Fuel Cells Bulletin
ISSN: 1464-2859

A Hybrid Parallel Techniques for Large-Scale Text Classification with Symmetric ADMM and 1-Factorization

Vijayakumar H. Bhajantri[1*], Shashikumar G. Totad[1, 2], Geeta R. Bharamagoudar[3]

[1]School of Computer Science and Engineering KLE Technological University, Hubballi, India.
[2]Department of Computer Science and Engineering, TKIET, Warananagar, India.
[3]Department of Computer Science and Engineering, KLE Institute of Technology, Hubballi, India.
*Corresponding Email: **Vijayakumar H. Bhajantri**

Abstract

Distributed computing has become essential for handling large-scale datasets in machine learning. This study focuses on the implementation of two prominent multiclass Support Vector Machine (SVM) algorithms—Crammer and Singer (CS) and Weston and Watkins (WW)—in a distributed computing environment using symmetric Alternating Direction Method of Multipliers (ADMM). Designed for multiclass classification, these algorithms extend traditional binary SVMs by optimizing a single objective function that captures all class relationships simultaneously. These algorithms are adapted to leverage the computational power of multi-node clusters, ensuring scalability and efficiency. Symmetric ADMM is employed to decompose the optimization problem across multiple nodes, enabling parallel processing and efficient convergence for large datasets. The implementation distributes the optimization problem across a multi-node cluster, enabling parallel computation for handling large-scale dataset LSHTC. Each node processes a subset of the data, and symmetric ADMM ensures coordination and convergence by exchanging updates between nodes in a balanced manner. The optimization leverages polynomial kernels to handle non-linear separability effectively. The distributed framework reduces computational overhead and memory constraints compared to traditional single-node methods, achieving efficient scaling with the number of nodes. Results from the implementation highlight improved training times and robust classification accuracy, demonstrating the effectiveness of symmetric ADMM in balancing workload and accelerating convergence. This implementation serves as a scalable solution for high-dimensional datasets in resource-intensive environments.

Keywords: Machine Learning, Support Vector Machine, Alternate Direction Method of Multipliers, Symmetric, Crammer and Singer, Weston and Watkins, Multiclass Classification, Large Scale Dataset, LSHTC, and Distributed Computing.

1. Introduction

The explosive growth of data in modern applications has driven the need for scalable and efficient machine learning algorithms. Among these, multiclass Support Vector Machines (SVMs) have proven to be particularly effective for classification tasks, owing to their strong theoretical foundations and capability to handle high-dimensional data. The Crammer-Singer (Crammer et al., 2001) [27] and Weston-Watkins (Weston, J., & Watkins, C, 1999) [7] multiclass SVM algorithms are two widely used approaches for addressing multiclass problems. Both methods aim to directly optimize classification boundaries for multiple classes simultaneously, rather than decomposing the problem into multiple binary classification tasks. However, their application to large-scale datasets remains a significant challenge due to computational complexity [1] and memory constraints.

Multiclass classification involves assigning a single label from a predefined set of categories to each input. Traditional approaches often decompose multiclass problems into binary classification tasks [7] [27], such as one-vs-rest or one-vs-one strategies. While effective for smaller datasets, these methods introduce redundancies and inefficiencies when scaling to larger datasets. The Crammer-Singer algorithm, introduced in 2001, formulates the multiclass SVM problem as a single optimization task, ensuring global margins between classes. Similarly, the Weston-Watkins algorithm, proposed in 1999, offers a margin-based approach that optimizes decision boundaries across all classes in a unified framework. Both algorithms provide elegant solutions for multiclass problems but face significant computational challenges when applied to datasets with millions of samples or high-dimensional features.

Support Vector Machines (SVMs) Hsu, C. W., Chang, C. et al., 2013[16] have been widely adopted for binary classification problems due to their robustness and effectiveness in handling high-dimensional data. Extending SVMs to multiclass classification, however, is non-trivial. Two prominent approaches for multiclass SVMs are:

1. One-vs-Rest (OvR): Decomposes the problem into multiple binary classification tasks [5], each separating one class from the rest.
2. All-in-One (or All-vs-All): Optimizes a single objective function to classify all classes simultaneously.

The Weston-Watkins and Crammer-Singer formulation falls under the All-in-One approach, where a unified objective function ensures simultaneous optimization of all classes. This model avoids the inconsistencies often seen in OvR (Wang, Y., & Zhou, X. 2015) [5], making it suitable for complex, large-scale datasets.

Distributed computing has emerged as a transformative solution to the scalability issues associated with large-scale datasets. By leveraging parallelism across multiple computational nodes [19], distributed frameworks such as Apache Spark, TensorFlow, and PyTorch enable efficient execution of machine learning algorithms. According to Geeta R. B and Totad S. G [4] for SVMs, distributed computing allows the partitioning of data and computational tasks, reducing memory bottlenecks and speeding up the optimization process. These frameworks also provide tools for handling data that exceed the capacity of a single machine, facilitating scalability and fault tolerance.

Despite the advantages of distributed computing, implementing multiclass SVMs in distributed environments introduces additional complexities. Communication overhead, synchronization, and fault tolerance are key challenges that must be addressed to achieve efficient distributed execution. Optimization techniques [18] also need to be adapted for distributed systems to ensure convergence and performance [24]. The Alternating Direction Method of Multipliers (ADMM) [1] has gained prominence as an effective optimization method for distributed machine learning. ADMM decomposes global optimization problems into smaller sub-problems that can be solved independently on distributed nodes. This iterative approach allows for parallel computation while maintaining coordination across nodes to ensure global convergence.

The Symmetric ADMM approach, Yang, L et al., 2019 [29] showed a variant of the traditional ADMM, is particularly suited for distributed implementations of multiclass SVMs. By enforcing symmetry in the updates of dual variables across computational nodes, Symmetric ADMM ensures balanced convergence and minimizes bottlenecks caused by heterogeneous workloads. This property is especially advantageous for training large-scale multiclass SVMs, where data and computational resources are distributed across multiple nodes. Integrating Symmetric ADMM with the Crammer-Singer and Weston-Watkins algorithms provides a robust framework for addressing the challenges of multiclass classification in large-scale settings.

Applications of multiclass SVMs in large-scale settings span diverse fields. For instance, in image classification, datasets such as LSHTC, ImageNet and CIFAR-10 consist of millions of labeled samples distributed across hundreds or thousands of categories. In NLP, tasks like text categorization and sentiment analysis involve processing large corpora of textual data with diverse labels [2]. Similarly, in bioinformatics, genomic data often include thousands of features and labels, requiring efficient classification algorithms. These applications highlight the need for scalable solutions that can handle the dual challenges of large data volumes and high feature dimensionality.

Despite their potential, the distributed implementation of multiclass SVMs using Symmetric ADMM presents several practical challenges. Communication overhead [2] [24] between nodes must be minimized to ensure that the computational gains from parallelism are not negated by synchronization delays. Additionally, distributed systems must be resilient to faults, ensuring that node failures do not compromise the overall optimization process. Effective hyper-parameter tuning is also critical, as it directly impacts the convergence speed and classification performance of the algorithms [26]. Addressing these challenges is essential for realizing the full potential of distributed multiclass SVMs in real-world scenarios.

Existing literature provides a strong foundation for this study. The Crammer-Singer and Weston-Watkins algorithms [7] [27] have been extensively analyzed for their effectiveness in multiclass classification. Weston and Watkins (1999) demonstrated the advantages of their unified margin-based approach, while Crammer and Singer (2001) introduced a globally consistent optimization framework for multiclass problems. Boyd et al. (2011) [1] formalized the ADMM optimization technique, showcasing its applicability to distributed systems. Subsequent research has explored the use of distributed ADMM for SVM training, highlighting its scalability and efficiency. However, the specific integration of Symmetric ADMM with the Crammer-Singer and Weston-Watkins algorithms in distributed settings remains an area with significant research potential.

