Supplementary Materials of "Limited-memory Common-directions Method for Distributed L1-regularized Linear Classification"

Wei-Lin Chiang* Yu-Sheng Li† Ching-pei Lee‡ Chih-Jen Lin§

I Introduction

In this document, we present additional details and more experimental results.

II Derivation of the Direction Used in LBFGS

By using only the information from the last m iterations, the definition of B_k in BFGS becomes the following in LBFGS.

(II.1)
$$\begin{aligned}B_k = & V_{k-1}^T \cdots V_{k-m}^T B_0^k V_{k-m} \cdots V_{k-1} \\ & + \rho_{k-m} V_{k-1}^T \cdots V_{k-m+1}^T \boldsymbol{s}_{k-m} \boldsymbol{s}_{k-m}^T V_{k-m+1} \cdots V_{k-1} \\ & + \cdots + \rho_{k-1} \boldsymbol{s}_{k-1} \boldsymbol{s}_{k-1}^T.\end{aligned}$$

Note that in BFGS, B_0^k is a fixed matrix, but in LBFGS, B_0^k can change with k, provided its eigenvalues are bounded in a positive interval over k. A common choice is

$$B_0^k = \frac{\boldsymbol{s}_{k-1}^T \boldsymbol{u}_{k-1}}{\boldsymbol{u}_{k-1}^T \boldsymbol{u}_{k-1}} I. \tag{II.2}$$

By expanding (II.1), $\boldsymbol{d}_k$ can be efficiently obtained by $\mathcal{O}(m)$ vector operations as shown in Algorithm II. The overall procedure of LBFGS is summarized in Algorithm I.

III More Details of Limited-memory Common-directions Method

A sketch of the procedure for L1-regularized problems is in Algorithm III.

IV Line Search in Algorithms for L2-regularized Problems

Here we present the trick mentioned in Section 2.3 in the paper. At each line search iteration, we obtain $\boldsymbol{w}^T \boldsymbol{x}_i, \boldsymbol{d}^T \boldsymbol{x}_i, \forall i$ first, and then use $\mathcal{O}(l)$ cost to calculate

$$(\boldsymbol{w} + \alpha \boldsymbol{d})^T \boldsymbol{x}_i = \boldsymbol{w}^T \boldsymbol{x}_i + \alpha \boldsymbol{d}^T \boldsymbol{x}_i, \forall i.$$

Algorithm I LBFGS.

Given $\boldsymbol{w}_0$, integer $m > 0$, and $\beta, \gamma \in (0, 1)$
$\boldsymbol{w} \leftarrow \boldsymbol{w}_0$
for $k = 0, 1, 2, \ldots$ **do**
  Calculate $\nabla f(\boldsymbol{w})$
  Calculate the new direction
$$\boldsymbol{d} \equiv -B \nabla f(\boldsymbol{w}) \approx -\nabla^2 f(\boldsymbol{w})^{-1} \nabla f(\boldsymbol{w})$$
  by using information of the previous m iterations (Algorithm II)
  Calculate $\nabla f(\boldsymbol{w})^T \boldsymbol{d}$
  $\alpha \leftarrow 1, \boldsymbol{w}^{\text{old}} \leftarrow \boldsymbol{w}$
  while true **do**
    $\boldsymbol{w} \leftarrow \boldsymbol{w}^{\text{old}} + \alpha \boldsymbol{d}$
    Calculate the objective value $f(\boldsymbol{w})$ in (2.2)
    if $f(\boldsymbol{w}) - f(\boldsymbol{w}^{\text{old}}) \leq \gamma \nabla f(\boldsymbol{w}^{\text{old}})^T (\boldsymbol{w} - \boldsymbol{w}^{\text{old}})$
      break
    $\alpha \leftarrow \alpha\beta$
  Update P with $\boldsymbol{w} - \boldsymbol{w}^{\text{old}}$ and $\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{w}^{\text{old}})$

Because $\boldsymbol{w}^T \boldsymbol{x}_i$ can be obtained from the previous iteration, $\boldsymbol{d}^T \boldsymbol{x}_i$ is the only $\mathcal{O}(\#\text{nnz})$ operation needed. The line search cost is reduced from

$$\#\text{line-search steps} \times \mathcal{O}(\#\text{nnz})$$

to

$$1 \times \mathcal{O}(\#\text{nnz}) + \#\text{line-search steps} \times \mathcal{O}(l).$$

This trick is not applicable to L1-regularized problems because the new point is no longer $\boldsymbol{w} + \alpha \boldsymbol{d}$.

