Computer Physics Communications 248 (2020) 107006

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Support vector machines on the D-Wave quantum annealer☆

D. Willsch [a,b,*], M. Willsch [a,b], H. De Raedt [c], K. Michielsen [a,b]

[a] Institute for Advanced Simulation, Jülich Supercomputing Centre, Forschungszentrum Jülich, D-52425 Jülich, Germany
[b] RWTH Aachen University, D-52056 Aachen, Germany
[c] Zernike Institute for Advanced Materials, University of Groningen, Nijenborgh 4, NL-9747 AG Groningen, The Netherlands

ARTICLE INFO

Article history:
Received 17 June 2019
Received in revised form 30 September 2019
Accepted 27 October 2019
Available online 4 November 2019

Keywords:
Support vector machine
Kernel-based SVM
Machine learning
Classification
Quantum computation
Quantum annealing

ABSTRACT

Kernel-based support vector machines (SVMs) are supervised machine learning algorithms for classification and regression problems. We introduce a method to train SVMs on a D-Wave 2000Q quantum annealer and study its performance in comparison to SVMs trained on conventional computers. The method is applied to both synthetic data and real data obtained from biology experiments. We find that the quantum annealer produces an ensemble of different solutions that often generalizes better to unseen data than the single global minimum of an SVM trained on a conventional computer, especially in cases where only limited training data is available. For cases with more training data than currently fits on the quantum annealer, we show that a combination of classifiers for subsets of the data almost always produces stronger joint classifiers than the conventional SVM for the same parameters.

1. Introduction

The growing interest in both quantum computing and machine learning has inspired researchers to study a combination of both fields, termed *quantum machine learning* [1–7]. Recently, it has been shown that using the D-Wave quantum annealer can yield advantages in classification performance over state-of-the-art conventional approaches for certain computational biology problems using a linear classifier [8]. In this paper, we improve on these results by replacing the linear classifier with a superior nonlinear classification approach, the kernel-based support vector machine (SVM) [9,10]. We introduce its formulation for a D-Wave quantum annealer and present training results for both synthetic data and real data. To distinguish between the SVM formulations, we use the word *classical* to denote the original version of an SVM as defined in [9].

The field of supervised machine learning deals with the problem of learning model parameters from a set of labeled training data in order to make predictions about test data. SVMs in particular are known for their stability (in comparison to decision trees or deep neural networks [11–14]), in the sense that small differences in the training data do not generally produce huge differences in the resulting classifiers. Moreover, kernel-based SVMs profit from the *kernel trick*, effectively maneuvering around the "curse of dimensionality" [9,15]. In contrast to Deep Learning, which often requires large amounts of training data, SVMs are typically used when only small sets of training data are available. But also in combination with Deep Learning, where SVMs are applied on top of neural networks to classify the detected features, SVMs have been found to yield significant gains in classification performance [16–19].

Quantum annealers manufactured by D-Wave Systems Inc. are available with about 2000 qubits [20–23]. They automatically produce a variety of close-to-optimal solutions to a given optimization problem [8,23,24]. This is particularly interesting in the context of machine learning, because any of the solutions produced for a given *training dataset* have the potential to perform well on new *test data*. For SVMs, for which the original solution is the *global optimum* of the underlying convex optimization problem for the training data [10], it is an interesting question whether the ensemble of different solutions from the quantum annealer can improve the classification performance for the test data.

We conduct our SVM experiments on a D-Wave 2000Q (DW2000Q) quantum annealer [23]. Quantum annealing (QA) is so far the only paradigm of quantum computing for which processors of a reasonable size are available. The other paradigm of quantum computing, i.e., the gate-based (or universal) quantum computer [25], is still limited to less than 100 quantum bits (qubits) [26]. It is worth mentioning that for gate-based quantum computers, a quantum algorithm for SVMs has already been proposed [27]. However, only a few very simple tasks, for which almost all classification was already done in the preprocessing step, have been studied experimentally [28].

☆ The review of this paper was arranged by Prof. D.P. Landau.

* Corresponding author at: Institute for Advanced Simulation, Jülich Supercomputing Centre, Forschungszentrum Jülich, D-52425 Jülich, Germany.
E-mail address: d.willsch@fz-juelich.de (D. Willsch).

https://doi.org/10.1016/j.cpc.2019.107006

QA requires the formulation of the computational problem as a quadratic unconstrained binary optimization (QUBO). A QUBO problem is defined as the minimization of the energy function

$$E = \sum_{i \le j} a_i Q_{ij} a_j, \tag{1}$$

where $a_i \in \{0, 1\}$ are the binary variables of the optimization problem, and Q is an upper-triangular matrix of real numbers called the QUBO weight matrix. Note that the size of the DW2000Q quantum processor and the Chimera topology [22] impose certain restrictions on this matrix. A popular alternative formulation of the problem in terms of variables $s_i \in \{-1, 1\}$ is known as the Ising model [29,30].

We present a formulation of SVMs as a QUBO defined by Eq. (1) and discuss certain mathematical properties in the training of SVMs that make it particularly appealing for use on a quantum annealer. In comparison to the classical SVM, we find that a combination of the solutions returned by the quantum annealer often surpasses the single solution of the classical SVM.

This paper is structured as follows: In Section 2, we introduce the classical SVM, our formulation of an SVM for QA, and the metrics we use to compare the performance of both. Section 3 contains the application of both SVM versions to synthetic two-dimensional data and real data from biology experiments, including the calibration, training, and testing phase. We conclude our study with a short discussion in Section 4.

2. SVMs on a quantum annealer

In this section, we first briefly review the classical SVM, and then introduce the QA version of an SVM. Finally, we discuss ways to evaluate the classification performance in the applications presented in the next section.

2.1. The classical SVM

An SVM is a supervised machine-learning algorithm for classification and regression. It operates on a dataset

$$D = \{(\mathbf{x}_n, t_n) : n = 0, \dots, N-1\}, \tag{2}$$

where $\mathbf{x}_n \in \mathbb{R}^d$ is a point in d-dimensional space (a *feature vector*), and t_n is the target label assigned to $\mathbf{x}_n$. We consider the task of learning a binary classifier that assigns a class label $t_n = \pm 1$ for a given data point $\mathbf{x}_n$. In the following, we call the class $t_n = 1$ "positive" and the class $t_n = -1$ "negative".

Training an SVM amounts to solving the quadratic programming (QP) problem [15]

$$\text{minimize} \qquad E = \frac{1}{2} \sum_{nm} \alpha_n \alpha_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) - \sum_n \alpha_n, \tag{3}$$

$$\text{subject to} \qquad 0 \le \alpha_n \le C, \tag{4}$$

$$\text{and} \qquad \sum_n \alpha_n t_n = 0, \tag{5}$$

for N coefficients $\alpha_n \in \mathbb{R}$, where C is a regularization parameter and $k(\cdot, \cdot)$ is the kernel function of the SVM [9]. The resulting coefficients α_n define a $(d-1)$-dimensional decision boundary that separates $\mathbb{R}^d$ in two regions corresponding to the predicted class label. A typical solution often contains many $\alpha_n = 0$. The decision boundary is then determined by the points corresponding to $\alpha_n \neq 0$ (the *support vectors* of the SVM). A prediction for an arbitrary point $\mathbf{x} \in \mathbb{R}^d$ can be made by evaluating the decision function

$$f(\mathbf{x}) = \sum_n \alpha_n t_n k(\mathbf{x}_n, \mathbf{x}) + b, \tag{6}$$

where a reasonable choice to determine the bias b is given by [15]

$$b = \frac{\sum_n \alpha_n (C - \alpha_n) \left[t_n - \sum_m \alpha_m t_m k(\mathbf{x}_m, \mathbf{x}_n) \right]}{\sum_n \alpha_n (C - \alpha_n)}. \tag{7}$$

Geometrically, the decision function $f(\mathbf{x})$ represents a signed distance between the point $\mathbf{x}$ and the decision boundary. Thus the class label for $\mathbf{x}$ predicted by the trained SVM is $\widetilde{t} = \operatorname{sign}(f(\mathbf{x}))$.

The formulation of the problem given in Eqs. (3)–(5) is the so-called dual formulation of an SVM (see [10] for more information). Since it represents a convex quadratic optimization problem, it is one of the rare minimization problems in machine learning that have a global minimum. Note, however, that the global minimum with respect to the training dataset D may not necessarily be optimal for generalizing to the test dataset.

