}^{(l)}(\theta_l) \, U_{\text{ent}} \dots U_{\text{loc}}^{(2)}(\theta_2) \, U_{\text{ent}} \, U_{\text{loc}}^{(1)}(\theta_1). \tag{31}$$

We apply a circuit of l repeated entanglers as depicted in Fig S3.b and interleave them with layers comprised of local single qubit rotations:

$$U_{\text{loc}}^{(t)}(\theta_t) = \otimes_{m=1}^n U(\theta_{m,t}) \quad \text{and} \quad U(\theta_{m,t}) = e^{i\frac{1}{2}\theta_{m,t}^z Z_m} e^{i\frac{1}{2}\theta_{m,t}^y Y_m}, \tag{32}$$

parametrized by $\theta_t \in \mathbb{R}^{2n}$ and $\theta_{i,t} \in \mathbb{R}^2$. In principle, there exist multiple choices for the entangling unitaries U_{ent} [16, 17]. For the feature map that we consider, however, we use the entangler that is comprised of products of control phase gates $\text{CZ}(i,j)$ between qubits i and j. The entangling interactions follow the interaction graph that $G = (E, V)$ that was used to generate the feature map in eqn. (27).

$$U_{\text{ent}} = \prod_{(i,j) \in E} \text{CZ}(i,j), \tag{33}$$

This short-depth circuit can generate any unitary if sufficiently many layers d are applied. The circuit Fig. S3.a
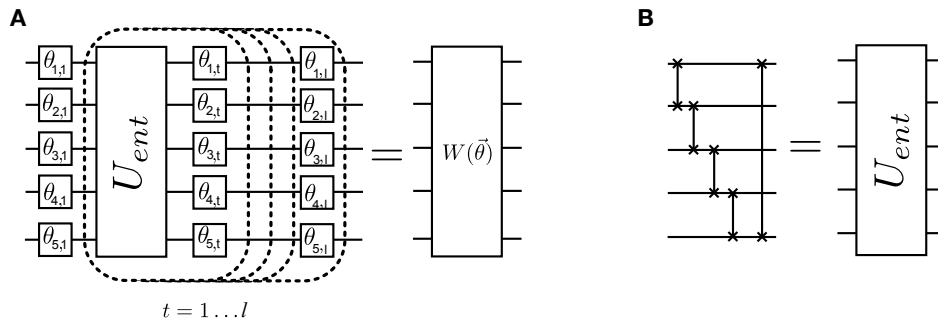

FIG. S3. (a) Circuit representation of short depth quantum circuit to define the separating hyperplane. The single qubit rotations $U(\theta_{i,t}) \in \text{SU}(2)$ are depicted by single line boxes parametrized by the angles θ_i, while the entangling operation U_{ent} is determined by the interaction graph of the superconducting chip. (b) Depiction of entangling gate as a product of $\text{CZ}_{i,i+1}$ for $i = 1, \dots, 5$ gates following the interaction graph of a circle $G = C_5$.

can be understood as a bang-bang controlled evolution of an Ising model Hamiltonian $H_0 = \sum_{(ij) \in E} J_{ij} Z_i Z_j$, c.f.

Fig. S3.b, interspersed with single qubit control pulses in $SU(2)$ on every qubit. It is known that this set of drift steps together with all single control pulses are universal, so we have that any state can be prepared this way with sufficient circuit depth [35]. For a general unitary gate sequence the entangling unitary has to be effectively generated from cross resonance gates, by applying single qubit control pulses.

Choosing the cost-function for the circuit optimization

The central goal is to find an optimal classifying circuit $W(\vec{\theta})$, c..f. eqn. (31) that separates the data sets with different labels. Since we can re-run the same classifying multiple times (R shots), we may consider a 'winner takes all' scenario, where we assign the label according to the outcome with the largest probability. We choose a cost function, so that the optimization procedure minimizes the probability of assigning the wrong label after having constructed the distribution after R shots.