V More on the Distributed Implementation

V.1 Complexity. The distributed implementation as mentioned in Section 5 is shown in Algorithm VI. Then we discuss the complexity below.

$$\frac{(2 + \#\text{line-search steps}) \times \mathcal{O}(\#\text{nnz}) + \mathcal{O}(lm^2) + \mathcal{O}(mn)}{K} + \mathcal{O}(m^3)$$

VI More Experiments

The data sets used in this section are shown in Table (I).

*National Taiwan University. b02902056@ntu.edu.tw
†National Taiwan University. b03902086@ntu.edu.tw
‡University of Wisconsin-Madison. ching-pei@cs.wisc.edu
§National Taiwan University. cjlin@csie.ntu.edu.tw

Algorithm II LBFGS Two-loop recursion.

$\boldsymbol{q} \leftarrow -\nabla f(\boldsymbol{w})$
for $i = k-1, k-2, \ldots, k-m$ **do**
  $\alpha_i \leftarrow \boldsymbol{s}_i^T \boldsymbol{q} / \boldsymbol{s}_i^T \boldsymbol{u}_i$
  $\boldsymbol{q} \leftarrow \boldsymbol{q} - \alpha_i \boldsymbol{u}_i$
$\boldsymbol{r} \leftarrow (\boldsymbol{s}_{k-1}^T \boldsymbol{u}_{k-1} / \boldsymbol{u}_{k-1}^T \boldsymbol{u}_{k-1}) \boldsymbol{q}$
for $i = k-m, k-m+1, \ldots, k-1$ **do**
  $\beta_i \leftarrow \boldsymbol{u}_i^T \boldsymbol{r} / \boldsymbol{s}_i^T \boldsymbol{u}_i$
  $\boldsymbol{r} \leftarrow \boldsymbol{r} + (\alpha_i - \beta_i) \boldsymbol{s}_i$
$\boldsymbol{d} \leftarrow \boldsymbol{r}$

Algorithm III Limited-memory common-directions method for L1-regularized problems.

while true **do**
  Compute $\nabla^{\mathrm{P}} f(\boldsymbol{w})$ by (2.6)
  Solve the sub-problem (3.21)
  Let the direction be $\boldsymbol{d} = P\boldsymbol{t}$
  for $j = 1, \ldots, n$ **do**
    Align d_j with $-\nabla_j^{\mathrm{P}} f(\boldsymbol{w})$ by (2.11)
  $\alpha \leftarrow 1$, $\boldsymbol{w}^{\mathrm{old}} \leftarrow \boldsymbol{w}$
  while true **do**
    Calculate $\boldsymbol{w}$ from $\boldsymbol{w}^{\mathrm{old}} + \alpha \boldsymbol{d}$ by (2.12)
    if $f(\boldsymbol{w}) - f(\boldsymbol{w}^{\mathrm{old}}) \le \gamma \nabla^{\mathrm{P}} f(\boldsymbol{w}^{\mathrm{old}})^T (\boldsymbol{w} - \boldsymbol{w}^{\mathrm{old}})$
      break
    $\alpha \leftarrow \alpha\beta$
  Update P and XP

VI.1 More Results by Using Different C Values. In Figure (I), we present more results with

$$C = \{0.1C_{\text{Best}}\ , C_{\text{Best}}\ , 10C_{\text{Best}}\ \},$$

where C_{Best} is the value to achieve the highest cross validation accuracy. The results are similar to C_{Best} presented in Section 6, but we observe that NEWTON converges slowly in larger C cases.

VI.2 More Results on Distributed Experiments. In Figure (II), we present more data sets for the distributed experiments. All settings are the same as in Section 6.

Table (I): Data statistics.

