Parallel Support Vector Machine

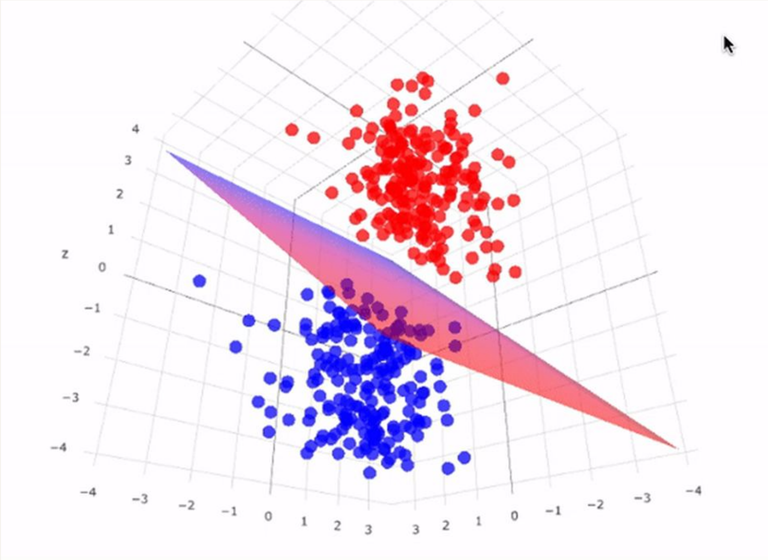
Wesley Osogo, Tianfan Xu, Connor Buchheit, Peter Chen

Problem – SVM

- **SVM: Finding the optimal hyperplane that separates a set of points belonging to two classes.**

- **Training an SVM model is computationally expensive, and scales poorly with dataset size, so training is *slow* on large datasets.**

- **Traditionally, coding SVM relies on Quadratic Programming (QP) for optimization, which is the main bottleneck. We aim to use existing techniques to reframe this problem as a parallel one, thus allowing for usage of OpenMP and MPI so that the model can accommodate larger datasets.**



Data Generation

- Data Fetching: dataset from the UCI repository, which includes features stored in X and targets stored in y.

- Standardization: The features X are standardized using StandardScaler from sklearn.preprocessing. This normalization ensures that each feature contributes equally to the analysis by giving them mean of zero and a standard deviation of one.

- Transformation of Targets: The target variable y is transformed based on a midpoint value calculated as the average of the maximum and minimum values of y. Each value in y is mapped to -1 if it is below this midpoint, otherwise to 1.

Math Model

Solve Quadratic Programming(QP)

$$\min \quad \mathcal{P}(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1} \xi_i$$

$$s.t. \quad 1 - y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \leq \xi_i, \quad \xi_i > 0,$$

$$\min \quad \mathcal{D}(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\mathbf{1}$$

$$s.t. \quad \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbf{C}, \ \mathbf{y}^T\boldsymbol{\alpha} = 0,$$

Interior Point Methods(IPM)

IPM: Solve QP with IPM, with computation bottleneck on matrix inverse in SVM

Parallel Incomplete Cholesky Factorization(ICF)

ICF: approximate a positive definite matrix (kernel matrix) with a lower triangular matrix
Parallel Factorization: Each machine performs ICF on its subset of the data independently

Parallel Interior Point Methods(IPM)

PIPM: Sherman-Morrison-Woodbury formula

$$\begin{aligned}
\Sigma^{-1}z &= (D+Q)^{-1}z \approx (D+HH^T)^{-1}z \\
&= D^{-1}z - D^{-1}H(I+H^TD^{-1}H)^{-1}H^TD^{-1}z \\
&= D^{-1}z - D^{-1}H(GG^T)^{-1}H^TD^{-1}z.
\end{aligned}$$

Parallel Support Vector Machine Implementation

Stages: Kernel Computation, Parallel ICF, Parallel IPM, Prediction

Scientific Library: Eigen(with row based memory layout)

Kernel Compute:

A single process read input data and computes kernel matrix locally through OpenMP

Parallel ICF:

Distribute different rows of computed kernel to distributed machines. Use OpenMp to parallelize local updates and MPI to decide pivot value/index and stopping condition

