Quantum Algorithms for Second Order Cone Programming

Iordanis Kerenidis [1,2]   Anupam Prakash [2]   Dániel Szilágyi [1]

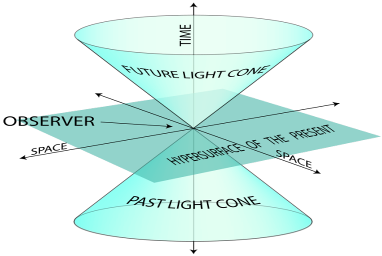
[1]CNRS, IRIF, Université Paris Diderot, Paris France

[2]QC Ware, Palo Alto, CA.

March 2, 2020

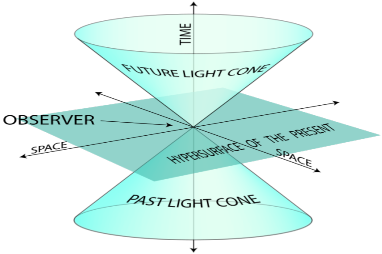
Second Order Cones

- Lorentz cone: The n-dimensional Lorentz cone, for $n \geq 1$ is defined as $\mathcal{L}^n := \{\vec{x} = (x_0; \vec{x}) \in \mathbb{R}^n \mid x_0 \geq \|\vec{x}\|\}$.



Second Order Cones

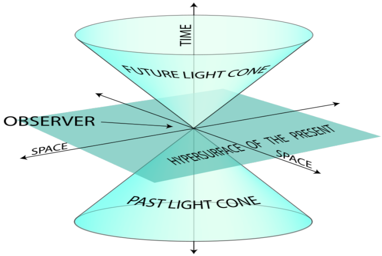
- Lorentz cone: The n-dimensional Lorentz cone, for $n \geq 1$ is defined as $\mathcal{L}^n := \{\vec{x} = (x_0; \vec{x}) \in \mathbb{R}^n \mid x_0 \geq \|\vec{x}\|\}$.



- $\mathcal{L}^3 = \{(x, y, z) \in \mathbb{R}^3 \mid z^2 \geq x^2 + y^2\}$.

Second Order Cones

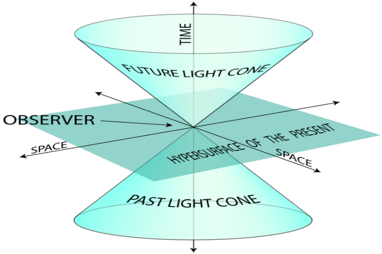
- Lorentz cone: The n-dimensional Lorentz cone, for $n \geq 1$ is defined as $\mathcal{L}^n := \{\vec{x} = (x_0; \vec{x}) \in \mathbb{R}^n \mid x_0 \geq \|\vec{x}\|\}$.



- $\mathcal{L}^3 = \{(x, y, z) \in \mathbb{R}^3 \mid z^2 \geq x^2 + y^2\}$.
- $\mathcal{L}^1 = \{x \in \mathbb{R} \mid x^2 \geq 0\}$.

Second Order Cones

- Lorentz cone: The n-dimensional Lorentz cone, for $n \geq 1$ is defined as $\mathcal{L}^n := \{\vec{x} = (x_0; \vec{x}) \in \mathbb{R}^n \mid x_0 \geq \|\vec{x}\|\}$.



- $\mathcal{L}^3 = \{(x, y, z) \in \mathbb{R}^3 \mid z^2 \geq x^2 + y^2\}$.
- $\mathcal{L}^1 = \{x \in \mathbb{R} \mid x^2 \geq 0\}$.
- Second order cone programs (SOCPs) have constraints of the form $\vec{x} \in \mathcal{L}^n$.

