Supplement: Distributed Box-constrained Quadratic Optimization for Dual Linear SVM

Ching-pei Lee LEECHINGPEI@GMAIL.COM
Dan Roth DANR@ILLINOIS.EDU
University of Illinois at Urbana-Champaign, 201 N. Goodwin Avenue, Urbana, IL 61801 USA

A. More Details on Convergence Analysis

In this section, we will establish a detailed convergence analysis for our methods. The main idea follows from the framework of Tseng & Yun (2009) and substitutes the local error bound with the global error bound given in Wang & Lin (2014). Note that the result in Yun (2014) does not directly apply here because $H = I$ is assumed in that work, but their analysis can also be modified for other symmetric, positive definite H by applying lemmas in Section 3 of Tseng & Yun (2009) to get the same result as ours.

For any matrix A, let $\lambda_{\min}(A)$ denotes the smallest eigenvalue of A and $\lambda_{\max}(A)$ denotes the largest eigenvalue of A.

A.1. L2-SVM

For L2-SVM, we have that $\lambda_{\min}(\bar{Q}) \geq 1/2C$, so clearly f is $1/2C$ strongly convex. In other words, $\forall \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \geq \mathbf{0}$,

$$\begin{aligned}\frac{1}{2C}\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|^2 \leq (\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2)^T \bar{Q}(\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2)\\ =(\nabla f(\boldsymbol{\alpha}_1) - \nabla f(\boldsymbol{\alpha}_2))^T(\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2).\end{aligned}$$

Also the gradient of f is $\lambda_{\max}(\bar{Q})$ Lipschitz continuous

$$\begin{aligned}\|\nabla f(\boldsymbol{\alpha}_1) - \nabla f(\boldsymbol{\alpha}_2)\| &= \sqrt{(\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2)^T \bar{Q}^2(\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2)}\\ &\leq \lambda_{\max}(\bar{Q})\|\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2\|. \qquad \text{(A.1)}\end{aligned}$$

Thus by Theorem 3.1 in Pang (1987), we have

$$\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\| \leq 2C(1 + \lambda_{\max}(\bar{Q}))\|\nabla^+ f(\boldsymbol{\alpha})\|,$$

where

$$\begin{aligned}\nabla^+ f(\boldsymbol{\alpha}) &\equiv \boldsymbol{\alpha} - [\boldsymbol{\alpha} - \nabla f(\boldsymbol{\alpha})]^+_{\mathcal{P}(\mathbf{0})},\\ [\boldsymbol{\alpha}]^+_{\mathcal{P}(\mathbf{0})} &\equiv \operatorname*{arg\,min}_{\boldsymbol{\beta}\in\mathcal{P}(\mathbf{0})} \|\boldsymbol{\alpha} - \boldsymbol{\beta}\|.\end{aligned}$$

Note that in our case, the definition of $\nabla^+ f(\boldsymbol{\alpha})$ is equivalent to that of $\boldsymbol{d}_I(\boldsymbol{\alpha})$ in Assumption 2(a) in Tseng & Yun (2009). Thus this assumption is satisfied with $\tau = 2C(1 + \lambda_{\max}(\bar{Q})), \epsilon = \infty$ for any ξ. Also note that Assumption 2(b) of Tseng & Yun (2009) always holds in convex optimization problems. Therefore, $\bar{k} = 0, \tau' = \tau$ in Equation (36), and $\hat{k} = 0$ in Equation (37) of Tseng & Yun (2009). Following their analysis and their Lemma 5(b), we then have

$$f(\boldsymbol{\alpha}^{t+1}) - f(\boldsymbol{\alpha}^*) \leq \frac{C_2}{1 + C_2}(f(\boldsymbol{\alpha}^t) - f(\boldsymbol{\alpha}^*)), \forall t \in \mathbf{N}\cup\{0\}, \qquad \text{(A.2)}$$

where

$$C_2 = C_1/(\sigma\beta \min\{1, \frac{1 - \sigma + \sigma\gamma}{C\lambda_{\max}(\bar{Q})}\}), \qquad \text{(A.3)}$$

and C_1 is a constant that only depends on $\lambda_{\max}(\bar{Q}), \lambda_{\min}(\bar{Q}), \lambda_{\max}(H), \lambda_{\min}(H)$, and C.

