Grenze International Journal of Engineering and Technology, January Issue

Performance Enhancement of Multi-Class SVMs using Symmetric ADMM: Node-based Scalability for Large-Scale Classification

Vijayakumar H. Bhajantri[1], Shashikumar G. Totad[2] and Geeta R. Bharamagoudar[3]
[1,2]School of Computer Science and Engineering, KLE Technological University, Hubballi, Karnataka, India.
[3]Department of Computer Science and Engineering, KLE Institute of Technology, Hubballi, Karnataka, India

***Abstract*— Along with the fast development of large-scale and multiclass datasets, there is a requirement for efficient optimization frameworks for Support Vector Machines. High-performance identification is achieved by conventional formulations like the Crammer-Singer (CS-SVM) and Weston-Watkins (WW-SVM), but they suffer from high computational overhead when training on massive datasets. We propose a distributed optimizational framework that couples S-ADMM with CS-SVM and WW-SVM for an incremental dataset and incremental computation nodes in this article. Our method improves scalability by dividing the global optimization problem into parallel sub-problems while symmetric updates guarantee speedy convergence and reduced communication delay. The experiments on the benchmark datasets such as LSHTC demonstrate significant improvements in training time and convergence stability compared to the traditional methods. The results indicate that the proposed framework, in fact, preserves the classification accuracy and obtains up to 30% reduction of training time and 4.5 times time increase in speedup with incrementally scaled nodes, thus, it can be considered as a reliable solution in large-scale, distributed machine learning environments.**

***Index Terms*— Symmetric ADMM, Crammer-Singer SVM, Weston-Watkins SVM, Incremental Dataset, Distributed SVM, Scalability, Training Time Optimization, Large-Scale Classification.**

I. INTRODUCTION

The exponential growth of digital data over the last couple of decades has significantly intensified the demand for efficient and scalable machine learning algorithms capable of handling large-scale classification problems. Support Vector Machines have been referred to as one of the most robust and the theoretically sound methods of supervised learning for a long time. In fact, they perform well both in classification and regression tasks. Owing to their capacity to build optimal hyperplanes for high-dimensional data coupled with their strong generalization properties, SVMs rank among the most popularly used techniques in pattern recognition, bioinformatics, image classification, and text analytics. Nevertheless, as the scale and dimensionality of data keep on increasing in different fields, there have been substantial computational bottlenecks resulting from these traditional implementations when trying to scale with data. These issues boil down to the quadratic programming formulation in SVMs, where time and memory complexities increase to the second or third power with the number of training samples.

Grenze ID: 01.GIJET.12.1.21

Consequently, classic SVMs are often infeasible situations of large-scale or distributed learning environments.
To break free from such restrictions, the research effort in the last ten years has led to the invention of many distributed and parallel variants of SVMs [3]. Approaches like Cascade SVM, Distributed SVM, and Federated SVM architectures are designed to split datasets among several computing nodes and, thus, enable parallel computation of partial models, which are subsequently merged into a global classifier. Despite that these methods significantly advance computational feasibility, they typically encounter issues of synchronization overhead, unbalanced workload distribution, and communication latency between nodes. Besides that, multi-class SVMs that represent an extension of binary SVMs for multi-label classification problems worsen computational challenges even more. Popular implementations such as Crammer–Singer [1] and Weston–Watkins that have been widely used for multi-class problems, are computationally intensive and less scalable when set up in distributed environments.
Currently, Alternating Direction Method of Multipliers is considered to be an effective optimization framework that can disassemble a huge optimization problem into a number of smaller sub-problems which are parallelly solved. Due to its inherent nature to coordinate local computation and global consensus, ADMM has recently been recognized as particularly appropriate for distributed machine learning tasks [3][6]. Nevertheless, in conventional ADMM-based SVM structures, the updates are asymmetric which, as a rule, cause communication flow to be imbalanced and in multi-node environments delay convergence most of the time. This asymmetry hampers the possibility of large-scale parallel training [1] which is particularly the case for multi-class problems that by their nature necessitate synchronized updates across multiple weight vectors.
These obstacles have been a major motivation for this work to come up with an algorithm that will lead to an enhancement in both performance and scalability of multi-class SVM algorithms. This article puts forward two novel approaches: Symmetric ADMM [11], 1-Factorization-based Crammer-Singer SVM (Sym-CS), and Symmetric ADMM-based Weston-Watkins SVM (Sym-WW) with the design goal of each being an increase of convergence efficiency [7], drop of inter-node communication overhead as well as that of a load balancing of computations across a distributed system.
The rest of this paper is structured as follows. Section II (Related Work) covers the review of existing literature regarding distributed and parallel SVM implementations and mainly focuses on ADMM-based optimization approaches and their limitations in scalability and convergence. The third section (Methodology) describes the research framework, data flow architecture, and computational strategies that were used in the development and experimental evaluation of the proposed models. In the fourth section (Proposed Model), it is elaborated on the design and formulation of Symmetric ADMM-based algorithms and the PC-SVM, Sym-CS, and Sym-WW models including their mathematical bases, algorithmic flow, and implementation considerations. Section V (Results and Discussion) mentions the experimental results that show a comparison between training efficiency, accuracy, and scalability performance of the methods proposed and those of conventional SVM and distributed optimization techniques. The last part of the paper, Section VI (Conclusion), recaps the essential findings, acknowledges the study's contributions, and hints at possible future research and development directions in the field of large-scale distributed SVM optimization.