Second Order Cone Programs

- A SOCP (Second Order Cone Program) is an optimization problem of the following form,

$$
\begin{aligned}
\min_{\vec{x}_1,\ldots,\vec{x}_r} \quad & \vec{c}_1^T \vec{x}_1 + \cdots + \vec{c}_r^T \vec{x}_r \\
\text{s.t.} \quad & A^{(1)}\vec{x}_1 + \cdots + A^{(r)}\vec{x}_r = \vec{b} \\
& \vec{x}_i \in \mathcal{L}^{n_i}, \ \forall i \in [r].
\end{aligned} \tag{1}
$$

Second Order Cone Programs

- A SOCP (Second Order Cone Program) is an optimization problem of the following form,

$$\begin{aligned} \min_{\vec{x}_1,\ldots,\vec{x}_r} \quad & \vec{c}_1^T \vec{x}_1 + \cdots + \vec{c}_r^T \vec{x}_r \\ \text{s.t.} \quad & A^{(1)}\vec{x}_1 + \cdots + A^{(r)}\vec{x}_r = \vec{b} \\ & \vec{x}_i \in \mathcal{L}^{n_i}, \ \forall i \in [r]. \end{aligned} \tag{1}$$

- Constraint matrices $A^{(i)} \in \mathbb{R}^{m \times n_i}$ for $i \in [r]$ and $b \in \mathbb{R}^m$.

Second Order Cone Programs

- A SOCP (Second Order Cone Program) is an optimization problem of the following form,

$$\begin{aligned} \min_{\vec{x}_1,\ldots,\vec{x}_r} \quad & \vec{c}_1^T\vec{x}_1 + \cdots + \vec{c}_r^T\vec{x}_r \\ \text{s.t.} \quad & A^{(1)}\vec{x}_1 + \cdots + A^{(r)}\vec{x}_r = \vec{b} \\ & \vec{x}_i \in \mathcal{L}^{n_i},\ \forall i \in [r]. \end{aligned} \tag{1}$$

- Constraint matrices $A^{(i)} \in \mathbb{R}^{m \times n_i}$ for $i \in [r]$ and $b \in \mathbb{R}^m$.
- The number of Lorentz constraints r is the rank of the SOCP.

Second Order Cone Programs

- A SOCP (Second Order Cone Program) is an optimization problem of the following form,

$$\begin{aligned} \min_{\vec{x}_1,\dots,\vec{x}_r} \quad & \vec{c}_1^T \vec{x}_1 + \cdots + \vec{c}_r^T \vec{x}_r \\ \text{s.t.} \quad & A^{(1)}\vec{x}_1 + \cdots + A^{(r)}\vec{x}_r = \vec{b} \\ & \vec{x}_i \in \mathcal{L}^{n_i},\ \forall i \in [r]. \end{aligned} \tag{1}$$

- Constraint matrices $A^{(i)} \in \mathbb{R}^{m \times n_i}$ for $i \in [r]$ and $b \in \mathbb{R}^m$.
- The number of Lorentz constraints r is the rank of the SOCP.
- The sum of dimensions of the vectors, $n := \sum_i n_i$ is the dimension of the SOCP.

Second Order Cone Programs

- The SOCP can be written as an optimization problem over $\mathcal{L} = \prod_{i\in[r]} \mathcal{L}^{n_i}$ by concatenating vectors x_i, c_i and matrices A^i.

Second Order Cone Programs

- The SOCP can be written as an optimization problem over $\mathcal{L} = \prod_{i \in [r]} \mathcal{L}^{n_i}$ by concatenating vectors x_i, c_i and matrices A^i.
- Standard form of primal and dual SOCP,

$$\begin{aligned} &\min \vec{c}^T \vec{x} & &\max \vec{b}^T \vec{y} \\ &A\vec{x} = \vec{b} & &A^T \vec{y} + \vec{s} = \vec{c} \\ &\vec{x} \in \mathcal{L} & &\vec{s} \in \mathcal{L}, \vec{y} \in \mathbb{R}^m \end{aligned} \tag{2}$$

Second Order Cone Programs

- The SOCP can be written as an optimization problem over $\mathcal{L} = \prod_{i \in [r]} \mathcal{L}^{n_i}$ by concatenating vectors x_i, c_i and matrices A^i.
- Standard form of primal and dual SOCP,

$$\begin{aligned} &\min \vec{c}^T \vec{x} & \quad\quad & \max \vec{b}^T \vec{y} \\ &A\vec{x} = \vec{b} & & A^T \vec{y} + \vec{s} = \vec{c} \\ &\vec{x} \in \mathcal{L} & & \vec{s} \in \mathcal{L}, \vec{y} \in \mathbb{R}^m \end{aligned} \tag{2}$$

- SOCPs generalize Linear Programs (LPs) and Convex Quadratic Programs (QPs).