A.2. L1-SVM

For the case of L1-SVM, we see that the problem is of the form

$$\begin{aligned}\min_{\boldsymbol{\alpha}\in\mathbf{R}^l} \quad & f(\boldsymbol{\alpha}) = g((YX)^T\boldsymbol{\alpha}) - \boldsymbol{e}^T\boldsymbol{\alpha}\\ \text{subject to} \quad & \boldsymbol{\alpha} \in \mathcal{P}(\mathbf{0}),\end{aligned}$$

where

$$g(\cdot) = \frac{1}{2}\|\cdot\|^2$$

is 1 strongly convex. From (A.1), we know that the gradient of f is $\lambda_{\max}(\bar{Q})$ Lipschitz continuous. In addition, it is clear that $\mathcal{P}(\mathbf{0})$ is a polyhedral set. Thus, according to Theorem 4.6 in Wang & Lin (2014), Assumption 2(a) of Tseng & Yun (2009) is also satisfied with a τ that depends on $C, \lambda_{\max}(\bar{Q}), \lambda_{\max}(\bar{H}), f(\boldsymbol{\alpha}^0) - f(\boldsymbol{\alpha}^*), \|\nabla f(\boldsymbol{\alpha}^*)\|, (\boldsymbol{\alpha}^*)^T Q\boldsymbol{\alpha}^*$ and $\tilde{\tau}$ in (6). We can then substitute this result into (A.3) to get a similar result to (A.2). Since the rates are related to the eigenvalues of H, it will be interesting to consider this property to construct H that has a better convergence rate upper bound. We leave this as a future research direction.

A.3. Discussion

Our analysis indicates that the convergence speed of our method in (A.2) is independent of both l and n. Thus the iteration complexity depends on l only via the optimal function value, which is upper bounded by $f^P(\mathbf{0}) = Cl$. If we

consider the scaled problem $f^P(\boldsymbol{w})/Cl$ and $f(\boldsymbol{\alpha})/Cl$ used in other works including Yang (2013); Ma et al. (2015), then the iteration complexity of our method is totally independent of l and n.

This result is not surprising, because we are considering the rounds of communication and outer iteration, while the overall training time might still be dependent on l and n. Note that the data dimension n does not affect the number of variables being optimized, and thus does not contribute to the iteration complexity. However, note that n affects the training time for solving the local-sub-problems at each iteration, and the communication cost is linear to n as long as a method synchronizes $\boldsymbol{w}$. Also, in the analysis of Wang & Lin (2014) for using cyclic coordinate descent method to train linear SVM, the training time depends on l because in this method, the definition of one iteration is passing through all instances once and thus the running time will be at least l times the iteration complexity. But since our framework can use any sub-problem solver, this is might be avoided by considering a solver whose training time is independent of l.

B. Additional Experiment Results

In this section, we provide more experimental results.

B.1. Results Appeared Partially in Section 4

We present more results under the same setting of Section 4. Figure (I) presents the dual objective values and accuracies of epsilon. Figure (II) shows the primal objective values of all data sets.

B.2. Training Time Profile

To better understand the bottleneck of the training time, we investigate the fraction of the total training time taken by computation, synchronization and communication. The result is presented in Figure (III). By synchronization cost, we mean the time amount spent when one machine finished solving its corresponding sub-problem (7) but has not started communication because it is idle to wait for other machines to finish solving (7). Note that since in our experimental setting, each solver conducts one round of size-n communication every time it goes through the whole data once, the behaviors in communication and synchronization of all solvers should be similar. Therefore, we only report the result of BQO-E. We can see that for sparse data sets url and webspam, the proportion of computation time is rather low and synchronization time is larger. This means that each machine spent a different amount of time to solve the local sub-problem. Thus, one might consider setting the number of instances being examined at each iteration to be identical for all machines to reduce the synchronization time. This will be another advantage over primal batch solvers that require exactly one pass through all instances at each round of communication.

B.3. Speedup On Primal Objective Value

In Section 4, the training time of the speedup experiments is obtained by the following. We reported the time for TRON to reach $\boldsymbol{w}^t$ satisfying

$$|\frac{f^P(\boldsymbol{w}^t) - f^P(\boldsymbol{w}^*)}{f^P(\boldsymbol{w}^*)}| \leq 0.01, \tag{B.1}$$

where $\boldsymbol{w}^*$ is the optimal solution of (1). For DSVM-AVE and BQO-E that minimize (2), we report the time for them to obtain $\boldsymbol{\alpha}^t$ such that

$$|\frac{f(\boldsymbol{\alpha}^t) - f(\boldsymbol{\alpha}^*)}{f(\boldsymbol{\alpha}^*)}| \leq 0.01.$$

Here we report the result of considering (B.1) for DSVM-AVE and BQO-E in Figures (IV)-(V). In this setting, both DSVM-AVE and BQO-E have better speedups. Also, DSVM-AVE and BQO-E have similar training time in epsilon, but BQO-E is significantly better in webspam in terms of both training time and speedup.