Kernel-based SVMs are particularly powerful since they allow for nonlinear decision boundaries defined by $f(\mathbf{x}) = 0$ (see Eq. (6)), implicitly mapping the feature vectors to higher-dimensional spaces [31]. Interestingly, the complexity of the problem does not grow with this dimension, since only the values of the kernel function $k(\mathbf{x}_n, \mathbf{x}_m)$ enter the problem specification (see Eq. (3)). This fact is known as the *kernel trick* [9,15].

The choice of the kernel function can have a significant impact on the results. Typical choices for SVMs are linear, polynomial, sigmoid, and radial basis function (rbf) kernels [10]. In general, an rbf kernel is a kernel for which $k(\mathbf{x}_n, \mathbf{x}_m)$ can be written as a function of the distance $\|\mathbf{x}_n - \mathbf{x}_m\|$ only [9]. The most common rbf kernel is the Gaussian kernel (often referred to as *the* rbf kernel),

$$\mathrm{rbf}(\mathbf{x}_n, \mathbf{x}_m) = e^{-\gamma \|\mathbf{x}_n - \mathbf{x}_m\|^2}, \tag{8}$$

where the value of the hyperparameter $\gamma > 0$ is usually determined in a calibration procedure prior to the training phase (if no particular set of values for γ is known for the data, a good strategy is to try exponentially growing sequences like $\gamma \in \{\dots, 2^{-3}, 2^{-2}, \dots\}$ [32]).

Gaussian kernels have the advantage of not suffering as much from numerical difficulties as polynomial kernels [32] and, in general, compare favorably to sigmoid or tanh kernels (which are, strictly speaking, not positive semi-definite) [33]. They implicitly map the feature vector onto an infinite-dimensional space [10]. In principle, a Gaussian kernel also includes the linear kernel as an asymptotic case [34]. However, we explicitly include a linear kernel for convenience, denoted by the special value $\gamma = -1$. Therefore, we formally define

$$k(\mathbf{x}_n, \mathbf{x}_m) := \begin{cases} \mathrm{rbf}(\mathbf{x}_n, \mathbf{x}_m) & (\gamma > 0) \\ \mathbf{x}_n \cdot \mathbf{x}_m & (\gamma = -1), \end{cases} \tag{9}$$

as the kernel function for our experiments.

> In the following, we symbolically write $\texttt{cSVM}(C, \gamma)$ to denote the training of the classical SVM defined by Eqs. (3)–(5) with the kernel function given in Eq. (9).

For the computational work associated with `cSVM`, we used the C++ library LIBSVM [35], the Python module Scikit-learn [36], and a quadratic programming solver from the Python package CVXOPT [37]. All packages produced identical results, i.e., the global optimum of the convex optimization problem.

2.2. The quantum SVM

The solution to Eqs. (3)–(5) consists of real numbers $\alpha_n \in \mathbb{R}$. However, the DW2000Q can only produce discrete, binary

solutions to a QUBO (see Eq. (1)). Therefore, we use an encoding of the form

$$\alpha_n = \sum_{k=0}^{K-1} B^k a_{Kn+k}, \tag{10}$$

where $a_{Kn+k} \in \{0, 1\}$ are binary variables, K is the number of binary variables to encode α_n, and B is the base used for the encoding. In practice, we obtained good results for $B = 2$ or $B = 10$ and a small number of K (see also the list of arguments given below).

To formulate the QP problem given in Eqs. (3)–(5) as a QUBO (see Eq. (1)), we use the encoding defined in Eq. (10) and introduce a multiplier ξ to include the second constraint given in Eq. (5) as a squared penalty term. We obtain

$$E = \frac{1}{2}\sum_{nmkj} a_{Kn+k} a_{Km+j} B^{k+j} t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) - \sum_{nk} B^k a_{Kn+k} + \xi \left(\sum_{nk} B^k a_{Kn+k} t_n \right)^2 \tag{11}$$

$$= \sum_{n,m=0}^{N-1} \sum_{k,j=0}^{K-1} a_{Kn+k} \widetilde{Q}_{Kn+k,Km+j} a_{Km+j}, \tag{12}$$

where $\widetilde{Q}$ is a matrix of size $KN \times KN$ given by

$$\widetilde{Q}_{Kn+k,Km+j} = \frac{1}{2} B^{k+j} t_n t_m (k(\mathbf{x}_n, \mathbf{x}_m) + \xi) - \delta_{nm}\delta_{kj} B^k. \tag{13}$$

Since $\widetilde{Q}$ is symmetric, the upper-triangular QUBO matrix Q required for the QUBO formulation given in Eq. (1) is defined by $Q_{ij} = \widetilde{Q}_{ij} + \widetilde{Q}_{ji}$ for $i < j$ and $Q_{ii} = \widetilde{Q}_{ii}$. Note that the constraint Eq. (4) is automatically included in Eq. (11) through the encoding given in Eq. (10), since the maximum for α_n is given by

$$C = \sum_{k=1}^{K} B^k, \tag{14}$$

and $\alpha_n \geq 0$ by definition.

Given K, each α_n can take only 2^K different values according to Eq. (10). At first, it may seem questionable why a small number of B and K should be sufficient. The following arguments and empirical findings for SVMs motivated us to try the QUBO approach:

1. A typical solution to Eqs. (3)–(5) consists of many $\alpha_n = 0$ with only a few $\alpha_m \neq 0$ (the corresponding data points $\{\mathbf{x}_m\}$ are the support vectors). On a digital computer using floating-point numbers, establishing convergence to exactly 0 is a subtle task, whereas the encoding in Eq. (10) directly includes this value.
2. The box constraint Eq. (4) is automatically satisfied by the choice of the encoding Eq. (10) (see Eq. (14)).
3. In principle, one can extend the encoding Eq. (10) to fractional numbers by replacing the base B^k with B^{k-k_0} for some $k_0 \in \mathbb{N}$. Eventually, this would yield the same range of floating-point numbers as used in conventional digital computers, namely the IEEE standard for floating-point arithmetic [38]. However, it was observed that this kind of precision is not required for SVMs to produce reasonable results (see [39]), and it would also not be feasible with the current generation of QA devices.
4. For the classification task addressed by an SVM, the global order of magnitude of all α_n is often not as important as the relative factors between different α_n. This can be understood by studying the effect of substituting $\alpha_n \mapsto S\alpha_n$ for some factor S in Eqs. (3)–(5). Since E and E/S^2 are optimal for the same $\{\alpha_n\}$, and the hyperparameters of the box constraint are calibrated separately, it only replaces the linear term in Eq. (3) by $-\sum_n \alpha_n/S$. This term only affects the size of the margin between the decision boundary and the support vectors (see also [10]). However, if this is still found to be an issue, one can simply adjust the encoding Eq. (10) accordingly.
5. Especially for the Gaussian kernel given in Eq. (8), points with a large distance $\|\mathbf{x}_n - \mathbf{x}_m\| \gg 1$ result in $k(\mathbf{x}_n, \mathbf{x}_m) \approx 0$. This can be used to reduce couplings between the qubits such that embedding the problem on the quantum annealer is less complex. This may either yield better solutions or allow larger problems to be embedded on the DW2000Q.
6. The constraint $\sum_n a_n t_n = 0$ mathematically corresponds to an optimal bias b in the decision function given in Eq. (6) (see [10]). We have included it in Eq. (11) through the multiplier ξ. However, the constraint need not be satisfied exactly for the classification task to produce good results. Since the bias b is only one parameter, it can easily be adjusted afterwards if necessary. For this reason, it can be that $\xi = 0$ already suffices to get reasonable results. Furthermore, the special value $\xi = 1$ yields the Mangasarian–Musicant variant of an SVM (see [40,41] for more information). This variant has been shown to produce equally good classifiers while, at the same time, being numerically much more tractable [15]. An alternative approach would be to include ξ in the parameter set that has to be optimized (as conventionally done for Lagrange multipliers) by choosing an additional encoding for ξ such as Eq. (10). In this case, it would suffice to replace the last term in Eq. (11) by the linear penalty term $\xi \sum_n a_n t_n$. We experimented with this approach and it yields similar but less robust results (data not shown). For this reason, and due to the (on present quantum annealers) small set of numbers represented by the encoding Eq. (10), and also because of the SVM's sensitivity to the bias, we found it more convenient to keep ξ as a hyperparameter, and if necessary adjust the bias afterwards (see also Appendix A).

The last step required to run the optimization problem on the DW2000Q is the embedding procedure [42,43]. It is necessary because in general, the QUBO given in Eq. (1) includes some couplers $Q_{ij} \neq 0$ between qubit i and qubit j for which no physical connection exists on the chip (the connectivity of the DW2000Q is given by the Chimera topology [22]). The idea of embedding is to combine several physical qubits to one logical qubit (also called *chain*) by choosing a large negative value for their coupling strengths to favor solutions where the physical qubits are aligned. This can be used to increase the logical connectivity between the qubits.