Second Order Cone Programs

- The SOCP can be written as an optimization problem over $\mathcal{L} = \prod_{i \in [r]} \mathcal{L}^{n_i}$ by concatenating vectors x_i, c_i and matrices A^i.
- Standard form of primal and dual SOCP,

$$
\begin{aligned}
&\min \vec{c}^T \vec{x} && \max \vec{b}^T \vec{y} \\
&A\vec{x} = \vec{b} && A^T \vec{y} + \vec{s} = \vec{c} \\
&\vec{x} \in \mathcal{L} && \vec{s} \in \mathcal{L}, \vec{y} \in \mathbb{R}^m
\end{aligned}
\tag{2}
$$

- SOCPs generalize Linear Programs (LPs) and Convex Quadratic Programs (QPs).
- The running time for classical SOCP algorithms is given in terms of n, r and the duality gap ϵ.

Reducing SVM to SOCP

- Support Vector Machines (SVM) are one of the most important classification algorithms in Machine Learning.

Reducing SVM to SOCP

- Support Vector Machines (SVM) are one of the most important classification algorithms in Machine Learning.
- Standard form of the SVM,

$$
\begin{aligned}
\min \|w\|^2 + C\,\|\xi\|_1 \\
y_i(w^T x_i + b) \geq 1 - \xi_i \\
\xi \geq 0
\end{aligned}
\tag{3}
$$

Reducing SVM to SOCP

- Support Vector Machines (SVM) are one of the most important classification algorithms in Machine Learning.
- Standard form of the SVM,

$$\begin{aligned} \min \|w\|^2 + C\|\xi\|_1 \\ y_i(w^T x_i + b) \geq 1 - \xi_i \\ \xi \geq 0 \end{aligned} \tag{3}$$

- The Lorentz constraint $\mathbf{t} := (t+1, t, w) \in \mathcal{L}^{n+2}$ is equivalent to $(2t+1) \geq \|w\|^2$, thus linearizing the quadratic constraint.

Reducing SVM to SOCP

- Support Vector Machines (SVM) are one of the most important classification algorithms in Machine Learning.
- Standard form of the SVM,

$$\begin{aligned} \min \|w\|^2 + C\|\xi\|_1 \\ y_i(w^T x_i + b) \geq 1 - \xi_i \\ \xi \geq 0 \end{aligned} \tag{3}$$

- The Lorentz constraint $\mathbf{t} := (t+1, t, w) \in \mathcal{L}^{n+2}$ is equivalent to $(2t+1) \geq \|w\|^2$, thus linearizing the quadratic constraint.
- The SVM reduces to an SOCP with variables $\mathbf{t} \in \mathcal{L}^{n+2}$ and $\xi_i \in \mathcal{L}^1$ with $r = n + m + 2$.

Main Results

- (Ben Tal-Nemirovski) There is a classical SOCP interior point method (IPM) based SOCP solver with running time $O(\sqrt{r}n^{\omega}\log(n/\epsilon))$.

Main Results

- (Ben Tal-Nemirovski) There is a classical SOCP interior point method (IPM) based SOCP solver with running time $O(\sqrt{r} n^{\omega} \log(n/\epsilon))$.

Theorem

There is a quantum IPM for SOCPs with running time $O(n^{1.5} \sqrt{r} \frac{\kappa}{\delta^2} \log(1/\epsilon))$ where δ bounds the distance of the intermediate solutions from the cone boundary, κ is the condition number of intermediate matrices and ϵ is the duality gap.

