Quantum Algorithms: Support Vector Machines

Jan Lohse

September 3, 2024

Contents

1. Introduction

Support vector machines (SVMs) are widely used supervised machine learning models for classifying data. The goal of SVMs is to find a hyperplane separating different classes of data points as uniformly as possible. Using the kernel trick they can efficiently be extended to non-linear classification, by mapping the data to a higher-dimensional space.

Linear SVMs can be implemented classically using second-order cone programming (SOCP) and then be solved using interior-point methods (IPMs). IPMs are closely related to Newton's method for finding the roots of functions. An optimal solution is approximated through iterative improvements by solving a linear system in each step.

This method is not commonly used for SVMs in practice. The alternative, also classical, sequential minimal optimization (SMO) algorithm allows for kernel tricks and has a better runtime than classical SOCP methods for SVMs. SMO typically achieves a worst-case runtime of $\mathcal{O}(n^3)$, while SOCP methods have a runtime of $\mathcal{O}(n^{\omega+0.5})$ with a matrix multiplication exponent ω of at least 2.37 and up to 3, depending on the implementation.

Kerenidis et al. [1] have proposed an IPM for SOCP making use of quantum computing to solve linear systems more efficiently and applied this technique to solving SVMs. Their method promises a polynomial speedup over both SMO and classic SOCP methods. They verified this experimentally through numerical simulations. Random SVM instances were solved using all three methods and an estimated complexity of $\mathcal{O}(n^{2.59})$ for their own quantum SOCP method, $\mathcal{O}(n^{3.11})$ using SMO, and $\mathcal{O}(n^{3.31})$ with a classical SOCP method were measured.

We will present their techniques and results. Additionally we suggest an extension of their method to the alternative SVM formulation of least-squares SVMs. Kerenidis et al. consider soft-margin SVMs, where the hyperplane is only affected by samples lying in a margin around it. Least-squares SVMs consider the squared error over all samples.

2. Preliminaries

2.1. Notation

By $(\boldsymbol{x}_1; ...; \boldsymbol{x}_n) \coloneqq \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix}$ we denote the column-wise conjunction of vectors.
$[m] \coloneqq \{1, ..., m\} \subseteq \mathbb{N}$. If not specified $\|\boldsymbol{x}\|$ refers to the Euclidean norm.

2.2. Second-order cone programming

The goal in SOCP is to minimize a linear function inside the intersection of second-order Lorentz cones. We follow the formulation of a SOCP with its dual from [2]. A Lorentz cone $\mathcal{L}^k \subseteq \mathbb{R}^k$ is defined as $\mathcal{L}^k = \left\{\boldsymbol{x} = (x_0; \tilde{\boldsymbol{x}}) \in \mathbb{R}^k \,\middle|\, \|\tilde{\boldsymbol{x}}\| \leq x_0\right\}$. For its interior $\operatorname{int} \mathcal{L}$ the inequality $\|\tilde{\boldsymbol{x}}\| < x_0$ has to hold. Over the product of Lorentz cones $\mathcal{L} = \mathcal{L}^{n_1} \times \cdots \times \mathcal{L}^{n_r}$ we define a SOCP problem (1) and its dual (2) given $\boldsymbol{c}_i \in \mathbb{R}^{n_i}, \boldsymbol{b} \in \mathbb{R}^m, A_i \in \mathbb{R}^{m \times n_i}$ as

$$\begin{aligned} \min \quad & \boldsymbol{c}_1^\mathsf{T}\boldsymbol{x}_1 + \cdots + \boldsymbol{c}_r^\mathsf{T}\boldsymbol{x}_r \\ \text{s.t.} \quad & A_1\boldsymbol{x}_1 + \cdots + A_r\boldsymbol{x}_r = \boldsymbol{b} \\ & \boldsymbol{x}_i \in \mathcal{L}_i, \end{aligned} \tag{1}$$

$$\begin{aligned} \max \quad & \boldsymbol{b}^\mathsf{T}\boldsymbol{y} \\ \text{s.t.} \quad & A_i^\mathsf{T}\boldsymbol{y} + \boldsymbol{s}_i = \boldsymbol{c}_i \\ & \boldsymbol{s}_i \in \mathcal{L}_i, \boldsymbol{y} \in \mathbb{R}^m. \end{aligned} \tag{2}$$

$n \coloneqq \sum_{i=1}^r n_i$ is the SOCP problem's *size* and r its *rank*. For each indexed variable when leaving away the index we refer to a conjunction like $\boldsymbol{x} = (\boldsymbol{x}_1; ...; \boldsymbol{x}_r)$. $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s})$ is *feasible* if it fulfills both (1) and (2), and *strictly feasible* if also $\boldsymbol{x}, \boldsymbol{s} \in \operatorname{int} \mathcal{L}$. For a feasible solution $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s})$ the duality gap is defined as $\mu \coloneqq \frac{1}{r}\boldsymbol{x}^\mathsf{T}\boldsymbol{s}$. The *Weak Duality Lemma* [3] states that $\mu \geq 0$. With

$$\boldsymbol{c}^\mathsf{T}\boldsymbol{x} - \boldsymbol{b}^\mathsf{T}\boldsymbol{y} = (\boldsymbol{y}^\mathsf{T}A + \boldsymbol{s}^\mathsf{T})\boldsymbol{x} - \boldsymbol{x}^\mathsf{T}A^\mathsf{T}\boldsymbol{y} = \boldsymbol{x}^\mathsf{T}\boldsymbol{s} = r\mu$$

and the optimization terms of (1) and (2) it then follows that the goal is to minimize μ. The *Strong Duality Theorem* [3] states that an optimal solution with $\mu = 0$ exists.

2.3. Euclidean Jordan algebras

In this section a mathematical structure over Lorentz cones is introduced, whose properties are essential for proving the correctness of the main algorithm presented in Section 3. A Jordan algebra is a commutative algebra that satisfies the Jordan identity

$$(x \circ y) \circ (x \circ x) = x \circ (y \circ (x \circ x)).$$

It is not necessarily associative. Over the Lorenz cone $\mathcal{L}^n$ a Jordan algebra is defined by the Jordan product

$$\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{x} \circ \boldsymbol{y} \coloneqq \begin{bmatrix} \boldsymbol{x}^\mathsf{T}\boldsymbol{y} \\ x_0\tilde{\boldsymbol{y}} + y_0\tilde{\boldsymbol{x}} \end{bmatrix} \text{ with identity } \boldsymbol{e} = \begin{bmatrix} 1 \\ 0^{n-1} \end{bmatrix}.$$

With the *arrow-shaped* matrix representation $\operatorname{Arw}(\boldsymbol{x}) \coloneqq \begin{bmatrix} x_0 & \tilde{\boldsymbol{x}}^\mathsf{T} \\ \tilde{\boldsymbol{x}} & x_0 I_{n-1} \end{bmatrix}$ we get

$$\boldsymbol{x} \circ \boldsymbol{y} = \operatorname{Arw}(\boldsymbol{x})\boldsymbol{y} = \operatorname{Arw}(\boldsymbol{x}) \operatorname{Arw}(\boldsymbol{y})\boldsymbol{e}.$$

We can induce an analogue to the matrix decomposition:

$$\lambda_1(\boldsymbol{x}) \coloneqq x_0 + \|\tilde{\boldsymbol{x}}\|, \boldsymbol{c}_1(\boldsymbol{x}) \coloneqq \frac{1}{2}\begin{bmatrix} 1 \\ \frac{\tilde{\boldsymbol{x}}}{\|\tilde{\boldsymbol{x}}\|} \end{bmatrix},$$

$$\lambda_2(\boldsymbol{x}) \coloneqq x_0 - \|\tilde{\boldsymbol{x}}\|, \boldsymbol{c}_1(\boldsymbol{x}) \coloneqq \frac{1}{2}\begin{bmatrix} 1 \\ \frac{-\tilde{\boldsymbol{x}}}{\|\tilde{\boldsymbol{x}}\|} \end{bmatrix}.$$

If $\boldsymbol{x}$ is clear from context, we do not have to write it out, e.g. $\lambda_1 = \lambda_1(\boldsymbol{x})$. From the decomposition we get $\boldsymbol{x} = \lambda_1\boldsymbol{c}_1 + \lambda_2\boldsymbol{c}_2$ and call the set of eigenvalues $\{\boldsymbol{c}_1, \boldsymbol{c}_2\}$ $\boldsymbol{x}$'s *Jordan frame*.

