Limited-memory Common-directions Method for Distributed L1-regularized Linear Classification

Wei-Lin Chiang (National Taiwan University), Yu-Sheng Li (National Taiwan University), Ching-pei Lee* (UW-Madison), Chih-Jen Lin (National Taiwan University)
{r06922166,b03902086,cjlin}@csie.ntu.edu.tw, ching-pei@cs.wisc.edu*

Introduction

- Nowadays linear models are a mature technique for classification problems, but training on **large-scale** problems (e.g., document classification, click-through-rate prediction) is still challenging.
- For such scale (may be up to billions or more), **distributed training** can be useful, and **L1 regularization** is often adopted to reduce model size.
- Challenges:
  - Since $\|\boldsymbol{w}\|_1$ is **non-differentiable**, it is more difficult to develop efficient optimization algorithms.
  - Distributed training creates high **communication costs** between machines. Currently state-of-the-art methods on single machine, such as coordinate descent method or its variants, may not be suitable for distributed computation.
- Currently, **OWL-QN** [?] is the most commonly used method for distributed L1-regularized problems (e.g., it is the main linear classifier in **Spark MLlib**).
- In this work, we investigate why OWL-QN has been successful and whether we can develop a better distributed optimization method.

Optimization Problem

- Given training data $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^l$, $y_i \in \{-1, 1\}$, $\boldsymbol{x}_i \in \mathbb{R}^n$. We denote the data matrix by
$$X = \left[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l\right]^T \in \mathbb{R}^{l \times n}.$$
- We consider L1-regularized logistic regression (LR) for the following problem
$$\min_{\boldsymbol{w} \in \mathbb{R}^n} \quad f(\boldsymbol{w}) \equiv \|\boldsymbol{w}\|_1 + L(\boldsymbol{w}), \tag{1}$$
where
$$L(\boldsymbol{w}) \equiv C \sum\nolimits_{i=1}^{l} \log(1 + \exp(-y_i \boldsymbol{w}^T \boldsymbol{x}_i)).$$
- OWL-QN is an extension of L-BFGS [?], which is a quasi-Newton method for **smooth** optimization.
- We start with introducing Newton method for smooth problems. In each iteration, Newton method obtains the direction $\boldsymbol{d}$ by minimizing the **second-order approximation** of a smooth $\tilde{f}$:
$$\min_{\boldsymbol{d}} \quad \nabla \tilde{f}(\boldsymbol{w})^T \boldsymbol{d} + \frac{1}{2} \boldsymbol{d}^T \nabla^2 \tilde{f}(\boldsymbol{w}) \boldsymbol{d} \approx \tilde{f}(\boldsymbol{w} + \boldsymbol{d}) - \tilde{f}(\boldsymbol{w}), \tag{2}$$
which is equivalent to solving
$$\nabla^2 \tilde{f}(\boldsymbol{w}) \boldsymbol{d} = -\nabla \tilde{f}(\boldsymbol{w}) \quad \Rightarrow \quad \boldsymbol{d} = -(\nabla^2 \tilde{f}(\boldsymbol{w}))^{-1} \nabla \tilde{f}(\boldsymbol{w}).$$
- Since $\nabla^2 \tilde{f}(\boldsymbol{w})$ may be **expensive** to calculate, L-BFGS approximate the Newton direction by
$$\boldsymbol{d} = -B \nabla \tilde{f}(\boldsymbol{w}), \text{ where } B \approx (\nabla^2 \tilde{f}(\boldsymbol{w}))^{-1}.$$
- B is obtained by using steps and gradient differences of the **past m iterations** (usually m is small, e.g., 10 or 20)
$$\boldsymbol{w}_{k-s+1} - \boldsymbol{w}_{k-s}, \nabla \tilde{f}(\boldsymbol{w}_{k-s+1}) - \nabla \tilde{f}(\boldsymbol{w}_{k-s}), \quad s = 1, \ldots, m.$$

From Smooth to L1-regularized Problem

- For smooth problems,
$$\boldsymbol{w} \text{ is optimal} \quad \Leftrightarrow \quad \nabla f(\boldsymbol{w}) = \boldsymbol{0}.$$
- When not optimal, $\nabla f(\boldsymbol{w}) \neq \boldsymbol{0}$ is often used to generate the direction:
  - Gradient descent: $-\nabla f(\boldsymbol{w})$
  - Newton method: $-\nabla^2 f(\boldsymbol{w})^{-1} \nabla f(\boldsymbol{w})$
- For L1-regularized problems,
$$\boldsymbol{w} \text{ is optimal} \quad \Leftrightarrow \quad \nabla^{\mathrm{P}} f(\boldsymbol{w}) = \boldsymbol{0},$$
where $\nabla^{\mathrm{P}} f(\boldsymbol{w})$ is the **projected gradient** defined by
$$\nabla_j^{\mathrm{P}} f(\boldsymbol{w}) \equiv \begin{cases} \nabla_j L(\boldsymbol{w}) + 1 & w_j > 0, \text{ or } w_j = 0, \nabla_j L(\boldsymbol{w}) + 1 < 0, \\ \nabla_j L(\boldsymbol{w}) - 1 & w_j < 0, \text{ or } w_j = 0, \nabla_j L(\boldsymbol{w}) - 1 > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$
which indicates that w_j can move along the direction $-\nabla_j^{\mathrm{P}} f(\boldsymbol{w})$ to decrease the function value.
- We choose
$$A \equiv \{j \mid \nabla_j^{\mathrm{P}} f(\boldsymbol{w}) \neq 0\}$$
to be the **"active set,"** the components that are allowed to change.
- In each iteration, we obtain the active set A, and then find a direction on the orthant face where the projected gradient $\nabla^{\mathrm{P}} f(\boldsymbol{w})$ lies.

