Report Week 3

Last week, I studied how Quantum Support Vector Machine is implemented through the paper Quantum Support Vector Machine. The paper showed that a quantum support vector machine can be implemented with $O(\log MN)$ run time in both training and classification stages. This report records what I did in week 3 and some of week 4.

Classical SVM

Given M data points $\{x^{(i)}, y^{(i)}\}$ where $x^{(i)} \in \mathbb{R}^N$ is the i^{th} input and $y^{(i)} \in \mathbb{R}$ is the i^{th} output. A classifier is constructed as follow:

$$\begin{gathered} w^T\phi(x^{(i)}) + b \geq 1 \quad \text{if } y^{(i)} = 1 \\ w^T\phi(x^{(i)}) + b \leq -1 \quad \text{if } y^{(i)} = -1 \quad \text{for i = 1,..,M} \\ \text{or, equivalently} \\ y^{(i)} * (w^T\phi(x^{(i)}) + b) \geq 1 \quad \text{for i = 1,..,M} \end{gathered} \tag{1}$$

$\phi(.)$ is a feature mapping that maps input space into a higher dimension. Add regularization term to make the algorithm less sensitive to outliers:

$$\begin{gathered} y^{(i)} * (w^T\phi(x^{(i)}) + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \quad \text{for i = 1,..,M} \end{gathered} \tag{2}$$

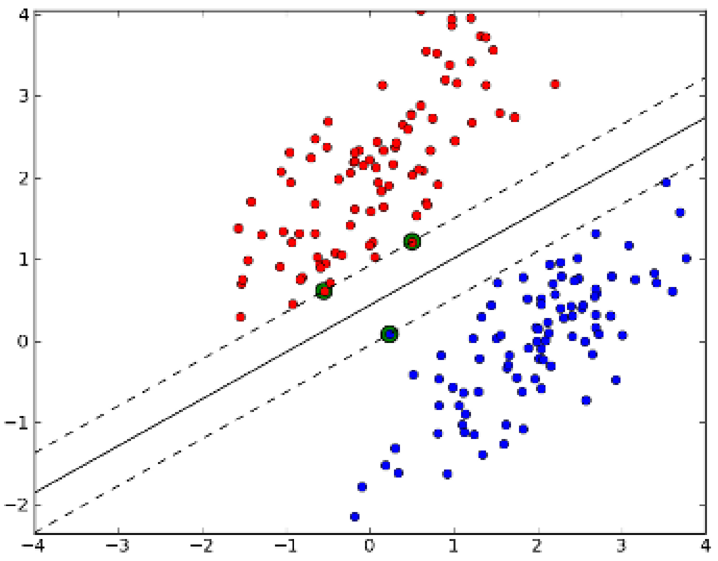

Figure 1: Support Vector Machine Hyperplane

SVM finds the hyperplane that maximizes the geometric margin of the dataset, where geometric margin of the dataset is defined as the minimum Euclidean distance between all data points to the hyperplane.

Optimization problem

$$\begin{aligned} \max_{\gamma,w,b} \quad & \gamma \\ \text{s.t.} \quad & y^{(i)}(w^T\phi(x^{(i)}) * b) \geq \gamma \quad \text{for i = 1, ..., M} \\ & \|w\| = 1 \end{aligned} \tag{3}$$

But $\|w\|$ is a non-convex function. So we transform the problem into a nicer one:

$$\begin{aligned} \max_{\gamma,w,b} \quad & \frac{\hat{\gamma}}{\|w\|} \\ \text{s.t.} \quad & y^{(i)}(w^T\phi(x^{(i)}) * b) \geq \hat{\gamma} \quad \text{for i = 1, ..., M} \end{aligned} \tag{4}$$

$\hat{\gamma}$ is the functional margin, defined as $\hat{\gamma}^{(i)} = y^{(i)}(w^T\phi(x^{(i)}) * b)$, and $\hat{\gamma} = \min_{i=1,...,M} \hat{\gamma}^{(i)}$. When $\|w\|$ equals 1, functional margin equals geometric margin. However, $\frac{\hat{\gamma}}{\|w\|}$ is a non-convex objective function. To get rid of this, we introduce a scaling constraint that the function margin of w, b with respect to the training set must be 1.

$$\hat{\gamma} = 1$$

So now, maximizing $\frac{\hat{\gamma}}{\|w\|} = \frac{1}{\|w\|}$ is the same as minimizing $\|w\|^2$

$$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y^{(i)}(w^T\phi(x^{(i)}) * b) \geq 1 \quad \text{for i = 1, ..., M} \end{aligned} \tag{5}$$

Optimization problem with regularization term:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T\phi(x^{(i)}) * b) \geq (1 - \xi_i) \quad \text{for i = 1, ..., M} \\ & \xi_i \geq 0 \quad \text{i = 1,...,m} \end{aligned} \tag{6}$$

