# Quantum Support Vector Machines: An Algorithmic Analysis

Harley Patton, Rob Giometti, Vijitha Sridhar, Marcus Lee

University of California, Berkeley

## Table of Contents

## Introduction

In quantum machine learning, quantum support vector machines achieve an exponential speedup in computation. Where classical support vector machines have complexity $O(M^2(M+N))$, with M the number of training samples and N the number of features, quantum support vector machines have complexity $O(\log MN)$. Quantum improvements in calculating a single inner product, as well as in matrix exponentiation, help create this speedup.

## 1 Least Squares SVM

A support vector machine (SVM) in classical machine learning is a supervised learning algorithm that attempts to find a separating hyperplane between datapoints with a binary label (-1 or 1). This hyperplane is to have the maximum margin from the positive and negative-labeled datapoints.

We can formulate this using a kernel function, which will allows us to compute the similarity between pairs of datapoints. The decision function for SVMs then becomes

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{M} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{1}$$

where the $M$ is the number of datapoints, $\alpha_i$ are the weights given by the hyperplane, $b$ is the bias of the hyperplane, and the kernel function is applied between the test datapoint $\mathbf{x}$ and each datapoint used for training $\mathbf{x}_i$.

Since sometimes the datapoints are not strictly separable, we want to accommodate violations of this decision function. We do so by introducing a slack variable $\xi_i$. We also want to make sure to prevent overfitting the training data, so we employ principles from structural risk minimization that tell us that the risk bound is minimized by the following optimization problem:

$$\min \frac{1}{2}\mathbf{u}^T\mathbf{u} + \frac{1}{2}\gamma\sum_{i=1}^{M}\xi_i^2 \tag{2}$$

subject to the condition based on the decision function - that

$$y_i(\mathbf{u}^T\phi(\mathbf{x}_i) + b) = 1 - \xi_i, \quad i = 1...N \tag{3}$$

This formulation of the optimization problem, which contains a minimization and a set of constraints to satisfy, can be solved by using Lagrange multipliers. We find the saddle point of the Lagrangian function, which will tell us the minimum value under the constraints. In seeking the saddle point, we arrive at the following least squares equation:

$$\begin{pmatrix} 0 & 1^T \\ 1 & K + \gamma^{-1}I \end{pmatrix}\begin{pmatrix} b \\ a \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{y} \end{pmatrix} \tag{4}$$

where y is the label vector, $K$ is the kernel function applied to each pair of vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\gamma$ is a variable attempting to minimize the complexity of the solution to the equation.

This is the least squares formulation of support vector machines. By using quantum mechanics to speed up the kernel matrix calculation and speed up the matrix inversion needed to solve this equation, we can achieve a significant speedup in the complexity of SVMs.

## 2 Kernel Matrix/Function Calculation

Classically, the computation of a single dot product of two training instances is $O(N)$ for constructing an element in the kernel function/matrix. However, through quantum methods we can reduce this to $O(\epsilon^{-1}\log N)$.

For any two training instances $|\mathbf{x}_i\rangle$ and $|\mathbf{x}_j\rangle$, we can reconstruct a particular state by querying quantum random access memory (QRAM) $O(\log N)$ times. Through such queries we construct a state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle|\mathbf{x}_i\rangle + |1\rangle|\mathbf{x}_j\rangle) \tag{5}$$

Then, we estimate another state $|\phi\rangle = \frac{1}{Z}(|\mathbf{x}_i||0\rangle - |\mathbf{x}_j||1\rangle)$ as well as the parameter $Z = |\mathbf{x}_i|^2 + |\mathbf{x}_j|^2$ by evolving the state

$$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes |0\rangle \tag{6}$$

with the Hamiltonian

$$H = (|\mathbf{x}_i| \left|0\right\rangle \left\langle 0\right| + |\mathbf{x}_j| \left|1\right\rangle \left\langle 1\right|) \otimes \sigma_x \tag{7}$$

This results in the state

$$\frac{1}{\sqrt{2}}[\cos|\mathbf{x}_i|t\left|0\right\rangle - \cos|\mathbf{x}_j|t\left|1\right\rangle] \otimes \left|0\right\rangle - \frac{i}{\sqrt{2}}[\sin|\mathbf{x}_i|t\left|0\right\rangle - \sin|\mathbf{x}_j|t\left|1\right\rangle] \otimes \left|1\right\rangle \tag{8}$$

By choosing $t$ such that $|\mathbf{x}_i|t, |\mathbf{x}_j|t << 1$, we then perform a projective measurement through a swap test (described in the Rigetti Forest section) on the state $\left|\psi\right\rangle$ to measure $\left|\phi\right\rangle$ with probability $\frac{1}{2}Z^2t^2$, and this allows us to estimate $Z$ within a predetermined error $\epsilon$ with complexity $O(\epsilon^{-1})$. $Z$ times this probability of successful measurement $\frac{1}{2}Z^2t^2$ gives us the Euclidean distance between the two training instances $|\mathbf{x}_i - \mathbf{x}_j|^2$, and with this we compute the dot product

$$\mathbf{x}_i^T\mathbf{x}_j = \frac{Z - |\mathbf{x}_i - \mathbf{x}_j|^2}{2} \tag{9}$$

with total complexity $O(\epsilon^{-1}\log N)$.