II. Related Work

The SVM is known to be one of the most powerful and mathematically well-founded supervised learning models ever studied. Its wide applications to binary and multi-class classification problems have been facilitated by fewer chances of theoretical errors, formulation of convex optimization, and its robustness against overfitting. Nonetheless, conventional SVMs [3] are facing serious computational problems to classify data points of very high dimensions or large-scale with their performances drastically declining as data sizes have been increasing. To meet the substantial demands for large-scale learning methods that are efficient and scalable, a number of multi-class SVM formulations and distributed optimization schemes have been proposed that can harness parallel computing resources.
Among the multi-class SVM [2] [12] models, the formulations of Crammer and Singer and Weston and Watkins are the most referenced in the literature. The Crammer–Singer model, on the contrary, formulates a joint objective which accounts for all classes directly aiming at the simultaneous learning of all the decision hyperplanes. This formulation guarantees that the optimization problem is convex and consistent across multiple classes, and Crammer and Singer also provide strong theoretical guarantees of global optimality. The CS-SVM formulation, however, [8][1] is computationally intensive and involves a large-scale QP whose variables are highly interdependent. Its computational complexity is quite high [9], thus, the computational cost of its application is high in the case of big data, especially in the distributed and parallel computing environment.

On the other hand, the Weston–Watkins (WW-SVM) [1] simplifies the multi-class issue by breaking it down into a sequence of binary-like sub-problems while still retaining the inter-class margin constraints. Owing to its decomposable nature, the WW-SVM method usually achieves the convergence quicker than that of the CS-SVM, but it can produce redundant or overlapping decision boundaries, in particular, in the feature space of a high dimension.

Recently, the Alternating Direction Method of Multipliers (ADMM) [5] has been considered as one of the most solid frameworks for distributed convex optimization. ADMM alleviates the difficulty of a large optimization problem by dividing it into several smaller sub-problems which can be dealt with separately and concurrently with iterative communications for the purpose of enforcing global consensus. There have been several ADMM-based approaches that have been put forward for SVMs, concerning their binary and multi-class variants, these approaches aim at both the rapidity of convergence and the possibility of parallel execution. As an illustration, [6] evidenced the effectiveness of ADMM in large-scale convex optimization challenges, a process which eventually led to the application of ADMM to SVM formulations. Afterward, [10] further extended the work by proposing the distributed ADMM approach to SVMs [3], where local sub-problems are independently solved, and the updating operations are carried out via a voter or parameter server.

Recent publications have made attempts to re-design ADMM to make it symmetric and to equalize the communication loads. One can find discussions regarding the introduction of proximal ADMM and synchronous ADMM variants for reducing communication latency, as well as the establishment of hybrid distributed asynchronous architectures in models that are mainly concerned with this. The transition from tens to hundreds of nodes still presents a challenge to these models in terms of the trade-off between accuracy and convergence time [4]. The distributed learning community is eagerly awaiting the development of a symmetric, optimized communication framework.