Least Squares SVM optimization problem can be stated as:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}w^Tw + \frac{1}{2} * \zeta \sum_{i=1}^{m}{\xi_i}^2 \\ \text{s.t.} \quad & y^{(i)}(w^T\phi(x^{(i)}) + b) = (1 - \xi_i) \quad \text{for i = 1, ..., M} \end{aligned} \tag{7}$$

We solve for the optimization problem by Lagrange Duality. The Lagrangian for the optimization problem can be stated as:

$$\begin{aligned} \mathcal{L}_1(w, b, \xi_i, \alpha_i) = & \frac{1}{2}w^Tw + \frac{1}{2} * \zeta * \sum_{i=1}^{M}{\xi_i}^2 \\ & - \sum_{i=1}^{M}\alpha_i\{y^{(i)}(w^T\phi(x^{(i)}) + b) - 1 + \xi_i\} \end{aligned} \tag{8}$$

KKT conditions for optimality are:

$$\begin{gathered}
\frac{\partial}{\partial w_i}\mathcal{L}_1 = 0 \longrightarrow w = \sum_{i=1}^{M} \alpha_i \phi(x^{(i)}) \\
\frac{\partial}{\partial b}\mathcal{L}_1 = 0 \longrightarrow \sum_{i=1}^{M} \alpha_i = 0 \\
\frac{\partial}{\partial \xi_i}\mathcal{L}_1 = 0 \longrightarrow \alpha_i = \zeta * \xi_i \quad \text{i = 1,..., m} \\
\frac{\partial}{\partial \alpha_i}\mathcal{L}_1 = 0 \longrightarrow y^{(i)} = (w^T\phi(x^{(i)}) + b) + \xi_i
\end{gathered} \tag{9}$$

We obtain the following matrix:

$$\begin{bmatrix} \vec{1} & 0 & 0 & -Z^T \\ 0 & 0 & 0 & \vec{1} \\ 0 & 0 & \zeta * \vec{1} & -\vec{1} \\ Z & \vec{1} & \vec{1} & 0 \end{bmatrix} * \begin{bmatrix} w \\ b \\ \xi \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ y \end{bmatrix}$$

where:
Z $= (\phi(x^{(1)}), \phi(x^{(2)}), \dots, \phi(x^{(m)}))$
$\alpha = (\alpha_1, \dots, \alpha_m)$
$\xi = (\xi_1, \dots, \xi_m)$ and $\vec{1} = (1, \dots, 1)$
$y = (y_1, \dots, y_n)$

By rule of substitution, we get the following:

$$\begin{bmatrix} 0 & \vec{1} \\ \vec{1} & \Omega + \zeta^{-1} * I \end{bmatrix} * \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}$$

where: $\Omega_{ij} = \phi(x^{(i)})^T \phi(x^{(j)})$, and $\Omega \in \mathbb{R}^{MxM}$

The problem can now be solved by solving the above linear equation to find for b and α. Then we can use the fact that $w = \sum_{i=1}^{M} \alpha_i \phi(x^{(i)})$ to find for w. Once we find the optimal w and b, we have already solved for the optimization problem.

Computational Cost for Classical SVM

Solving for dual problem involves evaluateing $\frac{M*(M-1)}{2}$ dot products for the kernel matrix, and finding for optimal α takes $O(M^3)$ in the non-sparse case. As each dot product takes time $O(N)$ to evaluate, the classical support vector algorithm takes time $O(\log \frac{1}{\epsilon} M^2(N+M))$ with accuracy ϵ

Quantum SVM

Speed up of QSVM comes from two sources: (1) fast quantum evaluation of inner products (2) fast matrix inversion algorithm.

Fast quantum evaluation of inner product: Quantum algorithm can do dot product of two vectors in $O(\frac{1}{\epsilon}\log N)$ time. The idea is: for two vectors $\vec{x_i}$ and $\vec{x_j}$, then $\vec{x_i}^T\vec{x_j} = \frac{Z-|\vec{x_i}-\vec{x_j}|^2}{2}$

where $Z = |\vec{x_i}|^2 + |\vec{x_j}|^2$

Fast Matrix Inversion Algorithm

Now reconsider:

$$F * \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 & \vec{1} \\ \vec{1} & \vec{K} + \zeta^{-1} * I \end{bmatrix} * \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}$$

For simplicitity, let's just now assume that $\vec{K} = \vec{x_i}^T * \vec{x_j}$. Now our goal is to invert F. Consider, $\hat{F} = \frac{F}{tr(F)}$, then $\hat{F}$ can be written as:

$$\hat{F} = \frac{(J + K + \zeta^{-1} * I)}{tr(F)}$$

where:

$$J = \begin{bmatrix} 0 & \vec{1} \\ \vec{1} & 0 \end{bmatrix}$$

Our goal is to invert the matrix F and multiply F^{-1} with $(0, y)^T$ to find for b and α. In order to find for inverse of F, we need to compute $e^{-i\hat{F}\Delta t}$ so that if we have $|b\rangle$, some quantum state, we can apply Quantum Phase Estimation Algorithm on $e^{-i\hat{F}\Delta t} |b\rangle |0\rangle$ to obtain $\hat{F}^{-1} |b\rangle$. The general idea is as follow:

