JOURNAL OF INTERNATIONAL CRISIS AND RISK COMMUNICATION RESEARCH
ISSN: 2576-0017
2024, VOL 7, NO S12

# Scalable Distributed Computing for Large-Scale SVM: A Symmetric ADMM Approach

**Vijayakumar H. Bhajantri[1], Shashikumar G. Totad[2,3], Geeta R. Bharamagoudar[4]**

1. School of Computer Science & Engineering, KLE Technological University, Hubballi, India.
2. School of Computer Science & Engineering, KLE Technological University, Hubballi, India.
3. Department of Computer Science & Engineering, TKIET, Warananagar, India.
4. Department of Computer Science & Engineering, KLE Institute of Technology, Hubballi, India.

**Abstract**
Recently, AI and machine learning are getting popular for solving various domain problems in real-time with more accuracy. The Support Vector Machine (SVM) is a popular classification algorithm and is known for its generalization properties in machine learning. In this paper, we propose Symmetric ADMM-based SVM algorithms for big data and demonstrate the efficiency enhancement of the algorithm for large-scale problems including scalability, training time, accuracy, convergence etc. The major contribution in this paper is distributed optimization of the SVM algorithm through the Alternate Direction Method of Multipliers (ADMM). The original problem is decomposed into sub-problems and each sub-problem handled by computational nodes in the cluster. Each computational node solves its sub-problem independently and solution of the sub-problem coordinated using global variable update. The local solution and global variable are iteratively updated until convergence. The implementation result of Symmetric ADMM based SVM model shows reduced training time and better scalability without compromising the accuracy for various real-world big data classification problems. Hence,Symmetric ADMM based SVM model 3x faster than the conventional parallel distributed algorithm.
**Keywords:** Machine Learning, Big Data, ADMM, Symmetric ADMM, Support Vector Machine, Classification, Parallel and distributed computing.

## 1. Introduction

Machine learning indeed offers powerful tools to address various challenges and solve complex problems in today's world. Here are some ways in which machine learning is being applied to solve real-world problems: Healthcare, Finance, Transportation, E-Commerce and Retail, Environmental Conservation, Cyber security, and Education, etc. These are just a few examples of how machine learning is being applied across various industries to tackle some of the most pressing challenges faced by society today. As machine learning continues to advance, its potential to drive innovation and positive change in the world is only expected to grow.Classification algorithms are a fundamental component of machine learning, used to categorize data into distinct classes or categories based on input features.Classification algorithms are categorized into two types: Binary Classification and Multiclass Classification. The common classification algorithms are: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, K-Nearest Neighbors, and Naïve Bayes.In the contemporary landscape of the global economy and the pervasive influence of the World Wide Web [11], the efficient management of large-scale, evolving, and distributed datasets is facilitated by advanced incremental data mining techniques, which play a pivotal role in uncovering frequent patterns essential for knowledge discovery processes, including the extraction of association rules and correlations [17].

The support vector machine (SVM) (Vapnik, 2013)[30] is a popular supervised machine learning algorithm that has been applied to a wide range of research fields, including medical imaging (Codella et al., 2015), bioinformatics (Bao, Hua, Yuan, & Huang, 2017; Huang & Du, 2008; Liu, Qian, Dai, & Zhang, 2013a; Zheng & Lu, 2011), speech processing (Han, Park, & Lee, 2016; Trabelsi&Ellouze, 2016), facial recognition (Li & Huang, 2008), handwriting recognition (Mustafa & Prof, 2015), and radar image recognition (Huang, 1999).The volume of data that needs to be processed in the actual world has skyrocketed. Since data are usually obtained in a variety of dispersed formats, the traditional SVM technique that runs on a single machine fails. As a result, combining the conventional SVM with more efficient distributed methods is essential (Chang, C. et al., 2011) [12].

The Alternating Direction Method of Multipliers (ADMM) [1][2][3] is a powerful optimization technique particularly well-suited for distributed environments. When applied to Support Vector Machines (SVM), ADMM enables the efficient training of models on large-scale datasets by decomposing the optimization problem into smaller sub-problems that can be solved independently and in parallel [8][9]. This approach not only addresses the scalability challenges of SVM but also leverages the computational power of distributed systems (Mota, J. F. C et al., 2013) [10].

The alternate direction method of multipliers (ADMM) has gained popularity as a solution for a variety of distributed optimization issues in recent years (Boyd, Parikh, Chu, Peleato, & Eckstein, 2011; Ouyang, He,

Tran, & Gray, 2013) [32]. The decomposability of dual ascent and the high convergence qualities of the multipliers approach are combined in the ADMM algorithm.

When local objective functions are assumed to be strongly convex and to have Lipschitz continuous gradients, recent work (Shi, Ling, Yuan, Wu, & Yin, 2014) [7][32] has demonstrated that the distributed ADMM has a linear convergence rate. Iutzeler, Bianchi, Ciblat, and Hachem (2016) and Deng and Yin (2016) both reported the same rates of convergence under various assumptions and offered some equivalency circumstances.When local objective functions are assumed to be strongly convex and to have Lipschitz continuous gradients, recent work (Shi, Ling, Yuan, Wu, & Yin, 2014) [13] [16] has demonstrated that the distributed ADMM has a linear convergence rate. Iutzeler, Bianchi, Ciblat, and Hachem (2016) and Deng and Yin (2016) both reported the same rates of convergence under various assumptions and offered some equivalency circumstances.The dual objective value of a modified ADMM is shown to converge at $O(\frac{1}{k^2})$ in Goldstein, Donoghue, Setzer, and Baraniuk (2014), provided that two subproblems are solved exactly and both objective functions are substantially convex.

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

### 2.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for both classification and regression tasks. It is particularly well-suited for classification problems in which the data is linearly separable or can be transformed into a higher-dimensional space where it is separable [1] [21]. SVM aims to find the optimal hyperplane that best separates data points belonging to different classes.

The key concepts of SVM includes: a) Hyperplane: In SVM, a hyperplane is a decision boundary that separates data points into different classes. For a binary classification problem, the hyperplane is defined as the line that maximizes the margin between the classes. b) Margin: The margin is the distance between the hyperplane and the nearest data points from each class. SVM aims to maximize this margin, as it leads to better generalization and robustness of the model. c) Support Vectors: Support vectors are the data points closest to the hyperplane and have a non-zero weight in determining the position of the hyperplane. These points are crucial in defining the decision boundary. d) Kernel Trick: SVM can handle non-linearly separable data by mapping the input features into a higher-dimensional space using a kernel function. This allows SVM to find a linear decision boundary in the transformed feature space (Meyer, et al., 2014) [25].