This study aims to address this gap by developing a distributed implementation of the Crammer-Singer and Weston-Watkins multiclass SVM algorithms using the Symmetric ADMM approach. The primary objectives of the research are as follows:

1. To design scalable implementations of the Crammer-Singer and Weston-Watkins multiclass SVM algorithms using distributed computing frameworks.
2. To adapt the Symmetric ADMM optimization technique to the requirements of multiclass SVMs, ensuring balanced convergence and reduced communication overhead.

3. To evaluate the performance of the proposed approach on large-scale datasets from diverse application domains, such as LSHTC.
4. To identify and address practical challenges in implementing the proposed framework, such as computational complexity, scalability, fault tolerance, resource allocation, and hyper-parameter tuning.

By achieving these objectives, this research aims to contribute to the growing body of knowledge on scalable machine learning algorithms for large-scale datasets. The findings of this study are expected to enhance the applicability of multiclass SVMs across various domains, providing insights into the integration of distributed computing and advanced optimization techniques.

The remaining sections of this paper is organized as follows: The section 2 covers the significance of parallelization of Support Vector Machine, distributed Support Vector Machine and standard ADMM. Section 3 covers the details of the proposed work Symmetric ADMM and distributed SVM and its mathematical model. Section 4 is the experimental evaluation that includes result discussion and Section 5 is the conclusion of the proposed work.

2. Literature Review

The scalability and efficiency of machine learning algorithms have become paramount with the increasing size and complexity of datasets in various domains as we discussed in our paper [27]. Multiclass Support Vector Machines (SVMs) play a significant role in classification tasks [12] [15], particularly due to their robust mathematical foundations and capacity to handle high-dimensional data. Among the multiclass SVM formulations, the Crammer-Singer and Weston-Watkins algorithms have been widely studied for their ability to address multiclass problems directly [7] [27] without relying on problem decomposition strategies such as one-vs-one or one-vs-rest. This section provides a comprehensive review of literature focusing on distributed computing frameworks, the Crammer-Singer and Weston-Watkins multiclass SVM algorithms, and the application of the Alternating Direction Method of Multipliers (ADMM), particularly its symmetric variant, to large-scale machine learning problems. The Smart Behavioral Driven Power Stasher is an advanced data storage and retrieval framework that dynamically adapts to user behavior and optimizes energy efficiency for seamless data management [6]. The cloud service provider helps in modeling management and allocation of various resources of cloud based on the customers' demand [14].

2.1 Multiclass SVMs: All-In-One Approach

Crammer and Singer (2001) introduced a multiclass SVM formulation that optimizes a single global objective to ensure margin separation among all classes [5]. This approach eliminates the redundancy and inefficiencies associated with binary decomposition methods. Similarly, Weston and Watkins (1999) [7] proposed a margin-based optimization technique that aligns well with the needs of multiclass classification. Both algorithms are recognized for their theoretical strengths but face computational challenges when applied to large datasets with numerous classes and features. Studies have emphasized that while these methods offer a unified framework [5] for multiclass classification, their implementation on large-scale datasets requires significant computational resources, particularly during training.

Given a dataset $\{(x_i, y_i)\}_{i=1,}^{n}$ $where\ x_i \in R^d is\ the\ feature\ vector, y_i \in \{1,2,\ldots,K\}$ is the class label, and K is the number of classes, the objective function of Weston-Watkins multiclass SVM formulation is:

$$\min_{w} \frac{1}{2} \|W\|_F^2 + C \sum_{i=1}^{n} \sum_{k \neq y_i} \max(0, 1 - w_{y_i}^T x_i + w_k^T x_i) \ldots\ldots\ldots (1)$$

Here: $W \in R^{Kxd}$: $A\ matrix\ where\ each\ row\ w_k is\ the\ weight\ vector\ for\ class\ k. W_F^2\ is\ the$ Frobenius norm of the regularization. $C > 0$ is the regularization parameter. The loss term sums the hinge losses for all incorrect classes $k \neq y_i$.
The reformulated constraints introducing slack variables $\xi_{i,k}$for all $i\ and\ k \neq y_i$

$$\min_{w,\xi} \frac{1}{2} \|w\|_F^2 + C \sum_{i=1}^{n} \sum_{k \neq y_i} \xi_{i,k} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

subject to:

$$w_{y_i}^T x_i - w_k^T x_i \geq 1 - \xi_{i,k},\ \xi_{i,k} \geq 0, \forall_i\ \forall k \neq y_i,$$

The objective function of Crammer-Singer multiclass SVM formulation is:

$$\min_{w} \frac{1}{2} \sum_{k=1}^{K} \|W_k\|^2 + C \sum_{i=1}^{n} \max_{j \neq y_i} (1 + W_j^T x_i - W_{y_i}^T x_i) \ldots\ldots\ldots (3)$$

With reformulated constraints

$$\min_{w} \frac{1}{2} \|W\|_F^2 + C \sum_{i=1}^{n} \xi_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

subject to

$$w_{y_i}^T x_i - w_k^T x_i \geq 1 - \xi_i, \forall k \neq y_i, \xi_i \geq 0, i = 1, \ldots, n$$

Here: $W \in R^{Kxd}$: *A matrix where each row* w_k *is the weight vector for class* k, ξ_i *is* the slack variables for misclassification. $C > 0$ is the regularization parameter. W_F^2 *is the* Frobenius norm of the regularization.

2.2 Challenges in Large-Scale Multiclass SVMs

The application of multiclass SVMs to large-scale datasets has highlighted several computational bottlenecks. First, the memory requirements for storing kernel matrices and intermediate variables grow quadratically with the number of data points, making the algorithms unsuitable for large datasets without modification (Cai, J et al.,2012) [13]. Second, the optimization process for multiclass SVMs involves solving complex quadratic programming problems [11], which become increasingly expensive as the size of the dataset increases. Finally, multiclass SVMs require careful tuning of hyper-parameters [26] such as regularization parameters and kernel functions, further complicating their application to large-scale problems.

To address these challenges, researchers have explored various approaches, including parallel computing and data partitioning. For instance, Chang and Lin (2011) [10] introduced LIBSVM, an efficient library for SVM training that incorporates problem decomposition techniques to handle larger datasets [3]. While effective for binary classification tasks, these techniques often fall short in multiclass settings, where the optimization problem's complexity increases exponentially with the number of classes.

2.3 Distributed Computing Frameworks

Distributed computing frameworks have emerged as a transformative solution to the scalability issues faced by traditional machine learning algorithms. By partitioning data and computational tasks across multiple nodes [23], distributed systems enable parallel execution of machine learning algorithms, reducing computation time and overcoming memory constraints. Frameworks such as Apache Spark [3], Hadoop, and TensorFlow have been widely adopted for their ability to handle massive datasets and provide fault-tolerant mechanisms.

Research has demonstrated the potential of distributed computing for SVM training. Zaharia et al. (2012) [8 - 9] introduced Resilient Distributed Datasets (RDDs), a fault-tolerant abstraction for in-memory cluster computing, which has been leveraged in distributed SVM implementations. Distributed computing has also facilitated the processing of streaming data and real-time classification tasks, further extending the applicability of SVMs to dynamic environments. However, the integration of distributed frameworks with multiclass SVMs remains challenging, primarily due to the complexity of global optimization tasks and the need for synchronization across nodes.