We use a function provided by D-Wave Systems Inc. to generate embeddings for the QUBOs given by Eq. (13) [44]. When no embedding can be found, we successively decrease the number of nonzero couplers n_{cpl} by setting the smallest couplers to zero until an embedding is found. This works especially well in combination with the Gaussian kernel given in Eq. (8), where points with a large squared distance $\|\mathbf{x}_n - \mathbf{x}_m\|^2$ only produce negligible contributions to the QUBO. Typical values for n_{cpl} for the applications discussed in Section 3 are between 1600 and 2500, while the number of required qubits ranges from 28 to 114 with peaks at 56, 58, 84, and 87.

We chose to test the default mode of operation of the DW2000Q with an annealing time of 20 μs and leave the analysis

of improving the QA results by advanced features like reverse annealing, spin-reversal transforms, special annealing schedules, or alternative embedding heuristics to the future [23,45,46].

To summarize, the final QA version of the SVM defined by the QUBO in Eq. (13) depends on the following hyperparameters: the encoding base B, the number K of qubits per coefficient α_n, the multiplier ξ, and the kernel parameter γ (the number n_{cpl} of strongest couplers embedded on the DW2000Q is different for every run and is not a parameter of the SVM itself).

> We denote the QA version of an SVM defined in Eq. (13) as $\mathtt{qSVM}(B, K, \xi, \gamma)$, by analogy with $\mathtt{cSVM}(C, \gamma)$ defined in Eqs. (3)–(5),

For each run on the DW2000Q, we consider the twenty lowest-energy samples from 10,000 reads, denoted by $\mathtt{qSVM}(B, K, \xi, \gamma)\#i$ for $i = 0, \ldots, 19$. Note that the cut at $i = 20$ is arbitrary; one could also consider 50 or more samples from the distribution if appropriate.

In principle, it can happen that a particular sample $\#i$ yields only $\alpha_n = 0$ or $\alpha_n = C$ such that the bias b in Eq. (7) is undefined. This reflects the rare situation that no support vectors have been found. In this case, one may simply discard the affected sample and consider only the remaining samples.

Note that the DW2000Q produces a variety of close-to-optimal solutions (i.e., a variety of different coefficients $\{\alpha_n\}^{(i)}$ obtained from Eq. (10)). Many of these solutions may have a slightly higher energy than the global minimum $\{\alpha_n\}^*$ found by cSVM, but still solve the classification problem for the training data as intended. The different solutions often emphasize different features of the training data. When applied to the test data, a combination of these solutions has the potential to solve the classification task better than cSVM, which only yields the global minimum for the training data.

For the computational work associated with qSVM, we used the D-Wave Ocean SDK [44], which provides the functionality to generate embeddings (see above) and produce results for the QUBO matrix defined in Eq. (13).

2.3. Using accuracy, AUROC, and AUPRC to assess the classification performance

To measure the classification performance, we consider a separation of the data D given in Eq. (2) into two disjoint subsets $D^{(\mathrm{train})}$ and $D^{(\mathrm{test})}$. The training data $D^{(\mathrm{train})}$ is used to train either $\mathtt{cSVM}(C, \gamma)$ or $\mathtt{qSVM}(B, K, \xi, \gamma)$. In both cases, the result of the training is the set of coefficients $\{\alpha_n\}$, which can be used to make class predictions by means of the decision function given in Eq. (6). The classifier is then evaluated for the test data $D^{(\mathrm{test})}$ by comparing the class prediction $\widetilde{t_n} = \mathrm{sign}(f(\mathbf{x}_n))$ with the true label t_n for each $(\mathbf{x}_n, t_n) \in D^{(\mathrm{test})}$ from the test data.

A straightforward method to assess the performance of a classifier is to count the number of correct predictions, i.e., the number of true positives TP for which $\widetilde{t_n} = t_n = 1$. Dividing this number by the total number of points $|D^{(\mathrm{test})}|$ yields the *classification accuracy*. However, in binary classification problems, the accuracy is generally considered a bad measure [47,48], because a higher accuracy does not necessarily imply that the classifier is better. As a simple example, consider a dataset with 80% negatives. A trivial all-negative classifier, which always returns -1, would already achieve an accuracy of 80%, even though it is practically useless. Instead, we are often interested in identifying good positives, especially if the dataset contains a lot of negatives.

To obtain a more robust measure, we first count the number of all cases that can occur when making the class prediction $\widetilde{t_n} = \mathrm{sign}(f(\mathbf{x}_n))$: the number TP of true positives where $\widetilde{t_n} = t_n = 1$, the number FP of false positives where $\widetilde{t_n} = 1$ but $t_n = -1$, the number TN of true negatives where $\widetilde{t_n} = t_n = -1$, and the number FN of false negatives where $\widetilde{t_n} = -1$ but $t_n = 1$ (note that the sum of these four numbers is equal to the number of test data points $|D^{(\mathrm{test})}|$). Given these counts, one can compute the true positive rate $\mathrm{TPR} = \mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$ (also known as Recall), the false positive rate $\mathrm{FPR} = \mathrm{FP}/(\mathrm{FP}+\mathrm{TN})$, and the Precision $= \mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$ (defined to be 1 if $\mathrm{TP}+\mathrm{FP} = 0$).

Unfortunately, simply using one of these ratios instead of the classification accuracy does not solve the above problem either. For instance, if we were to measure success by means of the smallest false positive rate FPR, we would be satisfied with the trivial all-negative classifier, since it would never produce a false positive such that $\mathrm{FPR} = 0$.

The solution to this kind of problem is to use more robust metrics such as AUROC (area under the Receiver Operating Characteristic curve) and AUPRC (area under the Precision–Recall curve) [48,49]. These metrics are not based on a single evaluation of the classifier, but rather on the performance of the classifier as a function of the bias b in Eq. (6). By sweeping b, the classifier is artificially moved from an all-negative classifier (corresponding to $b \to -\infty$, where $\mathrm{TPR} = \mathrm{FPR} = \mathrm{Recall} = 0$ and $\mathrm{Precision} = 1$) to an all-positive classifier (corresponding to $b \to \infty$, where $\mathrm{TPR} = \mathrm{FPR} = \mathrm{Recall} = 1$). In essence, this procedure moves the decision boundary through all test data points, thereby measuring the characteristic shape of the decision boundary.

By plotting TPR vs. FPR, one generates the ROC curve, and by plotting Precision vs. Recall one generates the Precision–Recall curve (see Fig. 3 below for an example of these curves). The area under both curves is termed AUROC and AUPRC, respectively, and represents a much more robust measure for the quality of a classifier than the classification accuracy. This means that optimizing a classifier for AUROC and AUPRC is unlikely to result in a useless classifier, which can happen when optimizing for the accuracy instead [48] (see the example given above).

Note, however, that there is a particular situation in which optimizing for the accuracy is appropriate, namely to obtain a good value for the bias b in the decision function given in Eq. (6). The reason for this is that, ultimately, we are interested in making a definite class prediction $t = \mathrm{sign}(f(\mathbf{x}))$ for an arbitrary point $\mathbf{x}$. Since AUROC and AUPRC are independent of b, we cannot use these metrics to obtain an optimal bias. Instead, a reasonable option is to use the value of b for which the accuracy with respect to the training data is maximal. This is especially true for qSVM, for which the candidate given in Eq. (7) may not be optimal. This is the case for the real test problem below (see also Appendix A).

In the following applications, we report accuracy, AUROC, and AUPRC to compare the classifiers and to measure the classification performance.

3. Applications

3.1. Two-dimensional synthetic data

As a proof of concept and to understand the power of qSVM, we consider a small set of two-dimensional synthetic data. This has the advantage that the results can be easily visualized and the quality of the many different classifiers returned by the quantum annealer can be compared.