Main Results

- (Ben Tal-Nemirovski) There is a classical SOCP interior point method (IPM) based SOCP solver with running time $O(\sqrt{r} n^{\omega} \log(n/\epsilon))$.

Theorem

There is a quantum IPM for SOCPs with running time $O(n^{1.5}\sqrt{r}\frac{\kappa}{\delta^2}\log(1/\epsilon))$ where δ bounds the distance of the intermediate solutions from the cone boundary, κ is the condition number of intermediate matrices and ϵ is the duality gap.

- Experimental results on random SVM instances: The quantum algorithm scales as $O(n^k)$ where $k \in [2.56, 2.62]$ with high probability while an external SOCP solver scales as $O(n^{3.31})$.

Jordan Algebras: The Spin factor

- Formally real Jordan algebra satisfies the axioms: (i) $xy = yx$. (ii) $x^p x^q = x^{p+q}$. (ii) $\sum_i x_i^2 = 0 \Rightarrow x_i = 0$.

Jordan Algebras: The Spin factor

- Formally real Jordan algebra satisfies the axioms: (i) $xy = yx$. (ii) $x^p x^q = x^{p+q}$. (ii) $\sum_i x_i^2 = 0 \Rightarrow x_i = 0$.
- (Jordan, Von Neumann, Wigner 34): Classified finite dimensional formally real Jordan algebras into 5 families.

Jordan Algebras: The Spin factor

- Formally real Jordan algebra satisfies the axioms: (i) $xy = yx$. (ii) $x^p x^q = x^{p+q}$. (ii) $\sum_i x_i^2 = 0 \Rightarrow x_i = 0$.
- (Jordan, Von Neumann, Wigner 34): Classified finite dimensional formally real Jordan algebras into 5 families.
- Special Jordan algebra: Algebra of matrices with product defined as $x \circ y = (xy + yx)/2$.

Jordan Algebras: The Spin factor

- Formally real Jordan algebra satisfies the axioms: (i) $xy = yx$. (ii) $x^p x^q = x^{p+q}$. (ii) $\sum_i x_i^2 = 0 \Rightarrow x_i = 0$.
- (Jordan, Von Neumann, Wigner 34): Classified finite dimensional formally real Jordan algebras into 5 families.
- Special Jordan algebra: Algebra of matrices with product defined as $x \circ y = (xy + yx)/2$.
- The spin factor is a Jordan algebra on $\mathbb{R}^n$ with product defined as,

$$\vec{u} \circ \vec{v} = (u_0, \tilde{u}) \circ (v_0, \tilde{v}) := (\vec{u}^T \vec{v}, u_0 \tilde{v} + v_0 \tilde{u})$$

Jordan Algebras: The Spin factor

- Formally real Jordan algebra satisfies the axioms: (i) $xy = yx$. (ii) $x^p x^q = x^{p+q}$. (ii) $\sum_i x_i^2 = 0 \Rightarrow x_i = 0$.
- (Jordan, Von Neumann, Wigner 34): Classified finite dimensional formally real Jordan algebras into 5 families.
- Special Jordan algebra: Algebra of matrices with product defined as $x \circ y = (xy + yx)/2$.
- The spin factor is a Jordan algebra on $\mathbb{R}^n$ with product defined as,

$$\vec{u} \circ \vec{v} = (u_0, \tilde{u}) \circ (v_0, \tilde{v}) := (\vec{u}^T \vec{v}, u_0 \tilde{v} + v_0 \tilde{u})$$

- The identity element is $\vec{e} := (1; 0^n)$.