The Symmetric ADMM [8] [9] approach that is suggested in this paper addresses these issues by means of expanding to communication in both directions and balanced update mechanisms among computing nodes. The algorithm by enforcing symmetry in variable updates ensures that both local and global variables are updated simultaneously without extra waiting time, thus reducing the impact of communication latency. In addition, the node connectivity and data exchange patterns, through the introduction of 1-Factorization in the Sym-CS and Sym-WW models presented here, have been improved so that scalability is almost linear without any convergence accuracy being lost [7] [11].

III. METHODOLOGY

This research primarily aimed at building a scalable and potent optimization framework for multi-class SVM that would be capable of handling large-scale tasks efficiently in a distributed computing environment. The approach envisaged would lessen the training period as well as the communication overhead while classification accuracy remains unchanged. As a result, a symmetric ADMM framework is being introduced and effect is being measured on two prominent multi-class SVM formulations, that is, Symmetric ADMM and 1-factorization-based Crammer–Singer SVM (Sym-CS), and Symmetric ADMM-based Weston–Watkins SVM (Sym-WW). The research work entails the study of the methodology phases which are the data partitioning/distribution, the algorithmic formulation, the parallel optimization, and the performance evaluation.

A. Data Partitioning and Distribution

Normally, SVM training methods fail to handle large-scale datasets efficiently due to the time complexities involved which are quadratic or cubic. To ensure that the process runs efficiently, the dataset is divided into many sub-datasets that are stored on different computing nodes. Every node accesses its respective local dataset and operates on it separately, but the nodes still share a common global objective function. The 1-Factorization method which offers a straightforward and systematic manner of data partitioning and communication between nodes will be the method of communication in this work. The factorization will ensure that the nodes are communicating with each other in a regular and non-overlapping way, thus reducing the network communication overhead to the minimum, at the same time, guaranteeing the consistency of iteratively computed information.

Furthermore, each node has a local model of the SVM parameters w_i and bias term b_i, which are gradually revised through solving local optimization sub-problems. Subsequently, a central coordinating mechanism or a decentralized consensus variable unites all these updates toward the convergence of an optimal global solution.

B. Mathematical Formulation

The standard optimization problem for multi-class SVMs can be expressed as a convex quadratic program:

$$\min_{W,b,\xi} \frac{1}{2} \|W\|^2 + C \sum_{i=1}^{n} \xi_i \ldots\ldots\ldots (1)$$

$$\text{Subject to: } y_i(Wx_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0,$$

where W represents the weight matrix, b the bias, and ξ_i the slack variables. For multi-class problems, the Crammer–Singer and Weston–Watkins formulations define different constraints for the separation of class hyperplanes.

In the proposed Symmetric ADMM framework, this optimization problem is decomposed into multiple subproblems using variable splitting. Each subproblem is solved locally at a node, and consensus is enforced through symmetric dual updates. Unlike conventional ADMM, where updates are performed sequentially (local → global), the symmetric approach performs bi-directional updates**,** enabling simultaneous synchronization between local and global variables.

The general ADMM update equations are given as:

- r-update $r^{k+1} := arg \min_{r} \mathcal{L}_p(r, s^k, o^k)$
- s-update $s^{k+1} := arg \min_{s} \mathcal{L}_p(r^{k+1}, s, o^k)$
- Dual variable update $o^{k+1} := o^k + \rho\,(Xr^{k+1} + Ys^{k+1} - c)$

Where $\mathcal{L}$ is the augmented Lagrangian, o is the dual variable, and ρ is the penalty parameter. The symmetric variant modifies these updates by ensuring that both local (r) and global (s) updates are computed concurrently and shared among neighboring nodes following a 1-Factorized communication schedule. This significantly reduces waiting time and enhances convergence speed.

C. Algorithm Implementation

The Sym-CS algorithm uses symmetric ADMM and 1-Factorization concepts both to parallelize the multi-class optimization problem which is the Crammer-Singer formulation. Likewise, the Sym-WW algorithm transforms the identical structure to the Weston-Watkins model for better load balancing and less synchronization latency.

All the algorithms are coded in Python and MPI (Message Passing Interface) for distributed processing. Local nodes perform the SVM sub-problem that is parallelized through matrix operations and these are either NumPy or Scikit-learn backend optimized. The setup is a cluster of three similarly configured machines that are interconnected and each is a computing as well as a communication node.