The dataset D consists of $n = 1, \ldots, 40$ points $(\mathbf{x}_n, t_n)$, where the first half corresponds to the negative class $t_n = -1$ representing an outer region, and the second half corresponds to the positive class $t_n = 1$ representing an inner region. It was generated according to

$$\mathbf{x}_n = r_n \begin{pmatrix} \cos\varphi_n \\ \sin\varphi_n \end{pmatrix} + \begin{pmatrix} s_n^x \\ s_n^y \end{pmatrix}, \tag{15}$$

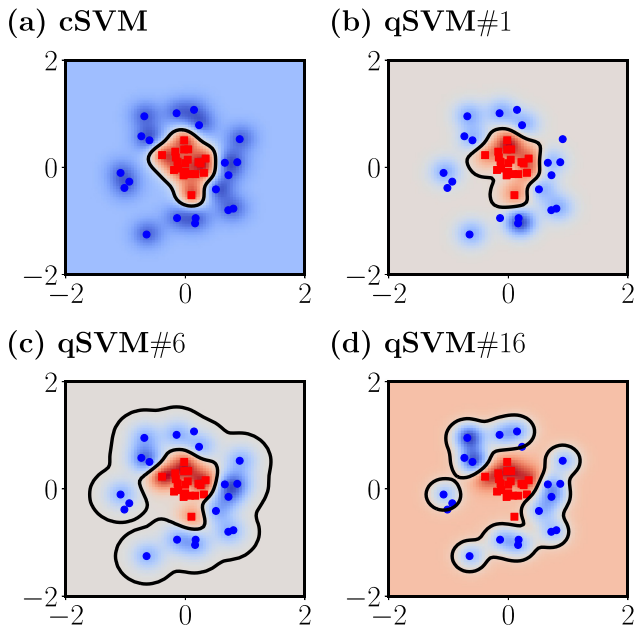

Fig. 1. Visualization of the classification boundary resulting from (a) the global optimum produced by the classical SVM, and (b)–(d) various solutions from the ensemble produced by the QA version of the SVM for the same problem (the identifier qSVM#*i* indicates the $(i+1)$th sample produced by the DW2000Q, starting at $i = 0$ and ordered by lowest energy). The parameters for the SVMs are $B = K = 2$, $\xi = 0$, $\gamma = 16$, and $C = 3$. The two classes for the two-dimensional synthetic data are plotted as red squares ($t_n = 1$) and blue circles ($t_n = -1$), respectively. The corresponding background color indicates the distance to the decision boundary. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where $r_n = 1$ if $t_n = -1$ and $r_n = 0.15$ if $t_n = 1$, φ_n is linearly spaced on $[0, 2\pi)$ for each class, and s_n^x and s_n^y are drawn from a normal distribution with mean 0 and standard deviation 0.2.

We visualize the resulting decision boundaries $f(\mathbf{x}) = 0$ for cSVM$(3, 16)$ in Fig. 1(a), and for three separate solutions from the ensemble found by qSVM$(2, 2, 0, 16)$ in Fig. 1(b)–(d). For demonstration purposes, the plotted data points do not come from a separate test set but are the same 40 points that the SVM versions have been trained on. The value of the decision function $f(\mathbf{x})$ given in Eq. (6) determines the background color, obtained by evaluating $f(\mathbf{x})$ for each point $\mathbf{x}$ in the two-dimensional plotting grid.

We see that cSVM shown in Fig. 1(a) satisfies all the properties expected from the global minimum of an SVM, i.e., separating the dataset into two regions where the decision boundary has a maximum margin to the closest data points (the support vectors).

The DW2000Q, however, automatically produces a variety of alternative classifiers shown in Fig. 1(b)–(d). Each of them solves the classification task of the training set as intended, and additionally highlights different features present in the training data. While sample #1 shown in Fig. 1(b) still resembles the properties of the global minimum, sample #6 shown in Fig. 1(c) yields a more narrow enclosure of the outer circle. The classifier from sample #16 shown in Fig. 1(d) is even sensitive to the gaps in the outer circle. This result suggests that a combination of the classifiers returned by qSVM may be more powerful than the single classifier produced by cSVM.

3.2. Application to real data

We compare the performance of both cSVM and qSVM when applied to real data obtained from the biology experiment studied in [8]. This data was provided to us on request. Briefly, the classification task is to decide whether a certain protein (a transcription factor labeled Mad, Max, or Myc) binds to a certain DNA sequence such as CCCACGTTCT (see also [50,51]).

The data consists of nine separate datasets labeled Mad50, Max50, Myc50, Mad70, Max70, Myc70, Mad80, Max80, and Myc80. The datasets consist of $N = 1655$ (Mad), $N = 1599$ (Max), and $N = 1584$ (Myc) data points, respectively. The data points $(\mathbf{x}_n, t_n)$ for $n = 1, \ldots, N$ consist of a 40-dimensional vector $\mathbf{x}_n \in \{-1, +1\}^{40}$ representing the DNA sequence, and a label indicating whether the protein binds to this DNA sequence ($t_n = +1$) or not ($t_n = -1$). The DNA sequence is encoded by mapping each base-pair in the DNA alphabet {A,C,G,T} according to A $\mapsto (+1,-1,-1,-1)$, C $\mapsto (-1,+1,-1,-1)$, G $\mapsto (-1,-1,+1,-1)$, and T $\mapsto (-1,-1,-1,+1)$, and concatenating all encoded base-pairs. An encoding of this type is sometimes called *one-hot encoding* (often using 0 instead of -1) since only one element in each encoded base-pair is $+1$ (cf. also [8,50]). For each dataset, the number behind the protein label indicates the percentage of negative classes such that e.g. the dataset Max80 contains 80% non-binding DNA sequences ($t_n = -1$) and 20% binding DNA sequences ($t_n = +1$).

We separate each of the nine datasets into 90% training data $D^{(\text{train})}$ and 10% test data $D^{(\text{test})}$. The training data is used for calibration of the hyperparameters and for training the classifiers. The test data is unseen during training and exclusively used to test the classifiers in the test phase. The entire data handling procedure is sketched in Fig. 2.

3.2.1. Calibration phase: Results for a small training dataset

To select the hyperparameters of qSVM, we use 10-fold Monte Carlo (or split-and-shuffle) cross-validation. This means that we train qSVM(B, K, ξ, γ) on 2% of $D^{(\text{train})}$ (approximately 30 data points) and evaluate its performance on the remaining data points of $D^{(\text{train})}$ for validation. The data is then shuffled and the process is repeated a total number of ten times (see Fig. 2).

The small fraction of 2% was chosen because of the size limitations of the quantum annealer (cf. also [8]). Since this is a very small amount of data, we performed some initial tests before systematically calibrating the hyperparameters. In these tests, we observed that qSVM can produce significantly stronger classifiers than cSVM for the same little training data and parameters. One example is shown in Fig. 3, where the ROC and PR curves are plotted for qSVM$(10, 3, 0, -1)$#14 (see Fig. 3(a)) and for cSVM$(111, -1)$ (see Fig. 3(b)), generated by sweeping the bias b as explained in Section 2.3. While the QA version produces almost optimal curves, the global optimum from the classical SVM obviously lacks precision when applied to the much larger validation data.

For each dataset, the hyperparameters are calibrated by evaluating qSVM for $B \in \{2, 3, 5, 10\}$ and $K \in \{2, 3\}$ (cf. Eq. (10)), $\xi \in \{0, 1, 5\}$ (cf. Eq. (11)), and $\gamma \in \{-1, 0.125, 0.25, 0.5, 1, 2, 4, 8\}$ (cf. Eq. (9)). We generically consider the classifiers $\{\alpha_n^{(i)}\}$ from the twenty best solutions qSVM(B, K, ξ, γ)#i for $i = 0, \ldots, 19$ as described in Section 2.2. The evaluation is repeated ten times for the Monte Carlo cross-validation. Therefore, each set of hyperparameters for each dataset results in a total of 200 values for AUROC, AUPRC, and accuracy.

An example of the calibration procedure for the dataset Max70 is shown in Fig. 4. For this dataset, we see that the linear kernels denoted by $\gamma = -1$ (see Eq. (9)) dominate (but Gaussian kernels perform still reasonably well). The selected set of hyperparameters in this case is $B = 10$, $K = 3$, $\xi = 5$, and $\gamma = -1$, corresponding to the leftmost points in Fig. 4. We also see fluctuations in the mean accuracy which are not reflected by AUROC and AUPRC. Since AUROC and AUPRC are insensitive to the