Jordan product, Arrow matrices

- The Jordan product is a linear operation, it has a matrix representation,

$$\vec{u} \circ \vec{v} = \begin{bmatrix} u_0 & \tilde{u}^T \\ \tilde{u} & u_0 I_{n-1} \end{bmatrix} \vec{v} := Arw(u)\vec{v}$$

Jordan product, Arrow matrices

- The Jordan product is a linear operation, it has a matrix representation,

$$\vec{u} \circ \vec{v} = \begin{bmatrix} u_0 & \tilde{u}^T \\ \tilde{u} & u_0 I_{n-1} \end{bmatrix} \vec{v} := Arw(u)\vec{v}$$

- The Jordan product and the Arrow matrices can be extended blockwise,

$$(\vec{u}_1; \ldots; \vec{u}_r) \circ (\vec{v}_1; \ldots; \vec{v}_r) := (\vec{u}_1 \circ \vec{v}_1; \ldots; \vec{u}_r \circ \vec{v}_r).$$

Jordan product, Arrow matrices

- The Jordan product is a linear operation, it has a matrix representation,

$$\vec{u} \circ \vec{v} = \begin{bmatrix} u_0 & \tilde{u}^T \\ \tilde{u} & u_0 I_{n-1} \end{bmatrix} \vec{v} := Arw(u)\vec{v}$$

- The Jordan product and the Arrow matrices can be extended blockwise,

$$(\vec{u}_1; \ldots; \vec{u}_r) \circ (\vec{v}_1; \ldots; \vec{v}_r) := (\vec{u}_1 \circ \vec{v}_1; \ldots; \vec{u}_r \circ \vec{v}_r).$$

- The well structured arrow matrices make the linear systems that arise in the IPM for SOCPs simpler than those for SDPs.

Interior Point Method for SOCP

- The central path for the SOCP is parametrized by $\nu > 0$ and is characterized by feasibility and complementary slackness conditions,

$$\begin{aligned} A\vec{x} &= \vec{b} \\ A^T\vec{y} + \vec{s} &= \vec{c} \\ \vec{x} \circ \vec{s} &= \nu\vec{e}, \end{aligned} \tag{4}$$

Interior Point Method for SOCP

- The central path for the SOCP is parametrized by $\nu > 0$ and is characterized by feasibility and complementary slackness conditions,

$$\begin{aligned} A\vec{x} &= \vec{b} \\ A^T\vec{y} + \vec{s} &= \vec{c} \\ \vec{x} \circ \vec{s} &= \nu\vec{e}, \end{aligned} \tag{4}$$

- The central path converges to the optimal solution as $\nu \rightarrow 0$.

Interior Point Method for SOCP

- The central path for the SOCP is parametrized by $\nu > 0$ and is characterized by feasibility and complementary slackness conditions,

$$\begin{aligned} A\vec{x} &= \vec{b} \\ A^T\vec{y} + \vec{s} &= \vec{c} \\ \vec{x} \circ \vec{s} &= \nu\vec{e}, \end{aligned} \tag{4}$$

- The central path converges to the optimal solution as $\nu \rightarrow 0$.
- A single iteration of the IPM finds $\Delta\vec{x}, \Delta\vec{y}, \Delta\vec{s}$ such that $\vec{x} + \Delta\vec{x}$, $\vec{y} + \Delta\vec{y}$ and $\vec{s} + \Delta\vec{s}$ are close to the central path for $\nu' = \sigma\nu$.

Interior Point Method for SOCP

- Linearizing the last equation and neglecting the term $\Delta\vec{x} \circ \Delta\vec{s}$ we get,

$$\vec{x} \circ \vec{s} + \vec{x} \circ \Delta\vec{s} + \vec{s} \circ \Delta\vec{x} = \sigma\nu\vec{e}.$$

Interior Point Method for SOCP

- Linearizing the last equation and neglecting the term $\Delta\vec{x} \circ \Delta\vec{s}$ we get,

$$\vec{x} \circ \vec{s} + \vec{x} \circ \Delta\vec{s} + \vec{s} \circ \Delta\vec{x} = \sigma\nu\vec{e}.$$