## 3 Optimizing the Least Squares Dual

In order to solve the least squares equation, we must invert

$$F = \begin{pmatrix} 0 & 1^T \\ 1 & K + \gamma^{-1}I \end{pmatrix} \tag{10}$$

Luckily, there is a fast quantum algorithm for matrix inversion. This algorithm relies on our ability to simulate the matrix exponential of F. We begin by splitting $F$ as $F = A + B$.

$$A = \begin{pmatrix} 0 & 1^T \\ 1 & 0 \end{pmatrix} B = \begin{pmatrix} 0 & 0 \\ 0 & K + \gamma^{-1}I \end{pmatrix} \tag{11}$$

After normalizing $F$ by its trace $\hat{F} = \frac{F}{\mathrm{tr}F} = \frac{F}{\mathrm{tr}K}$, we can use the Baker-Campbell-Hausdorff formula to get the matrix exponential.

$$e^{-i\hat{F}\Delta t} = e^{\frac{-iA\Delta t}{\mathrm{tr}B}} e^{\frac{-i\gamma^{-1}I\Delta t}{\mathrm{tr}B}} e^{\frac{-iK\Delta t}{\mathrm{tr}B}} + O(\Delta t^2) \tag{12}$$

Since the kernel matrix K is not sparse, we can use quantum self-analysis to obtain the exponential. Performing the quantum calculation of the dot product and querying the QRAM, we obtain the normalized kernel matrix

$$\hat{K} = \frac{K}{\mathrm{tr}K} = \frac{1}{\mathrm{tr}K}\sum_{i,j=1}^{M} \left\langle \mathbf{x}_i|\mathbf{x}_j\right\rangle |\mathbf{x}_i||\mathbf{x}_j| \left|i\right\rangle \left\langle j\right| \tag{13}$$

Since this is a normalized, hermitian matrix we can use quantum self-analysis to obtain the exponential

$$e^{-i\,\mathcal{L}_{\hat{K}}\ \Delta t} \approx I - i\Delta t[\hat{K}, .] + O(\Delta t^2) \tag{14}$$

Combining this with (12) to obtain the eigenvalues and eigenvectors, we can obtain the matrix inverse in $O(\log NM)$ time.

We now show another strategy to obtain the desired result. Using a density matrix formulation for a quantum state $\rho$, we obtain

$$e^{-i\hat{K}\Delta t}\rho e^{i\hat{K}\Delta t} = e^{\mathcal{L}_{\hat{K}}\Delta t}(\rho) \tag{15}$$

where $\mathcal{L}_K(\rho) = [K, \rho]$. We observe that

$$\begin{aligned} tr_P e^{-iS\Delta t}\rho \otimes \sigma e^{iS\Delta t} &= (cos^2\Delta t)\sigma + (sin^2\Delta t)\rho - isin\Delta t[\rho, \sigma] \\ &= \sigma - i\Delta t[\rho, \sigma] + O(\Delta t^2) \end{aligned} \tag{16}$$

where $tr_P$ is the partial trace over the first variable and $S$ is the swap operator. Since the swap operator is sparse, $e^{-iS\Delta t}$ can be calculated efficiently. Applying this equation to the previous one we obtain

$$\begin{aligned} e^{-i\mathcal{L}_{\hat{K}\Delta t}(\rho)} &\approx tr_1\{e^{-iS\Delta t}\hat{K} \otimes \rho e^{iS\Delta t}\} \\ &= \rho - i\Delta t[\hat{K}, \rho] + O(\Delta t^2) \end{aligned} \tag{17}$$

This shows that we can compute $e^{-i\hat{K}\Delta t}$ with error $O(\Delta t^2)$.

## Sources

1. “Quantum support vector machine for big data classification.”
   https://arxiv.org/pdf/1307.0471
2. “Understanding quantum support vector machines.”
   http://peterwittek.com/understanding-quantum-svms.html
3. “Quantum Recommendation Systems.”
   https://arxiv.org/abs/1603.08675
4. “Least squares support vector machine classifiers.”
   https://link.springer.com/article/10.1023/A:1018628609742
5. "Quantum algorithms for supervised and unsupervised machine learning"
   https://arxiv.org/pdf/1307.0411