2.4 Symmetric ADMM for Multiclass SVMs

Symmetric ADMM, a variant of traditional ADMM, introduces symmetry in the update rules for dual variables across distributed nodes. This symmetry ensures balanced convergence and reduces bottlenecks caused by heterogeneous workloads, making it particularly suited for multiclass SVM training [29] [26]. By enforcing uniform updates across nodes, Symmetric ADMM minimizes communication overhead and improves the scalability of distributed implementations. Recent studies have highlighted the potential of Symmetric ADMM for large-scale machine learning tasks [17]. For example, experiments on distributed implementations of logistic regression and binary SVMs have demonstrated significant improvements in training efficiency and model performance. However, the application of Symmetric ADMM to multiclass SVMs, particularly the Crammer-Singer and Weston-Watkins algorithms, remains underexplored. Addressing this gap requires a detailed investigation of how Symmetric ADMM can be tailored to the unique requirements of multiclass classification.

ADMM decomposes the optimization problem into sub-problems that can be solved iteratively. The symmetric ADMM formulation introduces auxiliary variables Z and Lagrange multipliers λ to handle constraints more effectively.

Introduce auxiliary variable $Z = W$ and rewrite the problem (2) and (4)

$$\min_{w,z} \frac{1}{2} \|W\|_F^2 + C \sum_{i=1}^{n} \xi_i$$

Subject to

$$w_{y_i}^T x_i - z_k^T x_i \geq 1 - \xi_i, \forall k \neq y_i, \xi_i \geq 0, i = 1, \ldots, n, W = Z$$

The augmented lagrangian is:

$$\zeta(W, Z, \lambda) = \frac{1}{2} \|W\|_F^2 + C \sum_{i=1}^{n} \xi_i + \frac{\rho}{2} \left\| W - Z + {}^{\lambda}/_{\rho} \right\|_F^2$$

Where $\rho > 0$ *is the penalty parameter*

2.5 Applications of Distributed Multiclass SVMs

The integration of distributed computing with multiclass SVMs has far-reaching implications for various domains. In image classification, large-scale datasets such as LSHTC [28], ImageNet and CIFAR-10 require efficient algorithms capable of processing millions of labeled samples across multiple categories. Similarly, NLP applications such as text classification and sentiment analysis involve processing massive corpora with diverse labels. In bioinformatics, genomic data analysis often involves datasets with thousands of features and categories, necessitating scalable machine learning solutions [25]. Studies have shown that distributed multiclass SVMs can achieve competitive performance in these domains, provided that the underlying optimization framework is robust and efficient. For instance, distributed implementations of the Crammer-Singer and Weston-Watkins algorithms have been evaluated on synthetic and real-world datasets, demonstrating their ability to handle large-scale classification tasks [20-21] [28]. However, further research is needed to optimize these implementations for practical applications, particularly in scenarios involving imbalanced datasets and high feature dimensionality.

3. Details of Proposed Work

The key components of the proposed work are: multiclass formulation, symmetric ADMM framework and distributed setup.

1. Multiclass SVM Formulations:
   - Crammer-Singer: Aims to ensure that the score for the correct class is consistently higher than the scores for all incorrect classes by a margin of 1. It uses a unified objective function for all classes, avoiding the redundancies of One-vs-Rest approaches [5].
   - Weston-Watkins: Penalizes all margin violations equally across incorrect classes, focusing on smoother decision boundaries by optimizing a unified objective.
2. Symmetric ADMM Framework:
   - Decomposes the global optimization problem into smaller, local problems for each computing node.
   - Synchronizes computations through global weight updates and dual variable updates to achieve consensus across all nodes.

   Considering Local Node Problem: Each node n solves the following optimization problem for its local dataset D_n

$$\min_{W_n} \frac{1}{2}\|W_n\|^2 + C \sum_{i \in D_n} \max_{j \neq y_i} \left(1 + W_j^T x_i - W_{y_i}^T x_i\right) + \frac{\rho}{2}\left\|W_n - W_{global} + Z_n\right\|^2$$

   Where W_n is the local weight matrix and $Z_n$is the dual variable.

   Considering Global Problem: The global weight matrix W_{global} is updated as the average of all local weights and dual variables:

$$W_{global} = \frac{1}{N} \sum_{n=1}^{N} (W_n + Z_n)$$

   The dual variable update: The dual variables Z_n are updated to enforce consensus between the local and global weights:

$$Z_n = Z_n + W_n - W_{global}$$

   Convergence Analysis: The convergence of the Symmetric ADMM is governed by the following residuals:

   i. Primal Residual: Measures the difference between local and global weights:

$$r_{primal} = \sum_{n=1}^{N} \left\|W_n - W_{global}\right\|$$

   ii. Dual Residual: Measures the change in global weights between iterations:

$$r_{dual} = \rho \sum_{n=1}^{N} \left\|W_{global}^{t+1} - W_{global}^{t}\right\|$$

   The algorithm stops when both residuals fall below predefined thresholds:

$$r_{primal} < \epsilon_{primal,} \quad r_{dual} < \epsilon_{dual}$$

3. Distributed Setup
   - Dataset is partitioned across multiple nodes (e.g., MPI-2 for 2 nodes, MPI-4 for 4 nodes).
   - Each node performs local computations independently, reducing computational overhead on a single machine.

To efficiently train the All-in-One multiclass SVM classifier for the LSHTC dataset [28], the cluster architecture shown in Figure 3 is designed to balance computational workload and minimize communication overhead. Below is a detailed explanation of the cluster setup and its integration with the model:
Each node acts as an independent compute unit responsible for handling a portion of the dataset and the computational workload. The logical architecture organizes the workflow into data distribution, parallel computation, and result aggregation stages:

Data Distribution:
The dataset is partitioned into subsets (one for each node) using MPI's MPI_Scatter function. Each subset contains a proportional number of examples and features to ensure load balancing.

Local Computation of Each Node:
Each node uses multi-threading (OpenMP) [31] to parallelize tasks such as: training binary classifiers for 1-factorization, computing local gradient updates [17] and threads share memory to avoid duplication of data and optimize performance. Task Decomposition: Binary classifiers for a subset of the K_r (complete graph with r nodes) factorization are assigned to each node. This reduces inter-node dependency, as most computations are local.

Global Communication and Synchronization:
Global Aggregation: Nodes communicate updates (e.g., dual variables or classifier weights) using MPI's collective communication functions like MPI_Allreduce and aggregation ensures consistency across nodes. Synchronization: After every ADMM iteration, nodes synchronize to share updated global parameters.