There are multiple ways of performing a multi-label classification. We only need to modify the final measurement M, to correspond to multiple partitions. This can be achieved by multiple strategies. For example one could choose to measure again in the computational basis, i.e. the basis in which Pauli Z are diagonal and then constructing classical labels form the measured samples, such as a labeling the outcome $z \in \{0,1\}^n$ according to a function $f : \{0,1\}^n \to \{1,\ldots,c\}$. The resulting $\{M_y\}_{y=1,\ldots,c}$ is therefore diagonal in the computational basis. Alternatively one could construct a commuting measurement akin to the syndrome check measurement for quantum stabilizers. For this approach we choose a set $\{g_i\}_{i=1\ldots\lceil\log_2(c)\rceil}$ of Pauli matrices $g_i \in \mathcal{P}_N$ that are commuting $[g_i, g_j] = 0$. The resulting measurement that would need to be performed is similar to that of an error correcting scheme. The measurement operators are given by $M_y = 2^{-1}\left(1 - \prod_{i=1}^{\lceil\log_2(c)\rceil} g_i^{y^i}\right)$. Here y^i denotes the i'th bit in the binary representation of y. In either case, the decision rule that assigns the labels can be written as

$$\tilde{m}_{|T}(\vec{x}) = \arg\max_{c'}\langle\Phi(\vec{x})\,|W(\vec{\theta})^\dagger M_{c'} W(\vec{\theta})|\,\Phi(\vec{x})\rangle. \tag{34}$$

This corresponds to taking R shots in order to estimate the largest outcome probability from the outcome statistics of the measurement M_y for $y = 1, \ldots, c$. Labelling T_c the subset of samples T labelled with c, the overall expected misclassification rate is given by:

$$P_{\text{err}} = \frac{1}{|T|}\left(\sum_c \sum_{s \in T_c} \Pr\left(\tilde{m}_{|T}(s) \neq c | s \in T_c\right)\right). \tag{35}$$

The error probability of misclassifying an individual datum is given by $\Pr\left(\tilde{m}_{|T}(s) \neq c | s \in T_c\right)$. This error probability is now used to define the empirical risk function $R_{\text{emp}}(\vec{\theta}) = P_{\text{err}}$. We now discuss of how to find suitable ways of evaluating the cost function for this classification scheme.

Binary label classification

Assume the programmer classifies into labels $y \in \{-1, 1\}$ by taking R shots for a single datapoint. She obtains an empirical estimates of probability of the datum being labeled by a label y

$$\hat{p}_y = \frac{r_y}{R}.$$

After $R = r_y + r_{-y}$ shots and a prior bias b, she misclassifies into a label y if

$$\hat{p}_y < \hat{p}_{-y} + yb \to r_y < r_{-y} + ybR \to r_y < \left\lceil\frac{1+yb}{2}R\right\rceil$$

The probability of her misclassifying a y sample according to the argmax rule is hence estimated by

$$\Pr\left(\tilde{m}_{|T}(s) \neq y | s \in T_y\right) = \Pr\left(r_y < r_{-y} + yb\right) = \sum_{j=0}^{\lceil\frac{1+yb}{2}R\rceil}\binom{R}{j}p_y^j(1-p_y)^{R-j}.$$

Assuming large R, computing this exactly may be difficult. Setting $Rp_y = a$, $Rp_y(1 - p_y) = \beta^2$ and $\lceil \left(\frac{1+yb}{2} \right) R \rceil = \gamma$, we can approximate the binomial CDF as an error function:

$$\Pr\left(\tilde{m}_{|T}(s) \neq y | s \in T_y \right) = \sum_{j=0}^{\lceil \left(\frac{1+yb}{2} \right) R \rceil} \binom{R}{j} p_c^{R-j} (1 - p_c)^j \approx \int_{-\infty}^{\gamma} dx \frac{1}{\sqrt{2\pi}\beta} \exp\left(-\frac{1}{2} \left(\frac{x-a}{\beta} \right)^2 \right)$$

$$= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\frac{\gamma-a}{\sqrt{2}\beta}} dz\, e^{-z^2} = \frac{1}{2} \mathrm{erf}\left(\frac{\gamma-a}{\sqrt{2}\beta} \right) + \frac{1}{2}$$

$$= \frac{1}{2} \mathrm{erf}\left(\sqrt{R} \frac{\frac{1+yb}{2} - p_y}{\sqrt{2(1-p_y)p_y}} \right) + \frac{1}{2}\,.$$

See Fig. S4. The error function can be consequently approximated with a sigmoid
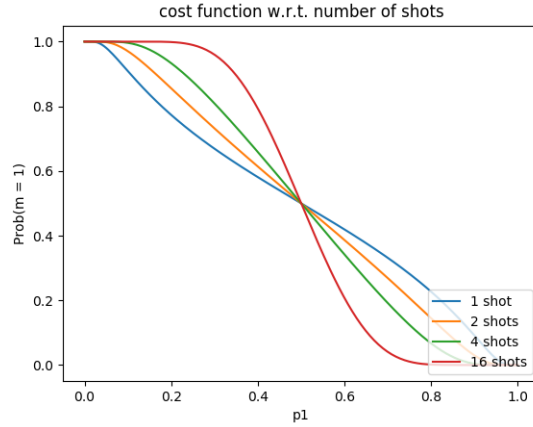

