Supplement Materials for "Large-scale Logistic Regression and Linear Support Vector Machines Using Spark"

Chieh-Yen Lin, Cheng-Hao Tsai, Ching-Pei Lee, and Chih-Jen Lin
Department of Computer Science
National Taiwan University, Taiwan
{r01944006, r01922025, r00922098, cjlin}@csie.ntu.edu.tw

1. Introduction

This document presents some materials not included in the paper. In Section 2, the details of applying the distributed TRON algorithm to solve L2-loss SVM are described. We then conduct empirical investigations on SVM in Section 3.

2. Distributed Algorithm for L2-Loss SVM

The distributed TRON algorithm for L2-Loss SVM is the same as that for LR, except the objective function and its derivatives are different. Note that (1) with L2 loss can be rewritten in the following form.

$$\begin{aligned} f(\boldsymbol{w}) &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i\in S(\boldsymbol{w})}(1-y_i\boldsymbol{w}^T\boldsymbol{x}_i)^2 \\ &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C(\boldsymbol{e}-YX\boldsymbol{w})^T D_{S(\boldsymbol{w})}(\boldsymbol{e}-YX\boldsymbol{w}), \end{aligned}$$

where

$$S(\boldsymbol{w}) = \{i \mid 1 \le i \le l, 1-y_i\boldsymbol{w}^T\boldsymbol{x}_i > 0\},$$

and $D_{S(\boldsymbol{w})}$ is a diagonal matrix such that

$$(D_{S(\boldsymbol{w})})_{i,i} = \begin{cases} 1 & \text{if } i \in S(\boldsymbol{w}) \\ 0 & \text{otherwise} \end{cases}, i = 1,\ldots,l.$$

Note that f is differentiable but not twice-differentiable. Therefore, we follow [1], [2] to consider a generalized Hessian for TRON.

$$\nabla^2 f(\boldsymbol{w}) = I + 2CX^T D_{S(\boldsymbol{w})} X.$$

Then in our distributed TRON algorithm, the function, gradient and Hessian-vector products for L2-loss SVM can be computed in the same way of (12)-(14), except $f_k(\boldsymbol{w})$, $\nabla f_k(\boldsymbol{w})$ and $\nabla^2 f_k(\boldsymbol{w})$ are replaced by the following functions.

$$f_k(\boldsymbol{w}) \equiv (\boldsymbol{e}_k - Y_k X_k \boldsymbol{w})^T D_k (\boldsymbol{e}_k - Y_k X_k \boldsymbol{w}), \quad (1)$$

$$\nabla f_k(\boldsymbol{w}) \equiv 2(X_k^T D_k X_k \boldsymbol{w} + X_k^T Y_k D_k \boldsymbol{e}_k), \quad (2)$$

$$\nabla^2 f_k(\boldsymbol{w})\boldsymbol{v} \equiv 2X_k^T\left(D_k\left(X_k\boldsymbol{v}\right)\right), \quad (3)$$

where D_k is the diagonal block of $D_{S(\boldsymbol{w})}$ corresponding to the k-th partition.

$$D_{S(\boldsymbol{w})} = \text{diag}(D_1,\ldots,D_p).$$

3. Experiments for L2-loss linear Support Vector Machine

We conduct experiments on L2-loss SVM with the same settings of LR. Figure I shows the scalability using 2, 4, 8 and 16 nodes. The comparisons with MLlib and MPI LIBLINEAR are shown in Figures II and III, respectively. The comparison with MLlib has very close results to the case of using LR. For the experiment comparing with MPI LIBLINEAR, the conclusion is in general similar to that of LR in the paper. The only exception is that Spark LIBLINEAR with multiple cores does not possess better training speed in comparison with Spark LIBLINEAR using only a single core per node in most cases. We can only observe significantly faster training of mult-core Spark LIBLINEAR in the dense data epsilon.

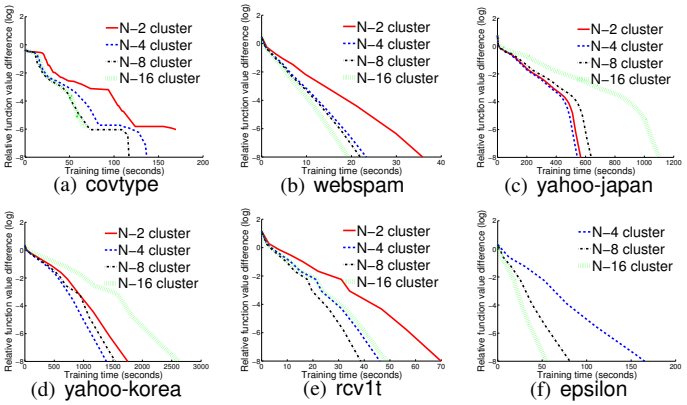

Fig. I. Scalability: We present running time (in seconds) versus the relative objective value difference. We run SVM with $C = 1$.