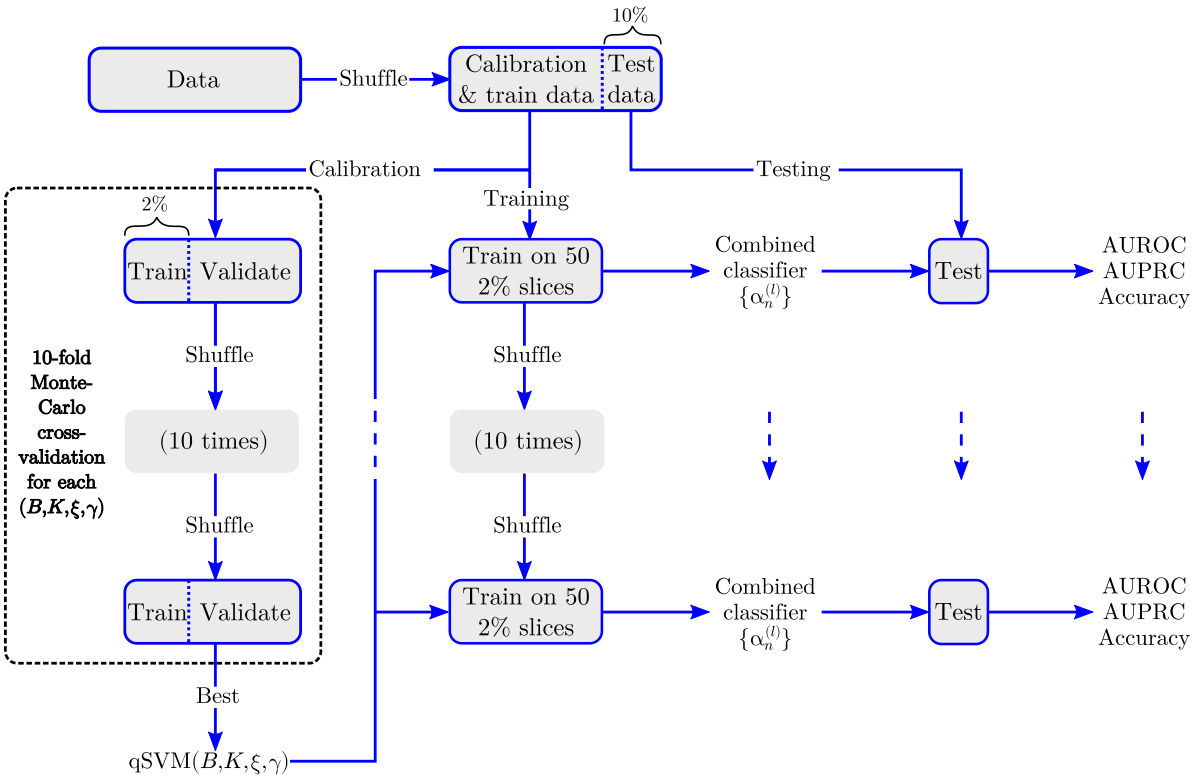

Fig. 2. Data handling procedure for the computational biology problem. Each of the nine datasets is split into 90% calibration and training data $D^{(\text{train})}$ and 10% test data $D^{(\text{test})}$. In the calibration phase, 10-fold Monte Carlo cross-validation is used to select the hyperparameters B, K, ξ, and γ (see Section 2.2), training on 2% of $D^{(\text{train})}$ and validating on the rest. In the test phase, the selected $\texttt{qSVM}(B, K, \xi, \gamma)$ is applied to every 2% slice of $D^{(\text{train})}$. The resulting classifiers are combined to classify the test data $D^{(\text{test})}$ to evaluate the AUROC, the AUPRC, and the classification accuracy (see Section 2.3). The test procedure is repeated 10 times to gather statistics.

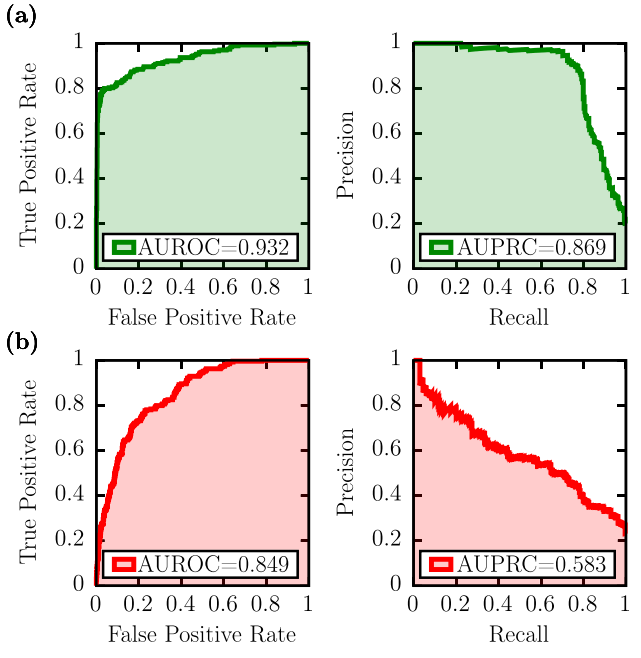

Fig. 3. (Color online) Example for the generated ROC and PR curves to measure the quality of the classifiers. (a) $\texttt{qSVM}(10, 3, 0, -1)\#14$ using $n_{\text{cpl}} = 2000$ couplers, and (b) $\texttt{cSVM}(111, -1)$ (note that $C = 111$ for $\texttt{cSVM}$ corresponds to $B = 10$ and $K = 3$, see Eq. (14)). Both SVMs have been trained and validated on the same data, taken from the fifth step in the 10-fold cross-validation procedure for the dataset Max80 [8].

Table 1
Selected hyperparameters for each dataset [8]. The parameters are the base B, the number K of qubits per coefficient α_n, the multiplier ξ, the kernel parameter γ, and the box constraint parameter C (see Section 2). The value of C is fixed by B and K through Eq. (14) and is given for reference only.

Dataset	B	K	ξ	γ	C
Mad50	2	3	5	0.125	7
Max50	2	3	5	0.125	7
Myc50	2	2	0	0.125	3
Mad70	10	3	5	−1	111
Max70	10	3	5	−1	111
Myc70	10	3	5	−1	111
Mad80	10	3	5	−1	111
Max80	10	3	0	−1	111
Myc80	10	3	5	−1	111

bias, this indicates that the choice for the bias b given by Eq. (7) may not always be optimal (see Appendix A for a way to improve the bias if the accuracy matters).

We selected the hyperparameters based on both mean AUROC and AUPRC. The reason for this is that we observed, when selecting exclusively based on the best AUPRC (cf. [8]), we sometimes obtained hyperparameters yielding AUROC ≈ 0.5 (the result for a random classifier [48]).

In Table 1, we list the best hyperparameters selected for each dataset. The trend from Gaussian kernels to linear kernels can be observed in all datasets: For Mad50, Max50, and Myc50, where half of the data is classified as positive and the other half as negative, only the Gaussian kernels can produce a reasonable decision boundary (see also Table 2 in Appendix B). But when going to higher class imbalances as present in the datasets Mad80, Max80,

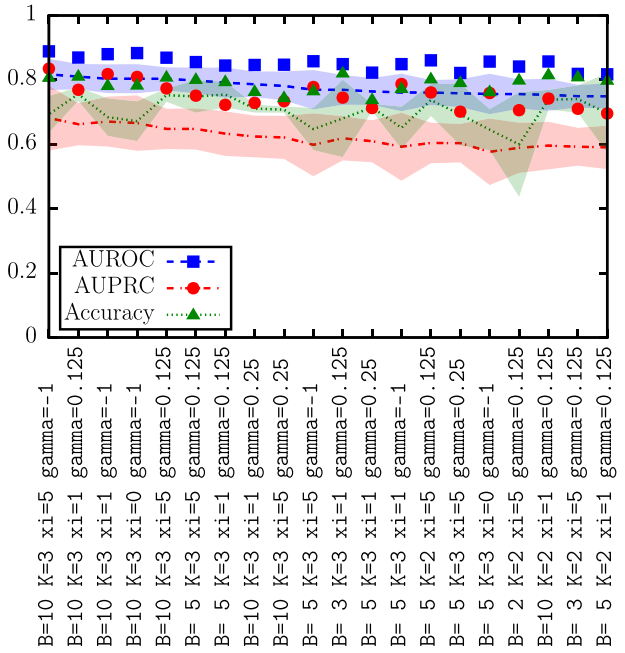

Fig. 4. Calibration performance of qSVM for the best sets of hyperparameters (B, K, ξ, γ), ordered by mean AUROC, for the dataset Max70 [8]. Shown are the AUROC (blue dashed line), the AUPRC (red dash-dotted line), the accuracy (green dotted line), and the respective standard deviations (shaded areas) over 200 classifiers (10 different calibration folds times 20 of the best solutions from the DW2000Q). Lines connecting the averages are guides to the eye. Squares, circles, and triangles denote the maximum performance among each of the 200 classifiers. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