- We thus obtain the Newton linear system for SOCPs,

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ \mathrm{Arw}(\vec{s}) & 0 & \mathrm{Arw}(\vec{x}) \end{bmatrix} \begin{bmatrix} \Delta\vec{x} \\ \Delta\vec{y} \\ \Delta\vec{s} \end{bmatrix} = \begin{bmatrix} \vec{b} - A\vec{x} \\ \vec{c} - \vec{s} - A^T\vec{y} \\ \sigma\nu\vec{e} - \vec{x} \circ \vec{s} \end{bmatrix} \tag{5}$$

Interior Point Method for SOCP

- Linearizing the last equation and neglecting the term $\Delta\vec{x} \circ \Delta\vec{s}$ we get,

$$\vec{x} \circ \vec{s} + \vec{x} \circ \Delta\vec{s} + \vec{s} \circ \Delta\vec{x} = \sigma\nu\vec{e}.$$

- We thus obtain the Newton linear system for SOCPs,

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ \mathrm{Arw}(\vec{s}) & 0 & \mathrm{Arw}(\vec{x}) \end{bmatrix} \begin{bmatrix} \Delta\vec{x} \\ \Delta\vec{y} \\ \Delta\vec{s} \end{bmatrix} = \begin{bmatrix} \vec{b} - A\vec{x} \\ \vec{c} - \vec{s} - A^T\vec{y} \\ \sigma\nu\vec{e} - \vec{x} \circ \vec{s} \end{bmatrix} \qquad (5)$$

- Analysis shows that if (x, y, s) is in a neighborhood $\mathcal{N}$ of the central path at ν, then $\vec{x} + \Delta\vec{x}, \vec{y} + \Delta\vec{y}, \vec{s} + \Delta\vec{s}$ remains feasible and in $\mathcal{N}$ at ν'.

Quantum Interior Point Method

- Quantum IPM uses quantum linear system solver to solve the Newton linear system and tomography to reconstruct the solutions.

Quantum Interior Point Method

- Quantum IPM uses quantum linear system solver to solve the Newton linear system and tomography to reconstruct the solutions.
- (Quantum data structures) Efficient unitary block encodings $\begin{bmatrix} A/\mu & . \\ . & . \end{bmatrix}$ can be constructed/updated in linear time.

Quantum Interior Point Method

- Quantum IPM uses quantum linear system solver to solve the Newton linear system and tomography to reconstruct the solutions.
- (Quantum data structures) Efficient unitary block encodings $\begin{bmatrix} A/\mu & . \\ . & . \end{bmatrix}$ can be constructed/updated in linear time.
- (Chakraborty, Gilyén, Jeffery 18): Given block encodings for input matrix A, the quantum linear system can be solved in time $O(\sqrt{n}\kappa \log(1/\epsilon))$.

Quantum Interior Point Method

- Quantum IPM uses quantum linear system solver to solve the Newton linear system and tomography to reconstruct the solutions.
- (Quantum data structures) Efficient unitary block encodings $\begin{bmatrix} A/\mu & . \\ . & . \end{bmatrix}$ can be constructed/updated in linear time.
- (Chakraborty, Gilyén, Jeffery 18): Given block encodings for input matrix A, the quantum linear system can be solved in time $O(\sqrt{n}\kappa \log(1/\epsilon))$.
- (Kerenidis, P. 18): The output of quantum linear system $|x\rangle = |A^{-1}b\rangle$ can be reconstructed in time $O(n \log n/\epsilon^2)$ queries to obtain $\tilde{x}$ such that $\|\tilde{x} - x\|_2 \leq \epsilon \|x\|_2$.

Analysis of QIPM

- Jordan algebra provides a dictionary/framework to transfer concepts from the analysis for the SDP to the SOCP setting.