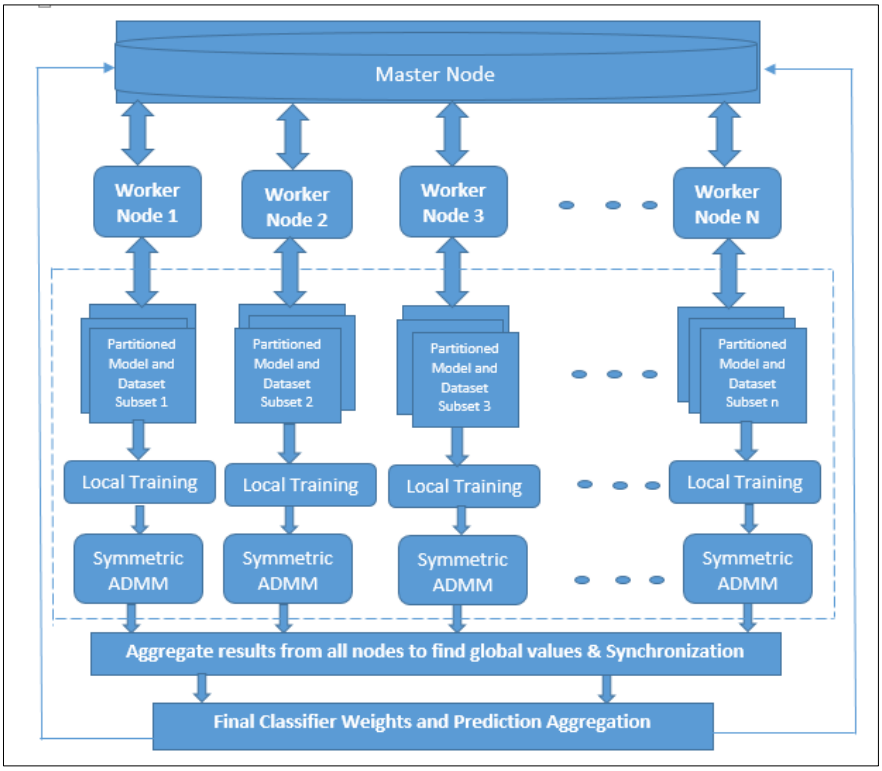


Figure 1. Architecture of the model

Workflow in the Cluster:

1. Initialization:
   - Each node loads its assigned subset of the dataset into memory.
   - MPI initializes communication channels among the nodes.
2. Training Phase:
   - Nodes independently compute local updates for their assigned binary classifiers using OpenMP.
   - Intermediate results are aggregated across nodes periodically to update global variables.
3. Prediction Phase:
   - Each node contributes predictions from its trained binary classifiers.
   - Results are aggregated across nodes using MPI to generate final predictions.

4. Evaluation:
   - The test dataset is replicated across nodes, allowing independent evaluation.
   - Accuracy and performance metrics are computed globally.

3.1 Symmetric ADMM

The symmetric variant ensures equal treatment of all nodes in distributed environments, reducing communication overhead and improving convergence.

Algorithm 1: Symmetric ADMM

1. Initialization
   $$Initialize \leftarrow W^{(0)}, Z^{(0)}, \lambda^{(0)} \; and \; \rho > 0$$
2. Update W: Solve the regularized least squares sub-problem:
   $$update\ W \leftarrow W^{(t+1)} = arg \min_{w} \frac{1}{2} \|W\|_F^2 + \frac{\rho}{2} \left\|W - Z^{(t)} + \lambda^{(t)}/\rho\right\|_F^2$$
3. Update Z: Solve the constrained optimization sub-problem for Z:
   $$update\ Z \leftarrow Z^{(t+1)} = arg \min_{z} \frac{\rho}{2} \left\|W^{(t+1)} - Z + \lambda^{(t)}/\rho\right\|_F^2$$
   subject to
   $$w_{y_i}^T x_i - z_k^T x_i \geq 1 - \xi_i, \forall k \neq y_i, \xi_i \geq 0$$
4. Dual variable update: Update Lagrange multipliers:
   $$\lambda^{(t+1)} = \lambda^{(t)} + \rho(W^{(t+1)} - Z^{(t+1)})$$
5. Convergence Check: Check convergence criteria for primal and dual residuals:
   $$\left\|W^{(t+1)} - Z^{(t+1)}\right\|_{F \leq \epsilon}, \left\|Z^{(t+1)} - Z^{(t)}\right\|_{F \leq \epsilon}$$

The Algorithm 1 steps are illustrated to understand local and global variables updates and finally setting and comparing the convergence.

Distributed Algorithm

1-Factorization

Using 1-factorization in the All-in-One multiclass SVM classifier provides a structured way to decompose and parallelize the training process [22], ensuring scalability and efficiency. This approach is particularly valuable for large datasets, where computational resources and time constraints are critical considerations.

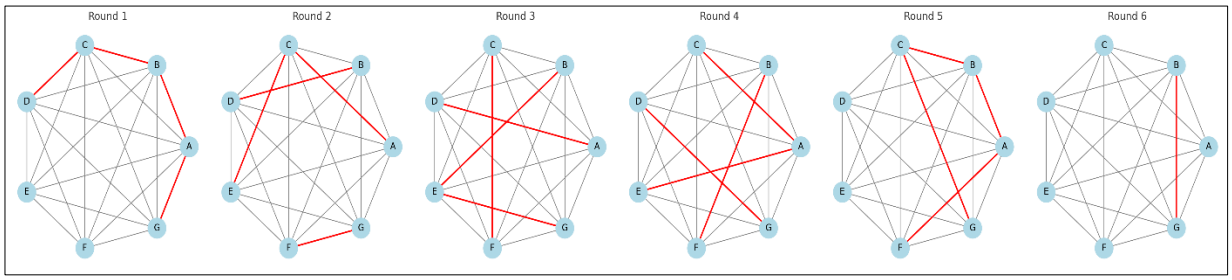


Figure 2. 1-Factorization

To better understand 1-factorization through a cricket match analogy referring the Figure 2, let's consider a cricket tournament with $r = 7\ teams: \{A, B, C, D, E, F, G\}$ each team needs to play against every other team exactly once, forming a round-robin tournament.

In graph terms, this is represented by a complete graph K_7, where:

- Nodes r represent the teams $\{A, B, C, D, E, F, G\}$ and edges (Red) represent the matches between two teams. Since there are 7 teams, in each round, one team does not play ("bye").

Since r is odd, every round consist of $\frac{r-1}{2}$ matches, leaving one team on bye. There are r rounds to ensure every team gets a chance to play against all other teams.

Example: Round 1: $A\ vs\ B, C\ vs\ D, E\ vs\ G\ (Team\ F\ is\ on\ bye)$ ... ... so on Round 7.

How it works in Multiclass SVM.

1. Pairwise Decomposition:
   - The All-in-One multiclass SVM involves training binary classifiers for all $\binom{r}{2}$ pairs of r classes.
   - Using 1-factorization, these binary classifiers are grouped into $r-1$ disjoint sets, where no two classifiers in the same set share a class.
2. Parallelization:
   - Each "matching" (set of binary classifiers) can be trained independently, as the classifiers in the matching do not interfere with one another.
   - This reduces the computational complexity by enabling parallel computation.
3. Efficient Predictions:
   - During prediction, scores from all binary classifiers are aggregated to determine the most likely class, similar to how all edges in the graph contribute to the complete classification process.

Symmetric ADMM with 1-Factorization

Reformulated problem: Introducing auxiliary variable Z and dual variable λ and we rewrite the problem as:

$$\min_{W,Z} \frac{1}{2} \|w\|_F^2 + C \sum_{i=1}^{n} \sum_{k \neq y_i} \xi_{i,k}$$

subject to:

Pairwise constraints for each matching F_j and $W = Z$. The augmented Lagrangian is:

$$\zeta(W, Z, \xi, \lambda) = \frac{1}{2} \|W\|_F^2 + C \sum_{i=1}^{n} \sum_{k \neq y_i} \xi_{i,k} + \frac{\rho}{2} \left\| W - Z + {}^{\lambda}/_{\rho} \right\|_F^2$$

Algorithm 2: Symmetric ADMM with 1-Factorization

1. Initialization:

$$Initialize \leftarrow W^{(0)}, Z^{(0)}, \lambda^{(0)}, \xi^{(0)}, and\ \rho > 0.$$
$$Decompose\ G\ into\ K-1\ matchings\ F_1, \ldots\ldots F_{K-1}$$

2. Iterative Updates:

   Update W: Solve the regularized least-squares problem

$$update\ W \leftarrow W^{(t+1)} = arg \min_{w} \frac{1}{2} \|W\|_F^2 + \frac{\rho}{2} \left\| W - Z^{(t)} + \lambda^{(t)}/\rho \right\|_F^2$$

