Quantum Sparse Support Vector Machines

Seyran Saeedi* and Tom Arodz†

Department of Computer Science, Virginia Commonwealth University
Richmond, VA, USA

Abstract

We analyze the computational complexity of Quantum Sparse Support Vector Machine, a linear classifier that minimizes the hinge loss and the L_1 norm of the feature weights vector and relies on a quantum linear programming solver instead of a classical solver. Sparse SVM leads to sparse models that use only a small fraction of the input features in making decisions, and is especially useful when the total number of features, p, approaches or exceeds the number of training samples, m. We prove a $\Omega(m)$ worst-case lower bound for computational complexity of any quantum training algorithm relying on black-box access to training samples; quantum sparse SVM has at least linear worst-case complexity. However, we prove that there are realistic scenarios in which a sparse linear classifier is expected to have high accuracy, and can be trained in sublinear time in terms of both the number of training samples and the number of features.

1   Introduction

Steadily increasing ability to measure large number of features in large number of samples leads to ongoing interest in fast methods for solving large-scale classification problems. One of the approaches being explored is training the predictive model using a quantum algorithm that has access to the training set stored in quantum-accessible memory. In parallel to research on efficient architectures for quantum memory [1, 2, 3], work on quantum machine learning algorithms and on quantum learning theory is under way (see [4, 5, 6, 7] and [8] for reviews and perspectives), and is attracting interest form the machine learning community [9, 10, 11, 12, 13].

In order to achieve quantum speedup, the choice of the variant of the machine learning problem often matters. For example, the pioneering quantum machine learning method – quantum Least-Squares Support Vector Machine (LS-SVM) [14] – focused on quadratic loss and quadratic regularizer, in order to be able to utilize solvers for quantum linear systems of equations, such as HHL [15]. This and other recent solvers based on quantum manipulation of eigenvalues [16, 17] can lead to exponential speedup compared to classical methods, as long as the eigenvalues of the linear kernel matrix are of similar magnitude. However, linear systems arise only from unconstrained, or equality-constrained quadratic problems, such as LS-SVM, but not from the standard, hinge-loss SVM. A more recent quantum supervised learning method [10] that can achieve sublinear training time involves efficient quantum primal-dual approach for solving minimax problems, and as a consequence focuses on minimizing the maximum – not the average – loss over the training set. For the hinge loss used in the original SVM, a quantum variant of SVM-perf algorithm that transforms samples from input space to kernel space and then uses quantum methods to approximate inner products in the kernel space, has been proposed recently [18], achieving complexity linear in the number of samples.

Here, we focus on the possibility of achieving quantum speedup for the Sparse SVM [19, 20, 21, 22], a linear classifier that combines the standard SVM hinge loss with the sparsity-promoting L_1 regularizer[1]

---

*Current address: Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, USA. e-mail: seyran@ucsb.edu

†Corresponding author. e-mail: tarodz@vcu.edu

[1]See Section 2 for background on binary classification and regularized models.

instead of the standard L_2 regularizer. Sparse SVM is useful in cases where the number of features approaches or exceeds the number of training samples, and where interpretability of the classifier matters, not just its predictive abilities. For example, in biomedicine it is important to know if, and how much, each feature contributes to the prediction, and often a small number of features is enough to tell the classes apart. Thus, linear models $h(x; \beta) = \beta^T x$ with weights $\beta, x \in \mathbb{R}^p$, which learn a single multiplicative weight β_j for each feature x^j of a p-dimensional sample x, are often preferred over non-linear approaches, such as kernel methods, ensembles, or neural networks. In these settings, we often expect that highly-accurate predictions can be made using just a few discriminative features; we expect that a well-performing model should be sparse, that is, there is a vector β composed mostly of zeros that achieves near-optimal accuracy. The remaining features either carry no information about the separation of classes, or the information is redundant.

1.1 Our Contribution

We propose Quantum Sparse SVM (QsSVM), an approach that minimizes the training-set objective function of the Sparse SVM model [19, 20, 21, 22] using a quantum algorithm for solving linear programming (LP) problems [23] instead of a classical solver. Our aim is to analyze whether using a quantum solver provides benefits in terms of computational complexity of model training, that is, if sublinear training time in terms of the number of training samples, m, and the number of features, p, be achieved. This problem is challenging, since quantum LP solver [23] and similar quantum SDP/LP solvers [24, 25, 26, 27] express the complexity not only in terms of m and p, but also in terms of characteristics of the primal and dual solution to the LP/SDP instance being solved (see Figure 1). So far, realistic application scenarios with characteristics that provably lead to quantum speedup for these solvers have been scarce.

We show that for arbitrary binary classification problems no quantum speedup is achieved using quantum LP solvers. More broadly, we prove that the lower bound for solving Sparse SVM using any quantum algorithm with black-box access to training data is $\Omega(m)$. However, we show there are realistic cases in which a sparse linear model will have high accuracy, and sublinear time in m and p can be guaranteed when using the quantum solver, while proving that any classical algorithm requires at least linear time.

2 Background on Binary Classification and Sparse Linear Models

Binary classification involves vector-scalar pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y} = \{-1, 1\}$ and $\mathcal{X} \subset \mathbb{R}^p$ is a compact subset of a p-dimensional feature space. Each pair describes an object of study, for example a brain scan or a tissue sample of a medical patient. Individual components x^j of a vector x are called
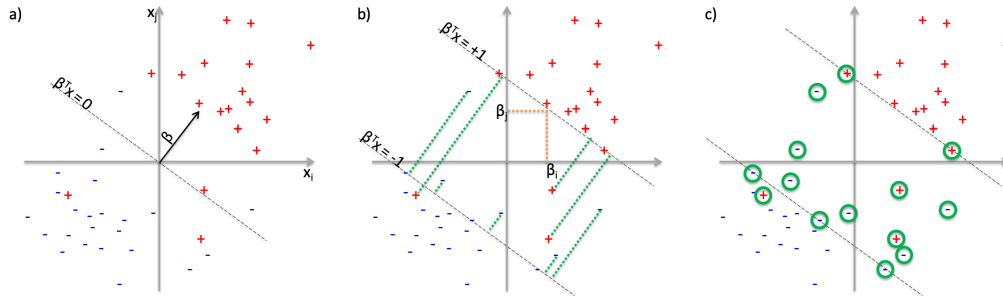

Figure 1: Graphical illustration of the factors contributing to the computational complexity of Quantum Sparse SVM for a classification problem with linear solution vector β (**a**). Complexity is proportional to the product Rr, where R is the sum of sample loss values (**b**: green lines) and feature weight magnitudes (**b**: orange lines), and r is the sum of weights of support vectors (**c**: green circles).

features. Each feature describes some numerical property of the object represented by x, for example signal intensity in a single voxel of a brain scan, or expression level of a single gene. The value of y tells us whether the object belongs to the positive or the negative class. In many scenarios, the feature vectors are easy to obtain, but the class variable is not. For example, we can measure the methylation status of each CpG base-pair in the patient's genome relatively easily, but deciding if the patient's prognosis is positive or negative is challenging.

In statistical learning [28], we assume that samples (x, y) come from a fixed but unknown distribution D over $\mathcal{X} \times \mathcal{Y}$. For a given feature vector x, the probabilities of either class are given by conditional distribution $D_{y|x}$ over $\mathcal{Y}$, and for a given class y, the probability density of feature vectors in that class is given by conditional distribution $D_{x|y}$ over $\mathcal{X}$. While the underlying distributions D, $D_{y|x}$, and $D_{x|y}$ are unknown, we have access to a training set Z consisting of m samples $z_i = (y_i, x_i)$ drawn independently from D. In the binary classification problem the goal is to use the training set to learn how to predict classes y for feature vectors x that are not in the training set.

The training set can be used to construct a predictive model, in a form of a hypothesis function $h : \mathcal{X} \to \mathbb{R}$, where the sign of $h(x)$ indicates the predicted class for input feature vector x. For a given sample (x, y), the prediction is considered correct if the signs of the predicted and the true class agree, that is, if $yh(x) > 0$. The predictive model should make as few errors as possible over samples $z = (x, y)$ sampled from distribution D, that is, it should minimize $\int_{\mathcal{X} \times \mathcal{Y}} I[yh(x) \leq 0]D(z)\,\mathrm{d}z$, where I is an indicator function over Boolean domain returning 1 for true and 0 for false.

