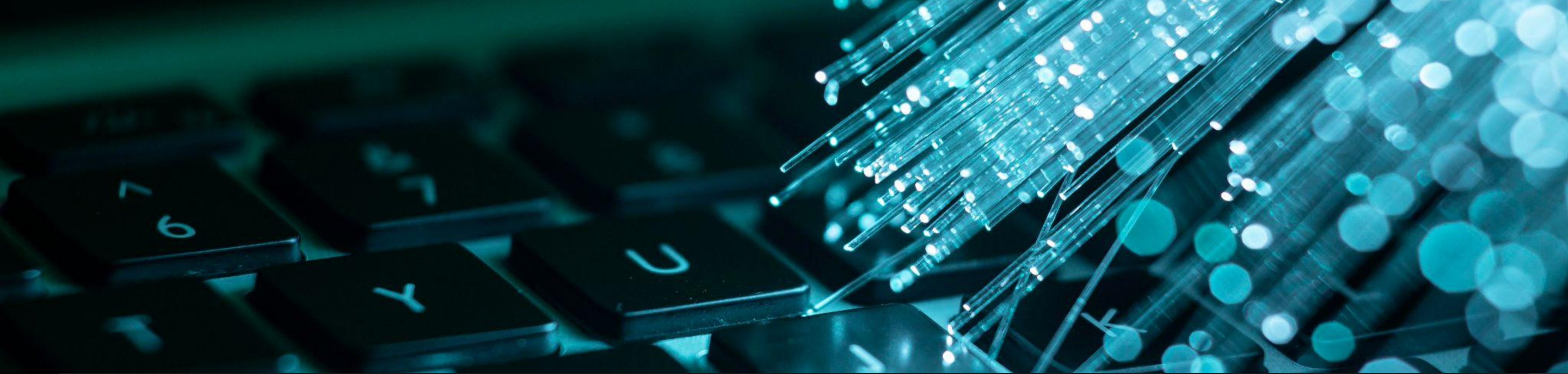


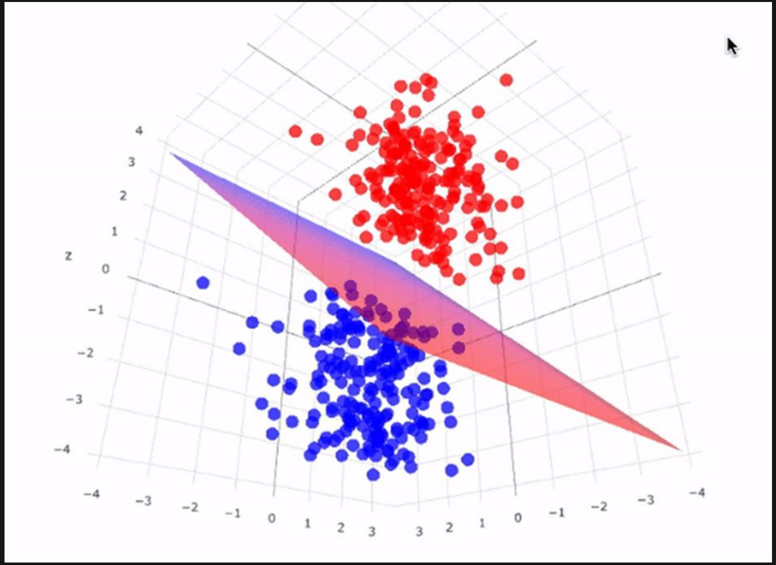
Parallelizing Support Vector Machines (SVM)

Project Proposal

Connor Buchheit, Peter Chen, Wesley Osogo, Tianfan Xu

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.**



Source: https://medium.com/low-code-for-advanced-data-science/support-vector-machines-svm-an-intuitive-explanation-b084d6238106

Math Model

Solve Quadratic Programming(QP)

$$
\begin{aligned}
\min \quad & \mathcal{P}(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|_2^2 + C \sum_{i=1} \xi_i \\
s.t. \quad & 1 - y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \le \xi_i, \quad \xi_i > 0, \\
\min \quad & \mathcal{D}(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha} - \boldsymbol{\alpha}^T \mathbf{1} \\
s.t. \quad & \mathbf{0} \le \boldsymbol{\alpha} \le \mathbf{C}, \ \mathbf{y}^T \boldsymbol{\alpha} = 0,
\end{aligned}
$$

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

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

WHY PARALLELIZE SVM

Handling Large Datasets

SVMs have a scalability problem in terms of memory use and computational time .

Requires **solving a complex quadratic programming optimization problem** during training process.

Non-linear kernel SVMs become expensive for large datasets. Most implementations **store the n*n matrix of distances** between training points.

Average case to train SVM is **usually between O(n^2) and O(n^3) in time and memory** for most implementations.

Improving Performance

Parallelizing SVM training enables the **quick training of multiple versions of SVMs** which is beneficial for hyperparameter tuning in **Grid Search.**

The **hyperparameters** that can be tuned include:

- C: The **regularization parameter**
- Kernel: appropriate **kernel function**
- Gamma: **Kernel coefficient** that controls shape of decision boundary

Our Plan

Work Package

1. Implement baseline SVM algorithm on our data set with non-linear kernel function in C++
2. Implement OpenMP to parallelize computations, calculate speedup with local parallelization
3. Implement parallel SVM (PSVM) with distributed machines using MPI

Memory System

- The central idea behind PSVM is to solve high requirement of memory needed for the computations by introducing a distributive memory model, so our ultimate goal is to achieve a **distributed memory system**.
- However, we would also like to test the speedup and space complexity with a shared memory system using OpenMP

Resources

- Chang, E. Y., Zhu, K., Wang, H., Bai, H., Li, J., Qiu, Z., & Cui, H. PSVM: Parallelizing Support Vector Machines on Distributed Computers Google Research. Beijing, China.
- Bottou, L., & Lin, C. J. (2007). Support vector machine solvers.
- https://www.kdnuggets.com/hyperparameter-tuning-gridsearchcv-and-randomizedsearchcv-explained

Thanks!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**