Proposition 1 (Properties of Jordan frames)**.** *For $\boldsymbol{x} \in \mathbb{R}^n$ with Jordan frame $\{\boldsymbol{c}_1, \boldsymbol{c}_2\}$, the following holds:*

1. $\boldsymbol{c}_1 \circ \boldsymbol{c}_2 = 0$
2. $\boldsymbol{c}_1 \circ \boldsymbol{c}_1 = \boldsymbol{c}_1$ *and* $\boldsymbol{c}_2 \circ \boldsymbol{c}_2 = \boldsymbol{c}_2$
3. $\boldsymbol{c}_1, \boldsymbol{c}_2$ *are of the form* $\left(\frac{1}{2}; \pm\tilde{\boldsymbol{c}}\right)$ *with* $\|\tilde{\boldsymbol{c}}\| = \frac{1}{2}$

Similarly to a matrix being (semi)definite iff all eigenvalues are positive (nonnegative), we get:

Proposition 2. *For $\boldsymbol{x} \in \mathbb{R}^n$ with eigenvalues λ_1, λ_2, the following holds:*

1. $\boldsymbol{x} \in \mathcal{L}^n$ *iff* $\lambda_1 \geq 0$ *and* $\lambda_2 \geq 0$.
2. $\boldsymbol{x} \in \operatorname{int} \mathcal{L}^n$ *iff* $\lambda_1 > 0$ *and* $\lambda_2 > 0$.

Using the decomposition we define power $\boldsymbol{x}^p$ for $p \in \mathbb{R}$ as $\boldsymbol{x}^p \coloneqq \lambda_1^p\boldsymbol{c}_1 + \lambda_2^p\boldsymbol{c}_2$ and as such in particular

$$\boldsymbol{x}^{-1} = \frac{1}{\lambda_1}\boldsymbol{c}_1 + \frac{1}{\lambda_2}\boldsymbol{c}_2, \text{if } \lambda_1, \lambda_2 \neq 0,$$
$$\boldsymbol{x}^{\frac{1}{2}} = \sqrt{\lambda_1}\boldsymbol{c}_1 + \sqrt{\lambda_2}\boldsymbol{c}_2, \text{if } \boldsymbol{x} \in \mathcal{L}^n.$$

We further define the Frobenius[1] and spectral norm

$$\|\boldsymbol{x}\|_F \coloneqq \sqrt{\lambda_1^2 + \lambda_2^2} = \sqrt{2}\|\boldsymbol{x}\|,$$
$$\|\boldsymbol{x}\|_2 \coloneqq \max\{|\lambda_1|, |\lambda_2|\} = |x_0| + \|\tilde{\boldsymbol{x}}\|.$$

Now we want to find an equivalent operation to $q_X : Y \mapsto XYX$. For this we define the *quadratic representation* $Q_{\boldsymbol{x}}$:

$$Q_{\boldsymbol{x}} \coloneqq 2\operatorname{Arw}^2(\boldsymbol{x}) - \operatorname{Arw}(\boldsymbol{x}^2) = \begin{bmatrix} \|\boldsymbol{x}\|^2 & 2x_0\tilde{\boldsymbol{x}}^\mathsf{T} \\ 2x_0\tilde{\boldsymbol{x}} & \lambda_1\lambda_2 I_n + 2\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^\mathsf{T} \end{bmatrix}$$

With this we define the shorthand $T_{\boldsymbol{x}} \coloneqq Q_{\boldsymbol{x}^{1/2}}$.

Finally, we extend the definitions to block vectors of rank r over the product of Lorentz cones:

1. $\boldsymbol{x} \circ \boldsymbol{y} \coloneqq (\boldsymbol{x}_1 \circ \boldsymbol{y}_1; \ldots; \boldsymbol{x}_r \circ \boldsymbol{y}_r)$

[1] *Frobenius norm* of a matrix $\|A\|_F \coloneqq \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}$

2. $\operatorname{Arw}(\boldsymbol{x}) := \operatorname{Arw}(\boldsymbol{x}_1) \oplus \cdots \oplus \operatorname{Arw}(\boldsymbol{x}_r), Q_{\boldsymbol{x}} := Q_{\boldsymbol{x}_1} \oplus \cdots \oplus Q_{\boldsymbol{x}_r}$
3. The $2r$ eigenvectors of $\boldsymbol{x}$ are the eigenvectors of its blocks extended with 0s in the other blocks.
4. $\boldsymbol{e} := (\boldsymbol{e}_1; \ldots; \boldsymbol{e}_r)$
5. $\|\boldsymbol{x}\|_F^2 := \sum_{i=1}^r \|\boldsymbol{x}_i\|_F^2, \|\boldsymbol{x}\|_2 := \max_i \|\boldsymbol{x}_i\|_2$
6. $\boldsymbol{x}^p := \left(\boldsymbol{x}_1^p; \ldots; \boldsymbol{x}_r^p\right)$

From these definitions we get some useful properties.

Claim 1 (Algebraic properties). *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{L}$, then*

1. $\|\boldsymbol{x} + \boldsymbol{y}\|_2 \leq \|\boldsymbol{x}\|_2 + \|\boldsymbol{y}\|_2$.
2. $\|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_F$.
3. $\lambda_{\min}(\boldsymbol{x} + \boldsymbol{y}) \geq \lambda_{\min}(\boldsymbol{x}) - \|\boldsymbol{y}\|_2$.

Proposition 3 (Properties of $Q_{\boldsymbol{x}}$ [3]). *For $x \in \operatorname{int} \mathcal{L}$ the following holds:*

1. $Q_{\boldsymbol{x}}\boldsymbol{e} = \boldsymbol{x}^2$, *and thus* $T_{\boldsymbol{x}}\boldsymbol{e} = \boldsymbol{x}$.
2. $Q_{\boldsymbol{x}^{-1}} = Q_{\boldsymbol{x}}^{-1}$, *and more generally* $Q_{\boldsymbol{x}^p} = Q_{\boldsymbol{x}}^p$ *for all* $p \in \mathbb{R}$.
3. $\|Q_{\boldsymbol{x}}\|_2 = \|\boldsymbol{x}\|_2^2$, *and thus* $\|T_{\boldsymbol{x}}\|_2 = \|\boldsymbol{x}\|_2$.[2]
4. $Q_{\boldsymbol{x}}$ *preserves* $\mathcal{L}$, *i.e.* $Q_{\boldsymbol{x}}(\mathcal{L}) = \mathcal{L}$ *and* $Q_{\boldsymbol{x}}(\operatorname{int} \mathcal{L}) = \operatorname{int} \mathcal{L}$.