   Update Z: Solve the constrained optimization for each matching F_j,

$$update\ Z \leftarrow Z^{(t+1)} = arg \min_{Z} \frac{\rho}{2} \left\| W^{(t+1)} - Z + \lambda^{(t)}/\rho \right\|_F^2$$

   subject to

$$w_{y_i}^T x_i - w_k^T x_i \geq 1 - \xi_{i,k},\ \xi_{i,k} \geq 0\ \forall (y_i, k) \in F_j$$

   Update ξ:

$$\xi_{i,k}^{(t+1)} \leftarrow \max(0, 1 - w_{y_i}^T x_i + w_k^T x_i)$$

   Dual update: Update the dual variables

$$\lambda^{(t+1)} \leftarrow \lambda^{(t)} + \rho(W^{(t+1)} - Z^{(t+1)})$$

3. Convergence Check: Monitor convergence for the primal and dual residuals:

$$\left\| W^{(t+1)} - Z^{(t+1)} \right\|_{F \leq \in,} \left\| \rho(Z^{(t+1)} - Z^{(t)}) \right\|_{F \leq \in}$$

The algorithm 2 is the key idea to integrate symmetric ADMM with 1-factorization to enhance the performance of the algorithm, here slack variables and other values are updated.

Algorithm 3: Crammer-Singer Multiclass SVM with 1-Factorization and Symmetric ADMM in a distributed environment.

1. Initialization
   - Input dataset $\{(x_i, y_i)\}_{i=1,}^{n}\ K\ Classes$
   - Initialize $W^{(0)}, Z^{(0)}, \xi^{(0)}, \lambda^{(0)}, and\ \rho > 0$
   - Decompose the complete graph $G \leftarrow (V, E)$

     $where\ V\ represents\ classes, into\ K-1\ disjoint\ matchings\ \{F_{1,} F_{2,} \ldots . F_{K-1}\}$

2. Iterative Updates: For $t = 0,1,2 \ldots . until\ convergence$
   - Step 1: W-update: Solve the regularized least-squares problem:
   $$W^{(t+1)} \leftarrow arg\ \min_{W} \frac{1}{2}\ \|W\|_F^2 + \frac{\rho}{2}\ \left\|W - Z^{(t)} + \lambda^{(t)}/\rho\right\|_F^2$$
   This is standard closed-form update:
   $$W^{(t+1)} \leftarrow \frac{1}{1+\rho}\ (Z^{(t)} - \lambda^{(t)}/\rho\ )$$
   - Step 2: Z-update: For each matching $F_j,\ Solve,$ the constrained optimization problem:
   $$Z^{(t+1)} \leftarrow arg\ \min_{Z} \frac{\rho}{2}\ \left\|W^{(t+1)} - Z + \lambda^{(t)}/\rho\right\|_F^2$$
   Subject to
   $$W_{y_i}^T x_i - Z_k^T x_i\ \geq 1 - \xi_i,\ \xi_i \geq 0 \quad \forall (i,k) \in F_j$$
   This step is distributed, with each worker solving its assigned matching
   - Step 3:Dual Variable Update: Update the dual variables:
   $$\lambda^{(t+1)} \leftarrow \lambda^{(t)} + \rho(W^{(t+1)} - Z^{(t+1)})$$
   - Convergence Check: Check for primal and dual residual:
   $$\left\|W^{(t+1)} - Z^{(t+1)}\right\|_{F \leq \in,}\ \left\|\rho(Z^{(t+1)} - Z^{(t)})\right\|_{F \leq \in}$$
3. Output: Optimized weight matrix W for classification.

The algorithm 4 demonstrated how the optimal weight matrix are determined through 1-factorization and global updated. The steps are iteratively executed so that convergence is achieved for the Crammer-Singer algorithm.

Algorithm 4: Weston-Watkins Multiclass SVM with 1-Factorization and Symmetric ADMM in a distributed environment.

1. Initialization
   - Input dataset $\{(x_i, y_i)\}_{i=1,}^{n}\ K\ Classes$
   - Initialize $W^{(0)}, Z^{(0)}, \xi^{(0)}, \lambda^{(0)}, and\ \rho > 0$
   - Decompose the complete graph
   - $G \leftarrow (V, E)\ into\ K - disjoint\ matchings\ \{F_{1,}F_{2,} \ldots . F_{K-1}\}$
2. Iterative Updates: For $t = 0,1,2 \ldots . until\ convergence$
   - Step 1: W-update: Solve the regularized least-squares problem:
   $$W^{(t+1)} \leftarrow arg\ \min_{W} \frac{1}{2}\ \|W\|_F^2 + \frac{\rho}{2}\ \left\|W - Z^{(t)} + \lambda^{(t)}/\rho\right\|_F^2$$
   This update is identical to the Crammer-Singer case:
   $$W^{(t+1)} \leftarrow \frac{1}{1+\rho}\ (Z^{(t)} - \lambda^{(t)}/\rho\ )$$
   - Step 2: Z-update: For each matching $F_j, Solve$:
   $$Z^{(t+1)} \leftarrow arg\ \min_{Z} \frac{\rho}{2}\ \left\|W^{(t+1)} - Z + \lambda^{(t)}/\rho\right\|_F^2$$
   Subject to
   $$w_{y_i}^T x_i - w_k^T x_i\ \geq 1 - \xi_{i,k},\ \xi_{i,k} \geq 0\ \forall (y_i, k) \in F_j$$
   - Step 3: Slack Variables $(\xi) update$: Update the slack variables for hinge loss:
   $$\xi_{i,k}^{(t+1)} \leftarrow \max(0, 1 - w_{y_i}^T x_i + w_k^T x_i)$$
   - Step 4:Dual Variable Update: Update the dual variables:
   $$\lambda^{(t+1)} \leftarrow \lambda^{(t)} + \rho(W^{(t+1)} - Z^{(t+1)})$$
   - Convergence Check: Check for primal and dual residual convergence:
   $$\left\|W^{(t+1)} - Z^{(t+1)}\right\|_{F \leq \in,}\ \left\|\rho(Z^{(t+1)} - Z^{(t)})\right\|_{F \leq \in}$$
3. Output: Optimized weight matrix W for classification.

The algorithm 5 demonstrated how the optimal weight matrix are determined through 1-factorization and global updated. The steps are iteratively executed until the convergence is achieved for the Weston-Watkins algorithm.

4. Experiments and Analysis

To evaluate the All-in-One multiclass SVM classifier using a 1-factorization approach with Symmetric ADMM, the experiment is conducted in a distributed environment with a cluster of 3 nodes. Below are the detailed steps, configurations, and observations:

Setup

We experimented the algorithms in the in the cluster of 3 machines: One is a master node and remaining two are the worker nodes. The hardware configuration of the cluster is 16 GB RAM of each machine with 500GB SSD, Intel Core i5 CPU, @ 2.6 GHz speed with 0.1 Gigabit Ethernet network. We adopted the polynomial kernel for the SVM model to capture non-linear patterns in the data, with hyper-parameters C set to 10 (regularization parameter), γ set to 10 (kernel coefficient) and penalty parameter of ADMM ρ is set to 1. Each instance in the dataset is represented as a feature vector, making it suitable for kernel-based methods like SVMs to capture complex relationships in the data.
We implemented the algorithms using MPI framework (OpenMPI 4.2) for inter-node communication and OpenMP for intra-node parallelism in the Python environment.