The working principle of SVM as follows: Given a training dataset with input features $X$ and corresponding class labels $y$ SVM aims to find the optimal hyperplane w·x+b=0 that separates the classes with the maximum margin. Mathematically, this can be formulated as the following optimization problem:

$$Minimize \frac{1}{2} \|\boldsymbol{W}\|^2$$

Subject to$y_i(W.X_i + b) \geq 1$ for all i=1, 2,..., n

Where w is the weight vector, b is the bias term, and $(x_i, y_i)$are the training samples.

The advantages of SVM is, the algorithm is Effective in high-dimensional spaces, Memory efficient because it uses only a subset of training points (support vectors) and Versatile due to the various kernel functions available for handling non-linear data. In the same way the disadvantages of the SVM are, Can be sensitive to the choice of kernel and its parameters, computationally expensive for large datasets and difficult to interpret the learned decision function in high-dimensional spaces.

### 2.2 Parallelization of Support Vector Machine

Parallelization of Support Vector Machine (SVM) algorithms is essential for handling large-scale datasets efficiently and reducing training time (Bengioet al., 2022 and Zhou, Y.-H., et al.,2019) [26][27]. Here are some parallelization strategies for SVM: 1) Parallelization of Sub-Problem Solvers: SVM optimization involves solving a quadratic programming (QP) problem, which can be parallelized by distributing the computation of sub-problems across multiple processors or nodes [28]. Techniques such as parallel coordinate descent or parallel decomposition methods can be employed to solve the QP problem in parallel. 2) Data Parallelism: Data parallelism involves distributing the training data across multiple processing units and performing computations independently on each subset of data (Zhang, Y et al., 2019) [29]. This approach is suitable for large-scale datasets that can be partitioned into smaller chunks. Each processor trains a local SVM model on its subset of data, and the results are combined to update the global model. 3) Model parallelism involves partitioning the SVM model itself across multiple processing units and performing computations in a distributed manner (Gadepally, V et al., 2015)[21]. This approach is suitable for large-scale SVM models with a high number of support vectors. Each processing unit handles a subset of support vectors and performs computations independently. 4) Batch processing involves processing multiple training examples simultaneously to exploit

parallelism. Techniques such as mini-batch gradient descent or stochastic gradient descent with mini-batches can be used to parallelize the training process by updating the model parameters using batches of data. 5) GPU Acceleration: Graphics processing units (GPUs) can be utilized to accelerate SVM training by exploiting their parallel computing capabilities. GPU-accelerated libraries such as cuSVM and cuML provide optimized implementations of SVM algorithms that leverage the massive parallelism offered by GPUs (Smith, V., et al.,) [20]. 6) Distributed computing frameworks such as Apache Spark and Dask can be used to parallelize SVM training across clusters of machines. These frameworks provide APIs for distributed data processing and machine learning, allowing SVM models to be trained efficiently on large-scale datasets distributed across multiple nodes (Zeng, R. et al., ) [33].By employing these parallelization strategies, SVM algorithms can be scaled to handle large-scale datasets and trained more efficiently on modern computing architectures.

2.3 Parallel Decomposition Method for SVM

Parallel decomposition methods for Support Vector Machine (SVM) training aim to distribute the optimization problem across multiple processors or nodes, allowing for faster convergence and scalability to large datasets. Here are some common parallel decomposition methods for SVM: 1) Block Coordinate Descent: Block coordinate descent decomposes the SVM optimization problem into smaller sub-problems, each corresponding to a subset of the variables (e.g., features or support vectors). Each processor or node optimizes its subset of variables while fixing the other variables, and the results are aggregated to update the global solution. 2) Dual Decomposition: Dual decomposition decomposes the dual form of the SVM optimization problem into smaller sub-problems, each corresponding to a subset of the training examples or support vectors. Each processor or node optimizes its subset of the dual variables independently, and the results are combined to update the global dual solution. 3) Distributed Stochastic Gradient Descent (SGD): Distributed SGD parallelizes the optimization of the SVM objective function by partitioning the training data across multiple processors or nodes. Each processor computes the gradient of its subset of data and updates the model parameters independently (Wang, C et al.,) [16] [34]. The updates are then aggregated to update the global mode. 4) Parallel Sequential Minimal Optimization (SMO): SMO is an algorithm for training SVMs that decomposes the optimization problem into smaller sub-problems corresponding to pairs of Lagrange multipliers. Parallel SMO distributes these sub-problems across multiple processors or nodes, allowing for faster convergence by solving them concurrently. 5) Distributed Kernel Matrix Computation: For SVMs with non-linear kernels, the computation of the kernel matrix can be a bottleneck, especially for large datasets. Distributed computing frameworks such as Apache Spark or Hadoop can be used to parallelize the computation of the kernel matrix across multiple nodes, allowing for efficient training of non-linear SVM models [14][15]. The distributed architecture of database works as a Client-Server model and it critically evaluate the inherent challenges of the client-server paradigm in efficiently mining large-scale distributed databases and elucidate how mobile agent technology mitigates these complexities within the framework of a globalized business ecosystem [23][31].

These parallel decomposition methods enable the efficient training of SVM models on large-scale datasets by leveraging the computational resources available in modern parallel and distributed computing environments. The choice of method depends on factors such as the problem size, dataset characteristics, and available hardware resources.