2.4. Interior-point methods

The IPM solves a SOCP by computing a sequence of feasible solutions with decreasing duality gap using Newton's method. First we reformulate the original problem to the Karush-Kuhn-Tucker optimality condition

$$\begin{aligned} A\boldsymbol{x} = \boldsymbol{b}, A^\mathsf{T}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c} \\ \boldsymbol{x} \circ \boldsymbol{s} = \nu\boldsymbol{e}, \boldsymbol{x} \in \mathcal{L}, \boldsymbol{s} \in \mathcal{L}. \end{aligned} \tag{3}$$

Because $\boldsymbol{x} \circ \boldsymbol{s} = \nu\boldsymbol{e}$, $\nu > 0$ is equal to the duality gap μ. Decreasing ν by a factor $\sigma < 1$ each iteration results in a sequence of feasible solutions $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s})$ of (3), converging towards an optimal solution of the original SOCP problem, as $\mu = \nu \to 0$. The trace of this sequence is called *central path*.

To obtain the next iteration we solve the *Newton system*

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^\mathsf{T} & I \\ \operatorname{Arw}(\boldsymbol{s}) & 0 & \operatorname{Arw}(\boldsymbol{x}) \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{y} \\ \Delta\boldsymbol{s} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} - A\boldsymbol{x} \\ \boldsymbol{c} - \boldsymbol{s} - A^\mathsf{T}\boldsymbol{y} \\ \sigma\mu\boldsymbol{e} - \boldsymbol{x} \circ \boldsymbol{s} \end{bmatrix} \tag{4}$$

and set $\boldsymbol{x}_{\text{next}} := \boldsymbol{x} + \Delta\boldsymbol{x}$, $\boldsymbol{y}_{\text{next}} := \boldsymbol{y} + \Delta\boldsymbol{y}$ and $\boldsymbol{s}_{\text{next}} := \boldsymbol{s} + \Delta\boldsymbol{s}$, giving us a new solution of (3) with $\nu = \sigma\mu$.

[2] *Matrix spectral norm* $\|A\|_2 = \max_\lambda \sqrt{\lambda(A^*A)}$, where λ is an eigenvalue of A^*A.

2.5. Quantum linear algebra

To solve linear systems like the Newton system (4) on quantum computers we need data structures for storing and loading vectors and matrices in quantum systems, as well as methods to perform basic linear operations on those.

For a unit vector $\boldsymbol{x} \in \mathbb{R}^{2^k}$ with $\|\boldsymbol{x}\| = 1$, $|\boldsymbol{x}\rangle$ refers to the quantum state $|\boldsymbol{x}\rangle := \sum_{i=0}^{2^k-1} x_i|i\rangle \in \mathbb{C}^{2^k}$. This gives a representation of a n-dimensional unit vector in a $\lceil \log n \rceil$-qubit system. To enable matrix operations on such quantum states we need a way to represent a matrix as a unitary transformation $U : \mathbb{C}^{2^k} \to \mathbb{C}^{2^k}$. For this we use the *block encoding* framework as described in [4].

Definition. *For a symmetric matrix $A \in \mathbb{R}^{n\times n}$, the ℓ-qubit unitary matrix $U \in \mathbb{C}^{2^\ell \times 2^\ell}$ is A's (ζ, ℓ)-block encoding if $U = \begin{bmatrix} A/\zeta & \cdot \\ \cdot & \cdot \end{bmatrix}$. The block encoding of a non-symmetric matrix $B \in \mathbb{R}^{n\times m}$ is a block encoding of its symmetric representation* $\operatorname{sym}(B) := \begin{bmatrix} 0 & B \\ B^\mathsf{T} & 0 \end{bmatrix}$.

What is contained in the dotted part of the unitary is left up to the implementation. Now that we have encodings for vectors and matrices we need a data structure implementing them efficiently on quantum computers, giving us access to matrix rows or columns in poly-logarithmic time. [4] provides one such structure built on top of the QRAM model. In QRAM an array $[\boldsymbol{b}_1, ..., \boldsymbol{b}_m]$ of w-bit strings can be accessed in poly-logarithmic time with the unitary transform

$$|i\rangle|0\rangle^{\otimes w} \mapsto |i\rangle|\boldsymbol{b}_i\rangle, \text{for } i \in [m].$$

Theorem 1 (Block encodings using QRAM [4, 5])**.** *There exists a QRAM data structure storing m vectors $\boldsymbol{v}_i \in \mathbb{R}^n$ and matrices $A \in \mathbb{R}^{n\times n}$, access to which enables the following:*

1. *Implement the unitary transform $|i\rangle|0\rangle \mapsto |i\rangle|\boldsymbol{v}_i\rangle$ for $i \in [m]$ in $\tilde{\mathcal{O}}(1)$.*
2. *For $\zeta(A) := \|A\|_2^{-1} \min(\|A\|_F, s_1(A))$ with $s_1 = \max_i \sum_j |A_{i,j}|$, a unitary $(\zeta(A), 2\log n)$-block encoding of A giving access to rows or columns of A in time $\mathcal{O}(\log n)$ can be constructed in a single pass over A. Each entry can be updated in $\mathcal{O}(\log^2 n)$ time.*

To solve linear systems with block encodings we make use of:

Theorem 2 (Quantum linear algebra with block encodings [6, 7])**.** *Given a $(\zeta, \mathcal{O}(\log n))$-block encoding of a matrix $A \in \mathbb{R}^{n\times n}$ with non-zero eigenvalues in the interval $\left[-1, -\frac{1}{\kappa}\right] \cup \left[\frac{1}{\kappa}, 1\right]$ and $\varepsilon > 0$, let t_A be the time it takes to access a row or column of A and let $t_{\boldsymbol{b}}$ be the time it takes to prepare state $|\boldsymbol{b}\rangle$ for a vector $\boldsymbol{b} \in \mathbb{R}^n$.*

A state ε-close to $\left|A^{-1}\boldsymbol{b}\right\rangle$ can be generated in time $\mathcal{O}((t_A\kappa\zeta + t_{\boldsymbol{b}}) \operatorname{polylog}(\kappa\zeta/\varepsilon))$.

Theorem 2 directly enables us to solve linear equations of the form $A\boldsymbol{x} = \boldsymbol{b}$ by generating $\left|A^{-1}\boldsymbol{b}\right\rangle$. To make practical use of this solution we require a way to efficiently extract classical data from this quantum state. In *quantum state tomography* multiple incomplete measurements on a quantum state are performed, which are then used to estimate the quantum state. By increasing the number of measurements higher accuracy can be gained. The tomography algorithm presented in [8] takes advantage of the fact that we are capable of preparing the quantum state arbitrarily often using a unitary transform.

Theorem 3 (Efficient vector state tomography [8])**.** *Given a procedure constructing* $|\boldsymbol{x}\rangle$ *in time* $t_{\boldsymbol{x}}$ *and precision* $\delta > 0$*, one can construct an estimate* $\overline{\boldsymbol{x}} \in \mathbb{R}^n$ *with* $\|\overline{\boldsymbol{x}}\| = 1$ *such that* $\|\boldsymbol{x} - \overline{\boldsymbol{x}}\| \leq \sqrt{7}\delta$ *with probability at least* $1 - 1/n^{0.83}$ *in time* $\mathcal{O}\left(t_{\boldsymbol{x}} \frac{n \log n}{\delta^2}\right)$.

This can be repeated $\tilde{\mathcal{O}}(1)$ times to get success probability at least $1 - 1/\text{poly } n$. Putting all of this together, we can solve $A\boldsymbol{x} = \boldsymbol{b}$ up to error δ in time $\tilde{\mathcal{O}}\left(n \cdot \frac{\kappa\zeta}{\delta^2}\right)$, assuming that A and $\boldsymbol{b}$ are already loaded in QRAM. For well conditioned matrices, especially those with large dimension n, and limited precision requirements, this can lead to a significant speedup over the classical $O(n^\omega)$. Here ω is the *matrix multiplication exponent* with a current theoretical lower bound of roughly 2.37, and a practical value in most implementations closer to 3.