A simple but often effective class of hypotheses is the class of linear functions $h(x; \beta, b) = \beta^T x + b = \sum_{j=1}^{p} \beta_j x^j + b$. A linear predictive model is parameterized by a vector of feature weights $\beta \in \mathbb{R}^p$ and a bias term $b \in \mathbb{R}$. To simplify the notation, we often add one more dimension to $\mathcal{X}$ with all samples having a value of one. The predictive model is then simply $h(x; \beta) = \beta^T x$, $\beta \in \mathbb{R}^{p+1}$, with β_{p+1} playing the role of bias.

Training of a linear model involves finding a suitable parameter vector β. For a single sample (x, y), the suitability of a model h with specific β will be captured by a loss function $\ell(y, h(x; \beta))$, which returns a nonnegative real number that we interpret as a measure of our dissatisfaction with the prediction $h(x; \beta)$. The natural *0/1 loss*, defined as $\ell(y, h(x; \beta)) = I[y\beta^T x \leq 0]$, that is, null for a correct prediction and one otherwise, is not a continuous function of the parameter vector β, and is flat almost everywhere, leading to problems with finding β that minimizes the loss. Instead of the 0/1 loss, a convex function that upper-bounds it is often used in training classification models. For example, the least-square loss $\ell(y, h) = (y - h)^2$ is used in Fisher's Linear Discriminant and in Least-Squares Support Vector Machine (LS-SVM) classifier [29]. However, the least-squares loss, while often used in regression problems, leads to high-magnitude loss if $h(x)$ has large magnitude, even if $h(x)$ and y agree on the sign, that is, the prediction is correct. Most classification loss functions involve a nonincreasing $\mathbb{R} \to \mathbb{R}_+$ function of the product yh. If the sign of prediction $h(x)$ and the target class y agree, then loss should not increase if $|h|$ increases. One prominent example of a convex, monotonic loss is the *hinge loss*, defined as $\ell(y, h) = [1 - yh]_+ = \max(0, 1 - yh)$, which is used in the original variant of the Support Vector Machine (SVM) classifier [30]. Hinge loss leads to hinge risk $\hat{R}_{SVM}(\beta) = \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_i \beta^T x_i)$.

Once the loss function is chosen, the goal of training a model is to find the parameter vector β that minimizes the expected loss $\mathbb{E}_{z \sim D}\ell(y, h(x; \beta))$, referred to as *risk* of the model, $R(\beta) = \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, h(x; \beta))D(z)\,\mathrm{d}z$. Since D is unknown, a surrogate goal is to search for β that leads to low loss on samples from the training set. For example, the *empirical risk minimization* strategy involves finding parameters β that minimize *empirical risk*, that is, the average loss on the training set, $\hat{R}(\beta) = \frac{1}{m} \sum_{i=1}^{m} \ell(y_i, h(x_i; \beta))$.

The model β that minimizes the empirical risk may have high generalization risk $R(\beta)$, that is, may fare poorly on samples outside of the training set, especially if the features are not statistically independent, or if the number of features p is of the same order or higher than the number of training samples, m. Often, the generalization error can be reduced if a penalty on the complexity of the model is introduced into the optimization problem. Typically, this penalty term, known as regularization term, is based on $\|\beta\|$, a norm of the vector of model parameters, leading to *regularized empirical risk minimization* strategy, which finds parameters that minimize $\hat{L}(\beta) = \hat{R}(\beta) + \lambda f(\|\beta\|)$. For example, most Support Vector Machines [30] use squared L_2 norm of β, $\|\beta\|_2^2$, as the regularizer.

3

The L_2 regularizer used in LS-SVM penalizes large-magnitude feature weights, but is unlikely to set any feature weights to null. In many real-world scenarios involving classification problems with large number of features we expect that highly-accurate predictions can be made using just a few discriminative features. The remaining features either carry no information about the separation of classes, or the information is redundant. For example, classification problems involving gene expression measured using microarrays or RNA-seq may have tens of thousands of features, and brain scans can have million of voxels, but only a small number may be enough to separate subjects with one subtype of a disease from another subtype, an information that is useful in choosing treatment. In these scenarios, we expect that a well-performing model should be sparse; that there is a vector β composed mostly of zeros that achieves near-optimal risk $R(\beta)$. The key problem is to decide which feature weights should be non-zero.

To find sparse solutions to classification problems, a regularization term in the form of L_1 norm of β is often included in the objective function. L_1 regularization is especially useful when working with a training set with large number of features compared to the number of training samples, which is referred to as the $p > m$ case. One method that uses L_1 regularization is Sparse SVM (sSVM) [19, 20, 22], a linear classifier based on regularized empirical risk minimization involving hinge loss and L_1 regularizer, $\hat{L}_{sSVM}(\beta) = \hat{R}_{SVM}(\beta) + \lambda\|\beta\|_1$, where $\lambda > 0$ is a hyperparameter specifying the strength of regularization. Training of a Sparse SVM model can be transformed into an optimization problem with linear objective function and linear inequality constraints.

3   Training Sparse SVM using Quantum LP Solvers

The training of Sparse SVM model using a training set $\{(x_i, y_i)\}$ with p features and m samples and a regularization constant $\lambda > 0$ involves solving a minimization problem

$$\arg\min_{\beta \in \mathbb{R}^p} \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_i \beta^T x_i) + \lambda \sum_{j=1}^{p} |\beta_j|. \tag{1}$$

Using standard techniques, for $\lambda > 0$, this non-linear unconstrained optimization problem can be transformed to an equivalent primal constrained linear program with $n = m + 2p$ nonnegative variables and m linear inequality constraints

$$\min_{\xi, \beta^+, \beta^-} \quad \frac{1}{m} \sum_{i=1}^{m} \xi_i + \lambda \sum_{j=1}^{p} \beta_j^+ + \lambda \sum_{j=1}^{p} \beta_j^- \tag{2}$$

$$\text{s.t.} \quad \sum_{j=1}^{p} y_i x_i^j \beta_j^+ - \sum_{j=1}^{p} y_i x_i^j \beta_j^- \geq 1 - \xi_i, \quad i \in [m]$$

$$\xi_i, \beta_j^+, \beta_j^- \geq 0,$$

where $[m] = \{1, ..., m\}$. We can read out the solution as $\beta_j = \beta_j^+ - \beta_j^-$. We also have $|\beta_j| = \beta_j^+ + \beta_j^-$. The value of the hinge loss of i-th training sample is equal to ξ_i. The dual problem is

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i \quad \text{s.t.} \quad -\lambda \leq \sum_{i} \alpha_i y_i x_i^j \leq \lambda, \quad j \in [p]$$

$$0 \leq \alpha_i \leq 1/m \quad i \in [m]. \tag{3}$$

A classification problem with m samples in p-dimensional feature space leads to a primal LP with $n = m + 2p$ nonnegative variables and m linear inequality constraints.

3.1   Linear Programs and Matrix Games

To solve the linear program associated with Sparse SVM, we employ a recently proposed quantum LP solver [23] based on zero-sum games. The solver is based on a series of zero-sum matrix games. For

a given LP, each game in the series is constructed to indicate if the optimal value of the LP lies in a certain numerical range.

Each zero-sum matrix game involves two players, Alice with n possible strategies and Bob with m possible strategies. If Alice chooses strategy $i \in [n]$ and Bob chooses $j \in [m]$, the pay-off to Alice is M_{ij} and to Bob is $-M_{ij}$, where M is the matrix defining the game. Given a randomized strategy for Alice, sampled from $u \in \mathbb{R}^n_+$, with $||u|| = 1$, and a randomized strategy for Bob, sampled from $v \in \mathbb{R}^m_+$, $||v|| = 1$, the expected payoff to Alice is $u^T M v$. Bob can assume optimal strategy distribution u on Alice's part and chose his strategy distribution v according to $\min_v \max_u u^T M v$ to minimize Alice's and maximize his own expected pay-off.