FIG. S4. Single shot to multi-shot decision rule for classification. The contribution to the cost-function interpolates from linear to logistic-normal CDF (approximately sigmoid). In the experiment the data was sampled with $R = \mathcal{O}(10^3)$, although the cost-function was evaluated with only $\tilde{R} = \mathcal{O}(10^2)$ to provide a smoother function to the optimization routine.

$$\mathrm{sig}(x) = \frac{1}{1 + \exp(-x)} \approx \frac{1}{2} \left(\mathrm{erf}(x) + 1 \right),$$

which gives

$$\Pr\left(\tilde{m}_{|T}(s) \neq y | s \in T_y \right) \approx \mathrm{sig}\left(\sqrt{R} \frac{\frac{1+yb}{2} - p_y}{\sqrt{2(1-p_y)p_y}} \right).$$

as an estimate for misclassifying a sample s. The cost function to optimize is then given by using this in eqn. (35).

Multi label classification

For multiple labels, one tries to optimize

$$P_{err} = \frac{1}{|T|} \sum_c \sum_{s \in T_c} \Pr\left(\tilde{m}_{|T}(s) \neq c | s \in T_c \right),$$

where:

$$\Pr\left(\tilde{m}_{|T}(s) \neq c | s \in T_c \right) = \Pr\left(n_c < \max_{c'}\left(\{n_{c'}\}_{c'/c} \right) \right).$$

We consider the case of three labels. For R samples with frequencies $\{n_0, n_1, n_2\}$, drawn independently from the output probability distribution, the probability of misclassifying a sample $s \in T_0$ by argmax is given by

$$\Pr\left(\tilde{m}_{|T}(s) \neq 0 | s \in T_0\right) = \Pr\left(n_0 < \max(n_1, n_2)\right) = \Pr\left(n_0 < \left\lceil \frac{N + |n_1 - n_2|}{3} \right\rceil\right),$$

where the last inequality is derived as follows

$$2n_0 < 2\max(n_1, n_2) = |n_1 - n_2| + n_1 + n_2 = |n_1 - n_2| + N - n_0.$$

Hence setting $\gamma = \frac{N + |n_1 - n_2|}{3}$, it follows that

$$\Pr\left(\tilde{m}_{|T}(s) \neq 0 | s \in T_0\right) = \sum_{k=0}^{k=\gamma} \binom{R}{k} p_0^k (1 - p_0)^{N-k} \approx \mathrm{sig}\left(\frac{\gamma - N p_0}{\sqrt{2N(1 - p_0)p_0}}\right).$$

This however still depends on n_1, n_2, which can't be simply eliminated. Additionally, for a general k-label case, there is no simple analytic solution for γ. For this reason, we therefore try to estimate the above probability by simply taking $\gamma = \max_{c'}\left(\{n_{c'}\}_{c'/c}\right)$. So for k-label case, the cost function terms are given by

$$\Pr\left(\tilde{m}_{|T}(s) \neq c | s \in T_c\right) \approx \mathrm{sig}\left(\sqrt{R}\frac{\max_{c'}\left(\{n_{c'}\}_{c'/c}\right) - n_c}{\sqrt{2(N - n_c)n_c}}\right).$$

Quantum kernel estimation

For the second classification method we only use the quantum computer to estimate the kernel $K(\vec{x}_i, \vec{x}_j) = |\langle \Phi(\vec{x}_i) | \Phi(\vec{x}_j) \rangle|^2$ for all the labeled training data $\vec{x}_j \in T$. Then we use the classical optimization problem as outlined again in eqn. (6) to obtain the optimal Lagrange multipliers α_i and support vectors N_S can be obtained. From this the classifier can be constructed, c.f .eqn. (14). To apply the classifier to a new datum $\vec{s} \in S$ the kernel $K(\vec{x}_i, \vec{s})$ between $\vec{s}$ and the support vectors in $i \in N_S$ has to be estimated. We discuss two methods to estimate this overlap for our setting.
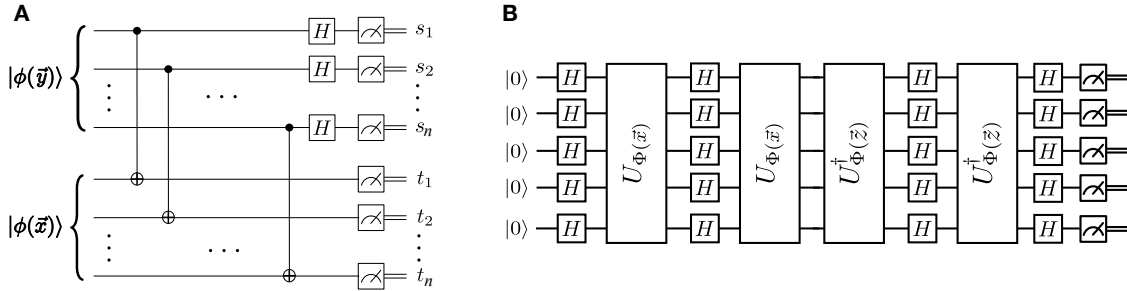