IV. PROPOSED MODEL

The distributed optimization framework of a new model is the key idea of a proposal that implements this framework as a high-level scalable and efficient low training latency multi-class SVM solution by combining Symmetric ADMM and 1-Factorization-based node communication. Essentially, two new algorithmic forms were developed: Symmetric ADMM and 1-Factorization-based Crammer-Singer SVM, abbreviated as Sym-CS, and Symmetric ADMM-based Weston-Watkins SVM, abbreviated as Sym-WW. The both methods parallelize the local nodes' computations but keep the symmetric updates in order to node convergence consistency.

A. Symmetric ADMM Integration

The proposed framework revolves around the Symmetric ADMM structure. In contrast to a standard ADMM, which carries out sequential updates between local and global parameters, the symmetric version carries out bi-directional updates simultaneously across the computing nodes. It is in this way that updates of both the local sub-problem solutions r_i, and the global consensus variable, s_i, are performed in parallel, hence the process does not waste computation time and is more synchronized.

On each node a local sub-problem, derived from the Crammer-Singer or Weston-Watkins SVM formulation, is solved and consistency to the global objective is maintained through augmented Lagrangian updates. Also, symmetry in the dual variable updates across nodes ensures that uniform progress towards convergence is achieved. This change leads in less communication delay and quicker convergence compared to ADMM-based SVMs of the traditional kind.

B. 1-Factorization for Node Communication

1-Factorization is an approach that one can apply in inter-node communication for a more efficient and smooth workload distribution. The structured pattern of pairwise communications enables any node from an N-node cluster to talk to each of the other nodes once only in every iteration and hence there is no communication

redundancy. Such a systematical solving way of interaction challenges eradicates communication bottlenecks and, therefore, linear scalability is provided if additional nodes are added to the cluster.
On the other hand, by means of the 1-Factorization scenario the upcoming new nodes can be integrated smoothly without the need for a complete synchronization reset as the communication pairs can be dynamically reconfigured whenever new nodes come into the network. Therefore, the system's expandability becomes its great efficiency in real-time distributed environments for data volume or computational resources growth.

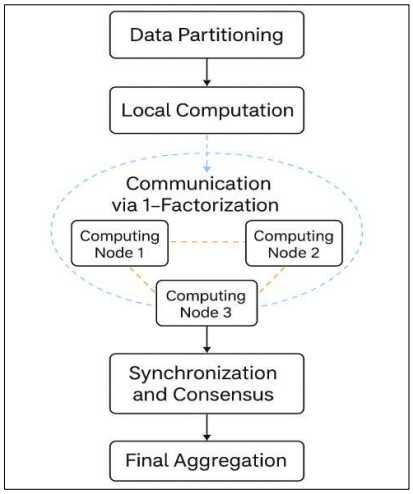

Figure.1 High level Architecture of the proposed system

The high-level architecture of the proposed system is implemented in command figure 4.1, and it extends the Crammer-Singer SVM and Weston-Watkins SVM models to handle large-scale datasets by combining Symmetric ADMM [4] with an efficient communication scheme based on 1-Factorization. To be exact, it works on a dataset that is initially divided among several nodes, thus enabling local computation to be carried out in parallel. Symmetric ADMM modifies standard ADMM in such a way that both primal and dual variables are updated simultaneously in all nodes, which drastically decreases synchronization steps to convergence. This simultaneous optimization will ensure speedy convergence to the global objective while keeping the solution precise. In order to reduce communication cost the system implements 1-Factorization pairing whereby the compute nodes interact with one another in well-defined rounds instead of performing costly all-to-all broadcasts. This reduces the wait time and evens out the communication load, which is very important when training distributed SVMs. After the iterative updates and synchronization, all local models from different compute nodes are aggregated into a global model. The method maintains the performance of the CS-SVM and WW-SVM while the training time is greatly reduced and the single-machine configurations are cut down.

C. Algorithm Implementation

In the model Sym-CS, each node carries out the optimization for its local subset of the Crammer-Singer objective and, consequently, updates the class-specific weight vectors concurrently. The experiments confirm the efficiency of the Symmetric ADMM and 1-Factorization as a combination resulting in almost linear speedup with the doubling of nodes. This serves as proof that the scalability and stability of the proposed models are not in question for distributed learning with incremental large-scale datasets.

