Scalable Machine Learning on Spark for multiclass problems

Goutham Kamath[†*] **Sauptik Dhar**[‡] **Naveen Ramakrishnan**[‡]
David Hallac[§] **Jure Leskovec**[§] **Mohak Shah**[‡]
[†]Georgia State University, GA [‡] Bosch Research Center, CA [§]Stanford University, CA
gkamath1@student.gsu.edu {jure, hallac}@cs.stanford.edu
{sauptik.dhar, naveen.ramakrishnan, mohak.shah}@us.bosch.com

Introduction The advent of big-data has seen an emergence of research on scalable machine learning (ML) algorithms and big data platforms [1]. Several software frameworks have been introduced to handle the data deluge like, MapReduce, Hadoop, Spark etc. Among them, Spark has been widely used by the ML community. Spark supports distributed in-memory computations and provides practitioners with a powerful, fast, scalable and easy way to build ML algorithms. Although there have been several Spark based ML libraries [3, 4], there are very few packages that cover a wide range of problems with fast and accurate results. Authors in [2] have shown that ADMM based approach can be used as a general framework to accurately solve several standard and variants of Linear, Logistic Regression and SVM. However, [2] is limited to binary classification problems. In this paper, we extend the ADMM based framework to solve *multiclass* $\ell 1/\ell 2/\ell 1$-$\ell 2$ regularized Logistic Regression, SVM etc. Currently, none of the existing big-data libraries provide a solution to most of the above multiclass algorithms. Hence, we compare our implementation with other available Spark ML packages like, MLLIB [3] and PhotonML [4], only for binary $\ell 2$ logistic regression; and use our multiclass solver to solve the binary problem. We additionally report the time complexity of our multiclass implementation for future references and comparisons. For the accuracy of the solutions we use scikit-learn implementation as ground truth.

Approach Most multiclass ML algorithms follow the following composite optimization formulation in eq. (1): Given training samples $\mathcal{T} := (\mathbf{x}_i, y_i)_{i=1}^N$, where $\mathbf{x} \in \mathcal{R}^d$, and $y \in \{1, \cdots, \mathcal{C}\}$. Solve,

$$\min_{\mathbf{w}} \frac{1}{N}\sum_{i=1}^{N} L(f_{\mathbf{w}}(\mathbf{x}_i), y_i) + \lambda R(\mathbf{w}) \equiv \quad \min_{\mathbf{w},\mathbf{z}} \frac{1}{N}\sum_{i=1}^{N} L(f_{\mathbf{w}}(\mathbf{x}_i), y_i) + \lambda R(\mathbf{z}) \quad s.t.\ \mathbf{w} = \mathbf{z} \tag{1}$$

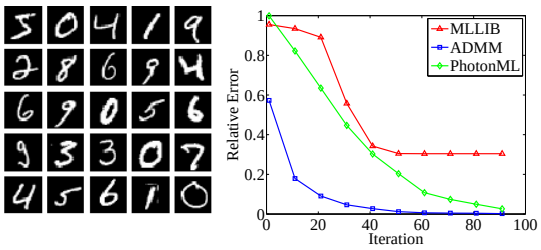
where, $L(\cdot)$ and $R(\cdot)$ are convex loss function and regularizer respectively. Eq.(1) allows us to formulate wide variety of multiclass ML problems like $\ell 1$,$\ell 2$,$\ell 1$-$\ell 2$ Logistic regression, SVM etc. In this paper we use an ADMM based approach to solve eq (1). The main idea in an ADMM based approach is to introduce additional *consensus* variable(s) $\mathbf{z}$ in eq. (1) and decompose the optimization problem into two separate sub-problems in $\mathbf{w}$ and $\mathbf{z}$ setting $\mathbf{w} = \mathbf{z}$ (shown in eq. (1)). A simple example for $\ell 1$-$\ell 2$ (elastic) multinomial logistic regression using the above decomposition results in the following ADMM steps:

$$(\mathbf{w}_1, \ldots, \mathbf{w}_c)^{k+1} = \underset{\mathbf{w}}{\operatorname{argmin}}\ \frac{1}{N}\sum_i \log\left(\frac{e^{\mathbf{x}_i^\top \mathbf{w_r}}}{\sum_{c=1}^{C} e^{\mathbf{x}_i^\top \mathbf{w_c}}}\right) + \frac{\rho}{2}\sum_c \|\mathbf{w_c} - \mathbf{z_c}^k + \mathbf{u_c}^k\|_2^2 \tag{2}$$

$$(\mathbf{z}_1, \ldots, \mathbf{z}_c)^{k+1} = \underset{\mathbf{z}}{\operatorname{argmin}}\ \lambda\sum_{c,j}\delta_j\left\{\alpha\,|z_{cj}| + (1-\alpha)\cdot\frac{z_{cj}^2}{2}\right\} + \frac{\rho}{2}\sum_c \|\mathbf{w_c}^{k+1} - \mathbf{z_c} + \mathbf{u_c}^k\|_2^2$$

$$\mathbf{u_c}^{k+1} = \mathbf{w_c}^{k+1} - \mathbf{z_c}^{k+1} + \mathbf{u_c}^k \quad \forall c$$

[*]work done during Goutham Kamath's internship at Bosch.