FIG. S5. a) Estimating the expectation value of the SWAP- matrix as derived in [27]. This circuit diagonalizes the SWAP - gate, when it is treated as a hermitian observables with eigenvalues ± 1. Averaging the eigenvalues, c.f. eqn. (36), with samples from the output distribution constructs an estimator for $|\langle \Phi(\vec{x}) | \Phi(\vec{y}) \rangle|^2$. b) The ciruit estimates the fidelity between two states in feature space directly by first applying the unitary $\mathcal{U}_{\Phi(\vec{x})}$ followed by the inverse $\mathcal{U}_{\Phi(\vec{y})}^{\dagger}$ and measuring all bits at the output of the circuit. The frequency $\nu_{0,\ldots,0} = |\langle \Phi(\vec{x}) | \Phi(\vec{y}) \rangle|^2$ of the all the zero outcome precisely corresponds to the desired state overlap.

The usual method of estimating the fidelity between two states is by using the swap test [26]. This circuit, however, is not a short depth circuit on a quantum computing architecture with geometrically local gates. It would require a sequence of controlled SWAP, also known as Fredkin gates, all conditioned on the state of the same ancilla qubit. A very nice protocol was recently developed in [27]. The authors have learned multiple ways of optimizing the conventional swap test. If only the value of the fidelity is needed, as is the case for our algorithm, the authors propose, c.f. [27] section III.C, a circuit that is constant depth if pairs of CNOT gates can be executed in parallel.

This circuit, c.f. Fig S5.a, evaluates the expectation value $\langle \psi | \langle \phi | \mathsf{SWAP} | \psi \rangle | \phi \rangle$ directly. The action of the algorithm can be understood as follows:

The SWAP gate is both a unitary gate and a hermitian observable $\mathsf{SWAP}^\dagger = \mathsf{SWAP}$ with eigenvalues ± 1. The expectation value on two product states as we said given by
$\langle \psi | \langle \phi | \mathsf{SWAP} | \psi \rangle | \phi \rangle = |\langle \phi | \psi \rangle|^2$. Now the gate can be decomposed in to a product of two qubit swap gates $\mathsf{SWAP} = \prod_{k=1}^{n} \mathsf{SWAP}_{s_k t_k}$ that act all in parallel. To evaluate the expectation value one has to diagonalize the full gate. This is achieved by diagonalizing the individual two-qubit swap gates by observing that $\mathsf{SWAP}_{ij} = \mathsf{CNOT}_{i \to j} \mathsf{CNOT}_{j \to i} \mathsf{CNOT}_{i \to j}$. Furthermore using the circuit identity $\mathsf{CNOT}_{j \to i} = H_j \mathsf{CZ}_{ji} H_j$, we see that SWAP_{ij} is diagonalized by $\mathsf{CNOT}_{j \to i} H_j$ and has eigenvalue $(-1)^{x_i x_j}$. For the full circuit Fig S5.a one first applies a transversal set of CNOT gates across both registers followed by a single layer of Hadamard gates H on the top register. Then the output is sampled and the average of the boolean function

$$f(s,t) = (-1)^{(s_1 t_1 + \ldots s_n t_n)} \tag{36}$$

is reported. The output bits on the top register are labeled by $s \in \{0,1\}^n$, while $t \in \{0,1\}^n$ is the output string on the lower register.