Analysis of QIPM

- Jordan algebra provides a dictionary/framework to transfer concepts from the analysis for the SDP to the SOCP setting.
- We can define a spectral decomposition for vectors,

$$\vec{x} = \frac{1}{2}\left(x_0 + \|\tilde{x}\|\right)\begin{bmatrix} 1 \\ \frac{\tilde{x}}{\|\tilde{x}\|} \end{bmatrix} + \frac{1}{2}\left(x_0 - \|\tilde{x}\|\right)\begin{bmatrix} 1 \\ -\frac{\tilde{x}}{\|\tilde{x}\|} \end{bmatrix} \tag{6}$$

Analysis of QIPM

- Jordan algebra provides a dictionary/framework to transfer concepts from the analysis for the SDP to the SOCP setting.
- We can define a spectral decomposition for vectors,

$$\vec{x} = \frac{1}{2}(x_0 + \|\tilde{x}\|)\begin{bmatrix} 1 \\ \frac{\tilde{x}}{\|\tilde{x}\|} \end{bmatrix} + \frac{1}{2}(x_0 - \|\tilde{x}\|)\begin{bmatrix} 1 \\ -\frac{\tilde{x}}{\|\tilde{x}\|} \end{bmatrix} \tag{6}$$

- We can thus define $\|x\|_2$, $\|x\|_F$ and prove familiar inequalities like $\|x \circ y\|_F \leq \|x\|_2 \|y\|_F$.

Analysis of QIPM

- Jordan algebra provides a dictionary/framework to transfer concepts from the analysis for the SDP to the SOCP setting.
- We can define a spectral decomposition for vectors,

$$\vec{x} = \frac{1}{2}\left(x_0 + \|\tilde{x}\|\right)\begin{bmatrix} 1 \\ \frac{\tilde{x}}{\|\tilde{x}\|} \end{bmatrix} + \frac{1}{2}\left(x_0 - \|\tilde{x}\|\right)\begin{bmatrix} 1 \\ -\frac{\tilde{x}}{\|\tilde{x}\|} \end{bmatrix} \tag{6}$$

- We can thus define $\|x\|_2$, $\|x\|_F$ and prove familiar inequalities like $\|x \circ y\|_F \leq \|x\|_2 \|y\|_F$.
- Matrix scaling $Y \rightarrow XYX$ has the Jordan algebra analog $2Arw^2(x) - Arw(x^2)$.

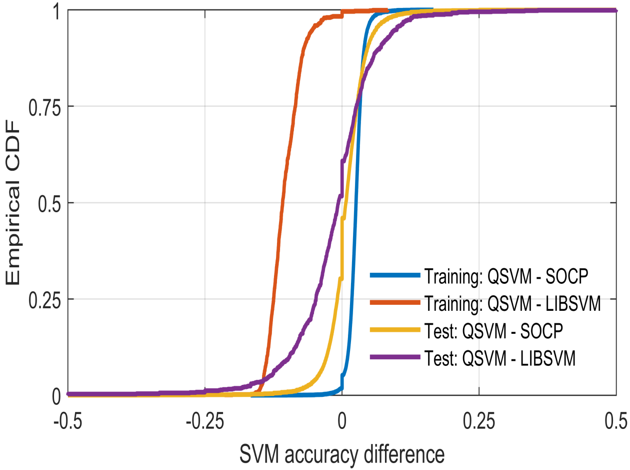
Experiments: Random SVM Instances

- Generate m points $\{\vec{x}_i \in \mathbb{R}^n \mid i \in [m]\}$ in the unit hypercube $[-1, 1]^n$.
- Generate a random unit vector $\vec{w} \in \mathbb{R}^n$ and assign labels to the points as $y^{(i)} = \text{sgn}(\vec{w}^T \vec{x}^{(i)})$.
- Corrupt a fixed proportion p of the labels, by flipping the sign of each $y^{(i)}$ with probability p.
- Shift the entire dataset by a vector $\vec{d} \sim \mathcal{N}(0, 2I)$, where $\mathcal{N}(\mu, \Sigma)$ denotes the multivariate normal distribution with mean μ and covariance Σ.
- Generate instances from $SVM(n, 2n, p)$ with n uniform in $[2, 2^9]$ and p uniform from $\{0, 0.1, \cdots, 0.9, 1\}$.