3. A Quantum Interior-point Method

[1] proposes a quantum IPM based on the classical IPM from Section 2.4, incorporating techniques from quantum linear algebra to solve the Newton system, the most computationally expensive part of IPMs. The main steps of this algorithm are as follows:

Given matrix A and vectors $\boldsymbol{b}, \boldsymbol{c}$ in QRAM, parameter ε.

1. Compute feasible $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}, \mu_0)$ and store it in QRAM.
2. For $\mathcal{O}(\sqrt{r}\log(\mu_0/\varepsilon))$ iterations:
   a. Compute $\sigma\mu\boldsymbol{e} - \boldsymbol{x} \circ \boldsymbol{s}$ classically and store it in QRAM.
   b. Prepare Newton system (3).
   c. Get $|(\Delta\boldsymbol{x}; \Delta\boldsymbol{y}; \Delta\boldsymbol{s})\rangle$ from solving the Newton system and get classical approximation $\overline{(\Delta\boldsymbol{x}; \Delta\boldsymbol{y}; \Delta\boldsymbol{s})}$ using tomography.
   d. Update $\boldsymbol{x} \leftarrow \boldsymbol{x} + \overline{\Delta\boldsymbol{x}}$, $\boldsymbol{s} \leftarrow \boldsymbol{s} + \overline{\Delta\boldsymbol{s}}$ and store it in QRAM.
   e. Update $\mu \leftarrow \frac{1}{r}\boldsymbol{x}^\mathsf{T}\boldsymbol{s}$.
3. Output $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s})$.

Algorithm 1: Quantum IPM for SOCP

The first step in showing that this algorithm solves an SOCP problem is to show that feasibility is preserved in each iteration and the solution stays close to the central path. Then asymptotic bounds for the required runtime to achieve a desired level of precision will be given.

From here on out most complete proofs will be omitted. Instead proof ideas will be provided. For further technical detail we refer to the original paper [1].

3.1. Central path

To start we take a closer look at the central path, tracing the exact iterative solutions of (3). As Algorithm 1 works with approximate solutions, we formalize how far we stray from the central path. Define the distance to the central path as $d(\boldsymbol{x}, \boldsymbol{s}, \nu) := \|T_{\boldsymbol{x}}\boldsymbol{s} - \nu\boldsymbol{e}\|_F$ and with this the η-neighborhood

$$\mathcal{N}_\eta(\nu) := \{(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}) \,|\, (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}) \text{ strictly feasible and } d(\boldsymbol{x}, \boldsymbol{s}, \nu) \leq \eta\nu\}.$$

Lemma 1 (Properties of the central path, [1] Lemma 1)**.** *For $\nu > 0$ and $\boldsymbol{x}, \boldsymbol{s} \in \operatorname{int}\mathcal{L}$ the following properties hold:*

1. *For all $\nu > 0$ the duality gap and distance from the central path are related by*

$$|\boldsymbol{x}^\mathsf{T}\boldsymbol{s} - r\nu| \leq \sqrt{\frac{r}{2}} \cdot d(\boldsymbol{x}, \boldsymbol{s}, \nu).$$

2. *The distance from the central path is symmetric in its arguments, i.e. $d(\boldsymbol{x}, \boldsymbol{s}, \nu) = d(\boldsymbol{s}, \boldsymbol{x}, \nu)$.*
3. *Let $\mu = \frac{1}{r}\boldsymbol{x}^\mathsf{T}\boldsymbol{s}$, then*

$$d(\boldsymbol{x}, \boldsymbol{s}, \mu) \leq \eta\mu \quad \Rightarrow \quad (1+\eta)\|\boldsymbol{s}^{-1}\|_2 \geq \|\mu^{-1}\boldsymbol{x}\|_2.$$

We show the first part by rewriting the scaled duality gap to $\boldsymbol{x}^\mathsf{T}\boldsymbol{s} = \frac{1}{2}\sum_{i=1}^{2r}\lambda_i$, where $\{\lambda_i\}_{i=1}^{2r}$ are the eigenvalues of $T_{\boldsymbol{x}}\boldsymbol{s}$. Because we take the absolute value, a lower and an upper bound are needed. With the Cauchy-Schwarz inequality and definition of $d(\boldsymbol{x},\boldsymbol{s},\nu)$ the bounds $\frac{1}{2}\sum_{i=1}^{2r}\lambda_i - r\nu \leq \sqrt{\frac{r}{2}}\cdot d(\boldsymbol{x},\boldsymbol{s},\nu)$ and $r\nu - \frac{1}{2}\sum_{i=1}^{2r}\lambda_i \leq \sqrt{\frac{r}{2}}\cdot d(\boldsymbol{x},\boldsymbol{s},\nu)$ are obtained.

The second part follows from Theorem 10.2 from [3]. For part 3 it is first shown using algebraic properties that $\eta\mu \geq \|T_{\boldsymbol{s}}\boldsymbol{x} - \mu\boldsymbol{e}\|_F \geq \frac{1}{\|\boldsymbol{s}^{-1}\|_2}\cdot\mu\cdot\|\mu^{-1}\boldsymbol{x} - \boldsymbol{s}^{-1}\|_2$ and thus

$$\eta\|\boldsymbol{s}^{-1}\|_2 \geq \|\mu^{-1}\boldsymbol{x} - \boldsymbol{x}^{-1}\|_2 \geq \|\mu^{-1}\boldsymbol{x}\|_2 - \|\boldsymbol{s}^{-1}\|_2,$$

from which the desired inequality directly follows.

3.2. Single iteration correctness

Central to showing the correctness of Algorithm 1 is that feasibility and a close distance to the central path are maintained in each iterative step. This is expressed by the following theorem:

Theorem 4 (Per-iteration correctness, [1] Theorem 4)**.** *For positive constants $\chi = \eta = 0.01$ and $\xi = 0.001$ and a feasible solution $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{s})$ of (1) with $\mu = \frac{1}{r}\boldsymbol{x}^\mathsf{T}\boldsymbol{s}$ and $d(\boldsymbol{x},\boldsymbol{s},\mu) \leq \eta\mu$, the Newton system (4) with $\sigma = 1 - \chi/\sqrt{r}$ has a unique solution $(\Delta\boldsymbol{x},\Delta\boldsymbol{y},\Delta\boldsymbol{s})$. For approximate solutions $\overline{\Delta\boldsymbol{x}}, \overline{\Delta\boldsymbol{s}}$ with*

$$\|\Delta\boldsymbol{x} - \overline{\Delta\boldsymbol{x}}\|_F \leq \frac{\xi}{\|T_{\boldsymbol{x}^{-1}}\|}, \|\Delta\boldsymbol{s} - \overline{\Delta\boldsymbol{s}}\|_F \leq \frac{\xi}{2\|T_{\boldsymbol{s}^{-1}}\|}$$

and $\boldsymbol{x}_{next} := \boldsymbol{x} + \overline{\Delta\boldsymbol{x}}$, $\boldsymbol{s}_{next} := \boldsymbol{s} + \overline{\Delta\boldsymbol{s}}$, the following holds:

1. *The updated solution is strictly feasible, i.e. $\boldsymbol{x}_{next}, \boldsymbol{s}_{next} \in \operatorname{int}\mathcal{L}$.*
2. *The updated solution satisfies $d(\boldsymbol{x}_{next},\boldsymbol{s}_{next},\overline{\mu}) \leq \eta\overline{\mu}$ and $\frac{1}{r}\boldsymbol{x}_{next}^\mathsf{T}\boldsymbol{s}_{next} = \overline{\mu}$ for $\overline{\mu} = \overline{\sigma}\mu, \overline{\sigma} = 1 - \frac{\alpha}{\sqrt{r}}$ and a constant $0 < \alpha \leq \chi$.*

To show Theorem 4 we will first rescale the vectors $\boldsymbol{x}$ and $\boldsymbol{s}$, so that they share the same Jordan frame. With these rescaled vectors we can then show that strict feasibility is maintained each iteration, showing part 1 of the theorem, and then in a second step that path closeness is kept, showing part 2.