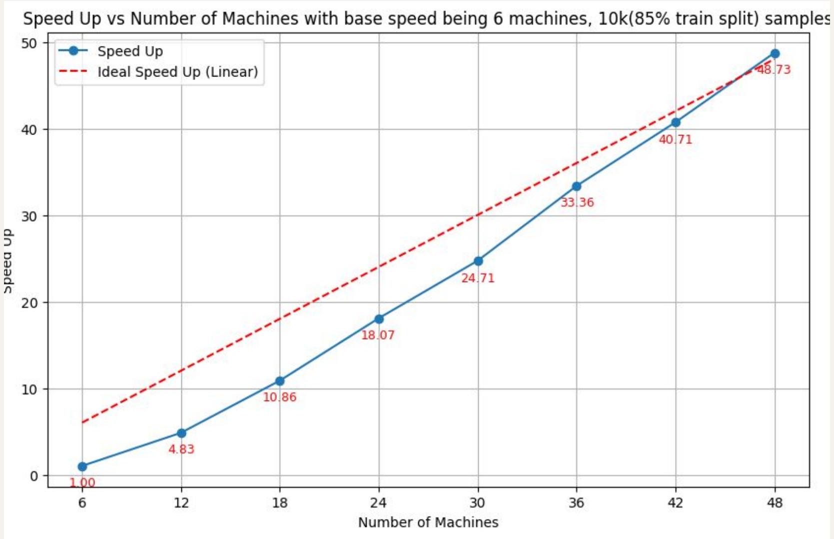
Parallel IPM:

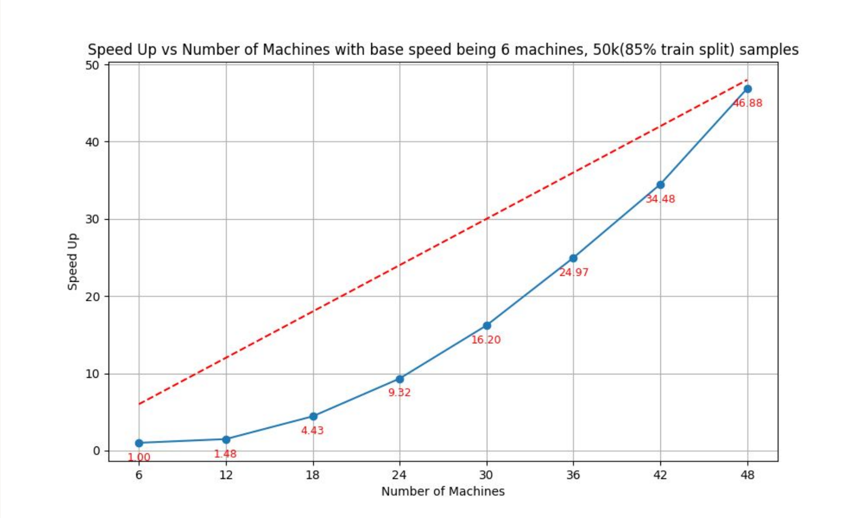
Use SMW formula to compute necessary parts locally and MPI to communicate, alleviating the heaviest computational task of solving inverse system

Prediction:

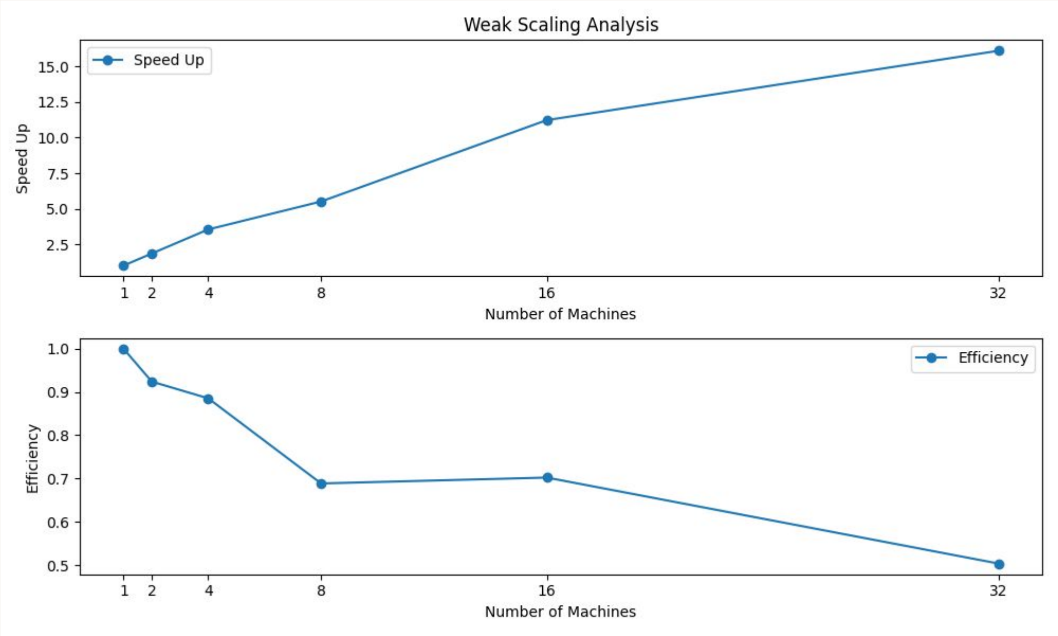
Each machine computes with the local weights and gathers result through MPI