Experiments: Comparisons with classical algorithms

- We compare with classical algorithms on $SVM(n, 2n, p)$ instances for $\epsilon = 0.1$ where these algorithms achieve high accuracy.
- SOCP solver (ECOS) scales empirically as $O(n^{3.314})$, this is consistent with using a Strassen like algorithm with exponent 2.8.
- LIBSVM with linear kernel scales empirically as $O(n^{3.112})$, it is consistent with state-of-the-art alternate approaches to SVM.
- The running time $\frac{n^2 \kappa}{\delta^2}$ for the quantum IPM empirically scales as $O(n^{2.591})$ with a 95% confidence interval $[2.56, 2.62]$.

Experiments



-

- The classification accuracy for the quantum algorithm is similar to that of the classical algorithms.

Conclusions

- Experiments indicate that the quantum IPM achieves a polynomial speedup for solving SOCPs with low and medium precision.
- For random SVM instances, it achieves a polynomial speedup with no detriment to the quality of the trained classifier.
- Similar results for the constrained portfolio optimization problem.
- Conclusion: Quantum optimization methods can achieve polynomial speedups for longer term algorithms.
- Open question: Improvements to the quantum IPM using tomography with ℓ_∞ guarantees?

Portfolio Optimization

- Portfolio optimization is the theory of optimal investment of wealth in assets that differ in expected return and risk [Markovitz 1952].
- Let $R(t) \in \mathbb{R}^m$ be returns for m assets over time epochs $t \in [T]$. Then, expected reward and risk can be estimated as,

$$\mu = \frac{1}{T} \sum_{t \in [T]} R(t)$$

$$\Sigma = \frac{1}{T-1} \sum_{t \in [T]} (R(t) - \mu)(R(t) - \mu)^T$$

- A portfolio is specified by $x \in \mathbb{R}^m$ with x_j being the investment in asset j.
- The expected reward and risk for x are $\mu^T x$ and $x^T \Sigma x$ respectively.

Portfolio Optimization

- Unconstrained portfolio optimization: Find portfolio that minimizes risk for a given reward.
- Constrained portfolio optimization: There are positivity $x_j \geq 0$ and budget constraints $Cx \geq d$. Introducing slack variables $s = Cx - d$, $s \geq 0$.
- The Constrained Portfolio Optimization problem reduces to SOCP,

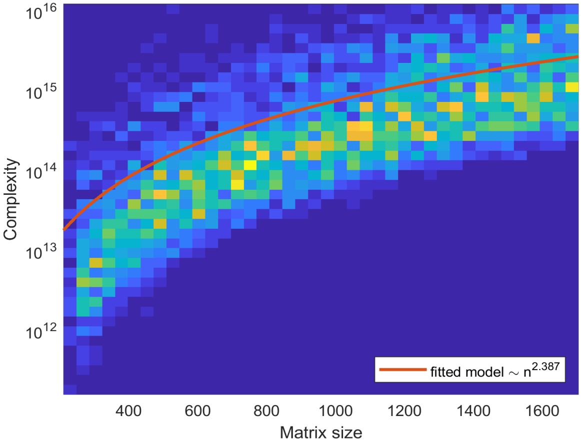
$$\begin{aligned} \min \quad & x^T \Sigma x \\ \text{s.t.} \quad & \mu^T x = R \\ & Ax = b \\ & x \geq 0. \end{aligned} \tag{7}$$

- (Lloyd-Rebentrost). The unconstrained problem is a least squares problem and has a closed form solution using a single linear system solver.

Experiments: Constrained Portfolio Optimization

- cvxPortfolio dataset: Stocks for the S&P-500 companies for each day over a period of 9 years (2007-2016).
- Subsample 100 companies and consider random interval of t days where t is uniform on [10,500]. Add positivity constraints on portfolio.
- Quantum algorithm can be simulated by adding Gaussian noise of magnitude δ, the duality gap $\epsilon = 0.1$ due to market stochasticity.

Experiments



- Observed complexity for $\epsilon = 0.1$ and power law fit for random portfolio optimization instances.