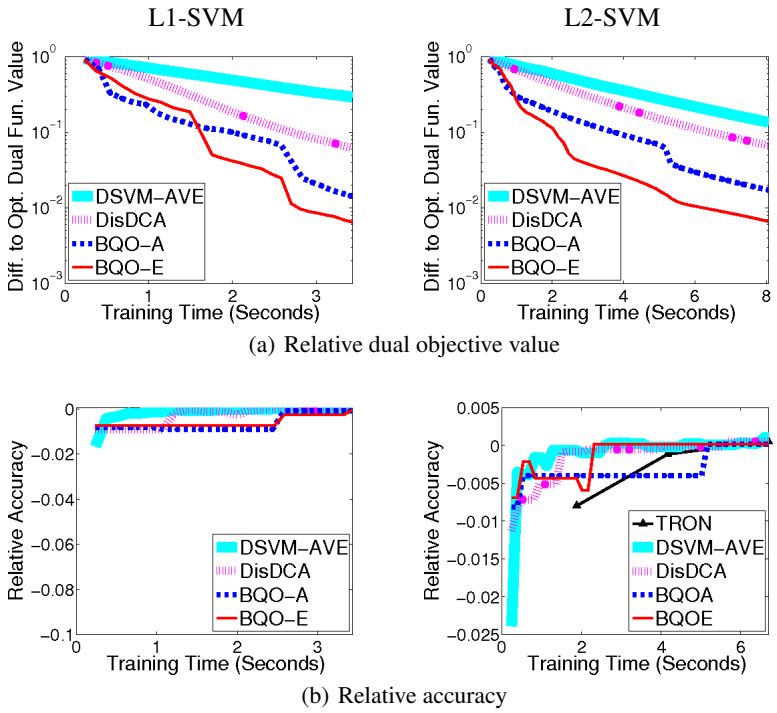
B.4. Experiments on Different Values of C

We also conduct additional experiments using different values of C. For the data sets we considered, $C = 1$ is already large enough because the number of instances is large and the loss term is summation instead of average over all instances. Thus we consider smaller C. Tables (I)-(II) present the result of $C = 1e-2$ while Tables (III)-(IV) present the result of $C = 1e-4$. We can see that when the problems are easier to solve, i.e., when the training time of all methods is short, DisDCA is usually the fastest, while our method is still faster than DSVM-AVE and TRON in most cases. However, in these situations, the training time does not affect the total running time too much, because in this case the data loading time is the bottleneck. Moreover, note that our line search methods are also applicable to DisDCA to further improve the training time in this situation.

References

Ma, Chenxin, Smith, Virginia, Jaggi, Martin, Jordan, Michael I, Richtárik, Peter, and Takáč, Martin. Adding vs. averaging in distributed primal-dual optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

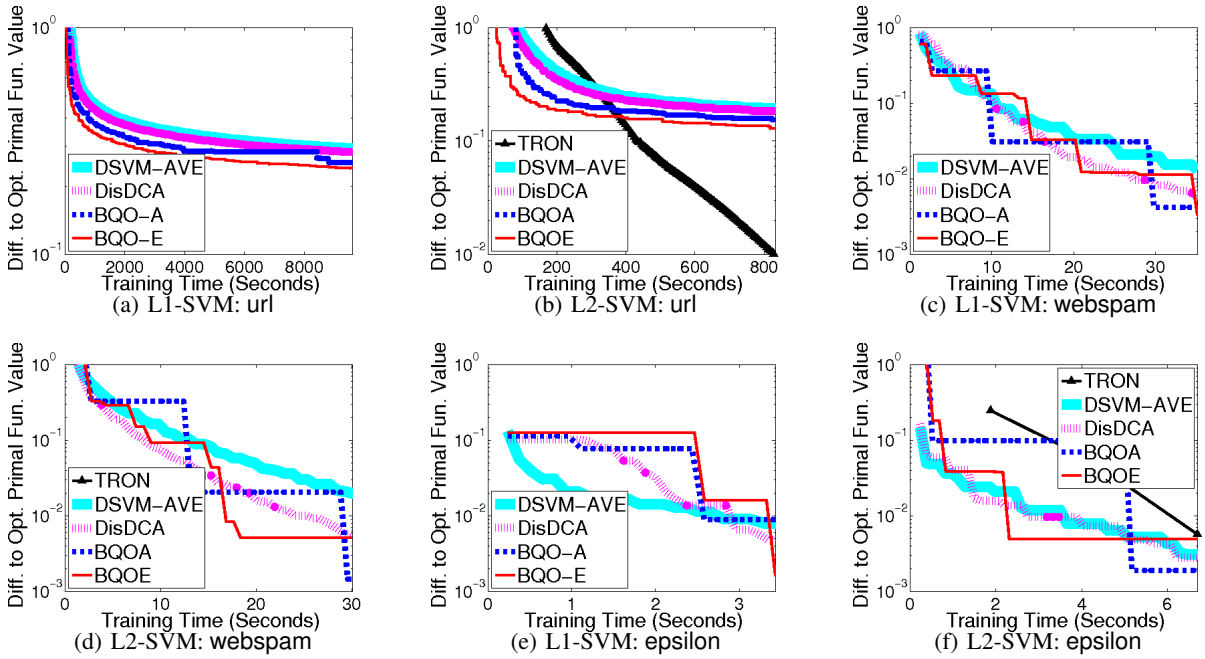
Pang, Jong-Shi. A posteriori error bounds for the linearly-constrained variational inequality problem. *Mathematics of Operations Research*, 12(3):474–484, 1987.