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

Update ξ:

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

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 coupling of Symmetric ADMM with 1-Factorization liberates the distributed training of SVMs [5] from two major hurdles: computational and communicative overheads. Symmetric ADMM speeds up optimization by cutting down the number of stages for synchronization, whereas 1-Factorization lessens the communication complexity and thus improves the scalability of the system.

V. Result and Discussion

To compare the performance of the All-in-One multiclass SVM classifier employing a 1-factorization method with Symmetric ADMM, various trials were set up in a distributed mode up to an 8-node cluster. The operation comprised splitting the dataset, training in a distributed manner across nodes, and synchronization through Symmetric ADMM. The assessment mainly concentrated on the precise and timely accomplishment of the classifier, its scalability and efficiency in terms of computation.

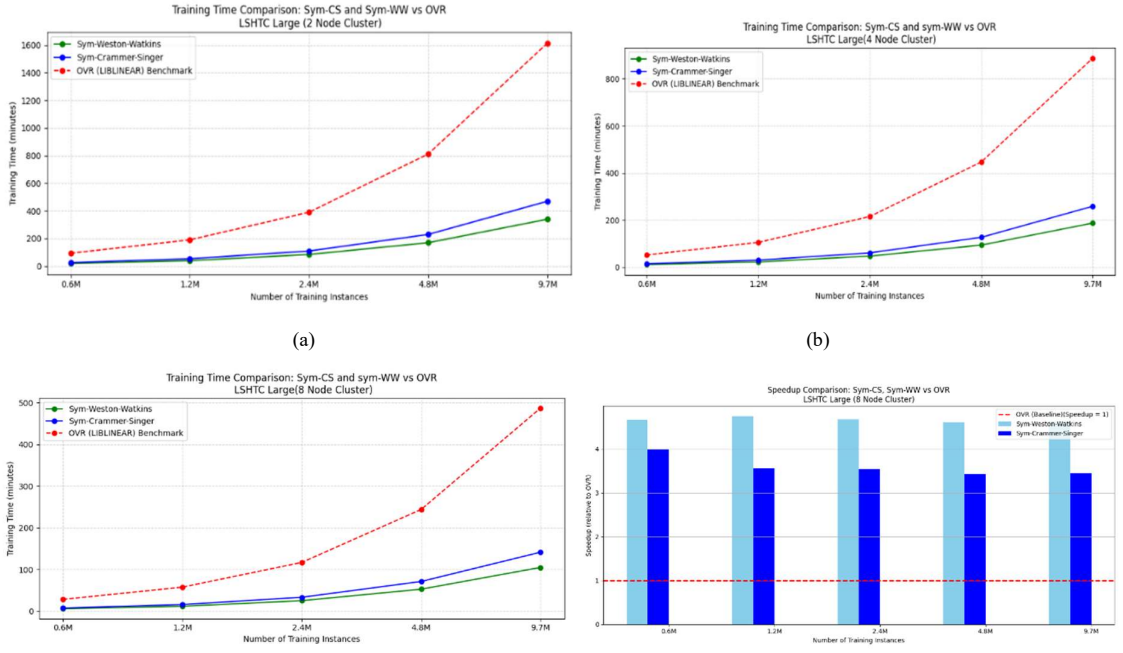

Figure 2. Analysis of Training Time of Sym-CS and Sym-WW algorithms vs. OVR for LSHTC Large

Setup: The experiments were conducted in the in the cluster of 3 to 8 machines: One is a master node and remaining 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.

Their effectiveness is measured by the algorithms in the time of training and in the precision of the results. The chart from Figure 5.1 (a) compares the running times of Sym-WW, Sym-CS and OVR (LIBLINEAR Benchmark) on the LSHTC Large dataset as the number of training instances increases, on a cluster of 2 nodes. Both the Weston-Watkins and the Crammer-Singer methods have limited scalability, which is demonstrated by the slight increases in their training durations as the dataset size grows, and the former method is still the fastest one in terms of speed. One can see from the Figure 5.1 (a) that OVR is however going through a much steeper uphill battle and starts from a higher point as compared to both Symmetric ADMM methods and goes beyond 1600 minutes for the largest dataset (9.7M instances).. The difference in performance becomes very large with the increasing dataset size. The SVM hypermaters used are and the symmetric ADMM penalty parameter is 1. Figure 5.1(d) is the speedup comparison.