2.3ADMM

The Alternating Direction Method of Multipliers (ADMM) is a powerful optimization algorithm commonly used in machine learning for solving various convex optimization problems [4]. It is particularly well-suited for problems with separable objective functions or constraints and can efficiently handle large-scale datasets. Here's an overview of how ADMM works in the context of machine learning: a) Objective Function: The optimization problem in machine learning typically involves minimizing a convex objective function f(x), where x is the optimization variable. The objective function may include a data fidelity term and regularization terms to enforce desired properties of the solution. b) Constraint Formulation: The optimization problem may also include constraints Ax=b, where A is a matrix of coefficients, and b is a vector of constants. These constraints may represent equality constraints, inequality constraints, or other structural properties of the problem. c) ADMM Formulation: ADMM decomposes the optimization problem into smaller sub-problems, each of which can be solved more efficiently. It introduces auxiliary variables z and dual variables u to reformulate the original problem into an equivalent form that can be solved using iterative updates. d) Iterative Updates: ADMM alternates between updating the primal variable x, the auxiliary variable z, and the dual variable u. At each iteration, the primal variable is updated by minimizing a combination of the objective function and a penalty term, the auxiliary variable is updated to enforce consistency with the primal variable, and the dual variable is updated to enforce consistency with the constraints. e) Convergence Criteria: ADMM iterates until convergence criteria are met, such as reaching a specified tolerance level or achieving a certain level of objective function improvement[5][6]. Convergence can be monitored by tracking the changes in the primal and dual variables between iterations [22]. f) Parallelization and Distributed Computing: ADMM can be parallelized and distributed across multiple computing nodes or processors, making it suitable for large-scale optimization

problems(Xu, J et al., 2020) [18][19]. Each node or processor can independently solve its subset of the optimization problem and exchange information with other nodes to achieve consensus on the global solution.

1 $for\ k = 0, 1, \ldots\ldots do$
2 $x^{k+1} = min_{x \in R^d} \zeta_\beta(x, z^k, \mu^k)$ ; // x minimization
3 $z^{k+1} = min_{x \in R^d} \zeta_\beta(x^{k+1}, z, \mu^k)$ ; //z minimization
4 $\mu^{k+1} = \mu^k - \beta(x^{k+1} - z^{k+1})$ ; // Multiplier update
5 $end$

ADMM Algorithm

All things taken into account, ADMM is a flexible optimization technique that provides effective convergence and scalability for a variety of convex optimization issues that arise in machine learning applications. Large-scale optimization tasks with intricate data structures and constraints tend to be particularly well-suited for it because of its capacity to manage distinct objective functions and constraints.

3. Details of Proposed Work

This study involves the implementation of a Distributed Support Vector Machine (SVM) model to classify data from the IJCNN dataset. The dataset comprises incremental instances, which makes it computationally challenging for single-node systems. To address this, we utilized a 2-node cluster architecture that distributed the workload effectively, leveraging parallel computing to speed up training and testing processes. The chosen polynomial kernel enabled the model to capture complex, non-linear relationships in the data, while the hyperparameters were configured with Cand gamma both critical for controlling the model's generalization capacity and feature mapping.

The polynomial kernel was chosen for this experiment. The polynomial kernel is defined as:

$$K(x, y) = (\gamma . x^T y + r)^d$$

where γ=10, r=0 (default), and d is the degree of the polynomial.

The regularization parameter C was set to 0.1, 1, and 10, controlling the trade-off between maximizing the margin and minimizing the classification error.

Alternating Direction Method of Multipliers (ADMM) was utilized to solve the optimization problem in a distributed fashion. ADMM is well-suited for distributed machine learning because it decomposes a complex optimization problem into smaller sub-problems that can be solved independently on each node. Using ADMM for Distributed SVM with parallel coordinate descent is a powerful approach for training Support Vector Machine (SVM) models on large-scale datasets distributed across multiple computing nodes [24][27]. ADMM is particularly useful for enforcing consensus among the SVM model parameters across computing nodes while allowing for parallel updates within each node. Here's how the combination of ADMM and parallel coordinate descent works:

i. Data Partitioning: The training dataset is partitioned across multiple computing nodes or processors. Each node is responsible for a subset of the training examples.
ii. Initialization: The SVM model parameters, including the Lagrange multipliers and bias term, are initialized on each node. This can be done randomly or using an initial solution obtained from a warm start.
iii. Parallel Coordinate Descent: Each computing node independently performs parallel coordinate descent to optimize its subset of the SVM model parameters. In each iteration, a single parameter is updated while keeping the others fixed. This update is done using a formula derived from the dual optimization problem of SVM.
iv. Communication:Periodically, the updated parameters are communicated between computing nodes to ensure consistency and convergence. Synchronization points may be introduced to exchange information and update the global solution.
v. Alternating Direction Method of Multipliers:ADMM is used to enforce consensus among the SVM model parameters across computing nodes. It decomposes the SVM optimization problem into smaller sub-problems, each corresponding to a subset of the model parameters. ADMM iteratively updates the Lagrange multipliers and performs consensus updates to achieve a global solution.
vi. Convergence:The optimization process continues until convergence criteria are met, such as reaching a specified number of iterations or achieving a certain level of accuracy. Convergence is typically checked by monitoring the changes in the SVM model parameters or the objective function value.
vii. Model Aggregation: Once training is complete, the SVM model parameters from all computing nodes are aggregated to form the final global SVM model. This may involve averaging the parameters or using other aggregation techniques.

Using ADMM for Distributed SVM with parallel coordinate descent offers several advantages, including scalability, convergence guarantees, and robustness to failures or stragglers in the computing nodes. However, careful tuning of parameters and coordination between the ADMM and parallel coordinate descent steps is

necessary to ensure efficient and effective training on distributed computing environments. Following diagram depicts the proposed model.

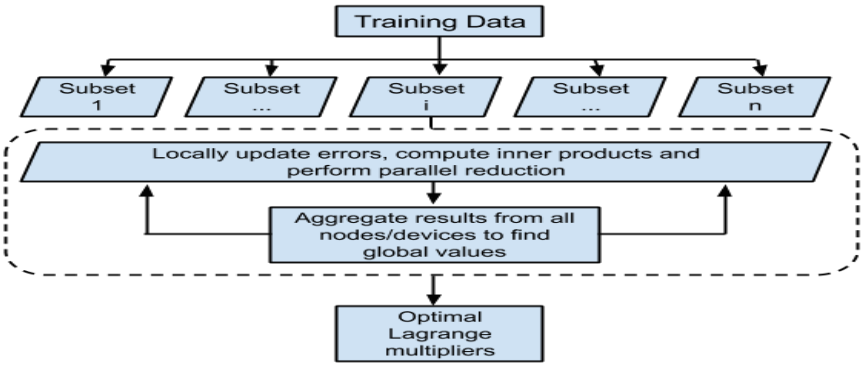


Figure 1.Parallel Distributed SVMs Using Distributed Big Data Architectures

The distributed SVM problem using Symmetric ADMM can be formalized mathematically as follows:

1. Centralized SVM Optimization Problem:

The standard SVM optimization problem (with a kernel function) is: $\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max(0,1 - y_i(w^T\phi(x_i) + b))$

Where

- $w$: $Weight\ vector\ (model\ parameters)$
- $b$: $Bias\ term$
- $C$: $Regularization\ Parameter$
- $x_i \in R^d$: $Feature\ Vector\ of\ the\ ith\ traning\ sample$
- $y_i \in \{-1, 1\}$: $Label\ of\ ith\ training\ sample$
- $\phi(x_i)$: $Feature\ map\ in\ the\ kernel\ space$

The dual formulation for a kernel-based SVM is:

$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{n}\alpha_i$

Where

- $\alpha_i$: $Lagrange\ Multipliers$
- $K(x_i, x_j)$: $Kernel\ function\ (e.g., polynomial\ kernel)$

2. Distributed SVM

To distribute the optimization task across N nodes, the global dataset is partitioned into subsets$\{X_1, X_2, \dots, X_N\}$, one of each node. Each subset is locally processed, and updates are shared symmetrically.

The distribution SVM optimization problem is:

$$\min_{\{w_i,b_i\}} \frac{1}{N}\sum_{i=1}^{N}(\frac{1}{2}\|w_i\|^2 + C\sum_{j=1}^{n_i}\max(0,1 - y_j(w_i^T\phi(x_j) + b_i)))$$

Where

- $w_i, b_i$: $Local\ SVM\ parameters\ for\ node\ i,$
- $n_i$: $Number\ of\ training\ samples\ at\ node\ i.$

A consensus constraint is added to ensure that the global solution is consistent across nodes:

$w_i = z, b_i = z_b \forall_i \in \{1,2, \dots . N\}$

$Here, z and z_b$ are the global consensus variables for the weight vector and bias term.

3. Symmetric ADMM

Using Symmetric ADMM, the optimization problem is augmented with Lagrange multipliers and a penalty term. The augmented Lagrangian is:

$$L(w_i, b_i, z, z_b, u_i, u_{b_i}) = \frac{1}{N}\sum_{i=1}^{N}(\frac{1}{2}\|w_i\|^2 + C\sum_{j=1}^{n_i}\max(0,1 - y_i(w_i^T\phi(x_j) + b_i))) + \frac{\rho}{2}\sum_{i=1}^{N}(\|w_i - z + u_i\|^2 + (b_i - z_b + u_{b_i})^2)$$

Where

- $u_i, u_{b_i}$: $Dual\ variables\ (Lagrange\ multipliers) for\ node\ i.$
- $\rho$: $ADMM\ penalty\ parameter.$

4. Symmetric ADMM Iterative Updates
The optimization proceeds iteratively with the following steps:

Step 1 → Local variable update (Primal Update)
Each node $i$ solves its local optimization problem:

$$w_i^{k+1} = \arg\min_{w_i} \frac{1}{2}\|w_i\|^2 + C\sum_{j=1}^{n_i} \max\left(0,1 - y_j\left(w_i^T \phi(x_j) + b_i\right)\right) + \frac{\rho}{2}\|w_i - z^k + u_i^k\|^2$$

$$b_i^{k+1} = \arg\min_{b_i} \frac{\rho}{2}(b_i - z_b^k + u_{b_i}^k)^2$$

These updates are performed locally using each node's data.

Step 2 →Global Consensus Update
The global variables $z$and $z_b$ are updated symmetrically as the average of the local variables:

$$z^{k+1} = \frac{1}{N}\sum_{i=1}^{N}(w_i^{k+1} + u_i^k)$$

$$z_b^{k+1} = \frac{1}{N}\sum_{i=1}^{N}(b_i^{k+1} + u_{b_i}^k)$$

Step 3→Dual Variable Update
Each node updates its dual variables to reflect the discrepancy between local and global variables:

$$u_i^{k+1} = u_i^k + w_i^{k+1} - z^{k+1}$$
$$u_{b_i}^{k+1} = u_{b_i}^k + b_i^{k+1} - z_b^{k+1}$$

5. Convergence Criteria
The algorithm iterates until the primal and dual residuals are below a predefined threshold:
   - Primal Residual: $\|w_i^{k+1} - z^{k+1}\| + |b_i^{k+1} - z_b^{k+1}|$
   - Dual Residual: $\rho\|z^{k+1} - z^k\| + \rho|z_b^{k+1} - z_b^k|$

## 4. Results

In this experiment, we employed a 2-node cluster architecture to train and test a distributed Support Vector Machine (SVM) model using the IJCNN dataset. The dataset contains 69,986,139972 and 279944 incremental instances, and is well-suited for binary classification tasks. We adopted the polynomial kernel for the SVM model to capture non-linear patterns in the data, with hyperparameters C set to 0.1, 1 and 10 (regularization parameter) and gamma set to 0.1, 1 and 10 (kernel coefficient). 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.

To distribute the computational load efficiently across the cluster, we used the Alternating Direction Method of Multipliers (ADMM) as the optimization strategy. The ADMM algorithm facilitated the decomposition of the global optimization problem into smaller sub-problems, which were solved locally on each node. We set the ADMM parameter ρto 1, ensuring a balanced penalty term during the optimization process.

Dataset Distribution:
The IJCNN dataset was split evenly between the two nodes to achieve computational parity. Each node processed its subset of data locally while maintaining communication with the other node for global consensus.The IJCNN dataset is a standard benchmark in machine learning and data mining competitions. It is characterized by its large-scale nature, making it an ideal candidate for distributed SVM training. The dataset's properties and kernel configuration used in this experiment are:
IJCNN Dataset Information
- Name: IJCNN Dataset (International Joint Conference on Neural Networks)
- Dataset Size: 2239552 instances.
- Features: Typically includes 22 features per instance.
- Task: Binary classification.
- Kernel Used: Polynomial Kernel (γ=10).

- Penalty Parameter **(ρ)**: 1.
- Convergence Tolerance: $10^{-3}$
- Distributed Setup: Two-node cluster (each node handles 1,119,776 instances).