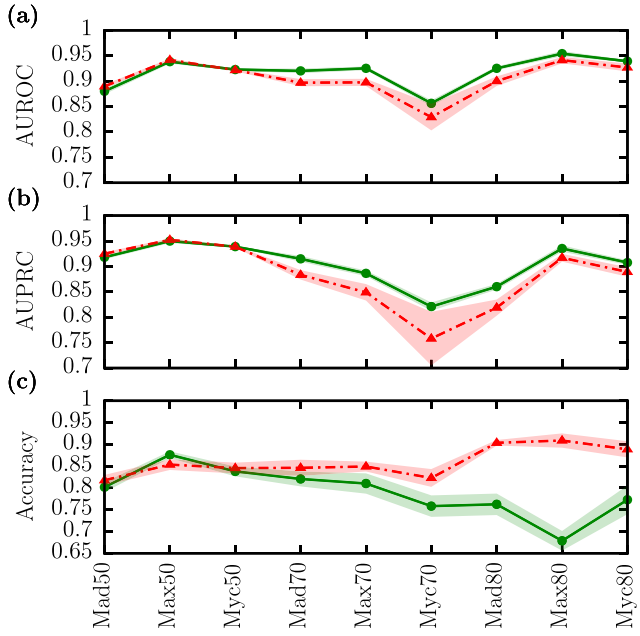

Fig. 5. Performance of qSVM (solid green line) and cSVM (dash-dotted red line) as measured by (a) AUROC, (b) AUPRC, and (c) accuracy (see Section 2.3) using the decision function given in Eq. (16) for each of the nine datasets from the computational biology problem [8]. The parameters for each dataset are taken from Table 1. The standard deviation over ten repetitions (see Fig. 2) is shown as shaded areas. Lines are guides to the eye. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

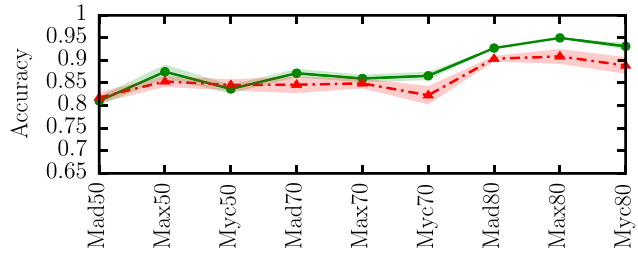

Fig. 6. Classification accuracy of qSVM (solid green line) and cSVM (dash-dotted red line) as shown in Fig. 5(c), after adjusting the suboptimal bias b to b^* where the accuracy for the training data is higher (see Appendix A). The metrics AUROC and AUPRC are the same as in Fig. 5(a) and (b). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and Myc80, a linear decision boundary suffices to classify the DNA sequences.

The numerical results of the calibration procedure for each dataset in comparison with the corresponding cSVM are listed in Table 2 in Appendix B.

3.2.2. Training and test phase: Results for a larger training dataset

In this section, we examine a way to overcome the size limitations of the DW2000Q for real applications with a bigger training dataset. We take the same nine DNA datasets as before, but now consider the full datasets $D^{(\text{train})}$ for training a classifier. The goal is to construct an aggregated classifier from the results of qSVM trained on each 2% slice of the available training data (see Fig. Fig. 2). Each of the $L = 50$ slices is labeled $D^{(\text{train},l)}$ for $l = 0, \ldots, 49$. The hyperparameters for each dataset are taken from the calibration results listed in Table 1.

The combined classifier is constructed in two steps. First, for each slice $D^{(\text{train},l)}$, the twenty best solutions from the DW2000Q (labeled qSVM(B, K, ξ, γ)#i for $i = 0, \ldots, 19$) are combined by averaging over the respective decision functions $f^{(l,i)}(\mathbf{x})$ (see Eq. (6)). Since the decision function is linear in the coefficients and the bias ($b^{(l,i)}$ is computed from $\alpha_n^{(l,i)}$ via Eq. (7)), this procedure effectively results in one classifier with an effective set of coefficients $\alpha_n^{(l)} = \sum_i \alpha_n^{(l,i)}/20$ and an effective bias $b^{(l)} = \sum_i b^{(l,i)}/20$.

The second step is to average over the $L = 50$ slices. Note, however, that the data points $(\mathbf{x}_n^{(l)}, t_n^{(l)}) \in D^{(\text{train},l)}$ are now different for each l. The full decision function is

$$F(\mathbf{x}) = \frac{1}{L} \sum_{nl} \alpha_n^{(l)} t_n^{(l)} k(\mathbf{x}_n^{(l)}, \mathbf{x}) + b, \tag{16}$$

where $b = \sum_l b^{(l)}/L$. As before, a decision for the class label of a point $\mathbf{x}$ is obtained through $\tilde{t} = \text{sign}(F(\mathbf{x}))$. We use this decision function to evaluate the metrics discussed in Section 2.3 for the test data $D^{(\text{test})}$ using the procedure illustrated in Fig. 2.

Note that in [8], instead of separating the training data into 50 disjoint subsets (each containing 2% of the data), an approach similar to bagging (bootstrap aggregating) [52] was used. In that approach, 50 subsets are constructed by drawing 2% of the training data *with replacement*. We also tested this bagging inspired approach (data not shown) and found that, although the results were similar, the fluctuations were much larger. This makes sense because drawing with replacement means that different subsets can share the same data points and also include a single point more than once. Consequently, one may expect that some points are not included in any of the datasets. In fact, the probability that a certain $x \in D^{(\text{train})}$ is not included in any of the $D^{(\text{train},l)}$ is $(1 - 1/N)^N \approx 36.8\%$ for $N = |D^{(\text{train})}| \approx 1500$. Apart from this counting argument, the general observation in [52] was that bagging is better suited for *unstable* classification algorithms, whereas SVMs

are stable. We therefore conclude that splitting the training data in disjoint, equally-sized subsets is superior.

As before, it is interesting to compare the results from the combined classifier with results from applying cSVM to the same data points and parameters. Note that Eq. (16) also applies to cSVM, but that $\alpha_n^{(l)}$ comes directly from the global minimum to Eqs. (3)–(5) and not from an average of the twenty best solutions produced by DW2000Q. The results for each dataset are shown in Fig. 5, where the mean and the standard deviation have been obtained from ten repetitions of the test procedure as sketched in Fig. 2.

Based on the resulting accuracy shown in Fig. 5(c), one could conclude that cSVM outperforms qSVM (especially for the dataset Max80 for which we studied one of the contributing classifiers in Fig. 3). However, from the metrics AUROC and AUPRC reported in Fig. 5(a) and (b), we find that the resulting classifiers from the QA version are in fact superior. This hints at a problem in the construction of the final decision function given in Eq. (16), which would have been overlooked if the accuracy had not been evaluated: Recall that AUROC and AUPRC are generated by sweeping the bias b in Eq. (16) to move the decision boundary through the feature space $\mathbb{R}^{40}$ from a full negative predictor to a fully positive predictor (see Section 2.3). If AUROC and AUPRC are better for qSVM, this means that the bias b has been chosen suboptimally and there must be some bias b^* for which the classifier produces better results.

The reason for this is that Eq. (7) from the original SVM may not be suited to obtain the optimal bias for the QA version of the SVM defined by Eq. (11). The condition for an optimal bias is the constraint Eq. (5), included through the multiplier ξ in Eq. (11). Since $\xi = 0$ for Max80 (cf. Table 1), this explains the particularly bad accuracy for this dataset despite better AUROC and AUPRC (see also the discussion under point 6 of the motivations given in Section 2.2).

We correct for the suboptimal bias by replacing b with the value of b^* for which the classification accuracy for the training data $D^{(\text{train})}$ is maximal. Note that this step does not require a new training of qSVM. It is a scan of a single parameter that can be done "offline", i.e., after obtaining the coefficients $\{\alpha_n^{(l)}\}$. As such, this step is simple and efficient and could, in principle, be directly added to the data handling scheme presented in Fig. 2. See Appendix A for more information.

The classification accuracy of qSVM after adjusting the bias for each dataset is shown in Fig. 6. It clearly improves the results for the linear kernel ($\gamma = -1$) with high class imbalance (Mad80, Max80, and Myc80). We also observe that the Gaussian kernel used for Mad50, Max50, and Myc50 was not affected as strongly by the suboptimal bias. As changing the bias of the decision function given in Eq. (16) does not affect AUROC and AUPRC, the results shown in Fig. 5(a) and (b) also apply to the adjusted version of qSVM.

To summarize, we observe a better or comparative performance of qSVM compared to cSVM for all datasets, as measured by AUROC, AUPRC, and classification accuracy. For completeness, the numerical results of the test are given in Table 2 in Appendix B.