This method works for arbitrary input states $|\psi\rangle, |\phi\rangle$. Our states, however, are structured and are all generated by the unitary eqn. (26) as shown in Fig S2. Writing out the kernel explicitly $K(\vec{y}, \vec{x}) = |\langle \Phi(\vec{y}) | \Phi(\vec{x}) \rangle|^2 = |\langle 0^n | \mathcal{U}_{\Phi(\vec{y})}^\dagger \mathcal{U}_{\Phi(\vec{x})} | 0^n \rangle|^2$ gives the indication of how to measure it. Simply apply the circuit $\mathcal{U}_{\Phi(\vec{y})}^\dagger \mathcal{U}_{\Phi(\vec{x})}$ to the state $|0^n\rangle$, c.f. Fig S5.b. Then sample R times the resulting state $\mathcal{U}_{\Phi(\vec{y})}^\dagger \mathcal{U}_{\Phi(\vec{x})} |0^n\rangle$ in the Z basis. Record the number of all observed zero $(0,\ldots,0)$ bit-strings and divide by the total number of shots R. The frequency $\nu_{(0,\ldots,0)} = \#\{(0,\ldots,0)\} R^{-1}$ then gives an estimator for $K(\vec{y}, \vec{x})$ up to a sampling error $\tilde{\epsilon} = \mathcal{O}(R^{-1/2})$. A rough estimate for the operator norm $\|\cdot\|$ of the deviation of the resulting estimator $\hat{K}$ from the true kernel matrix K is given by $\|K - \hat{K}\| \leq \|K - \hat{K}\|_F$. Here $\|A\|_F = \sqrt{\sum_{ij} |A_{ij}|^2}$ denotes the Frobenius norm of the matrix A. A crude bound can then be obtained from the largest sampling error $\tilde{\epsilon}$ of all matrix entries and setting $\|K - \hat{K}\|_F \leq \tilde{\epsilon}|T|$, since both matrices of the training set T are of size $|T| \times |T|$. Hence to ensure a maximum deviation of ϵ with high probability a number of $R = \mathcal{O}(\epsilon^{-2}|T|^2)$ shots have to be drawn for each matrix entry. Due to the symmetry of the K matrix and the trivial diagonals, $|T|(|T| - 1)2^{-1}$ matrix entries have to be estimated. Thus the full sampling complexity is expected to scale as $\mathcal{O}(\epsilon^{-2}|T|^4)$. A more careful analysis of the statistical error could be carried out by using one of the matrix-concentration results [36].

Note that the optimization problem, eqn. (6) is only concave, when the matrix $K \geq 0$ is positive semi-definite. It can happen, that the shot noise and other errors in the experiment lead to a $\hat{K}$ that is no longer positive semi-definite. We have indeed observed this multiple times in the experiment. A possible way of dealing with this problem is a method developed in [28], where an optimization problem is solved to find the closest positive semi-definite K-matrix in trace norm to $\hat{K}$ consistent with the constraint. In our experiments however, we have found this not to be necessary and the performance has been almost optimal without performing this method.

Device parameters

Our device is fabricated on a 720-μm-thick Si substrate. A single optical lithography step is used to define all CPW structures and the qubit capacitors with Nb. The Josephson junctions are patterned via electron beam lithography and made by double-angle deposition of Al.

The dispersive readout signals are amplified by Josephson Parametric Converters [37] (JPC). Both the quantum processor and the JPC amplifiers are thermally anchored to the mixing chamber plate of a dilution refrigerator.

The two qubit fundamental transition frequencies are $\omega_i/2\pi = \{5.2760(4), 5.2122(3)\}$ GHz, with anharmonicities $\Delta_i/2\pi = \{-330.3, -331.9\}$ MHz, where $i \in \{0,1\}$. The readout resonator frequencies used are $\omega_{Ri}/2\pi = \{6.530553, 6.481651\}$ GHz, while the CPW bus resonator connecting Q_0 and Q_1 was unmeasured and designed to be 7.0 GHz. The dispersive shifts and line widths of the readout resonators are measured to be $2\chi_i/2\pi = \{-1.06, -1.02\}$ MHz and $\kappa_i/2\pi = \{661, 681\}$ kHz, respectively.

The two qubit lifetimes and coherences were measured intermittently throughout our experiments. The observed mean values were $T_{1(i)} = \{55, 38\}$, $T_{2(i)}^* = \{16, 17\}$, $T_{2(i)}^{\text{echo}} = \{43, 46\}$ μs with $i \in \{0,1\}$

Gate characterization

Our experiments use calibrated $X-$rotations (X_π and $X_{\pi/2}$) as single-qubit unitary primitives. $Y-$rotations are attained by appropiate adjustment of the pulse phases, whereas $Z-$rotations are implemented via frame changes in software [38]. A time buffer is added at the end of each physical pulse to mitigate effects from reflections and evanescent waves in the cryogenic control lines and components.