For a given linear program $\text{LP}(c, A, b) = \min c^T x \text{ s.t.} Ax \leq b, \ x \geq 0$ and a real value α, finding the expected pay-off for the optimal strategy in a game defined by a matrix M that contains α as an element and A, b, c as sub-blocks can be used to determine if the optimal value of the LP is lower than α. If it exists, a feasible LP solution with optimal value below α can also be obtained from the optimal strategy vectors u and v. Iteratively, this process can solve the LP up to a pre-determined error.

A classical algorithm [31] can solve the game up to ε additive error in $\tilde{O}\left((n+m)/\varepsilon^2\right)$ time, where $\tilde{O}$ notation hides logarithmic factors. The quantum version replaces Gibbs sampling step in the classical algorithm with its quantum counterpart, achieving quadratic speedup in terms of n, m, and a $1/\varepsilon^3$ dependence on the desired additive error ε of the solution. A query about the optimal value of an LP with m variables and n constraints being in a certain range reduces to a game with $(n+3) \times (m+2)$ matrix M. Further, assuming that the LP primal and dual solution vectors are bounded in L_1 norm by R and r, respectively, bisection argument shows that only a logarithmic number of such range queries – matrix games – are needed, each solved to within $R(r+1)/\varepsilon$ error, in order to obtain the solution to the LP with at most ε additive error. Together, the matrix game-based quantum solver outlined above [23] has $\tilde{O}\left(\left(\sqrt{m} + \sqrt{n}\right)\left(R(r+1)/\varepsilon\right)^3\right)$ complexity of obtaining the solution to the LP to within ε additive error, where $\tilde{O}$ notation hides logarithmic factors. The solver achieves a polynomial speedup compared to classical methods, and lower exponent in dependence on R, r, and $1/\varepsilon$ than in previous quantum methods [24, 25, 26, 27]. The key question is whether the dependence on R and r, the L_1 norms of the primal and dual LP solutions, respectively, allows for achieving sublinear time complexity for linear programs corresponding to training Sparse SVMs.

3.2 Data Access Model

For a linear program $\text{LP}(c, A, b) = \min c^T x \text{ s.t.} Ax \leq b, \ x \geq 0$, the quantum LP solver requires read-only access to the vectors and matrices (c, A, b) defining the LP, and read/write access to internal data. The solver assumes access to data using a quantum oracle implemented using quantum random access memory (QRAM). For example, the access to element A_{ij} of the LP constraint matrix A is given by an oracle associated with A, a unitary linear operator O_A capable of performing the mapping $O_A |i\rangle |j\rangle |z\rangle \rightarrow |i\rangle |j\rangle |z \oplus A_{ij}\rangle$ in superposition. The operator that takes three qubits on input, corresponding to indices i, j and a placeholder z, and output produces i, j and the exclusive alternative ($\oplus$) of the binary representation of the matrix element A_{ij} and the placeholder value of the third qubit, z. The algorithm assumes the oracle can operate in superposition, that is, given a superposition of indices, returns a superposition of array elements

$$O_A |i\rangle \left(\alpha |j\rangle + \beta |j'\rangle\right) |z\rangle \rightarrow \alpha |i\rangle |j\rangle |z \oplus A_{ij}\rangle + \beta |i\rangle |j'\rangle |z \oplus A_{ij'}\rangle,$$

and similarly for the first index. The computational complexity of the algorithm is measured with respect to number of oracle calls and the number of two-qubit quantum gates required for further processing.

Availability of quantum random access memory is a typical assumption in quantum algorithms, including recent quantum machine learning methods [10, 9, 11]. Workable, large-scale QRAM does not exist yet and feasibility of its constructing is still debated, but algorithmic models [32, 33, 2, 34] and experimental demonstrations [35, 3, 36, 37] of quantum memory alone, and as part of quantum networks or quantum learning systems, are emerging. Recent results indicate that loading classical data into quantum RAM can be done in logarithmic time [38, 39], at the cost of introducing small perturbations into the data, which .

In a fully-quantum RAM, operations of both reading and writing in quantum superposition should be available. In quantum machine learning systems, typically the input data, as well as certain intermediate data, is processed classically; for example, we assume the feature values for each sample arrive over some classical information channel. Only the read operation is required to operate in quantum superposition, that is, with parallelism implied in linearity of the data access oracle. This type of access is referred to as quantum-read, classical-write RAM (QCRAM) [40, 41]. The standard model of QCRAM utilizes a tree structures over non-zero elements of an n-dimensional vector x to allow classical readout and update of a single vector entry x_i in $O(\log n)$ time. It also allows for accessing selected elements x_i in quantum superposition, sampling integers $i \in [n]$ according to $x_i/||x||$, and creating the quantum state corresponding to x.

For a linear program $\mathrm{LP}(c, A, b) = \min c^T x$ s.t.$Ax \leq b$, $x \geq 0$, the quantum LP solver requires read-only access to the vectors and matrices (c, A, b) defining the LP, and read/write access to the strategy distribution vectors u, v as it iterates through a sequence of matrix games. The solver assumes access to data via a quantum oracle implemented efficiently using QRAM. Through iterations, the game solution vectors u, v are stored in QCRAM. Solving each game involves an iterative algorithm that result in at most $O\left(1/\varepsilon^2 \log mn\right)$ elements of u, v being non-zero. The QCRAM stores only non-zero elements of u, v, thus, accessing all solution elements classically using sequential tree traversal adds $O\left(1/\varepsilon^2 \log mn\right)$ overhead. For sparse models, with few non-zero feature weights in the optimal solution, access can be even faster.

4   Lower Bound for Complexity of Quantum Sparse SVM

Assuming efficient oracle access to input, the computational complexity of quantum LP solver utilized in Quantum Sparse SVM shows improved dependence on n and m, but polynomial dependence on R and r may erase any gains compared to classical LP solvers.

For any training set, the minimum of the objective function of the SparseSVM optimization problem (eq. 1) is bounded from above by one, since an objective function value of one can be obtain by setting $\beta = 0$, which leads to unit loss for each training sample, and thus unit average loss. For some training sets, one is the minimum of the objective function – for example if training samples come in pairs, $(x, +1)$ and $(x, -1)$. In this case, the norm of the primal solution is $R = \sum_i |\xi_i| + \sum_j |\beta_j| = m$, and the norm of the dual solution is $r = \sum_i |\alpha_i| = 1$, since by eq. (3) and strong duality it is equal to the value of the primal objective function. The quantum solver we use includes $(R(r+1)/\varepsilon)^3$ term in its complexity, and $Rr = O(m)$ erases any speedups compared to classical solvers.

A more realistic case in which we see $R = O(m)$ is a regular XOR problem, for example involving two features and four training samples, $[+1, +1]$ and $[-1, -1]$ with $y = +1$ and $[+1, -1]$, $[-1, +1]$ with $y = -1$. For any $\beta \in \mathbb{R}^2$, if there is a sample with loss $1 - \delta$, there is another sample with loss $1 + \delta$. Thus, sum of ξ_i variables is one for any β, and again $\beta = 0$ is the minimizer of the regularized empirical risk, leading to $Rr = O(m)$ and a $O\left(m^{3.5}\right)$ term in the solver worst-case computational complexity.

The above negative results concern speedup of Sparse SVM utilizing a specific quantum LP solver. It can be shown that sublinear worst-case complexity, in terms of the smaller of m and p, is not possible in general. To provide lower bound on solving Sparse SVM using any quantum algorithm with black-box access to elements of feature vectors x and class variables y, we utilize reduction from majority problem, that is, the problem of finding the majority element in a binary vector f of size n. In the majority problem, vector f has t unity and $s = n - t$ null elements, and the algorithm should return true if $t > s$ or false otherwise. Alternatively, it suffices to return the value of t or s.