3.3. Rescaling

We rescale $\boldsymbol{x}$ and $\boldsymbol{s}$ to $\boldsymbol{x}'$ and $\boldsymbol{s}'$ to obtain two desired properties. They share the same Jordan frame and the duality gap is 1. From then on it suffices to work only with the rescaled version, as their feasibility and closeness to the path implies the same properties for the original entities. We scale by $T_{\boldsymbol{x}}$ to bring them into the same Jordan frame and by μ^{-1} to normalize the duality gap:

$$\boldsymbol{x}' := T_{\boldsymbol{x}}^{-1}\boldsymbol{x} = \boldsymbol{e} \text{ and } \boldsymbol{s}' := \mu^{-1}T_{\boldsymbol{x}}\boldsymbol{s}$$

This also results in

$$\Delta\boldsymbol{x}' = T_{\boldsymbol{x}}^{-1}\Delta\boldsymbol{x} \text{ and } \Delta\boldsymbol{x}' = \mu^{-1}T_{\boldsymbol{x}}\Delta\boldsymbol{s}.$$

The conditional bounds from Theorem 4 on $\overline{\Delta \boldsymbol{x}}$ and $\overline{\Delta \boldsymbol{s}}$ imply simplified bounds on their scaled counterparts:

$$\begin{aligned}
\|\Delta \boldsymbol{x}' - \overline{\Delta \boldsymbol{x}}'\|_F &= \|T_{\boldsymbol{x}^{-1}} \Delta \boldsymbol{x} - T_{\boldsymbol{x}^{-1}} \overline{\Delta \boldsymbol{x}}\|_F \\
&\leq \|T_{\boldsymbol{x}^{-1}}\| \, \| \Delta \boldsymbol{x} - \overline{\Delta \boldsymbol{x}}\|_F \leq \xi \\
\|\Delta \boldsymbol{s}' - \overline{\Delta \boldsymbol{s}}'\|_F &= \mu^{-1} \, \|T_{\boldsymbol{x}} \Delta \boldsymbol{s} - T_{\boldsymbol{x}} \overline{\Delta \boldsymbol{s}}\|_F \\
&\leq \mu^{-1} \, \|T_{\boldsymbol{x}}\| \, \| \Delta \boldsymbol{s} - \overline{\Delta \boldsymbol{s}}\|_F \\
&= \mu^{-1} \, \|\boldsymbol{x}\|_2 \, \| \Delta \boldsymbol{s} - \overline{\Delta \boldsymbol{s}}\|_F \\
&\leq (1+\eta) \, \|\boldsymbol{s}^{-1}\|_2 \, \| \Delta \boldsymbol{s} - \overline{\Delta \boldsymbol{s}}\|_F \text{ by Lemma 1.3} \\
&\leq 2 \, \|T_{\boldsymbol{s}^{-1}}\| \, \| \Delta \boldsymbol{s} - \overline{\Delta \boldsymbol{s}}\|_F \leq \xi
\end{aligned}$$

Claim 2 ([1] Claim 2)**.** *The following holds for the scaled vectors $\boldsymbol{x}'$ and $\boldsymbol{s}'$:*

1. *The scaled duality gap is $\frac{1}{r} \boldsymbol{x}'^{\mathsf{T}} \boldsymbol{s}' = 1$.*
2. *$d(\boldsymbol{x}, \boldsymbol{s}, \mu) \leq \eta\mu$ is equivalent to $\|\boldsymbol{s}' - \boldsymbol{e}\| \leq \eta$.*
3. *$d(\boldsymbol{x}, \boldsymbol{s}, \mu\sigma) = \mu \cdot d(\boldsymbol{x}', \boldsymbol{s}', \sigma)$ for all $\sigma > 0$.*

These properties directly follow from the rescaling. Assuming $\boldsymbol{x}'_{\text{next}}, \boldsymbol{s}'_{\text{next}} \in \operatorname{int} \mathcal{L}$ we also get $\boldsymbol{x}_{\text{next}} = T_{\boldsymbol{x}}^{-1} \boldsymbol{x}'_{\text{next}} \in \operatorname{int} \mathcal{L}$ and $\boldsymbol{s}_{\text{next}} = \mu T_{\boldsymbol{x}} \boldsymbol{s}'_{\text{next}} \in \operatorname{int} \mathcal{L}$. This in combination with Claim 2 implies that it is sufficient to show the properties of Theorem 4 in the scaled case.

3.4. Maintaining strict feasibility

Here we show the first part of Theorem 4.

Lemma 2 ([9] Lemma 6)**.** *For distance to the central path η and duality gap μ, we have the following bounds for the scaled direction:*

$$\|\Delta \boldsymbol{x}'\|_F \leq \frac{\Theta}{\sqrt{2}}, \;\; \|\Delta \boldsymbol{s}'\|_F \leq \Theta\sqrt{2},$$

$$where \; \Theta = \frac{2\sqrt{\frac{\eta^2}{2} + (1-\sigma)^2 r}}{1 - 3\eta}.$$

These bounds on the increments enable us to show that strict feasibility is preserved.

Lemma 3 ([1] Lemma 3)**.** *For $\eta = \chi = 0.01$ and $\xi = 0.001$, $\boldsymbol{x}'_{next}$ and $\boldsymbol{s}'_{next}$ are strictly feasible, i.e. $\boldsymbol{x}'_{next}, \boldsymbol{s}'_{next} \in \operatorname{int} \mathcal{L}$.*

Proof. By Lemma 2 and Claim 1, $\lambda_{\min}(\boldsymbol{x}'_{\text{next}}) \geq \lambda_{\min}(\boldsymbol{x}') - \|\overline{\Delta \boldsymbol{x}}'\|_F \geq 1 - \frac{\Theta}{\sqrt{2}} - \xi$. With $d(\boldsymbol{x}, \boldsymbol{s}, \mu) \leq \eta\mu$, it follows that $d(\boldsymbol{x}', \boldsymbol{s}', 1) \leq \eta$, and

$$\eta^2 \geq \|\boldsymbol{s}' - \boldsymbol{e}\|_F^2 = \sum_{i=1}^{2r} (\lambda_i(\boldsymbol{s}') - 1)^2.$$

This grants us $\lambda_i(\boldsymbol{s}') \in [1-\eta, 1+\eta] \; \forall i \in [2r]$. Again with Lemma 2 and Claim 1 we get

$$\begin{aligned}\lambda_{\min}(\boldsymbol{s}'_{\text{next}}) &\geq \lambda_{\min}(\boldsymbol{s}' + \Delta\boldsymbol{s}') - \|\overline{\Delta\boldsymbol{s}}' - \Delta\boldsymbol{s}'\|_F \\ &\geq \lambda_{\min}(\boldsymbol{s}') - \|\Delta\boldsymbol{s}'\|_F - \|\overline{\Delta\boldsymbol{s}}' - \Delta\boldsymbol{s}'\|_F \\ &\geq 1 - \eta - \Theta\sqrt{2} - \xi.\end{aligned}$$

With $\eta = \chi = 0.01$ and $\xi = 0.001$ this results in $\lambda_{\min}(\boldsymbol{x}'_{\text{next}}) \geq 0.8, \lambda_{\min}(\boldsymbol{x}'_{\text{next}}) \geq 0.8$. □

3.5. Maintaining path closeness

The second part of Theorem 4 is shown by the next two lemmas.