We use two sets of gate times in order to perform the Richardson extrapolation of the noise in our system [10]. For the first set of gate times we use 83 ns for single-qubit gates and 333 ns for each cross-resonance pulse. The buffer after each physical pulse is 6.5 ns. The single-qubit gates are gaussian shaped pulses with $\sigma = 20.75$ ns. The cross-resonance gates are flat with turn-on and -off gaussian shapes of $\sigma = 10$ ns. Our implementation of a CNOT has a duration of two single-qubit pulses and two cross-resonance pulses, giving a total of 858 ns for the first set of gate times, including buffers. For our second set of gate times we use the times of the first set but stretched by a factor of 1.5, including the pulses σs and the buffers. This gives a total CNOT time of 1.287 μs and single-qubit gates of $\sim$ 125 ns.

We experimentally verified our single- and two-qubit unitaries by Randomized Benchmarking (RB) [39, 40]. The following table shows the RB results for all single-qubit gates used in our experiments, including individual and simultaneous RB.

Qubit label	Q_0 (83 ns) $(\times 10^{-3})$	Q_1 (83 ns) $(\times 10^{-3})$	Q_0 (125 ns) $(\times 10^{-3})$	Q_1 (125 ns) $(\times 10^{-3})$
01	-	0.715 ± 0.005	-	1.244 ± 0.010
10	1.319 ± 0.017	-	1.410 ± 0.010	-
11	1.367 ± 0.011	0.763 ± 0.005	1.484 ± 0.014	1.271 ± 0.010

TABLE S1. RB of our single-qubit gates. Qubit labels indicate which qubit was bencharmked on each case, with label 11 indicating simultaneous RB.

Our two-qubit unitarias are CNOTs constructed from echo cross-resonance sequences [40, 41]. Each of the two cross-resonance pulses in a CNOT has durations of 333 and 500 ns for the two different gate lengths used in our experiments. For our two-qubit RB we obtain a CNOT error of $0.0373 \pm .0015$ ($0.0636 \pm .0021$) for the 333 (500) ns cross-resonance pulse.

Readout correction

Our readout assigned fidelity was $\sim 95\%$ for both qubits.

For each experiment, we run 4 (2^2) calibration sequences preparing our two qubits in their joint computational states. We gather statistics of these calibrations and create a measurement matrix $A_{ij} = P(|i\rangle || j\rangle)$ where $P(|n\rangle || m\rangle)$ is the probability of measuring state $|m\rangle$ having prepared state $|n\rangle$. We then correct the observed outcomes of our experiments by inverting this matrix and multiplying our output probability distributions by this inverse.

Support vectors

Here we show the support vectors and α_i as calculated for each of the three datasets studied from their K matrices.

Error mitigation for kernel estimation

The experimental estimation of the kernel matrices shown in Fig. S7 and in Fig. 4 in the main text involves running the experiments at different gate lengths and extrapolating the expectation value of the observable of interest to its zero-noise value. While this technique can be extremely powerful in scenarios where the noise is invariant under time rescaling, it is particularly sensitive to measurement sampling noise. In many cases it is the experimental readout assignment fidelity that determines the bound on how precisely the observable can be estimated.

Set I		
SV	α	y
[5.3407,2.9531]	2.4185	+1
[5.1522,4.6496]	2.6248	+1
[0.6912,0.7540]	3.1379	+1
[6.0947,5.7177]	1.2891	+1
[6.0947,4.1469]	1.3604	+1
[2.6389,4.8381]	1.7163	+1
[4.1469,2.0735]	1.3242	-1
[0.2513,3.2044]	0.9757	-1
[2.3248,3.9584]	1.4953	-1
[4.1469,6.2204]	2.4536	-1
[2.6389,1.6965]	0.5293	-1
[5.4664,4.6496]	4.0213	-1
[4.1469,3.0788]	1.7477	-1
bias = -0.1090		

Set II		
SV	α	y
[0.1885,3.3301]	2.1871	+1
[5.4035,1.7593]	1.443	+1
[5.2150,3.7071]	2.0330	+1
[4.71240,4.6496]	0.7559	+1
[0.4398,2.388]	1.0022	+1
[6.2204,5.7177]	2.3090	+1
[5.9062,4.2726]	1.6297	-1
[1.8221,2.1362]	2.9281	-1
[0.3770,3.8327]	3.0627	-1
[0.31416,1.6965]	1.3793	-1
[1.6336,3.2672]	0.2759	-1
[2.5761,3.2673]	0.4544	-1
bias = 0.2102		