Given arbitrary majority problem instance, we show a simple procedure to treat it as the input training set for a Sparse SVM classifier. The procedure introduces no overhead, one call to the training data oracle translates to $\Theta(1)$ computation and at most one call to the underlying majority problem oracle. We prove that solving the Sparse SVM classification problem, for values of the regularization constant λ leading to non-null feature weights vector β, allows us to provide the answer to the underlying instance of the majority problem. The lower bound for computational complexity of the majority problem, assuming black-box quantum oracle access to elements f_i in superposition, is $\Omega(n)$ [42], leading to the lower bound on the complexity of training Sparse SVM.

Theorem 1. *Assuming black-box quantum oracle access to a training set with m samples and p features, the lower bound for training Sparse SVM, for regularization constants λ that do not lead to all-null models $\beta = 0$, has complexity $\Omega(m)$, even if the optimization algorithm returns a suboptimal solution with objective value $\hat{L} = L^* + O(1)$.*

Proof. For given $m \geq 8$, $p \geq 1$, set $n = \lceil m/2 \rceil$; we have $2n \leq m \leq 2n + 1$, and $n \geq 4$. Let $f_{n,t}$ be an arbitrary instance of a majority problem of size n with $1 \leq t \leq n - 1$ ones and $n - t$ nulls. We construct an $m \times p$ training set based on majority problem instance $f_{n,t}$ in the following way.

We start with defining values for the first feature, $j = 1$, for all m samples. For $i \in \{1, ..., n\}$, the samples belong to the negative class, that is, $y_i = -1$. For obtaining their feature values $x_i^{j=1}$, we rely on the underlying majority problem. We set $x_i^{j=1} = 1$ if $f_{n,t}(i) = 1$, and set $x_i^{j=1} = -1$ if $f_{n,t}(i) = 0$. We define samples $i = n + 1, ..., m$ to have the same, unit value of the first feature, $x_i^{j=1} = 1$; these samples all belong to the positive class, $y_i = 1$. For the remaining features $j = 2, ..., p$, we assign null value for all m samples, that is, for $j \geq 2$, $x_i^j = 0$ for any $i \in [1, m]$.

The $m \times p$ training set is not constructed explicitly. Instead, the oracle access to its elements involves determining, using indices i, j and $\Theta(1)$ computation, whether a single call to the underlying majority problem black-box oracle is required (for $j = 1$, $i = 1, ..., n$) or whether to return a pre-determined value without the need for accessing the underlying oracle. Thus, the number of oracle calls to the training data is bounded from below by the number of calls to the underlying majority problem oracle.

For the training problem instance defined above, in the optimal Sparse SVM solution[2] $\arg\min_{\beta \in \mathbb{R}^p} \frac{1}{m} \sum_{i=1}^m [1 - y_i \beta^T x_i]_+ + \lambda \sum_{j=1}^p |\beta_j|$, features $j = 2, ..., p$ have null model weight $\beta_j = 0$, since the feature values x_i^j for $j \geq 2$ are all zeros and do not contribute anything to the classifier decision irrespective of β_j value, while L_1 regularization penalizes for non-zero values of feature weights.

We turn our focus to optimal model weight for the first feature, β_1. Consider $\beta_1 = 1$. The t samples corresponding to $f_{n,t}(i) = 1$ will be misclassified (prediction 1, true class -1), and each will contribute $[1 - y_i \beta_1 x_i^1]_+ / m = 2/m$ loss to the objective value. The $n - t$ samples corresponding to $f_{n,t}(i) = 0$, as well all the samples $n + 1, ..., 2n$, will be classified correctly (prediction vs. true class is -1 vs. -1 or 1 vs. 1), contributing null loss. Regularization term will contribute $\lambda|\beta_1| = \lambda$. In total, we will have $L = 2t/m + \lambda$.

Any negative β_1 has higher objective value than the same-magnitude positive β_1, allowing us to focus on nonnegative values of β_1. Increasing β_1 above 1 will increase the loss on the t samples and keep null loss for the $2n - t$ samples, and will increase the regularization term. Having $\beta_1 = 1 - \delta$ for $1 \leq \delta > 0$ will reduce loss on the t samples by δ/m per sample, increase loss on the $2n - t$ samples by the same amount, and decrease the regularizer term by $\lambda\delta$, for a total change of $(2n-t)\delta/m - t\delta/m - \lambda\delta = \delta(2n - 2t - \lambda m)/m$ in the objective value.

For regularization constants $\lambda \geq 2(n - t)/m$, minimal objective value is attained by setting $\beta_1 = 0$, that is, by an uninformative, all-null $\beta = 0$ model. For any $\lambda < 2(n - t)/m$, the optimal model is $\beta_1 = 1$, $\beta_{\geq 2} = 0$, with optimal objective value $L^* = 2t/m + \lambda$.

For the non-all-null models, the observed optimal objective value L^* can be used to solve the underlying majority problem by calculating t using

$$t = m(L^* - \lambda)/2.$$

Hence, among datasets with m samples and p features are those corresponding to all possible majority problems of size $n = \lceil m/2 \rceil$, some of which require number of oracle calls proportional to n to be solved. Thus, quantum Sparse SVM is $\Omega(m)$.

If the solver returns a suboptimal solution with objective value $\hat{L} = L^* + \varepsilon$ and we use it to obtain $\hat{t} = m(\hat{L} - \lambda)/2$, we make an error

$$\Delta t = \hat{t} - t = m\varepsilon/2.$$

Consider a promise variant of majority, in which the minority class is guaranteed to have at most k elements, which has $\Omega(k)$ complexity lower bound. For problems with the $k = n/4$-minority class

---

[2]We use the notation $[\cdot]_+ = \max(0, \cdot)$.

promise, $n/4$ oracle calls are needed in the worst case; furthermore, making smaller than $n/4$ error Δt in estimating t does not change the correctness of the solution.

To have the error $\Delta t < n/4$ that guarantees no mistakes in solving the $n/4$-promise majority problem, we need

$$\varepsilon < n/2m.$$

Noting that $2n \leq m \leq 2n + 1$ and that $m \geq 8$, the inequality is guaranteed if

$$\varepsilon \leq 1/5.$$

Any suboptimal solution with additive error of $1/5$ or less compared to the optimal objective value allows us to solve $n/4$-promise majority problems, some of which require $\Omega(n)$ oracle calls. Thus, allowing $O(1)$ solver error still leads to $\Omega(m)$ complexity lower bound. □

Note that $L^* \leq 1$, since $L^* = 1$ can always be achieved irrespective of the training set by an all-null feature weights vector β. The error in estimating L^* within which it is still possible to solve the promise majority problems does not decrease with the input problem size, neither in absolute terms, nor in relative terms when we focus on multiplicative error $\rho = (L^* + \varepsilon)/L^*$ instead of ε.

5   Speedup in Quantum Sparse SVM for Special Cases

Worst-case analysis of the proposed method utilizing quantum LP solver [23], and of quantum Sparse SVM in general, shows that it cannot achieve sublinear computational complexity. This does not preclude sublinear complexity for some families of classification problems for specific quantum SVM solvers.

In the quantum LP-based approach, sublinear time can only be achieved if the L_1 norms of the primal and dual solution vectors are kept in check as the number of samples and features grows. The regularizing term $||\beta||_1$ can be expected to grow slowly in sparse models, and the average loss is normalized by the number of training samples and thus always below one. Yet, sparsity assumption alone is not enough to achieve speedup when class distributions overlap in the features space and the optimal error is not null. While the bound on the dual solution norm is a direct consequence of sparsity of the model, the $\sum_i \xi_i$ term in the primal solution norm in principle grows in proportion to the number of training samples and to the generalization risk, and may be further inflated by the stochastic nature of the training set. We show that under realistic assumptions about the family of classification problems for growing number of features, where new features occasionally bring new discriminative information, a bound on the primal solution norm that leads to sublinear training time can be formulated.

5.1   Truncated Subgaussian Classification Problems

The first defining characteristic of the family of classification problems we explore here is limited overlap between class distributions. Since we are focusing on a linear, sparse SVM classifier, the overlap will be with respect to a linear decision boundary – we will assume that the tails of class conditional distributions that reach across the decision hyperplane are bounded, in a way formalized in Definition 2. The consequences of that assumption[3] on the expected loss of the model are analyzed in Lemma 4 in the distribution setting, and then extended in Lemma 5 to the empirical risk on finite-sample datasets of fixed size (m, p) that are actually seen during training, providing basis for asymptotic analysis building up to Theorem 3.