Data set	#instances	#features	#nonzeros	C_{Best}
real-sim	72,309	20,958	3,709,083	16
rcv1_test	677,399	47,226	49,556,258	4
news20	19,996	1,355,191	9,097,916	1024
yahoojp	176,203	832,026	23,506,415	2
url	2,396,130	3.231,961	277,058,644	8
yahookr	460,554	3,052,939	156,436,656	4
epsilon	400,000	2,000	800,000,000	0.5
webspam	350,000	16,609,143	1,304,697,446	64
KDD2010-b	19,264,097	29,890,096	566,345,888	0.5
criteo	45,840,617	1,000,000	1,787,773,969	0.5
avazu-site	25,832,830	999,962	387,492,144	1
kdd2012	149,639,105	54,686,452	1,646,030,155	2

Algorithm IV A distributed implementation of OWLQN.

for $k = 0, 1, 2, \ldots$ **do**

  Compute $\nabla^{\mathrm{P}} f(\boldsymbol{w})$ by (2.6) and

$$\nabla L(\boldsymbol{w}) = C \bigoplus_{r=1}^{K} (X_{J_r,:})^T \begin{bmatrix} \vdots \\ \xi'(y_i \boldsymbol{w}^T \boldsymbol{x}_i) \\ \vdots \end{bmatrix}_{i \in J_r}.$$

▷ $\mathcal{O}(\#\mathrm{nnz}/K)$; $\mathcal{O}(n)$ comm.

  Compute the search direction $\boldsymbol{d}_{\bar{J}_r}, r = 1, \ldots, K$ by Algorithm V ▷ $\mathcal{O}(nm/K)$; $\mathcal{O}(m)$ comm.

  An *allgather* operation to let each node has

$$\boldsymbol{d} = \begin{bmatrix} \boldsymbol{d}_{\bar{J}_1} \\ \vdots \\ \boldsymbol{d}_{\bar{J}_K} \end{bmatrix}$$

▷ $\mathcal{O}(n/K)$ comm.

  for $j = 1, \ldots, n$ **do**

    Align d_j with $-\nabla_j^{\mathrm{P}} f(\boldsymbol{w})$ by (2.11)

  $\alpha \leftarrow 1$, $\boldsymbol{w}^{\mathrm{old}} \leftarrow \boldsymbol{w}$

  while true **do**

    Calculate $\boldsymbol{w}$ from $\boldsymbol{w}^{\mathrm{old}} + \alpha \boldsymbol{d}$ by (2.12) and

$$f(\boldsymbol{w}) = \|\boldsymbol{w}\|_1 + C \bigoplus_{r=1}^{K} \sum\nolimits_{i \in J_r} \xi(y_i \boldsymbol{w}^T \boldsymbol{x}_i)$$

▷ $\mathcal{O}(\#\mathrm{nnz}/K)$; $\mathcal{O}(1)$ comm.

    if $f(\boldsymbol{w}) - f(\boldsymbol{w}^{\mathrm{old}}) \le \gamma \nabla^{\mathrm{P}} f(\boldsymbol{w}^{\mathrm{old}})^T (\boldsymbol{w} - \boldsymbol{w}^{\mathrm{old}})$

      break

    $\alpha \leftarrow \alpha\beta$

  $\boldsymbol{s}_k \leftarrow \boldsymbol{w} - \boldsymbol{w}^{\mathrm{old}}, \quad \boldsymbol{u}_k \leftarrow \nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{w}^{\mathrm{old}})$

  Remove 1st column of S and U if needed and

$$S \leftarrow \begin{bmatrix} S & \boldsymbol{s}_k \end{bmatrix}, \quad U \leftarrow \begin{bmatrix} U & \boldsymbol{u}_k \end{bmatrix}$$

  $\rho_k \leftarrow \bigoplus_{r=1}^{K} (\boldsymbol{u}_k)_{\bar{J}_r}^T (\boldsymbol{s}_k)_{\bar{J}_r}$ ▷ $\mathcal{O}(n/K)$; $\mathcal{O}(1)$ comm.