Set III		
SV	α	y
[5.4664,3.7071]	6.2649	+1
[1.5080,5.2779]	0.7777	+1
[5.9690,3.8956]	1.5619	+1
[5.3407,3.5186]	0.6482	+1
[5.7177,4.5239]	1.6387	+1
[0.0628,3.5186]	0.9379	+1
[6.1575,1.445]	0.3882	-1
[5.9690,0.9425]	0.8346	-1
[0.2513,1.3823]	2.5561	-1
[5.2779,2.0735]	4.8076	-1
[6.0947,2.0106]	3.2428	-1
bias = -0.1865		

TABLE S2. Suport vectors for all three data sets used for our kernel estimation method, as shown as green circles in Figs. 3 (b) and S6.
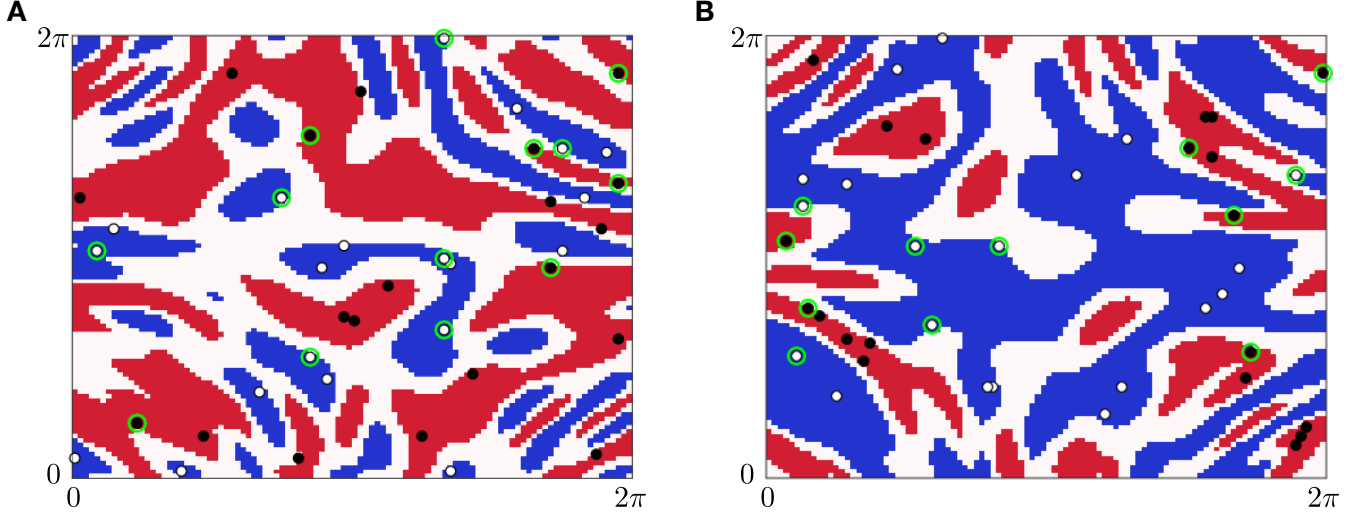

FIG. S6. Sets I (a) and II (b), including training data points (white and black circles) and support vectors (green circles).

Even though for our Sets I and II we attain 100 % classification success over 10 randomly drawn test sets in each case, we can quantify how close our experimentally determined separating hyperplane is to the ideal.

The optimal hyperplane for a given training set can be expressed as the linear combination $\sum_i \alpha_i y_i \vec{x}_i = \mathbf{w}$ (eqn. 7), where $\mathbf{w}$ is a vector orthogonal to the optimal separating hyperplane and $\vec{x}_i$ are the support vectors. We can therefore quantify the distance between the experimentally obtained hyperplane and the ideal hyperplane by calculating the inner product $\langle \mathbf{w}, \mathbf{w}_{\text{ideal}} \rangle = \sum_{i \in N_S} \sum_{j \in N'_S} y_i y_j \alpha_i^* \alpha_j |\langle \vec{x}_i \vec{x}_j \rangle|^2 / \|\mathbf{w}\| \|\mathbf{w}_{\text{ideal}}\|$ where N_S and N'_S are the sets of experimentally obtained and ideal support vectors, respectively.