(a) Relative dual objective value

(b) Relative accuracy

Figure (I). Experiment results on epsilon. Top: time versus relative dual objective value. Bottom: time versus relative accuracy.



(a) L1-SVM: url (b) L2-SVM: url (c) L1-SVM: webspam

(d) L2-SVM: webspam (e) L1-SVM: epsilon (f) L2-SVM: epsilon

Figure (II). Time versus relative primal objective value. Time is in seconds. Top: L1-SVM, bottom: L2-SVM. Note that in webspam, the function values of TRON for L2-SVM are too large to appear in the figure.

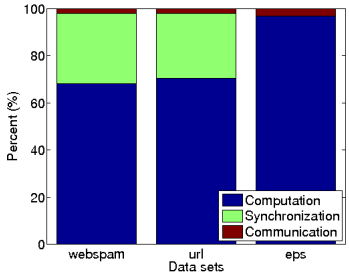


Figure (III). Percentage of computation, synchronization, and communication in the total training time.

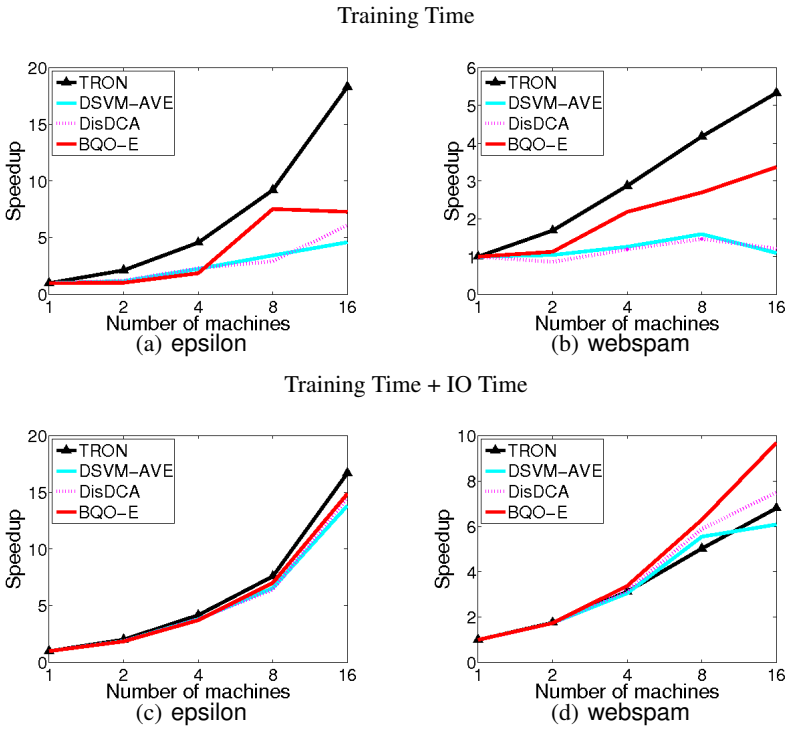


Figure (IV). Speedup of different methods training L2-SVM. All algorithms use (B.1) to decide training time. Top: speedup of training time. Bottom: speedup of the total running time including training and data loading time.