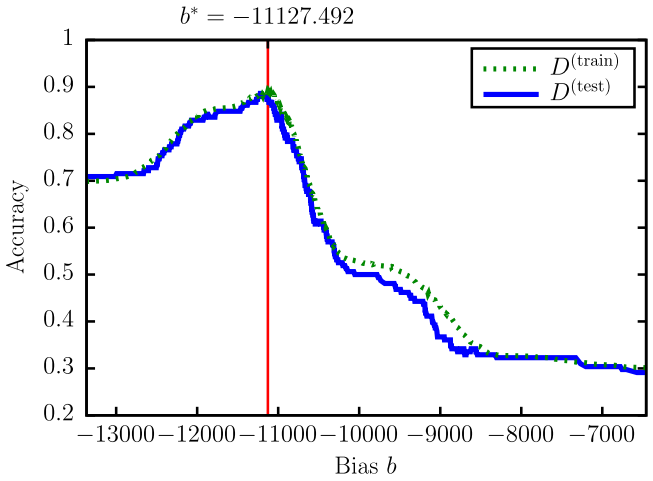

Fig. 7. Classification accuracy for the training data $D^{(\text{train})}$ (dotted green line) and the test data $D^{(\text{test})}$ (solid blue line) of the dataset Myc70 as a function of the bias b in the decision function $F(\mathbf{x})$ given in Eq. (16). The bias b^* is chosen to be optimal for the training data. The optimal bias for the test data (i.e. the peak of the solid blue line) is slightly smaller. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4. Conclusion

In this paper, we introduced and studied the implementation of kernel-based SVMs on a DW2000Q quantum annealer [23]. We found that the optimization problem behind the training of SVMs can be straightforwardly expressed as a QUBO and solved on a quantum annealer. The QUBO form exhibits certain mathematical advantages, such as its ability to produce exact zeros or the inherent inclusion of the box constraint. Each run of the training process on the quantum annealer yields a distribution of different classifiers that can later be used to classify arbitrarily many test data points.

Our results show that the ensemble of classifiers produced by the quantum annealer often surpasses the single classifier obtained by the classical SVM for the same computational problem as measured by AUROC, AUPRC, and accuracy. The advantage stems from the fact that the DW2000Q produces not just the global optimum for the training data, but a distribution of many reasonably good, close-to-optimal solutions to the given optimization problem. A combination of these has the potential to generalize better to the test data. This observation is in line with findings in other machine learning problems studied on a quantum annealer [8,24].

Therefore we conclude that the QA version of the SVM is a useful practical alternative to the classical SVM. If the capabilities of future quantum annealers continue to scale at the current pace, training SVMs on quantum annealers may become a valuable tool for classification problems, and can already be helpful for hard problems where only little training data is available.

An interesting project for future research would be to examine other approaches to building strong classifiers by constructing weighted sums of the class predictions from several SVMs as done in boosting methods like AdaBoost or QBoost [3,10,53,54]. It would also be valuable to examine how the QA results for SVMs can be further improved using advanced features offered by the DW2000Q like reverse annealing, spin-reversal transforms, special annealing schedules, or enhanced embeddings [23,45,46]. Furthermore, since SVMs can also be used for multi-class classification and regression tasks [9], it seems worthwhile to study corresponding applications to such problems using the QA formulation presented here. Finally, it would be a potentially interesting avenue to explore if suitable modifications to the original SVM can lead to an equally good distribution of solutions as the one produced by the quantum annealer.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table 2
Calibration and test results for all SVMs. The reported metrics are the mean area under the ROC curve, and the mean area under the Precision–Recall curve (see Section 2.3), and the mean classification accuracy. The parameters of the QA version of the SVM are qSVM(B, K, ξ, γ) where B is the encoding base, K is the number of qubits per coefficient α_n, ξ is a Lagrangian multiplier, and γ is the kernel parameter. The corresponding version of the classical SVM is cSVM(C, γ) where C is given by Eq. (14).

Dataset	SVM Parameters	Calibration			Testing		
		AUROC	AUPRC	Accuracy	AUROC	AUPRC	Accuracy
Mad50	qSVM(2,3,5,0.125)	0.71	0.71	0.63	0.88	0.92	0.81
	cSVM(7,0.125)	0.73	0.73	0.60	0.89	0.92	0.82
Max50	qSVM(2,3,5,0.125)	0.73	0.74	0.64	0.94	0.95	0.87
	cSVM(7,0.125)	0.73	0.74	0.63	0.94	0.95	0.85
Myc50	qSVM(2,2,0,0.125)	0.68	0.68	0.61	0.92	0.94	0.84
	cSVM(3,0.125)	0.69	0.70	0.58	0.92	0.94	0.85
Mad70	qSVM(10,3,5,−1)	0.75	0.58	0.65	0.92	0.91	0.87
	cSVM(111,−1)	0.70	0.47	0.67	0.90	0.88	0.85
Max70	qSVM(10,3,5,−1)	0.82	0.68	0.69	0.93	0.89	0.86
	cSVM(111,−1)	0.75	0.57	0.70	0.90	0.85	0.85
Myc70	qSVM(10,3,5,−1)	0.72	0.57	0.63	0.86	0.82	0.87
	cSVM(111,−1)	0.72	0.51	0.66	0.83	0.76	0.82
Mad80	qSVM(10,3,5,−1)	0.85	0.66	0.69	0.93	0.86	0.93
	cSVM(111,−1)	0.78	0.50	0.78	0.90	0.82	0.90
Max80	qSVM(10,3,0,−1)	0.85	0.62	0.67	0.95	0.94	0.95
	cSVM(111,−1)	0.78	0.47	0.77	0.94	0.92	0.91
Myc80	qSVM(10,3,5,−1)	0.73	0.48	0.60	0.94	0.91	0.93
	cSVM(111,−1)	0.71	0.37	0.71	0.93	0.89	0.89

Acknowledgments

We would like to thank Richard Li and Daniel Lidar for providing preprocessed data from TF-DNA binding experiments. We are grateful to Seiji Miyashita for helpful discussions. Access and compute time on the D-Wave machine located at the headquarters of D-Wave Systems Inc. in Burnaby (Canada) were provided by D-Wave Systems Inc. D.W. is supported by the Initiative and Networking Fund of the Helmholtz Association, Germany through the Strategic Future Field of Research project "Scalable solid state quantum computing (ZT-0013)".

Appendix A. Adjusting the bias in qsvm

The choice for the bias b given in Eq. (7) as a function of the coefficients $\{\alpha_n\}$ is based on the condition that the coefficients are the global minimum $\{\alpha_n\}^*$ of the QP problem given in Eqs. (3)–(5). In fact, it is the constraint given in Eq. (5) that identifies an optimal bias b [15].

However, for qSVM, a new classifier is generated by combining some of the lowest-energy solutions produced by the quantum annealer, which is in general not equal to $\{\alpha_n\}^*$. Moreover, the constraint for an optimal bias given in Eq. (5) is included through the multiplier ξ in Eq. (11), so it may not be satisfied for all solutions produced by the quantum annealer. Therefore, it can happen that the bias from Eq. (7) is not suitable for qSVM. This is what happened to the rightmost three datasets shown in Fig. 5 (especially for Max80 where $\xi = 0$, see Table 1). This problem only affects the actual accuracy and not the more robust metrics AUROC and AUPRC (see Section 2.3).

Since the bias is only a single parameter, this problem can easily be solved by replacing b with another bias b^*, for which the accuracy for the training data $D^{(\text{train})}$ is highest. Note that this step does not require a new training of qSVM. The result of the training is given by the set of coefficients $\{\alpha_n\}$, which enter the decision function in Eq. (6). The bias b can be adjusted "offline", i.e., independently of the $\{\alpha_n\}$.

Note that it is only allowed to use the training data $D^{(\text{train})}$ for adjusting the bias, and not the test data $D^{(\text{test})}$. Modifying a classifier as a function of its performance on the test data $D^{(\text{test})}$ would invalidate the statement that the classifier can generalize well to unseen data.

An example of such a scan of the bias b is shown in Fig. 7 for the dataset Myc70. It was taken from one out of ten repetitions of the test procedure (see Fig. 2). The classifier has been obtained from an average over 1000 decision functions (20 lowest-energy samples times 50 slices of the training data). One can see that the peak of the accuracy for $D^{(\text{train})}$ (dotted line) is close but not equal to the peak of the accuracy for $D^{(\text{test})}$ (solid line).

Appendix B. Calibration and test results

In Table 2, we list the numerical results for the calibration and the test phase for the application of cSVM and qSVM to the computational biology problem.

For the calibration phase, where 2% of the data was used for training, qSVM often produces stronger or equally strong classifiers. In the testing phase, where the classifiers for each of the 50 disjoint subsets of the training data were combined, qSVM almost always surpasses cSVM in all of the three metrics.