Lemma 4 ([1] Lemma 4). *Let $\eta = \chi = 0.01, \xi = 0.001$, and α any $0 < \alpha \leq \chi$, then for $\overline{\sigma} = 1 - \alpha/\sqrt{r}$ the distance to the central path is maintained with $d(\boldsymbol{x}'_{next}, \boldsymbol{s}'_{next}, \overline{\sigma}) < \eta\overline{\sigma}$.*

To show Lemma 4 the first step is to bound

$$d(\boldsymbol{x}'_{\text{next}}, \boldsymbol{s}'_{\text{next}}, \overline{\sigma}) \leq \left\|T_{\boldsymbol{x}'_{\text{next}}}\right\|_F \cdot \left\|\boldsymbol{s}'_{\text{next}} - \overline{\sigma} \cdot \boldsymbol{x}'^{-1}_{\text{next}}\right\|_F.$$

The left part becomes $\|T_{\boldsymbol{x}'_{\text{next}}}\|_F \leq 1 + \frac{\Theta}{\sqrt{2}} + \xi$ and for the right part we define $\boldsymbol{z} := \boldsymbol{s}'_{\text{next}} - \overline{\sigma} \cdot \boldsymbol{x}'^{-1}_{\text{next}}$, which then is split into three parts that are bounded individually:

$$\boldsymbol{z} = \underbrace{\boldsymbol{s}' + \overline{\Delta\boldsymbol{s}}' - \overline{\sigma}\boldsymbol{e} + \overline{\Delta\boldsymbol{x}}'}_{\boldsymbol{z}_1} + \underbrace{(\overline{\sigma} - 1)\overline{\Delta\boldsymbol{x}}'}_{\boldsymbol{z}_2} + \underbrace{\overline{\sigma}\left(\boldsymbol{e} - \overline{\Delta\boldsymbol{x}}' - \left(\boldsymbol{e} + \overline{\Delta\boldsymbol{x}}'\right)^{-1}\right)}_{\boldsymbol{z}_3}$$

1. $\|\boldsymbol{z}_1\|_F \leq 2\xi + \frac{\chi}{\sqrt{r}} + \frac{3}{\sqrt{2}}\eta\Theta$
2. $\|\boldsymbol{z}_2\|_F \leq \frac{\chi}{\sqrt{r}}\left(\frac{\Theta}{\sqrt{2}} + \xi\right)$
3. $\|\boldsymbol{z}_3\|_F \leq \overline{\sigma}\left(\frac{\Theta^2}{2-\sqrt{2}\Theta} + \xi + \frac{\xi}{\left(\left(1-\Theta/\sqrt{2}-\xi\right)\right)^2}\right)$

Plugging in $\chi = \eta = 0.01, \xi = 0.001$, this then gives $d(\boldsymbol{x}'_{\text{next}}, \boldsymbol{s}'_{\text{next}}, \overline{\sigma}) \leq 0.005\overline{\sigma} \leq \eta\overline{\sigma}$.

Lemma 5 ([1] Lemma 5). *For the same constants and $\alpha = 0.005$ the updated solution satisfies $\frac{1}{r}\boldsymbol{x}'^{\mathsf{T}}_{next}\boldsymbol{s}'_{next} = 1 - \frac{\alpha}{\sqrt{r}}$.*

With the fact that $\boldsymbol{x}^{\mathsf{T}}_{\text{next}}\boldsymbol{s}_{\text{next}} = \frac{\mu}{r}\boldsymbol{x}'^{\mathsf{T}}_{\text{next}}\boldsymbol{s}'_{\text{next}}$ and Lemma 1, a bound for $\frac{1}{r}\boldsymbol{x}'^{\mathsf{T}}_{\text{next}}\boldsymbol{s}'_{\text{next}}$ is found, into which $d(\boldsymbol{x}'_{\text{next}}, \boldsymbol{s}'_{\text{next}}, \sigma) \leq 0.005\sigma$ from the proof of Lemma 4 is inserted, to get the final bound.

3.6. Final complexity and feasibility

The desired precision for solving the Newton system is dependent on the norms

$$\begin{aligned}\|T_{\boldsymbol{x}^{-1}}\| = \|\boldsymbol{x}^{-1}\| = \lambda_{\min}(\boldsymbol{x})^{-1} \text{ and} \\ \|T_{\boldsymbol{s}^{-1}}\| = \|\boldsymbol{s}^{-1}\| = \lambda_{\min}(\boldsymbol{s})^{-1}.\end{aligned}$$

To satisfy Theorem 4 in the i-th iteration the tomography precision has to be at least

$$\delta_t := \frac{\xi}{4}\min\{\lambda_{\min}(\boldsymbol{x}_i), \lambda_{\min}(\boldsymbol{s}_i)\}$$

and the overall precision is chosen to be $\delta := \min_i \delta_i$. We do the same for the parameters of the block encoding of the Newton matrix $\kappa := \max_i \kappa_i$ and $\zeta := \max_i \zeta_i$.

Theorem 5 ([1] Theorem 5)**.** *For an SOCP as stated in (1) with $A \in \mathbb{R}^{m\times n}$ for $m \leq n$, and $\mathcal{L}$ of rank r, Algorithm 1 achieves duality gap ε in time*

$$\mathcal{O}\left(\sqrt{r}\log\left(\frac{\mu_0}{\varepsilon}\right)\cdot\frac{n\kappa\zeta}{\delta^2}\log\left(\frac{\kappa\zeta}{\delta}\right)\right).$$

This complexity follows as a product of the number of iterations and the complexity of the quantum tomography.

Throughout the execution, due to the imprecision of tomography, the linear constraints $A\boldsymbol{x} = \boldsymbol{b}$ and $A^\mathsf{T}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c}$ are never satisfied exactly. It is now left to show that this error is only dependent on the tomography precision and not accumulated, so that the final solution has an acceptable level of accuracy.

Theorem 6 ([1] Theorem 6)**.** *For an SOCP as stated in Theorem 5, after T iterations the (linear) infeasibility of the final iterate $(\boldsymbol{x}_T, \boldsymbol{y}_T, \boldsymbol{s}_T)$ is bounded as*

$$\|A\boldsymbol{x}_T - \boldsymbol{b}\| \leq \delta\|A\|,$$
$$\|A^\mathsf{T}\boldsymbol{y}_T + \boldsymbol{s}_T - \boldsymbol{c}\| \leq \delta(\|A\| + 1).$$

Proof. The Newton system in the T-th iteration has constraint $A\Delta\boldsymbol{x}_T = \boldsymbol{b} - A\boldsymbol{x}_{T-1}$. This can recursively be expanded to

$$A\Delta\boldsymbol{x}_T = \boldsymbol{b} - Ax_0 - \sum_{t=1}^{T-1}\overline{\Delta\boldsymbol{x}_t} = -\sum_{t=1}^{T-1}\overline{\Delta\boldsymbol{x}_t}. \tag{5}$$

From the iterative definition of solutions it follows that

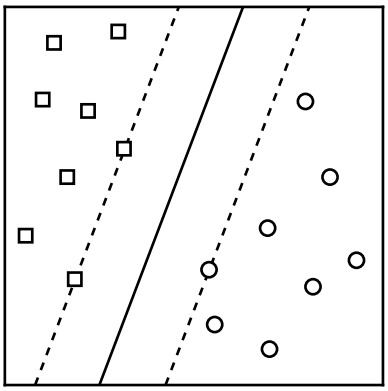
$$A\boldsymbol{x}_T - \boldsymbol{b} = A\sum_{t=1}^{T}\overline{\Delta\boldsymbol{x}_t} \overset{(5)}{=} A\Big(\overline{\Delta\boldsymbol{x}_T} - \Delta\boldsymbol{x}_T\Big).$$

Analogously we get

$$A^\mathsf{T}\boldsymbol{y}_T - \boldsymbol{b} = A\Big(\overline{\Delta\boldsymbol{y}_T} - \Delta\boldsymbol{y}_T\Big) + \Big(\overline{\Delta\boldsymbol{s}_T} - \Delta\boldsymbol{s}_T\Big).$$

Bounding the norms on these equations directly gives the desired inequalities. □

4. Quantum Support-Vector Machines



SVMs are supervised learning models for data classification. Given a set of *training vectors* $X = \{\boldsymbol{x}_i \in \mathbb{R}^n \mid i \in [m]\}$ and their *labels* $y_i \in \{-1, 1\}$, the goal is to find the *hyperplane* separating vectors with differing labels. For *linear* SVMs this hyperplane is defined by the set of $\boldsymbol{x} \in \mathbb{R}^n$ with

$$\boldsymbol{w}^\mathsf{T}\boldsymbol{x} - b = 0,$$

where $\boldsymbol{w}$ is the normal vector of the hyperplane and $\frac{b}{\|\boldsymbol{w}\|}$ its distance from the origin.