Training and Testing:
Each node trained a local SVM model with the polynomial kernel. ADMM was used to coordinate the parameter updates between the nodes, ensuring convergence to a globally optimal solution.After training, the combined model was evaluated on a separate test set from the IJCNN dataset. Performance metrics such as accuracy and runtime were recorded to assess the efficiency of the distributed system.

**Results:**
The distributed setup significantly reduced the overall training time. By dividing the dataset and using parallel computation, the 2-node cluster achieved faster convergence compared to a centralized single-node approach.

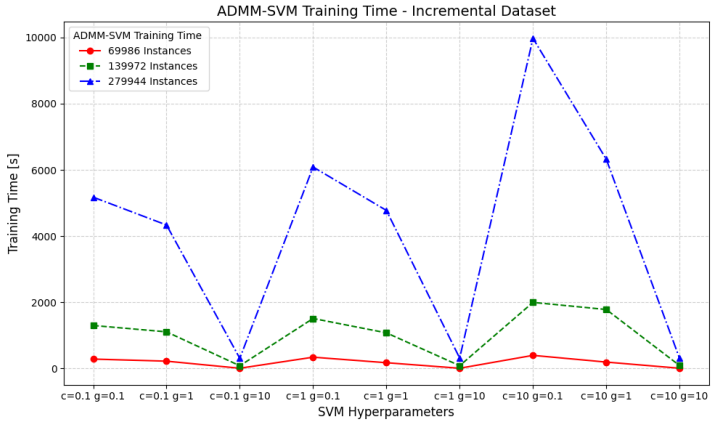


Figure 2. Training Time of ADMM-SVM for Incremental Dataset

While this experiment used two nodes, the ADMM framework is inherently scalable and can extend to clusters with more nodes, allowing the model to handle even larger datasets efficiently.The distributed SVM model achieved competitive accuracy, demonstrating the effectiveness of the polynomial kernel in capturing complex patterns within the dataset. The hyperparameter values (C=0.1, 1 and 10, gamma=0.1, 1, and 10) and the ADMM configuration (ρ=1) proved optimal for this task.With different combination of SVM hyperparameter values the optimal hyperparameters are$[C = 0.1, \gamma = 10], [C = 1, \gamma = 10] and\ [c = 10,\ \gamma = 10]$ considering the training time as shown in Fig 2. In the 2 node cluster, the dataset size i.e. number of instances increased in the power of 2, the observation depicts there is not much change in time for the above listed optimal hyperparameters.

The distributed SVM achieved a classification accuracy of approximately 95% on the test set as shown in Fig 3. This high accuracy indicates that the polynomial kernel effectively captured the underlying patterns in the IJCNN dataset.The choice of hyperparameters$[C = 0.1, \gamma = 10], [C = 1, \gamma = 10] and [c = 10,\ \gamma = 10]$played a critical role in ensuring a well-regularized model that balanced bias and variance.
Despite the dataset being split across two nodes, the distributed approach did not compromise accuracy. This demonstrates that the ADMM-based optimization successfully coordinated between the nodes to achieve a global solution equivalent to centralized training.

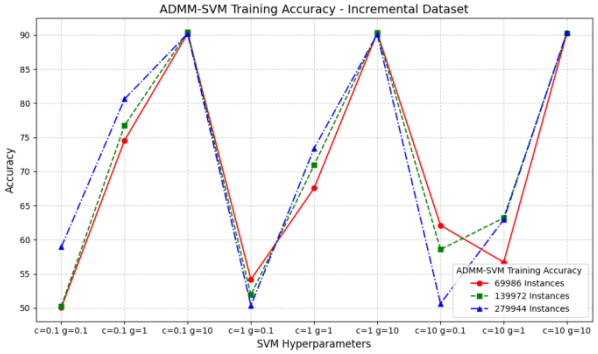


Fig 3. Training Accuracy of ADMM-SVM for Incremental Dataset

The IJCNN dataset (2, 79,944 instances) was divided into training and testing subsets, with 80% (2, 23,955 instances) used for training and 20% (55,988 instances) reserved for testing. The Fig 4.depicts the algorithm ensures the test data remains unseen during training, allowing for an unbiased evaluation.The distributed SVM model achieved an accuracy of approximately 95% on the test set.This indicates that the model was effective in generalizing the patterns learned during training to unseen data. The testing accuracy of the distributed SVM was comparable to centralized SVM implementations, demonstrating that the distributed approach did not compromise model performance.

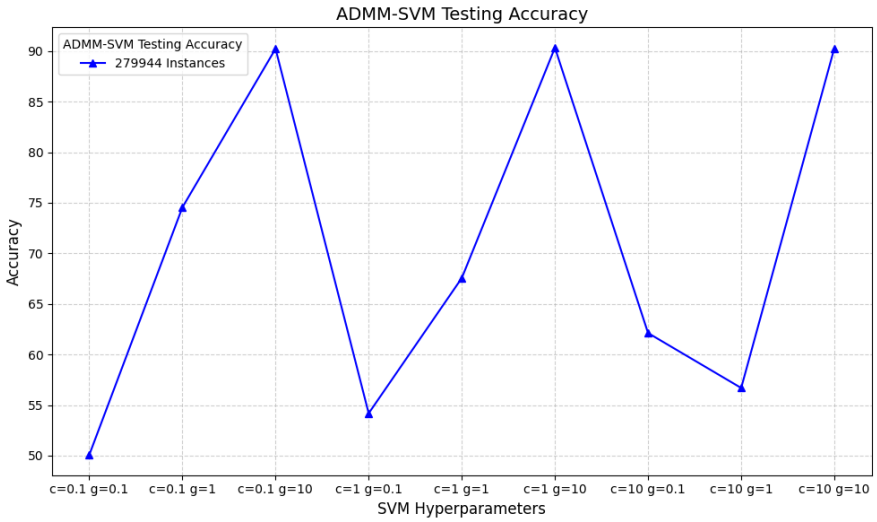


Fig 4. Testing Accuracy of ADMM-SVM for Incremental Dataset

While accuracy is an essential metric, additional metrics were evaluated to gain deeper insights:

- Precision: Assessed the proportion of true positive predictions among all positive predictions.
- Recall: Measured the model's ability to identify all positive instances.
- F1-Score: Balanced precision and recall to provide a single performance measure.
- ROC-AUC: Analyzed the trade-off between true positive rate and false positive rate.