References

[1] H. Neven, V.S. Denchev, G. Rose, W.G. Macready, 2008. arXiv:0811.0416.
[2] K.L. Pudenz, D.A. Lidar, Quantum Inf. Process. 12 (2012) 2027.
[3] H. Neven, V.S. Denchev, G. Rose, W.G. Macready, in: S.C.H. Hoi, W. Buntine (Eds.), Proceedings of the Asian Conference on Machine Learning, in: Proceedings of Machine Learning Research (PMLR), vol. 25, Singapore Management University, Singapore, 2012, pp. 333–348.
[4] S.H. Adachi, M.P. Henderson, 2015. arXiv:1510.06356.
[5] T.E. Potok, C. Schuman, S. Young, R. Patton, F. Spedalieri, J. Liu, K.-T. Yao, G. Rose, G. Chakma, J. Emerg. Technol. Comput. Syst. 14 (2018) 19:1.
[6] D. O'Malley, V.V. Vesselinov, B.S. Alexandrov, L.B. Alexandrov, PLoS One 13 (2018) 1.
[7] D. Ottaviani, A. Amendola, 2018. arXiv:1808.08721.
[8] R.Y. Li, R. Di Felice, R. Rohs, D.A. Lidar, npj Quantum Inf. 4 (2018) 14.
[9] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, USA, 2001.
[10] C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag, Berlin, Heidelberg, 2006.
[11] R.-H. Li, G.G. Belford, Instability of Decision Tree Classification Algorithms Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02, ACM, New York, NY, USA, 2002, pp. 570–575.
[12] X. Yuan, X. Yuan, F. Yang, J. Peng, B.P. Buckles, FLAIRS Conference, 2003.
[13] H. Xu, C. Caramanis, S. Mannor, J. Mach. Learn. Res. 10 (2009) 1485.
[14] E. Raczko, B. Zagajewski, Eur. J. Remote Sens. 50 (2017) 144.
[15] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes 3rd Edition: The Art of Scientific Computing, Cambridge University Press, New York, USA, 2007.
[16] Y. Tang, Proceedings of the International Conference on MachineLearning (ICML) Workshops, 2013.

[17] S. Kim, S. Kavuri, M. Lee, in: M. Lee, A. Hirose, Z.-G. Hou, R.M. Kil (Eds.), Neural Information Processing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 458–465.
[18] M. Lazri, S. Ameur, Atmos. Res. 203 (2018) 118.
[19] M. Zareapoor, P. Shamsolmoali, D.K. Jain, H. Wang, J. Yang, Pattern Recognit. Lett. 115 (2018) 4, multimodal Fusion for Pattern Recognition.
[20] R. Harris, M.W. Johnson, T. Lanting, A.J. Berkley, J. Johansson, P. Bunyk, E. Tolkacheva, E. Ladizinsky, N. Ladizinsky, T. Oh, F. Cioata, I. Perminov, P. Spear, C. Enderud, C. Rich, S. Uchaikin, M.C. Thom, E.M. Chapple, J. Wang, B. Wilson, M.H.S. Amin, N. Dickson, K. Karimi, B. Macready, C.J.S. Truncik, G. Rose, Phys. Rev. B 82 (2010) 024511.
[21] M.W. Johnson, M.H.S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A.J. Berkley, J. Johansson, P. Bunyk, E.M. Chapple, C. Enderud, J.P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M.C. Thom, E. Tolkacheva, C.J.S. Truncik, S. Uchaikin, J. Wang, B. Wilson, G. Rose, Nature 473 (2011) 194.
[22] P.I. Bunyk, E.M. Hoskinson, M.W. Johnson, E. Tolkacheva, F. Altomare, A.J. Berkley, R. Harris, J.P. Hilton, T. Lanting, A.J. Przybysz, J. Whittaker, IEEE Trans. Appl. Supercond. 24 (2014) 1.
[23] D-Wave Systems Inc, Technical Description of the D-Wave Quantum Processing Unit, Tech. Rep., D-Wave Systems Inc., Burnaby, BC, Canada, 2018, D-Wave User Manual 09-1109A-M.
[24] A. Mott, J. Job, J.-R. Vlimant, D. Lidar, M. Spiropulu, Nature 550 (2017) 375.
[25] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition, Cambridge University Press, New York, 2011.
[26] E. Grumbling, M. Horowitz (Eds.), Quantum Computing: Progress and Prospects, The National Academies Press, Washington, DC, 2018.
[27] P. Rebentrost, M. Mohseni, S. Lloyd, Phys. Rev. Lett. 113 (2014) 130503.
[28] Z. Li, X. Liu, N. Xu, J. Du, Phys. Rev. Lett. 114 (2015) 140504.
[29] E. Ising, Z. Phys. 31 (1925) 253.
[30] F. Barahona, J. Phys. A: Math. Gen. 15 (1982) 3241.
[31] C.J. Burges, Data Min. Knowl. Discov. 2 (1998) 121.
[32] C.-W. Hsu, C.-C. Chang, C.-J. Lin, A Practical Guide to Support Vector Classification, Tech. Rep., Department of Computer Science, National Taiwan University, 2003, http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.
[33] H.-T. Lin, C.-J. Lin, A Study on Sigmoid Kernels for SVM and the Training of Non-PSD Kernels By SMO-Type Methods, Tech. Rep., Department of ComputerScience, National Taiwan University, 2003, http://www.csie.ntu.edu.tw/~cjlin/papers/tanh.pdf.
[34] S.S. Keerthi, C.-J. Lin, Neural Comput. 15 (2003) 1667, http://dx.doi.org/10.1162/089976603321891855.
[35] C.-C. Chang, C.-J. Lin, ACM Trans. Intell. Syst. Technol. (TIST) 2 (2011) 27:1, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, J. Mach. Learn. Res. 12 (2011) 2825.
[37] M.S. Andersen, J. Dahl, L. Vandenberghe, 2018. https://cvxopt.org, version 1.2.2.
[38] IEEE Std 754-2008, Microprocessor Standards Committee of the IEEE Computer Society, 3 Park Avenue, New York, NY, USA, 2008, 10016-5997.
[39] B. Lesser, M. Mü, W.N. Gansterer, Proced. Comput. Sci. 4 (2011) 508, Proceedings of the International Conference on Computational Science, ICCS 2011.
[40] O.L. Mangasarian, D.R. Musicant, IEEE Trans. Neural Netw. 10 (1999) 1032.
[41] O.L. Mangasarian, D.R. Musicant, in: O.L. Mangasarian, J.-S. Pang (Eds.), Complementarity: Applications, Algorithms and Extensions, Springer US, Boston, MA, 2001, pp. 233–251.
[42] V. Choi, Quantum Inf. Process. 7 (2008) 193.
[43] J. Cai, W.G. Macready, A. Roy, 2014. arXiv:1406.2741.
[44] D-Wave Systems Inc, 2018. https://github.com/dwavesystems/dwave-ocean-sdk, release 1.2.0.
[45] M. Ohkuwa, H. Nishimori, D.A. Lidar, Phys. Rev. A 98 (2018) 022314.
[46] S. Boixo, T. Albash, F.M. Spedalieri, N. Chancellor, D.A. Lidar, Nature Commun. 4 (2013) 2067.
[47] F.J. Provost, T. Fawcett, R. Kohavi, Proceedings of the Fifteenth International Conference on Machine Learning, ICML '98, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998, pp. 445–453.
[48] C. Cortes, M. Mohri, Proceedings of the 16th International Conference on Neural Information Processing Systems, NIPS'03, MIT Press, Cambridge, MA, USA, 2003, pp. 313–320.
[49] J. Davis, M. Goadrich, Proceedings of the 23rd International Conference on Machine Learning, ICML '06, ACM, New York, NY, USA, 2006, pp. 233–240.
[50] T. Zhou, N. Shen, L. Yang, N. Abe, J. Horton, R.S. Mann, H.J. Bussemaker, R. Gordân, R. Rohs, Proc. Natl. Acad. Sci. USA 112 (2015) 4654.
[51] L. Yang, Y. Orenstein, A. Jolma, Y. Yin, J. Taipale, R. Shamir, R. Rohs, Mol. Syst. Biol. 13 (2017) 910.
[52] L. Breiman, Mach. Learn. 24 (1996) 123.
[53] Y. Freund, R.E. Schapire, Proceedings of the Thirteenth International Conference on Machine Learning (ICML 1996), ICML'96, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004, pp. 148–156.
[54] J. Friedman, T. Hastie, R. Tibshirani, Ann. Statist. 38 (2000) 337.