An Empirical Evaluation of The Efficacy of an ADMM Classification Technique in Modern Spark

Amonn Brewer
abrewe47@students.kennesaw.edu
Kennesaw State University
Marietta, Georgia, USA

Phillip Gregory
pgrego10@students.kennesaw.edu
Kennesaw State University
Marietta, Georgia, USA

Jennifer Felton
jfelto14@students.kennesaw.edu
Kennesaw State University
Marietta, Georgia, USA

Colin Pittman
cpittm24@students.kennesaw.edu
Kennesaw State University
Marietta, Georgia, USA

ABSTRACT

The scope of this project is to revisit the work of Xiaodong Su's 2020 paper, "Efficient Logistic Regression with L2 Regularization using ADMM on Spark," [3] which presents an ADMM implementation on Apache Spark 2.4 shown to provide a significant improvement over the standard MLLib implementation without requiring meta-parameter tuning. The purpose of this study is to assess the technical efficacy under further scrutiny by comparing it with the modern MLLib implementation in Spark 4.0, testing it under different datasets, and exploring alternatives to the original presupposed algorithm.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**; **Classification and regression trees**; **Supervised learning by classification**; • **Theory of computation** → *Distributed algorithms.*

KEYWORDS

ADMM, Alternating Direction Method of Multipliers, Logistic Regression, Apache Spark, Distributed Computing, L2 Regularization, Machine Learning Optimization, MLLib, Convergence Analysis

1 RESEARCH STATEMENT AND CONJECTURE

This research revisits and builds upon Su's 2020 implementation of ADMM for logistic regression with L2 regulation in Apache Spark, which introduced a dynamic penalty parameter to enhance coverage without requiring manual tuning. Although the ADMM-based approach demonstrated scalability and efficiency in Spark 2.4, it has not been thoroughly evaluated against modern Spark MLLib versions or across diverse datasets.

CS7265 Project, June 03–05, 2025, Marietta, GA
© 2025 ACM.
ACM ISBN 555-5-5555-5555-5/25/06
https://doi.org/This.Is.Only.A.Class.Project

This study focuses on extending and evaluating Su's improved ADMM algorithm in a modern Spark environment by:

- Benchmarking it against current MLLib logistic regression in Spark 4.0,
- Validating its adaptability and coverage across varied datasets,
- Exploring algorithmic modifications such as integrating stability terms to further enhance its convergence without sacrificing scalability.

By exploring enhancements such as incorporation of stability terms, we conjecture that it is possible to further stabilize the convergence trajectory of the algorithm without compromising scalability. Thus, our modified ADMM variant may demonstrate improved performance in both training efficiency and generalization under non ideal data conditions common in big data contexts.

2 PRELIMINARY LITERATURE SURVEY

To perform a literature review, the scope of the review consisted of the previous 10 years to maintain relevant results as they relate to ADMM. A 10-year period was chosen because there has not been a robust series of work to improve the algorithm with respect to big data in the past five years on the topic of ADMM; however, there has been some research on the application of ADMM in various industry contexts. Although there are many papers related to the application of ADMM, the research conducted here is restricted to improving the existing ADMM algorithm and not providing an additional application; therefore, this research attempted to review the most recent advancements in the ADMM algorithm.

In "ADMM-based Scalable Machine Learning on Spark" [2], Sauptik Dhar et al. presents a direct implementation of ADMM on Spark that allows the ADMM algorithm to be parallelizable on Spark. This appears to be the first implementation of ADMM in Spark based on the research carried out. ADMM was implemented in Spark by decomposing the objective function,

$$\min_{w\in\mathbb{R}^d} \left\{ \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{d} x_{ij} w_j \right)^2 + \lambda \sum_{j=1}^{d} |w_j| \right\},$$

into two specialized functions. The first function, $f(w)$, was created for smooth data-fitting terms:

$$f(w) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - x_i^T w)^2.$$

The second function, $g(z)$, was created to handle non-smooth regularization terms:

$$g(z) = \lambda \sum_{j=1}^{d} |z_j|.$$

The first function was decomposed to allow parallelization where $f(w)$ can be solved independently on each data partition while $g(z)$ can manage the global consensus. The limitation with this implementation is that it requires the user to manually adjust the tuning of the penalty parameter (p). Adjusting the penalty parameter has an effect on the number of iterations of the algorithm and the quality of the solution convergence; therefore, having a static penalty parameter is not ideal.

As described in "Efficient Logistic Regression with L2 Regularization using ADMM on Spark" [3], the proposed algorithm was to restructure the original ADMM algorithm on Spark for distributed computing. This implementation appears to be a direct improvement on the approach from Sauptik Dhar et al.[2] This was accomplished by creating three main classes, ADMMState, ADMMUpdater, and ADAMMOptimizer. The ADMMState class stores all variables for each Resilient Distributed Dataset (RDD) partition, the ADAMMUpdater consists of the update logic for each of the variables and implements the stopping criteria, which is based on the primal and dual residuals. Additionally, a penalty parameter, p, was created as a result of this implementation. The penalty parameter controls the consensus between all of the models across partitions. A small penalty parameter will have slower convergence, which will have a weak consensus across all models. This will increase the number of iterations required for convergence, while a large penalty parameter will have the opposite effect. The penalty parameter is dynamically adjusted according to the primal and dual residuals. If the primal residual is greater than the dual residual, it indicates that w and z are far apart. Therefore, p is increased to strengthen consensus among the models. Conversely, if the dual residual is greater than the primal residual, p is decreased to allow greater flexibility between all models; otherwise, p remains unchanged.