Algorithm V Distributed OWLQN Two-loop recursion

if $k = 0$ **return** $\boldsymbol{d}_{\bar{J}_r} \leftarrow -\nabla_{\bar{J}_r}^{\mathrm{P}} f(\boldsymbol{w})$

for $r = 1, \ldots, K$ **do in parallel**

  $\boldsymbol{q}_{\bar{J}_r} \leftarrow -\nabla_{\bar{J}_r}^{\mathrm{P}} f(\boldsymbol{w})$ ▷ $\mathcal{O}(n/K)$

for $i = k-1, k-2, \ldots, k-m$ **do**

  Calculate α_i by

$$\alpha_i \leftarrow \frac{\bigoplus_{r=1}^{K} (\boldsymbol{s}_i)_{\bar{J}_r}^T \boldsymbol{q}_{\bar{J}_r}}{\rho_i}$$

▷ $\mathcal{O}(n/K)$; $\mathcal{O}(1)$ comm.

  for $r = 1, \ldots, K$ **do in parallel**

    $\boldsymbol{q}_{\bar{J}_r} \leftarrow \boldsymbol{q}_{\bar{J}_r} - \alpha_i (\boldsymbol{u}_i)_{\bar{J}_r}$ ▷ $\mathcal{O}(n/K)$

Calculate

$$\boldsymbol{u}_{k-1}^T \boldsymbol{u}_{k-1} \leftarrow \bigoplus_{r=1}^{K} (\boldsymbol{u}_{k-1})_{\bar{J}_r}^T (\boldsymbol{u}_{k-1})_{\bar{J}_r}$$

▷ $\mathcal{O}(n/K)$; $\mathcal{O}(1)$ comm.

for $r = 1, \ldots, K$ **do in parallel**

  $\boldsymbol{r}_{\bar{J}_r} \leftarrow \frac{\rho_{k-1}}{\boldsymbol{u}_{k-1}^T \boldsymbol{u}_{k-1}} \boldsymbol{r}_{\bar{J}_r}$ ▷ $\mathcal{O}(n/K)$

for $i = k-m, k-m+1, \ldots, k-1$ **do**

  Calculate β_i by

$$\beta_i \leftarrow \frac{\bigoplus_{r=1}^{K} (\boldsymbol{u}_i)_{\bar{J}_r}^T \boldsymbol{r}_{\bar{J}_r}}{\rho_i}$$

▷ $\mathcal{O}(n/K)$; $\mathcal{O}(1)$ comm.

  for $r = 1, \ldots, K$ **do in parallel**

    $\boldsymbol{r}_{\bar{J}_r} \leftarrow \boldsymbol{r}_{\bar{J}_r} + (\alpha_i - \beta_i)(\boldsymbol{s}_i)_{\bar{J}_r}$ ▷ $\mathcal{O}(n/K)$

return $\boldsymbol{d}_{\bar{J}_r} \leftarrow \boldsymbol{r}_{\bar{J}_r}, r = 1, \ldots, K$

Algorithm VI Distributed limited-memory common-directions method.

while true **do**

  Compute $\nabla^{\mathrm{P}} f(\boldsymbol{w})$ by (2.6) and

$$\nabla L(\boldsymbol{w}) = C \bigoplus_{r=1}^{K} (X_{J_r,:})^T \begin{bmatrix} \vdots \\ \xi'(y_i \boldsymbol{w}^T \boldsymbol{x}_i) \\ \vdots \end{bmatrix}_{i \in J_r}.$$

  ▷ $\mathcal{O}(\#\mathrm{nnz}/K)$; $\mathcal{O}(n)$ comm.

  Calculate

$$X_{J_r,:} \nabla^{\mathrm{P}} f(\boldsymbol{w})$$

  ▷ $\mathcal{O}(\#\mathrm{nnz}/K)$

  Remove 1st column of P and U if needed and

$$P_{J_r,:} \leftarrow \begin{bmatrix} P_{J_r,:} & \nabla^{\mathrm{P}}_{J_r} f(\boldsymbol{w}) \end{bmatrix}$$
$$U_{J_r,:} \leftarrow \begin{bmatrix} U_{J_r,:} & X_{J_r,:} \nabla^{\mathrm{P}} f(\boldsymbol{w}) \end{bmatrix}$$

  Calculate

$$(XP)^T D_{\boldsymbol{w}} (XP) = \bigoplus_{r=1}^{K} (U_{J_r,:})^T (D_{\boldsymbol{w}})_{J_r,J_r} U_{J_r,:}$$

  ▷ $\mathcal{O}(lm^2/K)$, $\mathcal{O}(m^2)$ comm.

$$-P^T \nabla^{\mathrm{P}} f(\boldsymbol{w}) = -\bigoplus_{r=1}^{K} (P_{\bar{J}_r,:})^T \nabla^{\mathrm{P}}_{\bar{J}_r} f(\boldsymbol{w})$$

  ▷ $\mathcal{O}(mn/K)$; $\mathcal{O}(m)$ comm.