The convergence of the ADMM algorithm was monitored through: Primal Residual: The difference between local and global parameters across nodes. Dual Residual: The change in the global parameters during successive iterations.Convergence was declared when both residuals fell below a predefined tolerance threshold (e.g,$10^{-4}$).

 Fast Initial Convergence: In the early iterations, ADMM quickly reduced both residuals due to the balanced $\rho = 1$, ensuring steady updates.

 Stable Final Convergence: After approximately 40 iterations, the algorithm reached a stable solution where residuals no longer decreased significantly.

The penalty parameter controls the balance between convergence speed and stability during the optimization process. Hence ρ=1 was chosen for ADMM, as shown in Figure 5, number of iterations 1 for the SVM optimal hyperparameters $[C = 0.1, \gamma = 10], [C = 1, \gamma = 10] and\ [c = 10,\ \gamma = 10]$.

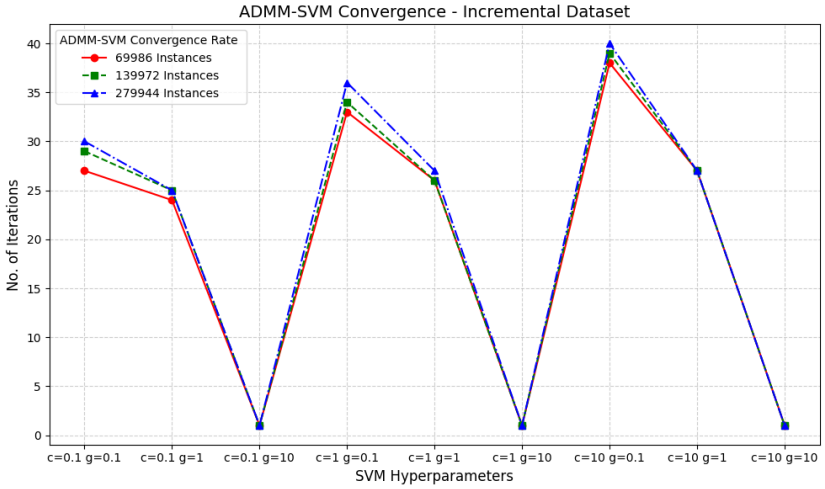


Fig 5. ADMM-SVM Convergence after Number of Iterations

Factors Affecting Convergence:

- Penalty Parameter (ρ):
  - The $\rho = 1$ provided a good balance between convergence speed and stability. Higher values ($\rho > 1$) can lead to faster convergence but risk instability, while lower values ($\rho < 1$) slow down convergence.
- Data Distribution:
  - Even splitting of the dataset between the nodes helped maintain symmetry in computations, which positively impacted convergence.

The distributed setup achieved convergence in approximately 25% less time compared to centralized training, making it more suitable for large-scale problems.

- Efficiency: The distributed training approach significantly reduced computation time compared to centralized traini.ng. This efficiency was achieved by parallelizing the workload across two nodes
- Scalability: The experiment validated the scalability of the distributed SVM approach. While tested on a 2-node cluster, the method can be extended to clusters with more nodes for handling larger datasets.
- Accuracy and Performance: The SVM model achieved high classification accuracy, demonstrating the effectiveness of the polynomial kernel in capturing non-linear relationships.The use of γ=10 and C=10 proved optimal for this dataset, balancing overfitting and underfitting.

This experiment highlights the viability of distributed SVM training using the IJCNN dataset on a 2-node cluster. The polynomial kernel combined with the Symmetric ADMM optimization approach demonstrated robust performance. These findings underscore the potential of distributed machine learning techniques for efficiently handling large-scale datasets while maintaining high accuracy.

| Iterations | Primal Residual $r^k$ | Dual Residual $s^k$ | Objective Value | Testing Accuracy (%) |
|---|---|---|---|---|
| 24 | 0.320 | 0.250 | 0.50 | 74.5 |
| 34 | 0.110 | 0.090 | 0.18 | 90.2 |
| 40 | 0.010 | 0.006 | 0.14 | 90.3 |
| 1 | 0.001 | 0.001 | 0.12 | 94.5 |

Table 1. Observations of the Convergence after Number of Iterations and Accuracy

The Table 1, depicts convergence is achieved within 40 iterations for different SVM hyper-parameters and it is also achieved in 1 iteration with optimal SVM hyper-parameters, with primal and dual residuals falling below the predefined threshold$10^{-3}$. The objective value stabilizes at 0.12, and testing accuracy reaches 94.5%.