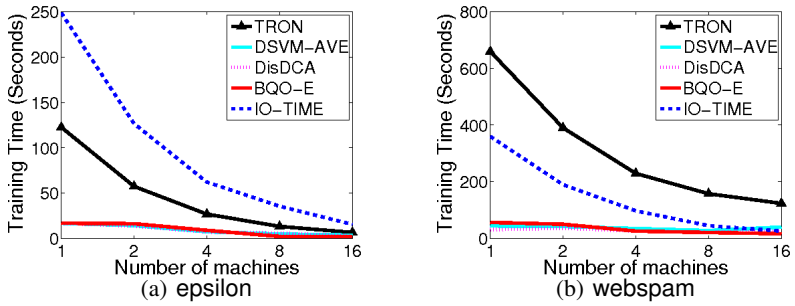


Figure (V). Training time of L2-SVM using different numbers of machines. All algorithms use (B.1) to decide training time.

Data set	ϵ	L1-SVM solvers				L2-SVM solvers				
		BQO-E	**BQO-A**	DisDCA	DSVM-AVE	**BQO-E**	**BQO-A**	DisDCA	DSVM-AVE	TRON
url	0.01	21.4	**21.4**	70.5	79.3	**42.7**	146.8	98.3	115.5	46.9
epsilon	0.01	2.1	1.2	**0.4**	2.2	1.6	3.3	**0.4**	1.6	3.1
webspam	0.01	17.1	14.1	**1.8**	15.2	6.6	9.0	**1.8**	12.0	35.9

Table (I). Training time required for a solver to reach $(f^P(\boldsymbol{w}^t) - f^P(\boldsymbol{w}^*)) \leq \epsilon f^P(\boldsymbol{w}^*)$. We present results of $C = 0.01$.

Data set	ϵ	L1-SVM solvers				L2-SVM solvers			
		BQO-E	**BQO-A**	DisDCA	DSVM-AVE	**BQO-E**	**BQO-A**	DisDCA	DSVM-AVE
url	0.01	**10.8**	11.3	36.4	41.3	**296.6**	593.1	1,025.6	1,203.0
epsilon	0.01	2.0	1.6	**0.5**	12.2	1.8	2.5	**0.6**	6.4
webspam	0.01	11.3	16.4	**2.4**	56.2	8.4	10.7	**2.6**	28.6

Table (II). Training time required for a solver to reach $(f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^t)) \leq \epsilon f(\boldsymbol{\alpha}^*)$. We present results of $C = 0.01$.

Data set	ϵ	L1-SVM solvers				L2-SVM solvers				
		BQO-E	**BQO-A**	DisDCA	DSVM-AVE	**BQO-E**	**BQO-A**	DisDCA	DSVM-AVE	TRON
url	0.01	3.3	5.9	**1.1**	3.1	1.3	9.1	**0.8**	2.7	9.2
epsilon	0.01	**0.3**	0.3	0.3	0.9	0.3	0.3	**0.3**	2.1	0.9
webspam	0.01	5.9	7.5	**1.9**	19.4	10.1	8.1	**2.1**	15.3	13.4

Table (III). Training time required for a solver to reach $(f^P(\boldsymbol{w}^t) - f^P(\boldsymbol{w}^*)) \leq \epsilon f^P(\boldsymbol{w}^*)$. We present results of $C = 0.0001$.

Data set	ϵ	L1-SVM solvers				L2-SVM solvers			
		BQO-E	**BQO-A**	DisDCA	DSVM-AVE	**BQO-E**	**BQO-A**	DisDCA	DSVM-AVE
url	0.01	2.7	6.1	**0.8**	11.9	**2.2**	3.6	5.2	8.9
epsilon	0.01	**0.3**	0.3	0.3	12.3	0.3	0.4	**0.3**	6.0
webspam	0.01	6.7	7.5	**1.9**	65.7	10.1	7.1	**2.1**	30.1

Table (IV). Training time required for a solver to reach $(f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}^t)) \leq \epsilon f(\boldsymbol{\alpha}^*)$. We present results of $C = 0.0001$.

Tseng, Paul and Yun, Sangwoon. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117:387–423, 2009.

Wang, Po-Wei and Lin, Chih-Jen. Iteration complexity of feasible descent methods for convex optimization. *Journal of Machine Learning Research*, 15:1523–1548, 2014.

Yang, Tianbao. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems 26*, pp. 629–637, 2013.

Yun, Sangwoon. On the iteration complexity of cyclic coordinate gradient descent methods. *SIAM Journal on Optimization*, 24(3):1567–1580, 2014.