  Solve

$$\left((XP)^T D_{\boldsymbol{w}} (XP)\right) \boldsymbol{t} = -P^T \nabla^{\mathrm{P}} f(\boldsymbol{w})$$

  ▷ $\mathcal{O}(m^3)$

  Let the direction be

$$\boldsymbol{d} = P\boldsymbol{t} = \left[P_{\bar{J}_1,:}\boldsymbol{t}, \ldots, P_{\bar{J}_K,:}\boldsymbol{t}\right]^T$$

  ▷ $\mathcal{O}(mn/K)$; $\mathcal{O}(n/K)$ comm.

  for $j = 1, \ldots, n$ **do**

    Align d_j with $-\nabla^{\mathrm{P}}_j f(\boldsymbol{w})$ by (2.11)

  $\alpha \leftarrow 1$, $\boldsymbol{w}^{\mathrm{old}} \leftarrow \boldsymbol{w}$

  while true **do**

    Calculate $\boldsymbol{w}$ from $\boldsymbol{w}^{\mathrm{old}} + \alpha \boldsymbol{d}$ by (2.12) and

$$f(\boldsymbol{w}) = \|\boldsymbol{w}\|_1 + C \bigoplus_{r=1}^{K} \sum_{i \in J_r} \xi(y_i \boldsymbol{w}^T \boldsymbol{x}_i)$$

  ▷ $\mathcal{O}(\#\mathrm{nnz}/K)$; $\mathcal{O}(1)$ comm.

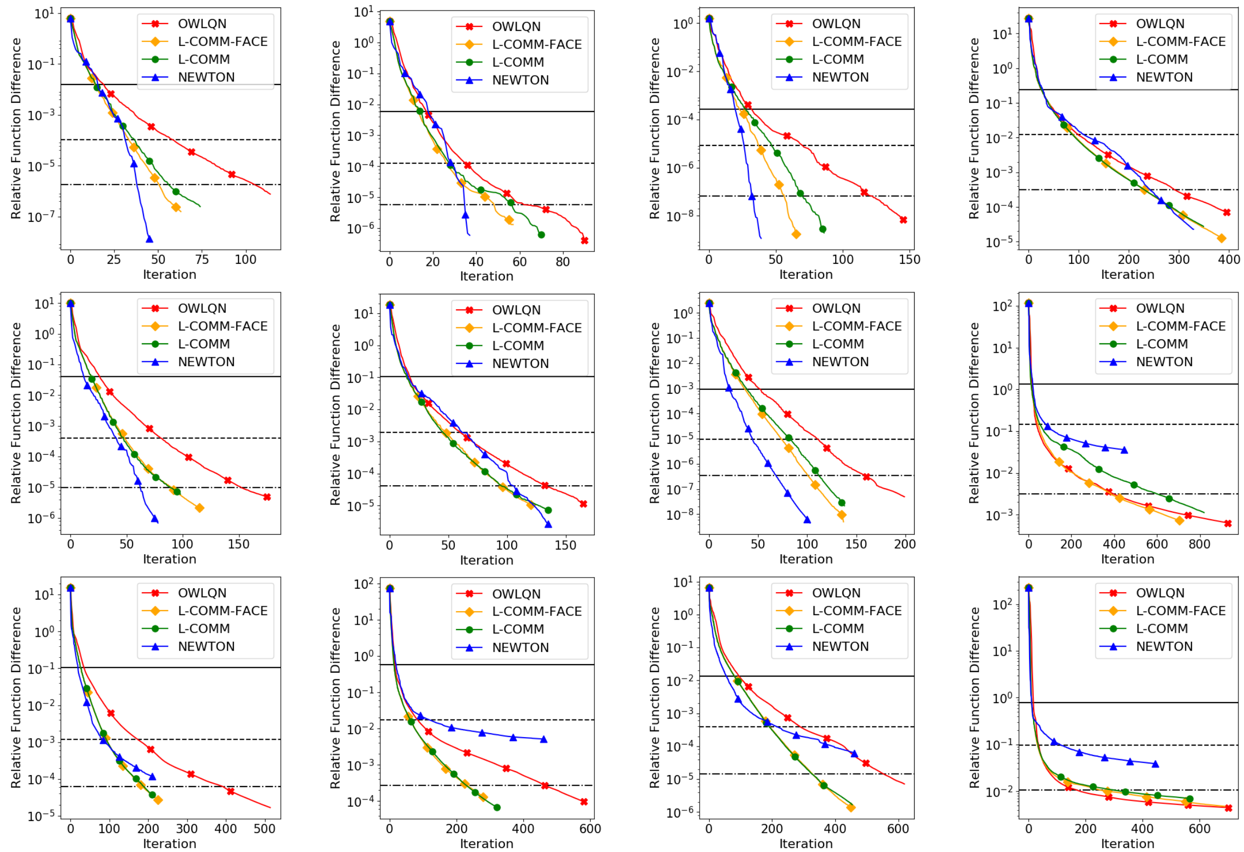
    if $f(\boldsymbol{w}) - f(\boldsymbol{w}^{\mathrm{old}}) \leq \gamma \nabla^{\mathrm{P}} f(\boldsymbol{w}^{\mathrm{old}})^T (\boldsymbol{w} - \boldsymbol{w}^{\mathrm{old}})$