In their most basic formulation, that of *hard-margin* SVMs, it is required that the data is linearly separable with such a hyperplane. The goal is then to find a hyperplane keeping as much distance from all samples and thus separating the two classes as clearly as possible. The area around this hyperplane, bounded by two parallel hyperplanes of equal distance, is called the *margin*. For all samples $\boldsymbol{x}$ with label 1 directly on the boundary, this gives us the equation $\boldsymbol{w}^\mathsf{T}\boldsymbol{x} - b = 1$ and analogously $\boldsymbol{w}^\mathsf{T}\boldsymbol{x} - b = -1$ for those labeled -1. Combining these equations and extending them to non-boundary samples we get the criterion

$$y_i(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i - b) \geq 1.$$

As the distance between the boundaries is $\frac{2}{\|\boldsymbol{w}\|}$, to get a wide margin we must minimize $\|\boldsymbol{w}\|$ and hence arrive at the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b) \geq 1, \ \ \forall i \in [m] \end{aligned}$$

To allow for cases where the data is not linearly separable this is extended to *soft-margin* SVMs, where we consider the *hinge loss*. For misclassified samples $\boldsymbol{x}_i$ it is $\xi_i = 1 - y_i(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i - b)$, proportional to the distance from the proper boundary of the sample. For correctly classified samples $\boldsymbol{x}_i$ it is $\xi_i = 0$. The goal is then not only to minimize $\|\boldsymbol{w}\|$, but also the cumulative hinge loss. To give control over how much to value a wide margin over correctly classifying more samples, the hyperparameter $C > 0$ is introduced. We get the final optimization problem

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0 \ \ \forall i \in [m]. \end{aligned} \tag{6}$$

Soft-margin can also be useful for linearly separable data, as it allows for the misclassification of outliers, that might otherwise prevent more meaningful hyperplanes.

While the results of [1] only apply to linear soft-margin SVMs, we also mention that SVMs can be extended to non-linear classifiers with the so called *kernel trick*. The idea is that we map the samples and $\boldsymbol{w}$ from our vector space $\mathcal{X}$ to another, often higher-dimensional space $\mathcal{V}$, with a function $\varphi : \mathcal{X} \to \mathcal{V}$. To keep performance reasonable by avoiding to explicitly map

all vectors to a higher-dimensional space, all dot products are replaced by a *kernel function* k satisfying $k(\boldsymbol{x}, \boldsymbol{y}) = \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{y}) \rangle_{\mathcal{V}}$.

4.1. Reduction

To solve SVMs with the quantum IPM from Section 3 we now have to reformulate the optimization problem (6) as an SOCP problem. The minimization term lets us use the primal form (1). Implementing $C \sum_{i=1}^{m} \xi_i = [C^m]\boldsymbol{\xi}$ is straightforward. C^m is C repeated m times. To minimize $\|\boldsymbol{w}\|^2$ we introduce the auxiliary variable t, which we minimize instead, and use it to create an upper bound for $\|\boldsymbol{w}\|^2$. Let $\boldsymbol{t} \coloneqq (t+1; t; \boldsymbol{w})$, then $\boldsymbol{t} \in \mathcal{L}^{n+2}$ grants us

$$\boldsymbol{t} \in \mathcal{L}^{n+2} \Leftrightarrow (t+1)^2 \geq t^2 + \|\boldsymbol{w}\|^2 \Leftrightarrow 2t + 1 \geq \|\boldsymbol{w}\|^2.$$

Thus, by minimizing t we also minimize $\|\boldsymbol{w}\|^2$. In practice this might affect the choice of C as $t \neq \|\boldsymbol{w}\|^2$. We put these parts together and rewrite the conditions from (6) to get the final reformulation

$$\begin{aligned} \min_{\boldsymbol{t},b,\boldsymbol{\xi}} \quad & \begin{bmatrix} 0 & 1 & 0^n & 0 & C^m \end{bmatrix} \begin{bmatrix} \boldsymbol{t} \\ b \\ \boldsymbol{\xi} \end{bmatrix} \\ \text{s.t.} \quad & \begin{bmatrix} 0 & 0 & & 1 & \\ \vdots & \vdots & X^\mathsf{T} & \vdots & \operatorname{diag}(\boldsymbol{y}) \\ 0 & 0 & & 1 & \\ 1 & -1 & 0^n & 0 & 0^m \end{bmatrix} \begin{bmatrix} \boldsymbol{t} \\ b \\ \boldsymbol{\xi} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ 1 \end{bmatrix} \\ & \boldsymbol{t} \in \mathcal{L}^{n+2}, b \in \mathcal{L}^1, \xi_i \in \mathcal{L}^1 \quad \forall i \in [m]. \end{aligned} \tag{7}$$

Here $X = [\boldsymbol{x}_1 \ \cdots \ \boldsymbol{x}_m] \in \mathbb{R}^{n \times m}$ is the matrix containing the samples as columns. Note that $a \in \mathcal{L}^1$ is equivalent to $a \geq 0$. This enforces $\xi_i \geq 0$ for $i \in [m]$. At the same time it also limits us to solutions $b > 0$. This poses no problem, as for any solution with $b < 0$ and weights $\boldsymbol{w}$ an equivalent one with bias $-b > 0$ and weights $-\boldsymbol{w}$ can be found. For $i \in [m]$ the i-fd row of the linear system is

$$\boldsymbol{x}_i^\mathsf{T} \boldsymbol{w} + b + y_i \xi_i = y_i \Leftrightarrow y_i(\boldsymbol{w}^\mathsf{T} \boldsymbol{x}_i + b) = 1 - \xi_i.$$

The final row makes sure that $\boldsymbol{t}$ is of the desired shape.

4.2. Least-squares SVM

In the original paper [1] it is mentioned that a similar formulation can also be made for *least-squares* SVMs (LS-SVMs). We will now suggest such a reduction ourselves.

LS-SVMs as introduced in [10] are an alternative version of SVMs. They consider not the sum of the hinge loss, but instead the least-squares error:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \|\boldsymbol{w}\|^2 + C\|\boldsymbol{\xi}\|^2 \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^\mathsf{T} \boldsymbol{x}_i + b) = 1 - \xi_i \quad \forall i \in [m] \end{aligned}$$

This results in all samples, except for those directly on their classes boundary, having a positive loss, i.e. samples can also be penalized for being "too correct". As such the notion of a

margin, in which there ideally would be no samples, is lost. This effects which solutions can be found using LS-SVMs. In turn the solution is more robust to outliers and can be optimized more efficiently.