In Fig. S8 we show the inner products between the ideal and all experimental hyperplanes, including the two sets
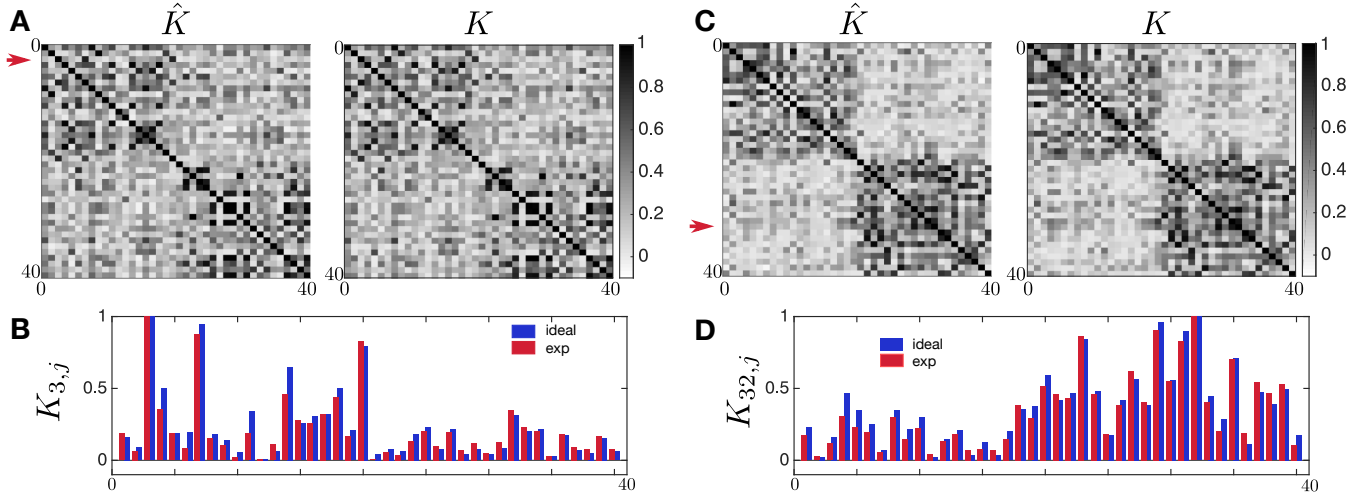
FIG. S7. Experimentally estimated and ideal kernel matrices for Sets I (a, b) and II (c, d). For both datasets we show a cut (b, d) through the row at which the maximum of $|\hat{K} - K|$ occurs, similarly as in Fig. 4 in the main text.

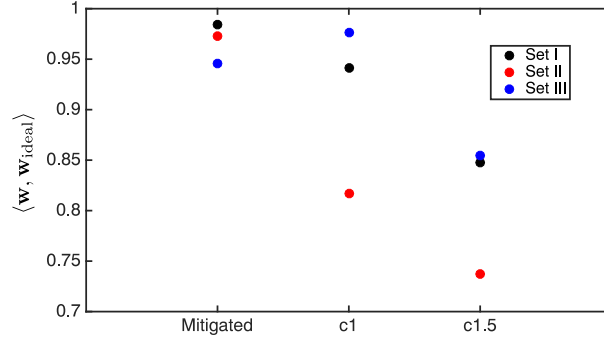of gate times used throughout our experiments, $c1$ and $c1.5$, as well as the error-mitigated hyperplanes.

FIG. S8. Inner products between experimentally obtained hyperplanes and the ideal for each training set. The $x-$axis shows the results for the two different time stretches used in our experiments, $c1$ and $c1.5$, corresponding to the faster and slower gates, respectively. We also show the error-mitigated hyperplanes inner products.

For Sets I and II, which classify at 100 % success, it is clear that error mitigation improves our results very significantly. This is not the case for Set III, which classifies at 94.75 % success. In fact, for Set III we see that error-mitigation worsens the hyperplane, as the results are closer to the ideal for the unmitigated experiments than both Sets I and II.

A look at the calibration data taken along the direct kernel estimation experiments for each set, we see that the readout assignment fidelities of $Q_0(Q_1)$ are 96.56% (96.31%) for Set I, 95.90% (96.36%) for Set II, and 93.99% (95.47%) for Set III. The slightly worse readout fidelities for Set III could partially explain the worse classification results for this set, but other aspects of the protocol might also contribute to this, like for example some gates operating on a somewhat non-linear regime after the calibrations in this set.

Another symptom for the degree of classification success in a given set can be observed by looking at the combined weight of negative eigenvalues in the kernel matrix, $\sum_{k_i < 0} |k_i|$, with k_i the eigenvalues of the kernel. We obtain 1.40, 1.27 and 2.41 for Sets I, II, and III, respectively.