Dataset Distribution

The Large Scale Hierarchical Text Classification (LSHTC) dataset is designed for large-scale, multi-class, and hierarchical text classification tasks. The dataset is derived from the Open Directory Project (ODP)**,** also known as DMOZ [28], which is a human-curated hierarchical directory of web pages. It contains hierarchical category labels for text documents. Each document is assigned to one or more categories within a large hierarchical taxonomy. It is a benchmark dataset used to evaluate machine learning models that handle a large number of classes and hierarchical relationships

Challenges of the LSHTC dataset

- Scalability: The large number of classes and instances requires efficient algorithms to handle memory and computation constraints.
- Class Imbalance: Some classes have significantly fewer instances than others, leading to skewed classification performance.
- Hierarchical Dependencies: Models must account for hierarchical relationships among classes, adding complexity to training and evaluation.
- Sparse Representation: The high scarcity of the feature matrix requires specialized storage and processing techniques, such as compressed sparse row (CSR) formats.

Training and Testing

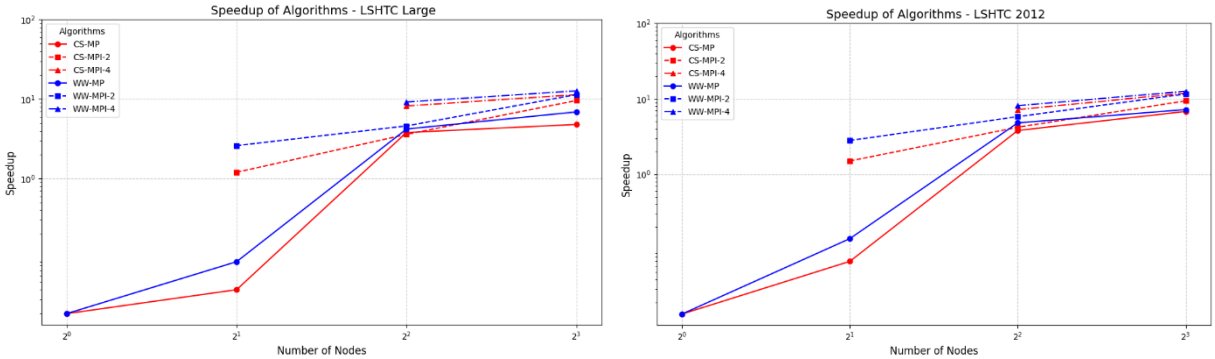
The LSHTC dataset is provided in multiple versions, in our experiment, we have used LSHTC- large and LSHTC-2012, details of the dataset provided in Table 3.1. These two datasets differing in the number of classes, instances, and hierarchy depth.

Table 1. Features of LSHTC Large and LSHTC-2012 Dataset

Feature	**LSHTC Large**	**LSHTC-2012**
Instances	~200,000 training, ~50,000 test	~93,000 training, ~7,000 test
Features	~50,000 sparse features	~16,000 sparse features
Classes	Over 20,000	~12,000
Dimension	~381,581	~575,555
Hierarchy Depth	Up to 15 levels	Up to 10 levels
Purpose	Scalability-focused benchmark	Hierarchical classification refinement
Challenges	Extreme scalability, deep hierarchies	Hierarchical consistency, feature sparsity

Results

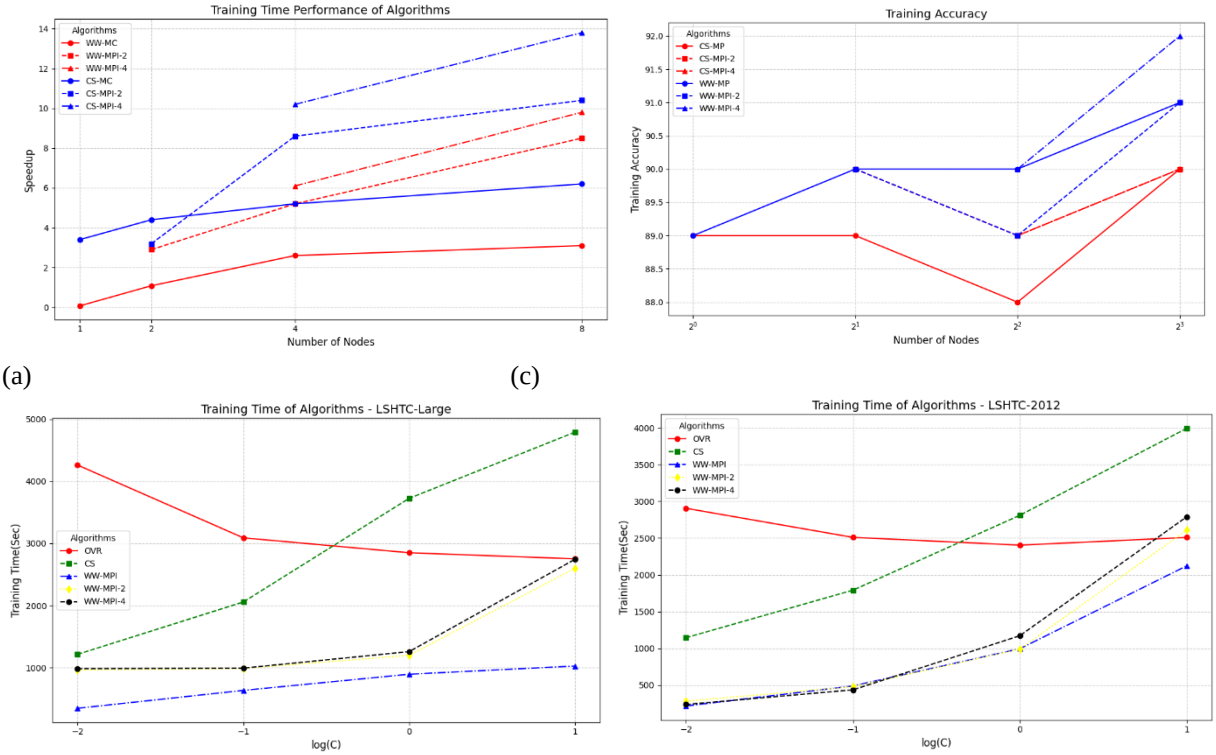
In order to measure the performance of the algorithms provided increasing the number of cores and run the algorithm for static number of iterations on the incremental size of the LSHTC dataset where the regularization parameter value is $[C = 10, and\ \gamma = 10]$ as polynomial kernel parameter and the ADMM penalty parameter$[\rho = 1]$. The dataset spread evenly on the worker nodes of the cluster to train locally by using MPI.



(a) (b)

Figure 3. Speedup of CS and WW algorithms for LSHTC Large (a) and 2012 (b)

The result of speedup of both algorithms shown in Figure 4.1 for the dataset LSHTC- Large (a) and LSHTC-2012 (b). Both the algorithms unveils the linear speedup and also shows a little cost of the communication. The observation shows that Weston-Watkins algorithm performed better compared to Crammer-Singer, eventually both algorithms exhibits linear speedup.