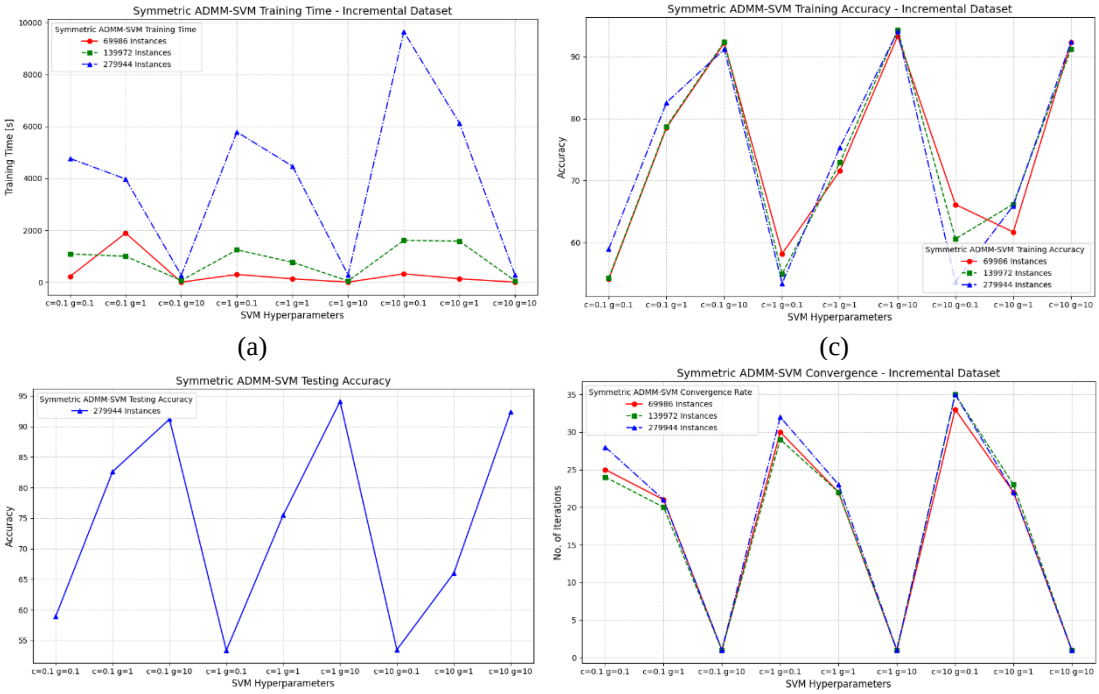


Figure 6. Symmetric ADMM-SVM for the incremental dataset (a) Training Time and (c) Training Accuracy for and (b) is Testing Accuracy and (d) Convergence after number of iterations.

The Figure 6 depicts the performance of Symmetric ADMM for the incremental dataset by varying the C and gamma value from [0.1, 1, and 10] with penalty parameter rho =1.

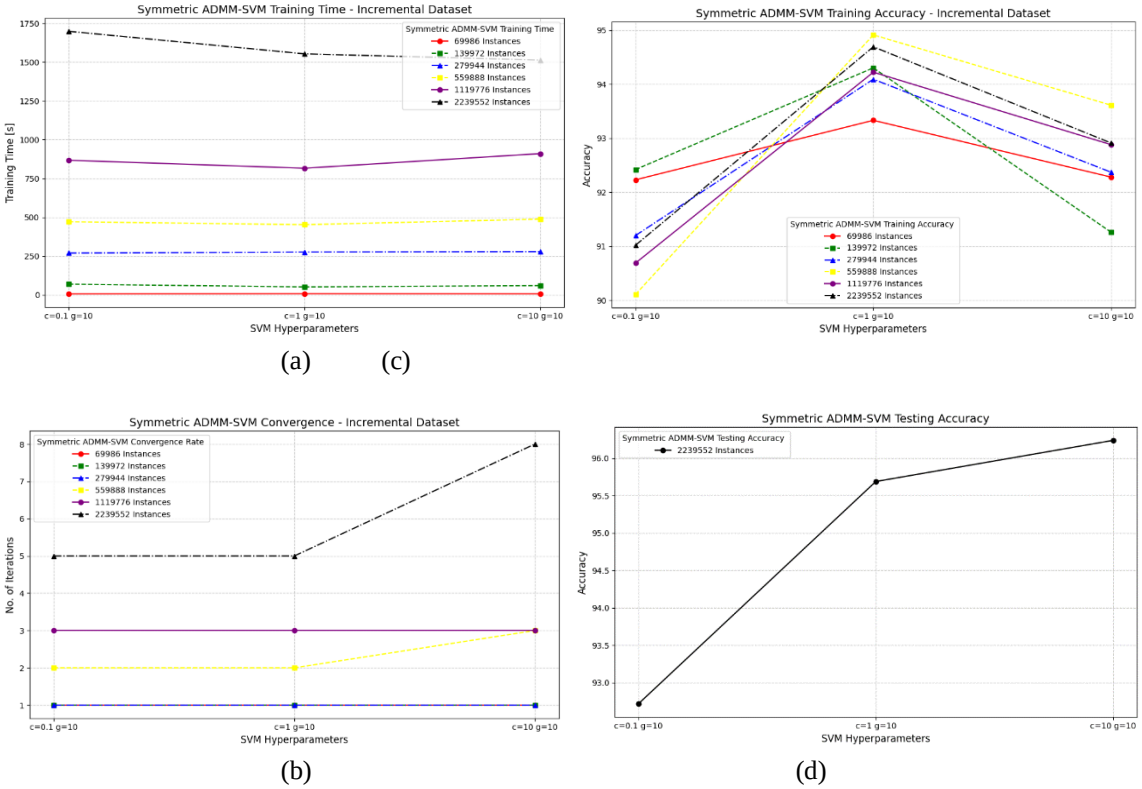


Figure 7. Symmetric ADMM-SVM with Optimal SVM Hyper-parameters (a) Training Time and (c) Training Accuracy and (b) Convergence after number of iterations (d) Testing accuracy for the 2239552 Instances.

Observing the optimal hyper-parameters we tested the algorithm for only optimal hyper-parameters [c and gamma] for the incremental dataset, the output is shown in the Figure 7. (a), (b), (c), (d), training and testing accuracy, training time and convergence after N iterations.

Advantages of Symmetric ADMM

- Decentralized Architecture:Each node contributes equally, and no central coordinator is required.
- Scalability:The algorithm scales well with an increasing number of nodes.
- Efficient Communication:Symmetric updates reduce communication overhead compared to standard ADMM.
- Convergence:Convergence is guaranteed under mild convexity assumptions, making it robust for distributed SVM.

## 5. Conclusion

The exploration of distributed Support Vector Machines (SVMs) using symmetric Alternating Direction Method of Multipliers (ADMM) for large-scale datasets, such as the IJCNN dataset, demonstrates significant advancements in scalability, efficiency, and accuracy. Symmetric ADMM enables distributed systems to achieve convergence efficiently by balancing computational loads across nodes and ensuring robust communication between them. This approach effectively handles the complexity of non-linear kernels, such as polynomial kernels, while preserving the accuracy of the classification. The adaptability of symmetric ADMM to various hyperparameters like ρ (rho), and its ability to optimize dual variables, showcases its superiority over traditional optimization methods in distributed settings.

Through experiments, it is evident that symmetric ADMM facilitates faster convergence rates without compromising the testing accuracy, making it a viable solution for real-world applications involving massive datasets. Furthermore, the framework provides a foundation for future research on incorporating advanced kernel functions and dynamic parameter tuning to further enhance performance in distributed environments. Thus, distributed SVM with symmetric ADMM emerges as a robust and scalable framework for tackling challenges

.