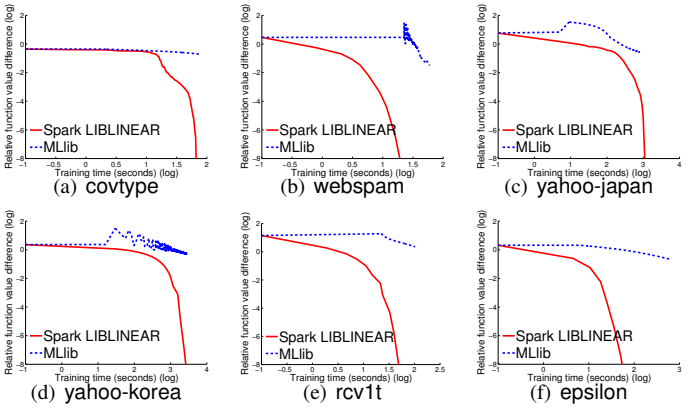

Fig. II. Comparison with MLlib: We present running time (in seconds, log scale) versus the relative objective value difference. We run SVM with $C = 1$ on 16 nodes.

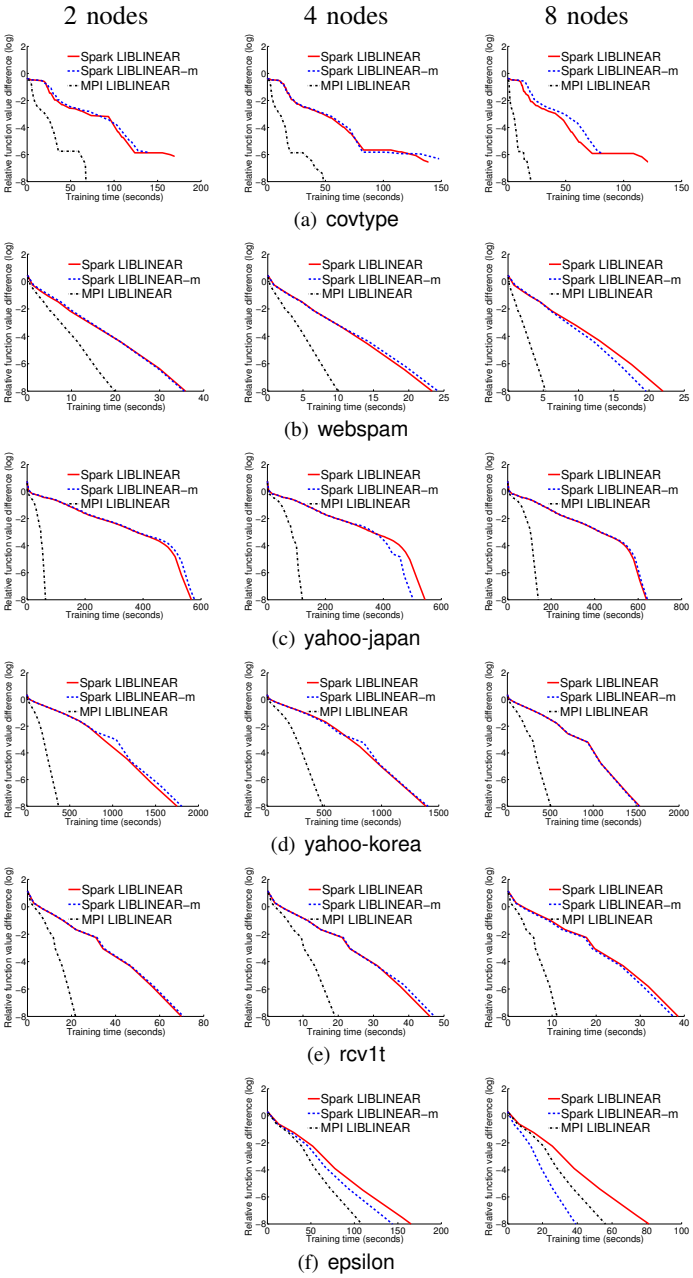

Fig. III. Comparison with MPI LIBLINEAR: We present running time (in seconds) versus the relative objective value difference. We run SVM with $C = 1$.

REFERENCES

[1] O. L. Mangasarian, "A finite Newton method for classification," *Optimization Methods and Software*, vol. 17, no. 5, pp. 913–929, 2002.

[2] C.-J. Lin, R. C. Weng, and S. S. Keerthi, "Trust region Newton method for large-scale logistic regression," *JMLR*, vol. 9, pp. 627–650, 2008.