To turn this into a SOCP problem, we modify problem (7) by applying the same trick used to implement the minimization of $\|\boldsymbol{w}\|^2$. We introduce the auxiliary vector $\boldsymbol{u} = (u+1; u; \boldsymbol{\xi})$.

$$\min_{\boldsymbol{t},b,\boldsymbol{\xi}} \quad [0 \;\; 1 \;\; 0^n \;\; 0 \;\; 0 \;\; C \;\; 0^m] \begin{bmatrix} \boldsymbol{t} \\ b \\ \boldsymbol{u} \end{bmatrix}$$

$$\text{s.t.} \quad \begin{bmatrix} 0 & 0 & & 1 & 0 & 0 & \\ \vdots & \vdots & X^\mathsf{T} & \vdots & \vdots & \vdots & \operatorname{diag}(\boldsymbol{y}) \\ 0 & 0 & & 1 & 0 & 0 & \\ 1 & -1 & 0^n & 0 & 0 & 0 & 0^m \\ 0 & 0 & 0^n & 0 & 1 & -1 & 0^m \end{bmatrix} \begin{bmatrix} \boldsymbol{t} \\ b \\ \boldsymbol{u} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ 1 \\ 1 \end{bmatrix}$$

$$\boldsymbol{t} \in \mathcal{L}^{n+2}, b \in \mathcal{L}^1, \boldsymbol{u} \in \mathcal{L}^{m+2}$$

As suggested in [1], this results in $\mathcal{O}(1)$ conic constraints and as such the IPM converges in $\tilde{\mathcal{O}}(1)$ iterations, showing the runtime advantages LS-SVMs can have over traditional SVMs.

4.3. Experimental results

In [1] a simulated experiment was performed to get insights into the realistic complexity and accuracy of solving SVM problems using Algorithm 1. The IPM is implemented classically and noise is added to each solution of the Newton system. The amount of noise was chosen according to the accuracy bounds from Theorem 4.

For each SVM instance the dimension $n \in [2^2, 2^9]$, number of samples $m = 2n$, and probability of a sample being misclassified by an optimal solution $p \in \{0, 0.1, ..., 0.9, 1\}$ were chose uniformly. The target duality gap was $\varepsilon = 0.1$, and C was chosen to be 1. A test set containing $\lfloor m/3 \rfloor$ samples from the same distribution was used to evaluate the resulting SVM classifier.

In total 16.000 instances were generated and each was solved using the simulated QSVM solver and the classical ECOS SOCP solver [11]. The complexity of both approaches was measured,

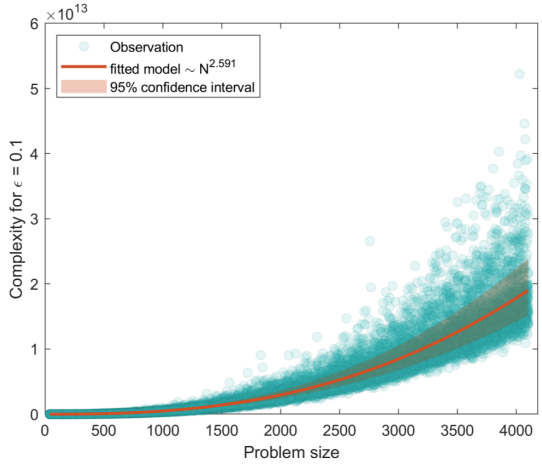


Figure 1: Observed complexity, power law fit, and 95% confidence interval, [1]

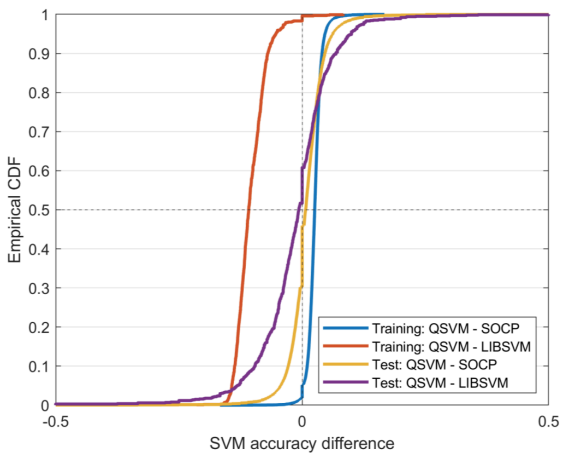


Figure 2: SVM accuracy difference, [1]

the results of which for Algorithm 1 are shown in Figure 1. Algorithm 1 achieved an estimated complexity of $\mathcal{O}(n^{2.591})$, a significant speedup over ECOS' $\mathcal{O}(n^{3.314})$. This polynomial speedup is preserved for classical SOCP solvers in general, as they have complexity $\mathcal{O}(n^{\omega+0.5})$, dependent on the matrix multiplication index ω with a current lower bound of roughly 2.37.

Another 1.000 instances were solved using Algorithm 1 and LIBSVM [12] with a linear kernel. LIBSVM uses the widespread *sequential minimal optimization* algorithm, that, unlike SOCP methods, supports the kernel trick. It achieves a measured complexity of $\mathcal{O}(n^{3.112})$ and therefore is also beat by the theoretical results of the QSVM.

The results of the QSVM were compared with those of ECOS and LIBSVM, and the accuracy difference was accumulated and shown in Figure 2. It can be seen that the QSVM tends to slightly outperform ECOS and on the test set is close to LIBSVM. The biggest difference is a worse performance on the training set compared to LIBSVM.

Bibliography

[1] I. Kerenidis, A. Prakash, and D. Szilágyi, "Quantum algorithms for Second-Order Cone Programming and Support Vector Machines," *Quantum*, vol. 5, p. 427–428, 2021, doi: 10.22331/Q-2021-04-08-427°.

[2] Y. Ye, M. J. Todd, and S. Mizuno, "An O(√nL)-Iteration Homogeneous and Self-Dual Linear Programming Algorithm," *Math. Oper. Res.*, vol. 19, no. 1, pp. 53–67, 1994, doi: 10.1287/MOOR.19.1.53°.

[3] F. Alizadeh and D. Goldfarb, "Second-order cone programming," *Math. Program.*, vol. 95, no. 1, pp. 3–51, 2003, doi: 10.1007/S10107-002-0339-5°.

[4] I. Chakrabarty, S. Khan, and V. Singh, "Dynamic Grover search: applications in recommendation systems and optimization problems," *Quantum Inf. Process.*, vol. 16, no. 6, p. 153–154, 2017, doi: 10.1007/S11128-017-1600-4°.

[5] I. Kerenidis and A. Prakash, "Quantum gradient descent for linear systems and least squares," *Physical Review A*, vol. 101, no. 2, Feb. 2020, doi: 10.1103/physreva.101.022316°.

[6] S. Chakraborty, A. Gilyén, and S. Jeffery, "The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation," in *ICALP*, in LIPIcs, vol. 132. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 1–14. doi: 10.4230/LIPICS.ICALP.2019.33°.

[7] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics," in *STOC*, ACM, 2019, pp. 193–204. doi: 10.1145/3313276.3316366°.

[8] I. Kerenidis and A. Prakash, "A Quantum Interior Point Method for LPs and SDPs," *CoRR*, 2018, [Online]. Available: http://arxiv.org/abs/1808.09266°

[9] R. D. C. Monteiro and T. Tsuchiya, "Polynomial convergence of primal-dual algorithms for the second-order cone program based on the MZ-family of directions," *Math. Program.*, vol. 88, no. 1, pp. 61–83, 2000, doi: 10.1007/PL00011378°.

[10] J. A. K. Suykens, L. Lukas, and J. Vandewalle, "Sparse least squares Support Vector Machine classifiers," in *ESANN*, 2000, pp. 37–42. [Online]. Available: https://www.esann.org/sites/default/files/proceedings/legacy/es2000-352.pdf°

[11] A. Domahidi, E. Chu, and S. P. Boyd, "ECOS: An SOCP solver for embedded systems," in *ECC*, IEEE, 2013, pp. 3071–3076. doi: 10.23919/ECC.2013.6669541°.

[12] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011, doi: 10.1145/1961189.1961199°.