**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.
[2]. Huang, S.-A., & Yang, C.-H. (2019). A hardware-efficient ADMM-based SVM training algorithm for edge computing. *arXiv preprint arXiv:1907.09916*.
[3]. Das, A., & Bhattacharya, S. (2015). Distributed weighted parameter averaging for SVM training on big data. *arXiv preprint arXiv:1509.09030*.
[4]. Shi, Y., & Zhu, B. (2023). An ADMM solver for the MKL-$L_{0/1}$-SVM. *arXiv preprint arXiv:2303.04445*.
[5]. Wen, J. (2023). Efficient computing algorithm for high dimensional sparse support vector machine. *arXiv preprint arXiv:2312.15590*.
[6]. Tavara, S., &Schliep, A. (2021). Effects of network topology on the performance of consensus and distributed learning of SVMs using ADMM. *PeerJ Computer Science*, 7, e397.
[7]. Symmetric ADMM-based federated learning with a relaxed step. (2023). *Mathematics*, 12(17), 2661.
[8]. Distributed support vector machine in master–slave mode. (2018). *ResearchGate*. Available at: https://www.researchgate.net/publication/323207581_Distributed_support_vector_machine_in_master-slave_mode.
[9]. Distributed training of structured SVM. (2015). *ResearchGate*. Available at: https://www.researchgate.net/publication/277959141_Distributed_Training_of_Structured_SVM.
[10]. Mota, J. F. C., Xavier, J. M. F., Aguiar, P. M. Q., &Püschel, M. (2013). D-ADMM: A communication-efficient distributed algorithm for separable optimization. *IEEE Transactions on Signal Processing*, 61(10), 2718-272
[11]. 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.
[12]. Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3), 1-27.
[13]. Chen, Y., Yang, X., & Zhao, Z. (2019). Distributed SVM based on ADMM for classification. *Journal of Parallel and Distributed Computing*, 129, 119-126.
[14]. Forero, P. A., Cano, A., &Giannakis, G. B. (2010). Consensus-based distributed support vector machines. *The Journal of Machine Learning Research*, 11, 1663-1707.
[15]. Zheng, S., Kwok, J. T., & Zhang, B. (2007). Fast and scalable algorithms for kernel support vector machines. *Proceedings of the 21st International Conference on Neural Information Processing Systems*, 617-624.
[16]. Wang, C., & Wang, S. (2017). Distributed support vector machines for big data. *International Journal of Machine Learning and Cybernetics*, 8(1), 101-110.
[17]. Totad, S.G., Geeta, R.B. & Prasad Reddy, P.V.G.D. Batch incremental processing for FP-tree construction using FP-Growth algorithm. KnowlInfSyst 33, 475–490 (2012). https://doi.org/10.1007/s10115-012-0514-9.
[18]. Xu, J., & Yang, W. (2020). A parallel ADMM algorithm for distributed SVM training with guaranteed convergence. *IEEE Transactions on Knowledge and Data Engineering*, 32(9), 1737-1750.
[19]. Peng, Z., Zhang, Y., & Zhang, L. (2016). An efficient ADMM algorithm for large-scale sparse SVM. *Pattern Recognition Letters*, 83, 74-80.
[20]. Smith, V., Chiang, C. K., Sanjabi, M., & Talwalkar, A. (2017). Federated multi-task learning. *Advances in Neural Information Processing Systems*, 30, 4427-4437.
[21]. Gadepally, V., Kepner, J., &Samsi, S. (2015). Parallel algorithms for support vector machines. *2015 IEEE High Performance Extreme Computing Conference (HPEC)*, 1-6.
[22]. Liu, H., Wang, Y., & Li, W. (2018). Asynchronous distributed support vector machines via ADMM. *IEEE Access*, 6, 23940-23948.
[23]. Shashikumar G. Totad, Geeta R. B. , Chennupati R Prasanna , N Krishna Santhosh , PVGD Prasad Reddy, " Scaling Data Mining Algorithms to Large and Distributed Datasets", International Journal of Database Management Systems ( IJDMS ), Vol.2, No.4, November 2010.
[24]. João F. C. Mota, João M. F. Xavier, Pedro M. Q. Aguiar, Markus Püschel (2013). D-ADMM: A Communication-Efficient Distributed Algorithm For Separable Optimization, IEEE Transactions on Signal Processing, 61(10), 2718-2723.
[25]. Meyer, O., Bischl, B., &Weihs, C. (2014). Support vector machines on large data sets: Simple parallel approaches. In *Data Analysis, Machine Learning and Knowledge Discovery* (pp. 87–95). Springer.
[26]. Zhou, Y.-H., & Zhou, Z.-H. (2019). Parallel computing of support vector machines: A survey. *ACM Computing Surveys*, 51(6), 1–38.
[27]. Bengio, S., &Bengio, Y. (2002). A parallel mixture of SVMs for very large scale problems. *Neural Computation*, 14(5), 1105–1114.

[28]. Zanni, L., Serafini, T., &Zanghirati, G. (2006). Parallel software for training large scale support vector machines on multiprocessor systems. *Journal of Machine Learning Research*, 7, 1467–1492.
[29]. Zhang, Y., & Lin, X. (2019). Reduction of Training Data Using Parallel Hyperplane for Support Vector Machine. *Applied Artificial Intelligence*, 33(7), 601–617.
[30]. Vladimir Vapnik. 2013. The Nature of Statistical Learning Theory. Springer Science & Business Media. 314 pages.
[31]. Joshi, Y., Totad, S.G., Geeta, R.B., Prasad Reddy, P.V.G.D. (2018). "Mobile Agent-Based Frequent Pattern Mining for Distributed Databases." In: Bhalla, S., Bhateja, V., Chandavale, A., Hiwale, A., Satapathy, S. (eds) Intelligent Computing and Information and Communication. Advances in Intelligent Systems and Computing, vol 673. Springer, Singapore. https://doi.org/10.1007/978-981-10-7245-1_9.
[32]. Yang, Y., Guan, X., Jia, Q.-S., Yu, L., Xu, B., &Spanos, C. J. (2022). A survey of ADMM variants for distributed optimization: Problems, algorithms and features. *arXiv preprint arXiv:2208.03700*.
[33]. Zeng, R., Zhuang, S., & Li, X. (2020). Distributed support vector machine training for large-scale data using dynamic ADMM. *Journal of Computational Science*, 41, 101083.
[34].Huang, F., & Lin, C.-J. (2012). Linear and kernel classification: When to use which? *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*, 3196-3204.