In the asymptotic analysis of time complexity, we will assume that we are given a series of classification problems where both the number of samples, m, and the number of features, p, grows. The second defining characteristic of the scenarios leading to sublinear time complexity is the assumption that p', the number of discriminative features used by the sparse model, also grows, but at a slower rate than m or p. In an illustrative idealized case, these discriminative features have means $+c$ and $-c$

---

[3]In this section we motivate Definition 2 and present the final complexity result in Theorem 3, and we defer the detailed Lemmas to Section 5.3
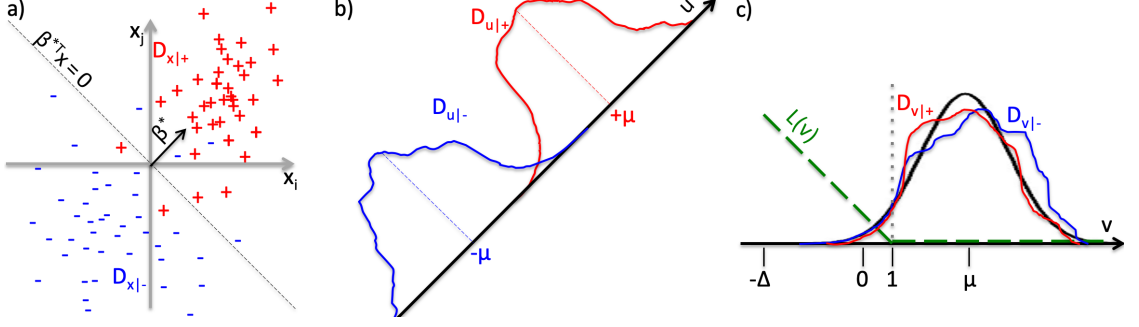
Figure 2: Graphical illustration of Definition 2: **a)** the initial multivariate two-class problem; **b)** intermediate step involving projection $u = \beta^{*T} x$ into one dimension; **c)** final univariate distributions after projection $v = yu$.

in the positive and the negative class, respectively – though the situation does not change if the signs of the means are swapped for some of the discriminative features. Thus, c represents how discriminative a feature is. With increasing number p' of discriminative features, each with means differing by at least $2c$, the distance between the means of the two multivariate distributions increases at the rate of at least $2c\sqrt{p'}$. Moving beyond the idealized case of means of each feature at $\pm c$, we will simply require that the multivariate class distribution means diverge at this rate. This defining feature of our scenario is key to asymptotic analysis starting in Lemmas 6 and 7 and concluding with the final complexity result in Theorem 3.

Definition 2. *A (Δ, μ)-truncated subgaussian classification problem, for $\mu > 1$, $\Delta > 0$, is defined by distribution D such that there is an underlying vector $\beta^\star \in \mathbb{R}^p$ with $\|\beta^\star\|_2 = 1$, for which*

- *the conditional distributions $D_{x|+}$ and $D_{x|-}$ of the samples from the positive and negative class, respectively, give rise to univariate distributions $D_{v|+}$ and $D_{v|-}$ on a line resulting from the projection $v = y\beta^{\star T} x$,*

- *the tails of $D_{v|+}$ and $D_{v|-}$ are bounded from above, in the region $v \in (-\infty, 1]$, by the normal probability density function $\mathcal{N}_{\mu,1}(v)$ with unit standard deviation, centered at μ,*

- *the tails of $D_{v|+}$ and $D_{v|-}$ have zero mass for $v < -\Delta$.*

A p-dimensional (Δ, μ)-truncated subgaussian problem is called sparse if the number of non-zero components in the vector $\beta^\star$ is small compared to the number of features, p.

A graphical illustration of the definition is shown in Figure 2. Sparse (Δ, μ)-truncated subgaussian problems often arise in dataset coming from natural science. For example, in molecular biology, levels of raw mRNA transcripts of individual genes approximately follow a log-normal distribution within a class of samples [43, 44]. Raw data is typically processed via a log-like transform prior to analyses, then each class of a multi-gene dataset approximately follows a multivariate normal distribution, leading to sub-Gaussian tail. Biomedical data often has large number of features, and the samples are rarely sparse, that is, vast majority of feature values are non-zero. For example, classification problems involving gene expression measured using RNA-seq may have tens of thousands of features, only a relatively small number of genes contribute to between-class differences, leading to a sparse problem – sparse linear models have been successful in analyzing brain activity [45], microbiome [46], and in other biomedical settings [47, 48, 49]. These application scenarios also typically lead to datasets in which the number of features is larger or at least comparable to the number of samples, $p \gtrsim m$.

We focus on the $p \gtrsim m$ scenarios common in differential gene expression or methylation analysis, or classification of 3D brain scans, in which L_1 regularization is especially useful. Training the Quantum Sparse SVM for sparse (Δ, μ)-truncated subgaussian problems, assuming datasets with $p \gtrsim m$, has complexity that is sublinear in terms of number of samples m and number of features p.

9

Theorem 3. *For $p \to \infty$, consider a family of p-dimensional (Δ_p, μ_p)-truncated subgaussian problems D_p with underlying vectors $\beta_p^\star$. Assume that the vector $\beta_p^\star$ is sparse, it only has $p' = 1 + 2 \log p$ non-zero coefficients. Further, assume that the mean μ_p diverges with the number of discriminative features p' as $\mu_p > c\sqrt{p'}$ for some $c > 1$ that captures how separated class centers are for individual informative features. As p grows, we allow scattering of the samples farther into the region dominated by the other class, $\Delta_p \leq 2 \log p$. For growing p, assume efficient oracle access to the training data, with the training set sizes m growing proportionally with p, $m/p = O(1)$. Then, training QsSVM has computational complexity of*

$$\tilde{O}\left(\sqrt{m + 2p}\operatorname{poly}\left(\log p, 1/\varepsilon\right)\right).$$

Proof. Training QsSVMs translates to solving an LP problem (eq. 2) with m constraints and $n = 2p + m$ variables. The quantum LP solver proposed of van Apeldoorn and Gilyén [23] has complexity $\tilde{O}\left(\left(\sqrt{m} + \sqrt{n}\right)\left(R_p(r_p + 1)/\varepsilon\right)^3\right)$. When $m/p = O(1)$, by virtue Lemma 7 (see Section 5.3), neither R_p nor r_p grow with m, and both grow with p as $O(\log p)$, yielding the complexity result. $\square$

5.2 Speedup Compared to Classical Sparse SVM for Truncated Subgaussian Problems

We analyze here how the sublinear complexity of Quantum Sparse SVM for truncated subgaussian problems compares to classical algorithms. Specifically, we show that classical algorithms for this class of problems cannot be faster than linear, through reduction from the search problem. As a result, Quantum Sparse SVM holds polynomial complexity advantage over any classical algorithm for Sparse SVM on this class of problems.

Consider the following truncated subgaussian problem with m samples and p features. All samples from the positive class are vectors with p ones. On the other hand, all samples from the negative class have negative one on the same subset of $p' = O(\log p)$ features, and have the value of positive one for all other features. This setup meets the criteria for a sparse truncated subgaussian classification problem according to Definition 2. Finding the optimal classification model involves finding at least one of the p' features where a positive sample differs in value from a negative sample, out of p possibilities. Successfully training the classifier for this particular classification problem translates to successfully solving the search problem. For small values of the p'/p ratio considered here, the search problem has classical complexity of $O(p)$, providing lower bound on classical algorithms for training Sparse SVMs for truncated subgaussian problems. In the quantum setting, the problem can be solved by Groveer's search algorithm, which attains the quantum lower bound of $O(\sqrt{p})$. The quantum lower bound shows that the proposed method involving a quantum LP solver is optimal, up to the slower growing polylogarithmic terms.