Speed up (10k and 50k)

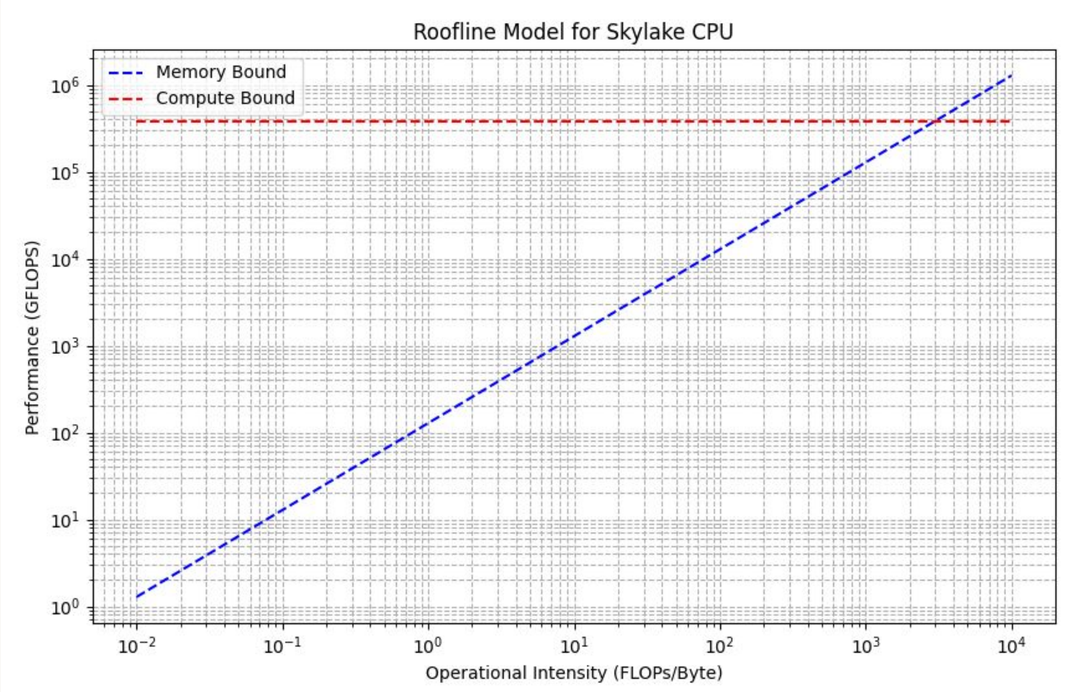




Performance – Scaling Analysis



Performance – Roofline Model



Compute Bound

8 cores
* 3 GHz
* 16 Flops/Cycle/Core

Memory Bandwidth

50 Gb/s

Next Steps

1) CUDA — Our code relies heavily on matrix multiplication. Given more time and access to GPUs, applying what we learned about CUDA and running our code on a GPU would result in even greater speedups than observed.
2) Performance — We saw a test accuracy of ~80% for most applications. Although we are very pleased with this performance as an initial result, experimenting with different SVM kernels could lead to even better performance.
3) Extreme scalability — Taking measures such as dimensionality reduction for our features to reduce problem complexity would likely yield even better results than we achieved. Given more time, we would have experimented with such techniques to make a much more fine-tuned model.