GPU Accelerated Support Vector Machines via Quadratic Programming

Armaan Kohli

May 14, 2020

The Cooper Union
ECE453

Objective

Solve classification problems quickly

- Support Vector Machines
- ADMM Algorithm
- CPU and GPU Implementations
- Results

Paper from Oxford Control Group (part of osqp)

GPU Acceleration of ADMM for Large-Scale Quadratic Programming [2]

Michel Schubiger, Goran Banjac, and John Lygeros - 2019

Support Vector Machine

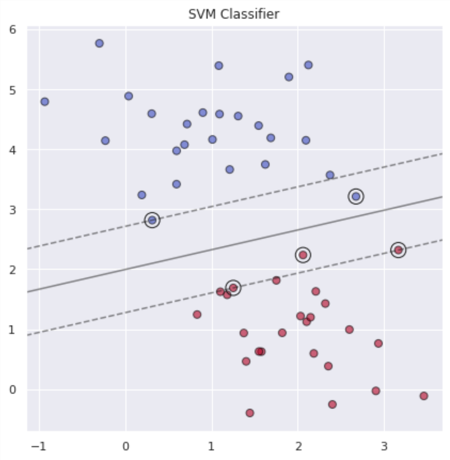


Figure 1: Cartoon of an SVM classification problem

Support Vector Machine

We can write the SVM problem as one that relies on convex optimization, specifically *quadratic programming*

$$\{x : f(x) = x^T\beta + 1 = 0\} \tag{1}$$

Support Vector Machine

We can write the SVM problem as one that relies on convex optimization, specifically *quadratic programming* [1]

$$L_P = \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i\left(x_i^T\beta + 1\right) - (1-\xi_i)\right] - \sum_{i=1}^{N}\mu_i\xi_i \quad (2)$$

$$L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j \quad (3)$$

$$\alpha_i[y_i(x_i^T\beta + 1) - (1-\xi_i)] = 0 \quad (4)$$

$$mu_i\xi_i = 0 \quad (5)$$

$$y_i(x_i^T\beta + 1) - (1-\xi_i) \geq 0 \quad (6)$$

This allows us to solve the SVM problem with an algorithm called ADMM

ADMM

Alternating Direction Method of Multipliers

- A technique to minimize convex functions
- Intuition: Try to minimize a convex function by alternatively minimize Lagrangian in different directions

ADMM

Algorithm 1: ADMM algorithm as presented in [3]

given: x^0, z^0, y^0 *and parameters* $\rho > 0,\ \sigma > 0,\ \alpha \in [0, 2]$

while *not terminated* **do**

$$(\tilde{x}^{k+1}, v^{k+1}) \leftarrow \begin{bmatrix} P + \sigma I & A^T \\ A & -\rho^{-1} I \end{bmatrix} \begin{bmatrix} \tilde{x}^{k+1} \\ v^{k+1} \end{bmatrix} = \begin{bmatrix} \sigma x^k - q \\ z^k - \rho^{-1} y^k \end{bmatrix}$$

$$\tilde{z}^{k+1} \leftarrow z^k + \rho^{-1}(v^{k+1} - y^k)$$

$$x^{k+1} \leftarrow \alpha \tilde{x}^{k+1} + (1 - \alpha) x^k$$

$$z^{k+1} \leftarrow \prod \left(\alpha \tilde{z}^{k+1} + (1 - \alpha) z^k + \rho^{-1} y^k\right)$$

$$y^{k+1} \leftarrow y^k + \rho(\alpha \tilde{z}^{k+1} + (1 - \alpha) z^k - z^{k+1}$$

end

ADMM

Issue: solving this linear system is challenging

$$\begin{bmatrix} P + \sigma I & A^T \\ A & -\rho^{-1} I \end{bmatrix} \begin{bmatrix} \tilde{x}^{k+1} \\ v^{k+1} \end{bmatrix} = \begin{bmatrix} \sigma x^k - q \\ z^k - \rho^{-1} y^k \end{bmatrix} \tag{7}$$

ADMM

Issue: solving this linear system is challenging

$$\begin{bmatrix} P + \sigma I & A^T \\ A & -\rho^{-1} I \end{bmatrix} \begin{bmatrix} \tilde{x}^{k+1} \\ v^{k+1} \end{bmatrix} = \begin{bmatrix} \sigma x^k - q \\ z^k - \rho^{-1} y^k \end{bmatrix} \tag{8}$$

- Use LDL Factorization
- Use Preconditioned Conjugate Gradient (PCG)

PCG

Algorithm 2: PCG algorithm as presented in [2]

initialise: $r^0 = Kx^0 - b,\ y^0 = M^{-1}r^0,\ p^0 = -y^0,\ k = 0$

while $||r^k|| > \epsilon||b||$ **do**

$$\alpha^k \leftarrow -\frac{\left(r^k\right)^T y^k}{\left(p^k\right)^T K p^k}$$

$$x^{k+1} \leftarrow x^k + \alpha^k p^k$$

$$r^{k+1} \leftarrow r^k + \alpha^k K p^k$$

$$y^{k+1} \leftarrow M^{-1} r^{k+1}$$

$$\beta^{k+1} \leftarrow -\frac{\left(r^{k+1}\right)^T y^{k+1}}{\left(r^k\right)^T y^k}$$

$$p^{k+1} \leftarrow -y^{k+1} + \beta^{k+1} p^k$$

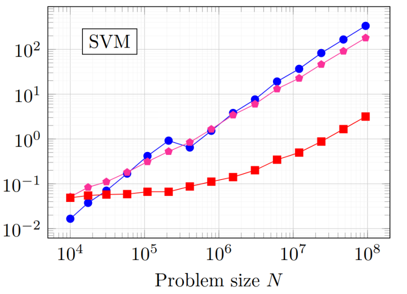
$$k \leftarrow k + 1$$

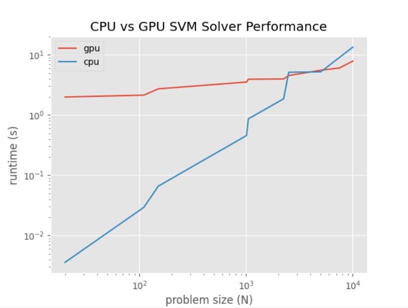
end

GPU Optimizations

- matrix representation - CSR
- `cuBLAS`
- `cuSPARSE`

Results


SVM
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{4}
10^{5}
10^{6}
10^{7}
10^{8}
Problem size N


CPU vs GPU SVM Solver Performance
gpu
cpu
runtime (s)
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{2}
10^{3}
10^{4}
problem size (N)

References

T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

M. Schubiger, G. Banjac, and J. Lygeros.
GPU acceleration of ADMM for large-scale quadratic programming.
arXiv:1912.04263, 2019.

B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd.
OSQP: An operator splitting solver for quadratic programs.
Mathematical Programming Computation, 2020.