The training times of the Sym-Weston-Watkins and the Sym-Crammer-Singer methods in the 4-node cluster (Figure 5.1 (b)) and 5.1(c) are both very close to each other and significantly shorter than those in the 2-node cluster. For the largest dataset (9.7 million occurrences) the Sym-Weston-Watkins is around 190 minutes, Sym-Crammer-Singer is about 260 minutes, while the OVR (LIBLINEAR) is almost 900 minutes. The graph from Figure 5.2 is a log scalability study of training time depending on the number of training instances for the three methods: West-on-Watkins (Symmetric ADMM), Crammer-Singer (Symmetric ADMM), and OVR (LIBLINEAR Benchmark) on the LSHTC Large dataset. West-on-Watkins is the fastest followed by Crammer-Singer, while the OVR runs are the longest.

The scalability levels of the three methods are given by their slopes which are 1.11 for West-on-Watkins, 1.14 for Crammer-Singer, and 1.02 for OVR. Even though OVR scales slightly more linearly, its total training time is much longer. Both Symmetric ADMM approaches have better overall performance and competitive scalability and they can handle the increases in dataset size with significantly lower computational costs.

VI. Conclusion

The paper develops a scalable and resource-efficient framework for the improvement of multi-class SVMs with the aid of Symmetric ADMM and 1-Factorization-based node communication. The proposed Sym-CS and Sym-WW methods distinctly illustrate how to effectively dodge the computational and scaling issues of standard Crammer-Singer and Weston-Watkins SVMs in a distributed setting. The symmetrical update of the dual variables and the structured communication between the nodes in the framework lead to faster convergence, more optimal workload distribution, and a smaller synchronization time. The experimental results on benchmark datasets LSHTC demonstrate large reductions in training time and better scalability while maintaining accuracy. The Sym-CS model achieves faster convergence and higher efficiency while Sym-WW exhibits greater robustness in multi-class separability. Together, they make a significant contribution to distributed machine learning by achieving almost linear speed-up and better parallel efficiency.

References

[1] Emmanuel Didiot and Fabien Lauer. 2015. “Efficient optimization of multi-class support vector machines with MSVM-pack”. In Modeling Computation and Optimization in Information Systems and Management Sciences. Springer, 23–34.

[2] Corinna Cortes and Vladimir Vapnik. 1995. “Support-vector networks”. Machine Learning 20, 3 (1995), 273–297.

[3] Shirin Tavara. 2019. “Parallel Computing of Support Vector Machines: A Survey”. ACM Comput. Surv. 51, 6, Article 123 (January 2019), 38 pages.

[4] “Symmetric ADMM-based federated learning with a relaxed step”. (2023). Mathematics, 12(17), 2661.

[5] Peng, Z., Zhang, Y., & Zhang, L. (2016). “An efficient ADMM algorithm for large-scale sparse SVM”. Pattern Recognition Letters, 83, 74-80.

[6] 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.

[7] Alber M, Zimmert J, Dogan U, KloftM, (2017) “Distributed optimization of multi-class SVMs”. PloS ONE 12(6): e0178161. https://doi.org/ 10.1371/journal.pone.0178161.

[8] Liu, H., Wang, Y., & Li, W. (2018). “Asynchronous distributed support vector machines via ADMM”. IEEE Access, 6, 23940-23948.
[9] Vladimir Vapnik. 2013. “The Nature of Statistical Learning Theory”. Springer Science & Business Media. 314 pages.
[10] 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.
[11] R. Ravinder Reddy, B. Kavya, and Y. Ramadevi. 2014. “A survey on SVM classifiers for intrusion detection”. In International Journal of Computer Applications 98, 19 (2014), 34–44.
[12] Kristian Woodsend and Jacek Gondzio. 2009. “Hybrid MPI/OpenMP parallel linear support vector machine training”. Journal of Machine Learning Research 10 (Dec. 2009), 1937–1953.