Suppose we want to invert a square Hermitian matrix A. For some quantum state, $|b\rangle = \sum_j \beta_j \psi_j$, where ψ_j is the eigenvector of A. Let $|0\rangle$ be another register and ρ is a momentum operator, then:

$$e^{-iA\otimes\rho t} |b\rangle |0\rangle \overset{\text{QPE}}{=} \sum_j \beta_j |\psi\rangle_j |\lambda_j t\rangle$$

We can get the phase operator $e^{i\Delta t \lambda_j{}^{-1}}$ and multiply it with the above result, then we get $\sum_j \beta_j e^{i\Delta t \lambda_j{}^{-1}} |\psi\rangle_j |\lambda_j t\rangle$. Apply the inverse Quantum Phase Estimation on this we get $\sum_j \beta_j e^{i\Delta t \lambda_j{}^{-1}} |\psi\rangle_j |0\rangle = e^{i\Delta t A^{-1}} |b\rangle |0\rangle$. For more explaination

Back to our problem, so now we need to compute $e^{-i\hat{F}\Delta t}$. By Lie Product, we have that:

$$e^{-i\hat{F}\Delta t} = e^{-iI\Delta t/tr(F)} * e^{-iJ\Delta t/tr(F)} * e^{-iK\Delta t/tr(F)}$$

"Computing $e^{-iI\Delta t}$ and $e^{-iJ\Delta t}$ is trival". The hard part is computing $e^{-iK\Delta t}$.

Starting from the initial state $\frac{1}{\sqrt{M}} \sum_{i=1}^{M} |i\rangle$, the training oracles are used to prepare the state $|\chi\rangle = \frac{1}{\sqrt{N_x}} \sum_{j=1}^{N} |\vec{x_i}| |i\rangle |\vec{x_i}\rangle$, with $N_x = \sum_{i=1}^{M} |\vec{x_i}|^2$ in $O(\log MN)$ run time. The Kernel matrix K is the density matrix reduced to a constant factor tr(K): $\hat{K} = \frac{K}{tr(K)} = tr_2\{|\chi\rangle\langle\chi|\} = \frac{1}{K} \sum_{i,j=1}^{M} \langle x_i | x_j \rangle |x_i||x_j| |i\rangle \langle j|$

We can now obtain $e^{-i\hat{K}\Delta t}$ by: (Density Matrix Exponentiation)

$$\begin{aligned} e^{-i\mathcal{L}_{\hat{K}}\Delta t} &\approx tr_1(e^{-iS\Delta t} \hat{K} \otimes \rho e^{iS\Delta t}) \\ &= \rho - i\Delta t[\hat{K}, \rho] + O(\Delta t^2) = e^{-i\hat{K}\Delta t} \rho e^{i\hat{K}\Delta t} \end{aligned} \tag{10}$$

where: $\mathcal{L}_{\hat{K}}(\rho) = [\hat{K}, \rho]$, super-operator; $\Delta t = \frac{t_o}{T}$ (t_o is the total evolution time determining the error of the phase estimation, and T is the number of time steps in the phase estimation; $S = \sum_{m,n=1}^{M} |m\rangle \langle n| \otimes |n\rangle \langle m|$ is the swap matrix of dimension M^2 x M^2.

Once F^{-1} is obtained, we can use it to find for b and α. The quantum state is first initialized into

$$|0, \vec{y}\rangle = \frac{1}{\sqrt{N_{0,y}}}(|0\rangle + \sum_{i=1}^{M} y_i |i\rangle)$$

where $N_{0,y} = 1 + \sum_i {y_i}^2$

Then, by performing the matrix inversion of F, the quantum state is transferred to:

$$|b, \vec{\alpha}\rangle = \frac{1}{\sqrt{N_{b,\alpha}}}(b\,|0\rangle + \sum_{i=1}^{M} \alpha_i |i\rangle), \text{ where } N_{b,\alpha} = b^2 + \sum_{i=1}^{M} {\alpha_i}^2$$

Computational Cost of QSVM

- Preparation of the Kernel matrix: $O(\log MN)$
- Computing $e^{-iK\Delta t}$: $O(\frac{{t_o}^2}{\epsilon} \log MN)$ (because $\epsilon = O(T\Delta t = O({t_o}^2/T)$ so the run time is $T = O(\frac{{t_o}^2}{\epsilon}))$
- Inverting $\hat{F}$ costs: $O(\log N)$
- Total cost: $O(\log MN)$

References

[1] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. *Quantum Support Vector Machine*

[2] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. *Quantum Principal Component Analysis*

[3] Video on Solving for Linear Equations. Quantum Algorithm for Solving Linear Equation

[4] Seth Lloyd, Massoud Mohseni, and Patrick Rebentrost. Quantum Algorithms for Supervised and Unsupervised Machine Learning

[5] Convex Optimization Notes. Notes 1, Notes 2, More on KKT conditions.