We also note that since all the sample feature vectors are composed of elements with the same, unit magnitude, and thus all rows and all columns have the same norm, it is not possible to use norm-based sampling methods to construct a dequantized version of Quantum Sparse SVM with sublinear complexity for the truncated subgaussian problems.

5.3 Complexity Analysis of Quantum Sparse SVM for Truncated Subgaussian Problems

The path from Definition 2 to Theorem 3 leads through a series of technical Lemmas, from generalization risk to the characteristics of the empirical solution.

For the hinge loss, the generalization risk $R(\beta^\star)$ associated with model $h(x) = \beta^{\star T}x$ on the (Δ, μ)-truncated subgaussian problem D is bounded through the following lemma.

Lemma 4. *Let D be a (Δ, μ)-truncated subgaussian classification problem with underlying vector $\beta^\star$ leading to univariate distributions $D_{v|+}$ and $D_{v|-}$ as described above. Let $L = \max(0, 1 - v)$ be a univariate random variable capturing hinge loss of the model $h(x) = \beta^{\star T}x$ for samples from D. Then,*

the expectation and standard deviation of L are bounded from above by

$$R(\beta^\star) = \mathbb{E}[L] \leq \frac{1}{\sqrt{2\pi}} e^{-\frac{(1-\mu)^2}{2}} = \mathcal{N}_\mu(1), \tag{4}$$

$$\mathbb{V}\mathrm{ar}[L] \leq \left[(1-\mu)^2 + 1\right] \left[1 + \mathrm{erf}\left(\frac{1-\mu}{\sqrt{2}}\right)\right]. \tag{5}$$

Also, values of L are in the range $[0, \Delta + 1]$.

Proof. The proof relies on properties of integrals of $x^k \mathcal{N}_0(x)$. Let

$$G(x) = \mathcal{N}_0(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, \quad G_k(x) = \int_{-\infty}^{x} t^k G(t)\, \mathrm{d}t.$$

Then, we have Let p_+ and p_- by the probabilities, under D, of the positive and the negative class, respectively. We have

$$\mathbb{E}[L] = \int_{-\infty}^{\infty} \max(0, 1-v)[p_+ D_{v|+}(v) + p_- D_{v|-}(v)]\, \mathrm{d}v$$

$$\leq (1-\mu)G_0(1-\mu) - G_1(1-\mu) \leq \frac{1}{\sqrt{2\pi}} e^{-\frac{(1-\mu)^2}{2}}.$$

$$\mathbb{E}[L^2] \leq [(1-\mu)^2 + 1]G_0(1-\mu) + (1-\mu)G(1-\mu)$$

$$\leq \left[(1-\mu)^2 + 1\right] \left[1 + \mathrm{erf}\left(\frac{1-\mu}{\sqrt{2}}\right)\right].$$

The range of L follows immediately from null mass of $D_{v|y}$ for $v \leq -\Delta$, and from null loss for any $v \geq 1$. $\qquad \square$

The result above gives the bound on the expected value of the hinge loss for the model $h(x) = \beta^{\star T} x$ on the distribution D, that is, it bounds from above the generalization risk of that model, $R(\beta^\star) = \mathbb{E}[L]$. However, it does not give an upper bound on the empirical risk for the model $h(x) = \beta^{\star T} x$ on a specific training set with m samples and p features, sampled from D. This bound is given be the following lemma.

Lemma 5. *Let D be a (Δ, μ)-truncated subgaussian problem based on $\beta^\star$. Let $\hat{R}(\beta^\star)$ be the empirical risk associated with model $\beta^\star$ over a m-sample training set sampled* i.i.d. *from D. Then, with probability at least $1 - \delta$*

$$\hat{R}(\beta^\star) \leq \frac{1}{\sqrt{2\pi}} e^{-\frac{(1-\mu)^2}{2}} + 4\frac{(\Delta+1)\log(2/\delta)}{m} \tag{6}$$

$$+ 4\frac{\sqrt{\log(2/\delta)}}{\sqrt{m}} \left[(1-\mu)^2 + 1\right] \left[1 + \mathrm{erf}\left(\frac{1-\mu}{\sqrt{2}}\right)\right]$$

Proof. Consider m values $l_1, ..., l_m$ drawn from a univariate random variable L taking values in range in $[a, b] = [0, \Delta + 1]$, and with finite variance $s = \mathbb{V}\mathrm{ar}[L]$ and finite mean $R = \mathbb{E}[L]$. Let $\hat{R} = \frac{1}{m}\sum_{i=1}^{m} l_i$ be the empirical mean. Bernstein's inequality states that

$$\mathbb{P}(|\hat{R} - R| \geq t) \leq 2\exp\left(\frac{mt^2}{2(s^2 + (b-a)t)}\right).$$

That is, with probability at least $1 - \delta$,

$$\hat{R} \leq R + 4s\sqrt{\frac{\log(2/\delta)}{m}} + \frac{4(b-a)\log(2/\delta)}{m}.$$

We thus have

$$\hat{R}(\beta^\star) \leq \mathbb{E}[L] + 4\mathbb{V}\mathrm{ar}[L]\sqrt{\frac{\log(2/\delta)}{m}} + \frac{4(\Delta+1)\log(2/\delta)}{m}$$

The bound follows from plugging in the bounds on expected value (eq. 4) and variance (eq. 5) of the loss. $\qquad \square$

11

We are now ready to analyze the behavior of empirical risk of models $\beta_p^\star$ on problems D_p as the number of all features p and the number of discriminative features p' grow.

Lemma 6. *For $p \to \infty$, consider a family of p-dimensional (Δ_p, μ_p)-truncated subgaussian problems D_p with underlying vectors $\beta_p^\star$. Assume that the vector $\beta_p^\star$ is sparse, it only has $p' = 1 + 2\log p$ non-zero coefficients. Further, assume that the mean μ_p diverges with the number of discriminative features p' as $\mu_p > c\sqrt{p'}$ for some $c > 1$ that captures how separated class centers are for individual informative features. As p grows, we allow scattering of the samples farther into the region dominated by the other class – specifically, we allow $\Delta_p \leq 2\log p$. Then, with probability at least $1 - \delta$, we have*

$$\hat{R}(\beta_p^\star) \leq \frac{1}{\sqrt{2\pi p}} + 4\frac{(2\log p + 1)\log(2/\delta)}{m}. \tag{7}$$

Proof. Under the assumption that μ_p grows at least as $c\sqrt{p'} = c\sqrt{1 + 2\log p}$, we have $\mu_p \geq 1 + \sqrt{2\log p}$, which leads to the bound on the first term of eq. (6), and to the second term approaching null limit.

Specifically, we have the following limit $\lim_{x \to \infty} c\sqrt{1 + kx}/[1 + \sqrt{kx}] = c$. Thus, under the assumption that μ_p grows at least as $c\sqrt{p'} = c\sqrt{1 + 2\log p}$, for $c > 1$, for sufficiently large p, we have $\mu_p \geq c\sqrt{1 + 2\log p} \geq 1 + \sqrt{2\log p}$, that is, $1 - \mu_p \leq -\sqrt{2\log p}$.

For the first term in eq. (6), e^x is an increasing function of x, we thus have the following upper bound

$$\frac{1}{\sqrt{2\pi}}e^{-\frac{(1-\mu_p)^2}{2}} \leq \frac{1}{\sqrt{2\pi}}e^{-\log p} = \frac{1}{\sqrt{2\pi}}\frac{1}{p}.$$

For the second term in eq. (6), we can show that $1 + \text{erf}\left(\frac{1-\mu_p}{\sqrt{2}}\right)$ approaches null with the rate faster than $\frac{1}{p^2}$. Since $1 - \mu_p \leq -\sqrt{2\log p}$ and $\text{erf}(x)$ is an increasing function of x, we have $1 + \text{erf}\left(\frac{1-\mu_p}{\sqrt{2}}\right) \leq 1 + \text{erf}\left(-\sqrt{2\log p}\right)$. We also have

$$\frac{\mathrm{d}\left[1 + \text{erf}\left(-\sqrt{2\log p}\right)\right]}{\mathrm{d}p} = -\frac{\sqrt{\frac{2}{\pi}}}{p^3\sqrt{\log p}}.$$

From the L'Hôpital's rule,

$$\lim_{p \to \infty}\frac{1 + \text{erf}\left(-\sqrt{2\log p}\right)}{p^{-2}} = \lim_{p \to \infty}\frac{1}{2\sqrt{\pi}}\frac{1}{\sqrt{\log p}} = 0.$$

Thus, $\left[(1 - \mu_p)^2 + 1\right]\left[1 + \text{erf}\left(\frac{1-\mu_p}{\sqrt{2}}\right)\right] = O\left(\left[(-\sqrt{2\log p})^2 + 1\right]/p^2\right) = O\left(\log p/p^2\right)$. The second term quickly approaches null as p grows. $\square$

Sparse SVM involves regularized empirical risk, that is, minimization of a weighted sum of the empirical risk and the L_1 norm of the model β. Under the scenario of slowly increasing number of discriminative features, the Sparse SVM regularized empirical risk minimization is characterized by the following lemma.