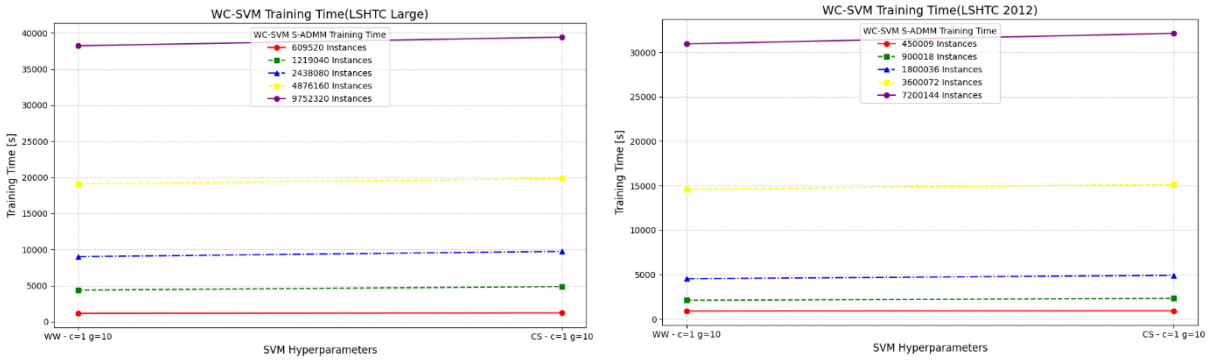


(a) (c)

(b) (d)

Figure 4. Training time accuracy of the algorithm for LSHTC-Large and LSHTC-2012

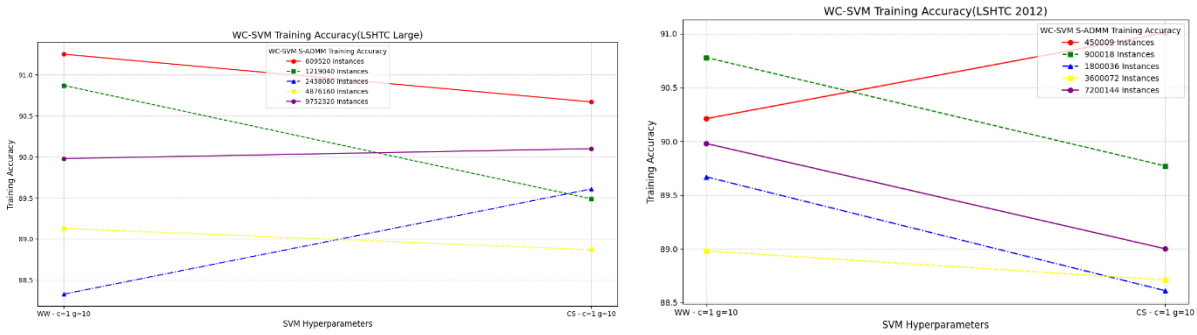
The performance of the algorithms is measured in terms of training time and accuracy. The above graphs Figure 4.2 (a) depicts the speedup of the algorithms as we increase the numbers of nodes.
Weston-Watkins algorithms achieves reasonable speed as compared with Crammer-Singer as communication overhead might be the reason and it is future research scope. Figure 4.2 (b) illustrates the training time of the algorithms including OVR for the different regularization parameter value [-2, -1, 0, 1] similarly from Figure 4.2 (d) both shows training time for LSHTC Large and LSHTC-2012 with polynomial kernel value and symmetric ADMM penalty parameter [$\gamma = 10$ and $\rho = 1$], WW takes less training time. It shows good result for SVM hyper-parameter$C = 1$ and $\gamma = 10$. Figure 4.2 (c) shows the training accuracy there is not much deviation in the accuracy.



(a) (b)

Figure 5. Training time of WW and CS algorithm for LSHTC-Large and LSHTC-2012

The training time of the Weston-Watkins and Crammer-Singer algorithms for LSHTC-Large and LSHTC-2012 is evaluated with following SVM hyper-parameters as shown.
$[C = 1, \gamma = 10, \rho = 1]$the penalty parameter of SADMM is set to 1. The Figure 4.3(a) is for LSHTC-Large and Figure 4.3 (b) is for LSHTC-2012 and the instances are incremented in power of 2 and trained up to 6 sets of the instances. It is observed that WW algorithms performs better than CS in the cluster of 3 nodes.



(a) (b)

Figure 6. Training Accuracy of WW and CS algorithm for LSHTC-Large and LSHTC-2012

The training time of the Weston-Watkins and Crammer-Singer algorithms for LSHTC-Large is shown in the Figure 4.4(a) as the dataset size increases both the algorithm shows almost constant accuracy. In case of LSHTC-2012 the Figure 4.4(b) shows there is little variations in the accuracy as the number of instances increases but it is in a range from 1 to 2 %. Our understanding and observation is that as dataset increases there is a slight variation in the accuracy of both the algorithms.

Table 2. F1-Score of both Algorithms

Dataset	**Micro-F1**				**Macro-F1**	
	OVR	**CS**	**WW**	**OVR**	**CS**	**WW**
LSHTC-Large						
log (c) : -2	15.88	40.16	39.21	2.86	25.29	20.16
-1	23.14	41.17	42.06	4.11	25.71	25.82
0	37.19	**44.38**	**44.20**	12.98	30.74	**36.62**
1	**41.87**	43.19	43.02	**25.17**	**36.11**	36.19
LSHTC-2012						
log (c) : -2	25.48	50.17	49.11	0.19	20.06	15.80
-1	40.01	52.86	54.62	2.10	22.91	25.15
0	51.11	**57.83**	**56.10**	13.26	**34.09**	**34.20**
1	**53.76**	54.91	53.91	**26.71**	32.55	30.11

The F1-Score, Micro-F1 and Macro-F1 is achieved by the both Crammer-Singer and Weston-Watkins algorithms on the LSHTC dataset. The best result across C values highlighted in the bold. Comparing WW and CS, WW performs marginally

better at classifying. To the best of our knowledge this is a first time comparison of well-known multiclass SVM algorithm implemented using symmetric ADMM with 1-factorization.

Table 3. Aspects of CS and WW algorithm comparison

Aspect	Crammer-Singer SVM	Weston-Watkins SVM
Loss Function	Penalizes misclassification margins.	Penalizes hinge loss for incorrect classes.
Number of Constraints	$O\ (n\ .\ K\ )$	$O\ (n\ .\ K\ )$, but fewer active constraints.
Slack Variables	Global ξ_i per sample	Pairwise $\xi_{i,k}$ for each incorrect class.
Computational Complexity	Slightly higher due to tighter constraints.	Lower due to simpler constraints.

Computational Complexity

- Local Updates: Solving the optimization problem at each node involves iterating over the local dataset. Complexity depends on the size of D_n and the number of classes K.
- Global Updates: Aggregating local weights and dual variables has a communication overhead proportional to the number of nodes 3 and considering cores we used up to 16 cores.
- Overall Complexity: For N nodes and M total instances in the dataset, the per-iteration complexity is approximately:

$$O\left(\frac{M}{N}\ .\ K\ .\ d\right)$$, where d is the feature dimension.

The computational time of Crammer-Singer (~56 sec) is marginally more compared with Weston-Watkins (~44) with convergence of 20 iterations and 56 iterations respectively.

5. Conclusion

This paper introduces an efficient and scalable framework for multi-class classification using symmetric ADMM with a 1-factorization approach, implemented in a distributed environment leveraging OpenMPI and OpenMP. The framework was evaluated on the LSHTC dataset, demonstrating strong performance in handling high-dimensional, sparse, and hierarchical data. Both the Crammer-Singer (CS) and Weston-Watkins (WW) formulations were implemented and analyzed. While WW exhibited superior performance in terms of classification accuracy and computational efficiency, CS performed also well but marginally introduces communication overhead. The combination of distributed computation with symmetric ADMM effectively addressed challenges associated with large-scale classification tasks. However, future work should focus on improving model generalization, particularly for imbalanced classes, and extending the framework's adaptability to diverse data modalities. Exploring deep learning-based approaches or hybrid models that integrate traditional SVMs with neural networks may provide further advancements in classification performance and computational efficiency.