Finally, the ADAMMOptimizer class orchestrates the distributed execution of the algorithm. It manages the ADMMState and ADMMUpdater in parallel, a process in which each partition first updates its local model variables. These local variables are then aggregated to update the global variables and form the global consensus. While the results show that this implementation successfully implements ADMM on Spark, the main drawback is that the implementation does not address any other aspects of ADMM. The authors state this as a limitation, but one that provides a foundation for further enhancements.

In "Optimizing Logistic Regression with Enhanced Convergence: A Modified ADMM Approach" [1], Al-Zamili et al. introduced a modified ADMM algorithm that incorporates a novel "stability term" alongside standard regularization. The purpose of the regularization term is to prevent overfitting by penalizing large model weights. The stability term serves to prevent large changes in weights between iterations, resulting in improved stability and convergence. This term is defined as $\frac{1}{2\eta}\|x - x^k\|_2^2$ which computes the sum of the squared difference between the weight vectors of the current and previous k iterations of the algorithm. The experimental results showed that the proposed algorithm had an F1 score increase by 20% via a synthetic dataset. Although the proposed algorithm does increase F1, it does not improve the scalability of the algorithm as it relates to big data.

3 METHODOLOGY

This project will utilize a benchmarking framework built on Apache Spark to assess and contrast the performance of several optimization algorithms in a distributed computing setting. Our approach involves three main phases:

- Replicating the original findings to establish a baseline.
- Benchmarking the algorithm against its modern counterpart in Spark 4.0.
- Implementing and evaluating our own modified ADMM variant that incorporates a stability term.

Optimization Algorithms

We will put the following optimization techniques into practice and contrast them:

- **ADMM:** The implementation of ADMM for logistic regression as described by Su (2020).
- **Modern MLLib:** The standard logistic regression solver, which uses L-BFGS, provided in the modern Spark 4.0 MLlib.
- **Proposed ADMM-Stability Variant:** Our modification of the base ADMM algorithm from the original paper with a stability term.

Classification Problems

The performance of each algorithm will be evaluated using a consistent benchmark task applied across two distinct datasets. This approach will test the algorithms' effectiveness on both the original problem and a new problem with different data characteristics. The two datasets are:

- **RCV1 (Reuters Corpus Volume 1):** This is the dataset used in the original Su (2020) study, for direct comparison and replication of the baseline results.
- **Higgs Dataset:** This features different data distributions and density which will be used to gain insights on how our algorithms perform under different conditions.

Execution Environment

All trials will be conducted with a containerized environment using Docker. This will enable team members to rapidly get into an equal environment and be able to export test results during the exploratory and analysis phases of the research. The two environments will be configured for Spark 2.4 and Spark 4.0.

Evaluation Metrics

Every algorithm will be evaluated according to a few key metrics, including:

- **Convergence Speed:** The number of iterations required to reach a stable loss value.
- **Model Accuracy:** Measured by classification accuracy, F1-score, and log-loss on a test set.

- **Robustness:** A measure of performance consistency between the accuracy results, based on the standard deviations.

Observational Protocol

For consistency, the same model parameters will be used to initialize each procedure. During each trial, we will run the algorithm variants in both the Spark 2.4 and Spark 4.0 environments under the controlled docker environments. Throughout each run, we will record performance data at regular intervals, including convergence progress, iteration count, and elapsed runtime. The data will be logged to output files for analysis. At the end of each trial, we will save final model accuracy, runtime, and convergence information so that we can later compare the behavior of the three algorithm variants across the two environments and datasets

4 PROJECT TIMELINE

The following is a tentative project plan with defined execution dates for each phase.

Table 1: Deliverables Schedule

Phase	Description	Due Date
1	Literature research & algorithm analysis	20-Jun-2025
2	Alignment of proposed algorithm	25-Jun-2025
3	Implementation of proposed algorithm	04-Jul-2025
4	Draft project progress report	05-Jul-2025
5	Final approved progress report	07-Jul-2025
6	Algorithm Comparison	15-Jul-2025
7	Initial draft of final report	18-Jul-2025
8	Final approved report	21-Jul-2025

5 PROJECT MANAGEMENT

The project consists of four (4) members, Amonn Brewer, Jennifer Felton, Colin Pittman, and Phillip Gregory. In terms of communication, a group channel has been created for direct communication between all team members. Specifically, we will use Microsoft Teams for drafting documents, GitHub for version control of all code, and Overleaf for collaborative drafting of the final report. The team will meet at least once per week in Microsoft Teams to discuss progress on the team members' respective responsibilities, alignments on the direction of the project, and to determine a path forward when there are issues. All work performed will be kept in a shared folder, so all members can see real-time progress of the respective aspects of the project. This will aid in collaboration so all members can deliver real-time feedback as the project is being completed. Phillip Gregory will be the group leader, and he will be responsible for keeping the momentum of the project, turning the assignments, and communicating any misalignments to Dr. Lo. Colin Pittman will take on the role of technical lead, focusing on both the details that guide the project's experimental design and maintaining a frictionless and collaborative software environment for the group to work within. Amonn Brewer and Jennifer Felton will assist in research, writing, and carrying out the evaluations and analysis.

REFERENCES

[1] Ayad Al-Zamili and Ali Aljilawi. 2025. Optimizing Logistic Regression with Enhanced Convergence: A Modified ADMM Approach. *International Journal of Mathematics and Computer Science* 20 (2025), 9–15.

[2] Sutanay Dhar, Chen Yi, Naren Ramakrishnan, and M. Arif Shah. 2015. ADMM based Scalable Machine Learning on Spark. In *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 1174–1182.

[3] Xiaodong Su. 2020. Efficient Logistic Regression with L2 Regularization using ADMM on Spark. In *Proceedings of the 2020 International Conference on Machine Learning and Big Data Analytics (ICMLBDA)*. 23–28. https://doi.org/10.1145/3409073.3409077