Lemma 7. *For $p \to \infty$, consider a family of p-dimensional classification problems D_p as described in Lemma 6. For each D_p, consider the SparseSVM regularized empirical minimization problem (eq. 1)*

$$\arg\min_{\beta} \quad \frac{1}{m}\sum_{i=1}^{m}\max(0, 1 - y_i\beta^T x_i) + \lambda\|\beta\|_1,$$

involving m-sample training set sampled from D_p. Then, for each p, with probability $1 - \delta$, there exist an empirical minimizer $\hat{\beta}_p$ of the problem above that can be found using a linear program (eq. 2), with L_1 norms of the primal and dual solutions, R_p and r_p, respectively, bounded from above by

$$R_p \leq \frac{1}{\sqrt{2\pi}}\frac{m}{p} + 4(1 + 2\log p)\log(2/\delta)] + \lambda\sqrt{1 + 2\log p},$$

$$r_p \leq \frac{1}{\sqrt{2\pi}}\frac{1}{p} + \frac{4(1 + 2\log p)\log(2/\delta)}{m} + \lambda\sqrt{1 + 2\log p}.$$

12

Proof. For any unit L_2-norm vector β with p' non-zero entries, the highest L_1 norm is achieved if all the p' coordinates are equal to $1/\sqrt{p'}$. Thus, $\|\beta_p^\star\|_1 \leq \sqrt{p'}$ for $\beta_p^\star$ with unit L_2-norm and at most p' non-zero coefficients. On the training set, the minimizer $\hat{\beta}_p$ has lowest objective function of all possible β, including $\beta_p^\star$. Thus, we have

$$
\begin{aligned}
\hat{R}(\hat{\beta}_p) + \lambda\|\hat{\beta}_p\|_1 &\leq \hat{R}(\beta_p^\star) + \lambda\|\beta_p^\star\|_1 \\
&\leq \frac{1}{\sqrt{2\pi}}\frac{1}{p} + 4\frac{(1 + 2\log p)\log(2/\delta)}{m} + \lambda\sqrt{1 + 2\log p}.
\end{aligned}
$$

From strong duality, we have $r_p = \hat{R}(\hat{\beta}_p) + \lambda\|\hat{\beta}_p\|_1$. The norm R_p of the primal solution does not involve averaging the losses $\max(0, 1 - y_i\beta^T x_i)$. Instead, the losses are added up, that is, R_p includes the term $m\hat{R}(\hat{\beta}_p)$ instead of the empirical risk. $\qquad\square$

Lemma 7 directly leads to Theorem 3.

6   Related Work

Considerable effort has been devoted into designing fast classical algorithms for training SVMs, although the focus is on the traditional SVM that involves the L_2 regularizer, which results in a strongly convex objective function even for hinge loss, as opposed to Sparse SVM, where the objective function is piece-wise linear, and thus convex but not strongly convex. The decomposition-based methods such as SMO [50] are able to efficiently manage problems with large number of features p, but their computational complexities are super-linear in m. Other training strategies [29, 51, 52] are linear in m but scale quadratically in p in the worst case. In more recent work, the number of features p is replaced by the average sparsity of samples, s, that is, the average number of non-zero feature values per sample. The SVM^{perf} algorithm [53] for linear SVM scales as $O(sm)$. The Pegasos algorithm [54] improves the complexity to $\tilde{O}\left(s/(\lambda\epsilon)\right)$, where λ, and ϵ are the regularization parameter of SVM and the error of the solution, respectively. However, sublinear sparsity of input samples is a strong assumption, and holds only in some special scenarios, such as Bag-of-Words encoding in natural language processing, but rarely holds for data describing physical phenomena, including biomedical data.

Beyond the classical realm, instead of the solver [23] we used, two alternative quantum solvers based on the interior point approach were also introduced recently, an SDP/LP [55] and a dedicated LP [56] solver. However, both of the quantum interior point solvers come with computational complexity challenges that make them less appealing for training sparse models. The interior point dedicated LP solver [56] has complexity of $O(\sqrt{n}L/\epsilon)$, where $L = O(\log n)$ on average but $L = mn$ in the worst case. Crucially, the complexity also depends on the sparsity of the constraint matrix A. However, in SVMs, A is composed of elements $y_i x_i^j$, the feature values, which are unlikely to be spare in many applications, including those involving biomedical data. The SDP/LP solver [55] has cubic dependence on the condition number κ of the intermediate solution matrices of the SDP. In the LP variant, κ is ratio of largest to smallest feature weight magnitude, which can easily explode in machine learning applications, where some feature weights are null or close to null. Even if the problem is well-conditioned, the interior-point SDP/LP solver has complexity of $\tilde{O}(m^2)$.

An alternative to using a quantum LP solver is to use quantum gradient descent. However, most quantum gradient descent approaches only work reliably for small number of steps, with probability of following the gradient path decreasing exponentially with the number of gradient updates. Recently, a quantum descent method circumventing this problem has been proposed [57], but only for objective functions where the norm of the gradient decreases with each step, such as quadratic functions. Sparse SVM objective function is piece-wise linear and non-differentiable, with possibility of subgradient norm increase when a step from one linear region to another is made.

Recently, construction of classical algorithms based on quantum machine learning methods using sampling-based data structure has been proposed [58], and has led to a dequantization of LS SVM and other machine learning methods under low-rank assumption [59]. A dequantized SDP solver has also been proposed [60] for SDP problems with low rank input matrices. While a linear program $LP(c, A, b)$

can be seen as an SDP with diagonal matrices formed by c and rows of A, for the LP resulting from Sparse SVM these diagonal matrices are full-rank. This provides further evidence that the proposed quantum Sparse SVM cannot be dequantized using existing approaches.

Acknowledgments

References

[1] Miles Blencowe. Quantum computing: Quantum RAM. *Nature*, 468(7320):44, 2010.

[2] Daniel K Park, Francesco Petruccione, and June-Koo Kevin Rhee. Circuit-based quantum random access memory for classical data. *Scientific Reports*, 9(1):1–8, 2019.

[3] N Jiang, Y-F Pu, W Chang, C Li, S Zhang, and L-M Duan. Experimental realization of 105-qubit random access quantum memory. *npj Quantum Information*, 5(1):1–6, 2019.

[4] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195, 2017.

[5] Vedran Dunjko and Hans J Briegel. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics*, 81(7):074001, 2018.

[6] M. Schuld and F. Petruccione. *Supervised Learning with Quantum Computers*. Springer Nature, 2018.

[7] Srinivasan Arunachalam and Ronald de Wolf. A survey of quantum learning theory. *ACM SIGACT News*, 48(2):41–67, 2017.

[8] Vedran Dunjko and Peter Wittek. A non-review of quantum machine learning: trends and explorations. *Quantum Views*, 4:32, 2020.

[9] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. q-means: A quantum algorithm for unsupervised machine learning. In *Advances in Neural Information Processing Systems*, pages 4136–4146, 2019.

[10] Tongyang Li, Shouvanik Chakrabarti, and Xiaodi Wu. Sublinear quantum algorithms for training linear and kernel-based classifiers. In *International Conference on Machine Learning*, pages 3815–3824, 2019.