References:

[1]. Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1), 1–122. https://doi.org/10.1561/2200000016

[2]. Shi, X., Chen, W., & Zhang, Q. (2020). Distributed multiclass SVM for large-scale text classification. *Journal of Machine Learning Research*, 21(1), 1-32.

[3]. Zhang, R., & Zhou, D. (2012). Parallel training of SVMs on large datasets using MapReduce. *IEEE Transactions on Big Data*, 2(1), 1-14.

[4]. Geeta, R. B., Totad, S. G., Reddy, P., & Shobha, R. B. (2015). "Big data structure and usage mining coalition", International Journal of Services Technology and Management, 21(4/5), 6.

[5].Wang, Y., & Zhou, X. (2015). A scalable OvR multiclass SVM for distributed frameworks. *ACM Transactions on Knowledge Discovery from Data*, 9(3), 1-18.

[6]. Geeta R.B., Shobha R.B, Shashikumar G. Totad and Prasad Reddy , "Smart Behavioral Driven Power Stasher" in Global Journal of Advanced Engineering Technologies, Vol3, Issue3-2014, ISSN:2277-6370, pp-274-282

[7]. Weston, J., & Watkins, C. (1999). Support vector machines for multi-class pattern recognition. In *Proceedings of the 7th European Symposium on Artificial Neural Networks (ESANN)* (pp. 219–224).Crammer, K., & Singer, Y. (2001). On the Algorithmic Implementation of Multiclass SVMs.

[8].Shashikumar G. Totad, Geeta R.B, and Prasad Reddy PVGD, "Scaling Data Mining Algorithms to Large and Distributed Datasets", International Journal of Database Management Systems(IJDMS), Vol.2, No.4, pp 26-35, Nov. 2010, ISSN: 0975-5705.

[9].Zaharia, M., Chowdhury, M., Das, T., et al. (2012). Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing.

[10].Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A Library for Support Vector Machines.

[11].Vijaykumar Bhajantri, Shashikumar G Totad, Geeta R. B, "Enhancement of Scalability of SVM Classifiers for Big Data" , Book chapter(no-9) in edited Book ", Advances in Data Science and Analytics", Wiley(Online library), Aug-2022, ISBN: 9781119791881 .

[12].Praveen M Dhulavvagol, S G Totad, "Comparative Study analysis of Classification Algorithms", Feb-2022 International Conference on Expert Clouds and Applications (ICOECA 2022) held at RV Institute of Technology & Management, Bangalore, during 3-4 Feb 2022.

[13].Cai, J., & He, K. (2012). A distributed algorithm for training large-scale hierarchical text classifiers. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management* (pp. 175–184). https://doi.org/10.1145/2396761.2396791

[14]. Preeti Jigalur, Geeta R. B, Shashikumar G. Totad, "Optimized Resource Utilization in Cloud Computing", International Journal of Computer Science, Vol-5,Issue 4, Jul-Aug 2017. pp. 123-125, ISSN: 2347-8578 , http://www.ijcstjournal.org/volume-5/issue-4/IJCST-V5I4P20.pdf, https://www.academia.edu/34350436/_IJCST_V5I4P20

[15].Neha Deshpande, Shashikumar G Totad, Karibasappa K. G. "Comparative analysis of optimization techniques on Multi-class SVM", IEEE 3rd International Conference of Emerging Technologies, Jain College , Belgaum, 27th May – 29th MAY 2022.

[16].Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification. *Technical Report, Department of Computer Science, National Taiwan University*.

[17].Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for SVM**.** In *Proceedings of the 24th International Conference on Machine Learning (ICML)* (pp. 807–814). https://doi.org/10.1145/1273496.1273598

[18].Shashikumar G. Totad, Geeta R.B, and Prasad Reddy PVGD, "Scaling Data Mining Algorithms to Large and Distributed Datasets", International Journal of Database Management Systems (IJDMS), Vol.2, No.4, pp 26-35, Nov. 2010, ISSN: 0975-5705

[19].Yu, H., Yang, S., & Zhu, S. (2012). Parallel distributed processing of massive hierarchical SVMs. In IEEE Transactions on Neural Networks and *Learning Systems*, 23(2), 192–200. https://doi.org/10.1109/TNNLS.2011.2178448

[20].Yang, L., Song, H., & Lu, Z. (2015). Efficient multi-class classification for large-scale hierarchical text datasets. In *Proceedings of the 23rd ACM International Conference on Multimedia* (pp. 1231–1235). https://doi.org/10.1145/2733373.2806258

[21]. Shashikumar G. Totad, Geeta R.B, and Prasad Reddy PVGD, "Batch incremental processing for FP-tree construction using FP-Growth algorithm", in Springer's KAIS journal(DOI 10.1007/s10115-012-0514-9) Vol. 33, Issue 2, pp 475–490, 2012 (IF:2.22)

[22].Wang, Y., & Gu, J. (2015). *A graph-based approach to multi-class classification problems*. *IEEE Transactions on Neural Networks and Learning Systems*, 26(3), 637 648. DOI: 10.1109/TNNLS.2014.2336615

[23].Yashaswini Joshi, Geeta R.B, Shashikumar G. Totad, and Prasad Reddy PVGD, "Mobile Agent based Frequent Pattern Mining for Distributed Databases", (Springer sponsored) International Conference on Intelligent Computing and Communication (ICICC - 2017) on 2 to 4th August 2-4, 2017 at MAEER's MIT College of Engineering, Pune; Intelligent Computing and Information and Communication(Springer BookChapter), 20-01-2018,pp:77-85,DOI:10.1007/978-981-10-7245-1_9, ISBN: 978-981-10-7244-4.

[24]. Alber, M., Zimmert, J., Dogan, U., & Kloft, M. (2017). *Distributed optimization of multi-class SVMs*. *PLOS ONE*, 12(6), e0178161

[25]. Fang, C.-H., Kylasa, S. B., Roosta, F., Mahoney, M. W., & Grama, A. (2018). *Newton-ADMM: A distributed GPU-accelerated optimizer for multiclass classification problems*. *arXiv preprint arXiv:1807.07132*

[26]. Vijayakumar H. Bhajantri, Shashikumar G. Totad, Geeta R. Bharamagoudar. (2024). Scalable Distributed Computing for Large-Scale SVM: A Symmetric ADMM Approach. Journal of International Crisis and Risk Communication Research, 1167–1178. Retrieved from https://jicrcr.com/index.php/jicrcr/article/view/2547

[27]. Crammer, K., & Singer, Y. (2001). *On the algorithmic implementation of multiclass kernel-based vector machines*. *Journal of Machine Learning Research*, 2, 265–292.

[28]. Partalas, I., Kosmopoulos, A., Baskiotis, N., Paliouras, G., Gaussier, E., Androutsopoulos, I., & Klasinas, I. (2015). LSHTC: A benchmark for large-scale text classification. arXiv preprint arXiv:1503.08581. https://arxiv.org/abs/1503.08581

[29] Yang, L., Li, J., & Fang, Z. (2019). Symmetric ADMM for distributed machine learning. *Proceedings of the Neural Information Processing Systems (NeurIPS)*, 2019.

[30] Geeta R.B, Shashikumar G. Totad and Prasad Reddy PVGD " Amalgamation of Web usage Mining and Web structure Mining" International Journal of Recent Trends in Engineering, Vol. 1, No. 2, May 2009

[31] Gabriel, E., Fagg, G. E., Bosilca, G., Angskun, T., Dongarra, J. J., Squyres, J. M., & Lumsdaine, A. (2004). Open MPI: Goals, concept, and design of a next-generation MPI implementation. In Proceedings of the 11th European PVM/MPI Users' Group Meeting (pp. 97-104). Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-540-30218-6_19