      break

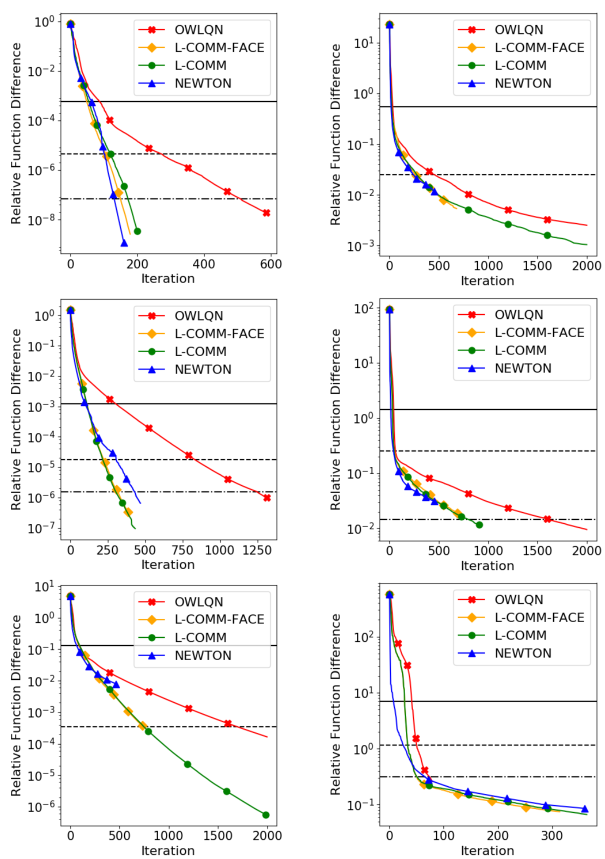
    $\alpha \leftarrow \alpha\beta$

  Remove 1st column of P and U if needed and

$$P \leftarrow \begin{bmatrix} P & \boldsymbol{w} - \boldsymbol{w}^{\mathrm{old}} \end{bmatrix}$$
$$U_{J_r,:} \leftarrow \begin{bmatrix} U_{J_r,:} & X_{J_r,:}(\boldsymbol{w} - \boldsymbol{w}^{\mathrm{old}}) \end{bmatrix}$$