[11] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. Quantum algorithms for deep convolutional neural networks. In *International Conference on Learning Representations*, 2020.

[12] Johannes Bausch. Recurrent quantum neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[13] Hayata Yamasaki, Sathyawageeswar Subramanian, Sho Sonoda, and Masato Koashi. Learning with optimized random features: Exponential speedup by quantum machine learning without sparsity and low-rank assumptions. *Advances in Neural Information Processing Systems*, 33, 2020.

[14] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(13):130503, 2014.

[15] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009.

[16] Rolando Somma, Andrew Childs, and Robin Kothari. Quantum linear systems algorithm with exponentially improved dependence on precision. In *APS Meeting Abstracts*, 2016.

[17] Sathyawageeswar Subramanian, Steve Brierley, and Richard Jozsa. Implementing smooth functions of a Hermitian matrix on a quantum computer. *arXiv:1806.06885*, 2018.

[18] Jonathan Allcock and Chang-Yu Hsieh. A quantum extension of SVM-perf for training nonlinear SVMs in almost linear time. *Quantum*, 4:342, 2020.

[19] Kristin Bennett. Combining support vector and mathematical programming methods for induction. *Advances in Kernel Methods: Support Vector Learning*, pages 307–326, 1999.

[20] Vojislav Kecman and Ivana Hadzic. Support vectors selection by linear programming. In *Proc. International Joint Conference on Neural Networks (IJCNN'00)*, volume 5, pages 193–198. IEEE, 2000.

[21] Jinbo Bi, Kristin Bennett, Mark Embrechts, Curt Breneman, and Minghu Song. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3(Mar):1229–1243, 2003.

[22] Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. 1-norm support vector machines. In *Advances in Neural Information Processing Systems (NIPS'04)*, pages 49–56. MIT Press Boston, MA, 2004.

[23] Joran van Apeldoorn and András Gilyén. Quantum algorithms for zero-sum games. *arXiv:1904.03180*, 2019.

[24] Fernando GSL Brandão and Krysta M Svore. Quantum speed-ups for solving semidefinite programs. In *IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS'17)*, pages 415–426. IEEE, 2017.

[25] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum SDP-solvers: Better upper and lower bounds. In *Proc. IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS'17)*, pages 403–414. IEEE, 2017.

[26] FGSL Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M Svore, and Xiaodi Wu. Quantum SDP solvers: Large speed-ups, optimality, and applications to quantum learning. *arXiv:1710.02581*, 6, 2017.

[27] Joran van Apeldoorn and András Gilyén. Improvements in quantum SDP-solving with applications. *arXiv:1804.05058*, 2018.

[28] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer, 2001.

[29] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[30] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[31] Michael D. Grigoriadis and Leonid Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18:53–58, 1994.

[32] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical Review Letters*, 100(16):160501, 2008.

[33] Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan. On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, 17(12):123010, 2015.

[34] Connor T. Hann, Chang-Ling Zou, Yaxing Zhang, Yiwen Chu, Robert J. Schoelkopf, S. M. Girvin, and Liang Jiang. Hardware-efficient quantum random access memory with hybrid quantum acoustic systems. *Physical Review Letters*, 123:250501, 2019.

[35] Michał Parniak, Michał Dabrowski, Mateusz Mazelanik, Adam Leszczyński, Michał Lipka, and Wojciech Wasilewski. Wavevector multiplexed atomic quantum memory via spatially-resolved single-photon detection. *Nature Communications*, 8(1):1–9, 2017.

[36] Yong Yu, Fei Ma, Xi-Yu Luo, Bo Jing, Peng-Fei Sun, Ren-Zhou Fang, Chao-Wei Yang, Hui Liu, Ming-Yang Zheng, Xiu-Ping Xie, Wei-Jun Zhang, Li-Xing You, Zhen Wang, Teng-Yun Chen, Qiang Zhang, Xiao-Hui Bao, and Jian-Wei Pan. Entanglement of two quantum memories via metropolitan-scale fibers. *Nature*, 578:240–245, 2020.

[37] X-L Ouyang, X-Z Huang, Y-K Wu, W-G Zhang, X Wang, H-L Zhang, L He, X-Y Chang, and L-M Duan. Experimental demonstration of quantum-enhanced machine learning in a nitrogen-vacancy-center system. *Physical Review A*, 101(1):012307, 2020.

[38] Zhikuan Zhao, Jack K Fitzsimons, Patrick Rebentrost, Vedran Dunjko, and Joseph F Fitzsimons. Smooth input preparation for quantum and quantum-inspired machine learning. *arXiv:1804.00281*, 2018.

[39] Ryan LaRose and Brian Coyle. Robust data encodings for quantum classifiers. *arXiv:2003.01695*, 2020.

[40] Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv quant-ph/0208112*, 2002.

[41] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. *arXiv:1603.08675*, 2016.

[42] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001.

[43] Martin Bengtsson, Anders Ståhlberg, Patrik Rorsman, and Mikael Kubista. Gene expression profiling in single cells from the pancreatic islets of langerhans reveals lognormal distribution of mrna levels. *Genome Research*, 15(10):1388–1392, 2005.

[44] Vasilis Ntranos, Lynn Yi, Páll Melsted, and Lior Pachter. A discriminative learning approach to differential expression analysis for single-cell rna-seq. *Nature Methods*, 16(2):163–166, 2019.

[45] Yunzhe Liu, Raymond J Dolan, Zeb Kurth-Nelson, and Timothy EJ Behrens. Human replay spontaneously reorganizes experience. *Cell*, 178(3):640–652, 2019.

[46] Jennifer M Fettweis, Myrna G Serrano, J Paul Brooks, David J Edwards, Philippe H Girerd, Hardik I Parikh, Bernice Huang, Tom J Arodz, Laahirie Edupuganti, Abigail L Glascock, et al. The vaginal microbiome and preterm birth. *Nature Medicine*, 25:1012–1021, 2019.

[47] Helen Davies, Dominik Glodzik, Sandro Morganella, Lucy R Yates, Johan Staaf, Xueqing Zou, Manasa Ramakrishna, Sancha Martin, Sandrine Boyault, Anieta M Sieuwerts, et al. HRDetect is a predictor of BRCA1 and BRCA2 deficiency based on mutational signatures. *Nature Medicine*, 23(4):517, 2017.

[48] Margaret E Ackerman, Jishnu Das, Srivamshi Pittala, Thomas Broge, Caitlyn Linde, Todd J Suscovich, Eric P Brown, Todd Bradley, Harini Natarajan, Shu Lin, et al. Route of immunization defines multiple mechanisms of vaccine-mediated protection against SIV. *Nature Medicine*, 24(10):1590, 2018.

[49] Peggie Cheung, Francesco Vallania, Hayley C Warsinske, Michele Donato, Steven Schaffert, Sarah E Chang, Mai Dvorak, Cornelia L Dekker, Mark M Davis, Paul J Utz, et al. Single-cell chromatin modification profiling reveals increased epigenetic variations with aging. *Cell*, 173(6):1385–1397, 2018.

[50] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft, 1998.

[51] Glenn M Fung and Olvi L Mangasarian. Multicategory proximal support vector machine classifiers. *Machine Learning*, 59(1-2):77–97, 2005.

[52] S Sathiya Keerthi and Dennis DeCoste. A modified finite newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

[53] Thorsten Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

[54] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30, 2011.

[55] Iordanis Kerenidis and Anupam Prakash. A quantum interior point method for LPs and SDPs. *arXiv:1808.09266*, 2018.

[56] PAM Casares and MA Martin-Delgado. A quantum IP predictor-corrector algorithm for linear programming. *arXiv:1902.06749*, 2019.

[57] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Physical Review A*, 101(2):022316, 2020.

[58] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proc. 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC'19)*, pages 217–228, 2019.

[59] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. *arXiv:1910.06151*, 2019.

[60] Nai-Hui Chia, Tongyang Li, Han-Hsuan Lin, and Chunhao Wang. Quantum-inspired classical sublinear-time algorithm for solving low-rank semidefinite programming via sampling approaches. *arXiv:1901.03254*, 2019.