Methods	ADMM	PhotonML
Binary	321.54 (12.52)	118.81 (1.39)
MultiClass	1229.56 (50.44)	N/A

Figure 1: (left) MNIST dataset, (center) Convergence for different implementations of L2-Logistic Regression, (right) Computation time comparison between ADMM and PhotonML (std dev in bracket) to reach relative error of 0.1

Here, $L(f_{\mathbf{w}}(\mathbf{x}_i), y_i) = \log\left(\frac{e^{\mathbf{x}_i^\top \mathbf{w_r}}}{\sum_{c=1}^{C} e^{\mathbf{x}_i^\top \mathbf{w_c}}}\right)$ and $R(\mathbf{z}) = \sum_{c,j} \delta_j \left\{\alpha\,|z_{cj}| + (1-\alpha)\cdot\frac{z_{cj}^2}{2}\right\}$. This ADMM based approach, provides a simple strategy to, (i) Plug in several Loss functions $L(\cdot)$, like multinomial logistic, multi-class hinge etc ; as well as regularizers $R(\cdot)$ like, L1,L2,L1-L2 and (ii) Solve the composite problem by decomposing the overall larger problem into smaller sub-problems (see eq. (2)). Following [2], it can be shown that the smaller sub-problems can in turn be iteratively solved using a system of distributed quadratic programs (QP).

Experiment Settings Next, we compare the performance of ADMM vs MLLib and PhotonML in terms of their model accuracy, convergence time and iteration. In this experiment we only compare L2 logistic regression algorithm using 1G MNIST dataset (Fig. 1 left) for binary (5 vs 8) and multiclass (3 vs 5 vs 8). We assume scikit-learn's solution ($\mathbf{w}^{opt}$) as the ground truth and compare the relative accuracy of the estimated models ($\mathbf{w}^*$) i.e $\frac{\|\mathbf{w}^{opt}-\mathbf{w}^*\|}{\mathbf{w}^{opt}}$. The hadoop cluster configuration follows: No. of Nodes = 6 (Apache 1.1.1), No. of cores (per node) = 12 core (Intel Xeon @ 3.20GHz), RAM size (per node) = 32 GB, Hard Disk size (per node) = 500 GB. Further, we use the following settings for ADMM: $\rho = 0.5$, $\delta = 1$. We use default parameters for MLLIB and PhotonML. λ is fixed to 1.

Results The results (in Fig 1 center) shows that the MLLib implementation does not converge to the scikit-learn solution. We are investigating the cause of this behavior and hence avoid any timing comparison with other packages. In comparison, both the Photon ML and ADMM based implementation converge to the optimal solution. PhotonML uses LBFGS and TRON for their fast and accurate solution. However, one caveat of using PhotonML is that it requires an AVRO input format which incurs a huge timing overhead of $\sim$ 110 sec to convert 1G data to AVRO format. However the underlying model estimation is extremely fast $\sim$4 sec. On the other hand, the ADMM based approach although takes fewer iterations to reach the same accuracy, but it is slower because of the QP solver in $\mathbf{w}$ - update. Further, the current implementation of ADMM approach in pyspark incurs additional overhead due to native calls to the underlying scala libraries. However, the current work extends the available algorithms in [2] for multiclass problems and is probably the biggest repository of distributed ML algorithms and can serve as a baseline for future research on big-data ML algorithms. Moreover, the current implementation in pyspark (Python platform for Spark) makes this distributed ML library easily accessible to the huge python based ML community.

Ongoing Work Currently we are extending our experiment results for other multiclass loss functions like hinge loss etc. Finally, we are extending this generic framework to solve several advanced learning problems that involve non-convex formulations.

References

[1] Cevher, V., et. al. "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics." Signal Processing Magazine, IEEE, 2014.

[2] Dhar, S., et al. "ADMM based scalable machine learning on Spark." IEEE Big Data, 2015.

[3] Meng, X., et al. "Mllib: Machine learning in apache spark." JMLR (2016).

[4] Photon ML https://engineering.linkedin.com/blog/2016/06/open-sourcing-photon-ml