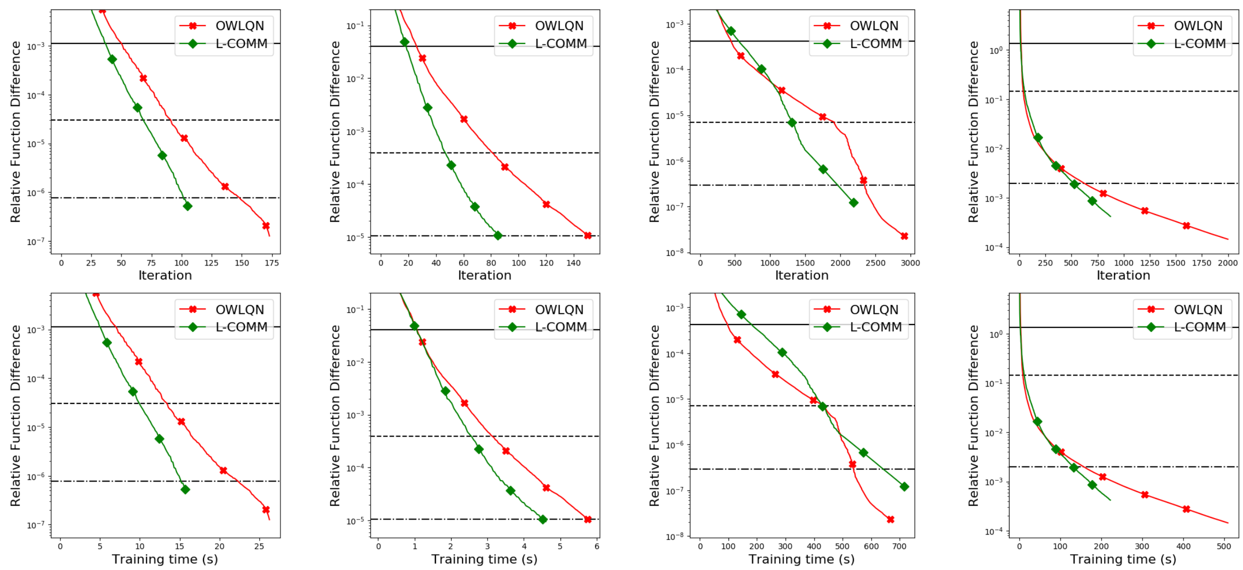


(a) rcv1_test (b) real-sim (c) yahoojp (d) news20

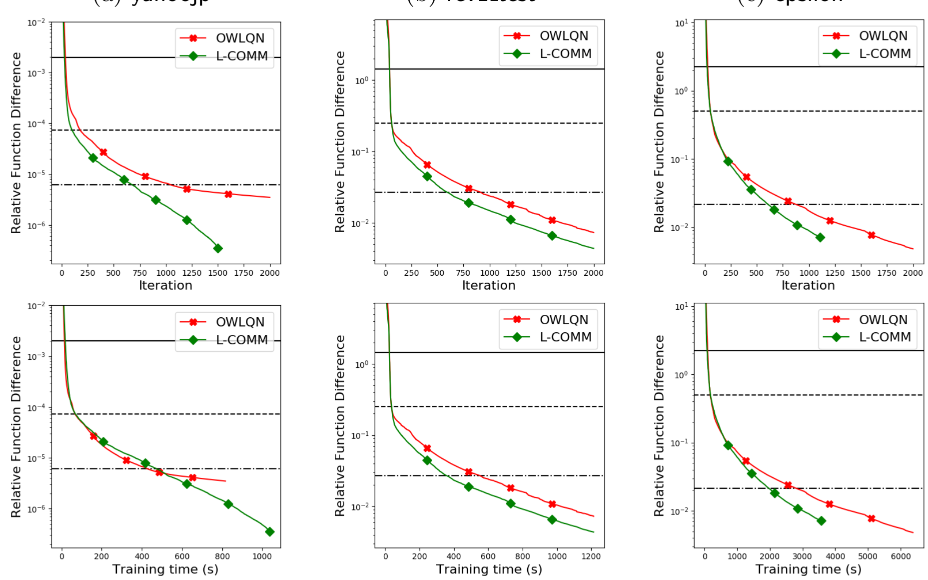


(e) yahookr (f) url

Figure (I): Comparison of different algorithms with $0.1C_{\text{Best}}$, C_{Best} , $10C_{\text{Best}}$, respectively from top to below for each data set. We show iteration versus the relative difference to the optimal value. Other settings are the same as in Figure 3



(a) yahoojp (b) rcv1_test (c) epsilon (d) news20



(e) avazu-site (f) url (g) webspam

Figure (II): Comparison of different algorithms by using 32 nodes. Upper: iterations. Lower: running time in seconds. Other settings are the same as Figure 4.