Existing Method: OWL-QN

- Now we aim to (approximately) minimize
$$\min_{\boldsymbol{d}_A} \quad f(\boldsymbol{w} + \begin{bmatrix} \boldsymbol{d}_A \\ \boldsymbol{0} \end{bmatrix}) = \|\boldsymbol{w} + \begin{bmatrix} \boldsymbol{d}_A \\ \boldsymbol{0} \end{bmatrix}\|_1 + L(\boldsymbol{w} + \begin{bmatrix} \boldsymbol{d}_A \\ \boldsymbol{0} \end{bmatrix}).$$
- We consider Newton method on A, and (2) becomes
$$\min_{\boldsymbol{d}_A} \quad \nabla_A^{\mathrm{P}} f(\boldsymbol{w})^T \boldsymbol{d}_A + \frac{1}{2} \boldsymbol{d}_A^T \nabla_{AA}^2 L(\boldsymbol{w}) \boldsymbol{d}_A, \tag{4}$$
which is equivalent to solving the linear system
$$\nabla_{AA}^2 L(\boldsymbol{w}) \boldsymbol{d}_A = -\nabla_A^{\mathrm{P}} f(\boldsymbol{w}). \tag{5}$$
- The solution to (5) is
$$-(\nabla_{AA}^2 L(\boldsymbol{w}))^{-1} \nabla_A^{\mathrm{P}} f(\boldsymbol{w}).$$
- As solving (5) is expensive, we follow the idea of L-BFGS to find
$$\bar{B} \approx (\nabla_{AA}^2 L(\boldsymbol{w}))^{-1} \text{ and calculate } \boldsymbol{d}_A = -\bar{B} \nabla_A^{\mathrm{P}} f(\boldsymbol{w}). \tag{6}$$
- Instead of (6), OWL-QN uses $B \approx \nabla^2 L(\boldsymbol{w})^{-1}$ and calculate
$$\boldsymbol{d}_A = -(B \nabla^{\mathrm{P}} f(\boldsymbol{w}))_A \approx -(\nabla^2 L(\boldsymbol{w})^{-1})_{AA} \nabla_A^{\mathrm{P}} f(\boldsymbol{w}).$$
- The main complexity of OWL-QN per iteration is only **2 data passes** (one **function** and one **projected gradient** evaluation).
- Note that Newton direction is much more expensive. It can cost up to
$$\mathcal{O}(n \times \text{ data passes}), \quad n = \#\text{features}.$$
- This explains why OWL-QN is practically viable.

Common-directions Method for L1-regularized Problems

- Instead of minimizing (4) over $\boldsymbol{d}_A \in \mathbb{R}^{|A|}$, we extend recent works [??] for smooth optimization by restricting the direction to be **a linear combination of only some vectors**. Then (4) becomes
$$\min_{\boldsymbol{t}} \quad \nabla_A^{\mathrm{P}} f(\boldsymbol{w})^T (P\boldsymbol{t})_A + \frac{1}{2} ((P\boldsymbol{t})_A)^T \nabla_{AA}^2 L(\boldsymbol{w}) (P\boldsymbol{t})_A,$$
where $P \in \mathbb{R}^{n \times m}$ contains m vectors as its columns and $\boldsymbol{t} \in \mathbb{R}^m$ is the coefficient vector corresponding to columns of P.
- Like OWL-QN, the columns of P are chosen to be past projected gradients or step difference
$$P = \left[\nabla^{\mathrm{P}} f(\boldsymbol{w}_k), \nabla^{\mathrm{P}} f(\boldsymbol{w}_{k-1}), \ldots, \boldsymbol{w}_k - \boldsymbol{w}_{k-1}, \boldsymbol{w}_{k-1} - \boldsymbol{w}_{k-2}, \ldots\right].$$
- The minimization problem is equivalent to solving the $m \times m$ linear system
$$(P_{A,:})^T \nabla_{AA}^2 L(\boldsymbol{w}) (P_{A,:}) \boldsymbol{t} = -(P_{A,:})^T \nabla_A^{\mathrm{P}} f(\boldsymbol{w}), \tag{7}$$
which can be done in $\mathcal{O}(m^3)$.
- Because of the special structure of linear classification, we have
$$\nabla^2 L(\boldsymbol{w}) = C X^T D_{\boldsymbol{w}} X, \tag{8}$$
where $D_{\boldsymbol{w}}$ is a diagonal matrix (details not shown here).
- By (8), we have
$$\nabla_{AA}^2 L(\boldsymbol{w}) = C X_{:,A}^T D_{\boldsymbol{w}} X_{:,A},$$
and thus in (7)
$$(P_{A,:})^T \nabla_{AA}^2 L(\boldsymbol{w}) (P_{A,:}) = C (X_{:,A} P_{A,:})^T D_{\boldsymbol{w}} (X_{:,A} P_{A,:}). \tag{9}$$
- However, obtaining $\boldsymbol{X_{:,A} P_{A,:}}$ in (9) costs m **data passes**, which is expensive.
- If the active sets in the past m iterations are similar (details not explained), then
$$\boldsymbol{X_{:,A} P_{A,:} \approx X P}.$$
- Therefore, we can approximate $X_{:,A} P_{A,:}$ by XP in (9), and hence (7) becomes
$$C (XP)^T D_{\boldsymbol{w}} (XP) \boldsymbol{t} = -(P_{A,:})^T \nabla_A^{\mathrm{P}} f(\boldsymbol{w}). \tag{10}$$
- Note that only one column of P is updated per iteration,
$$[\boldsymbol{p}_1 \cdots \boldsymbol{p}_m] \rightarrow [\boldsymbol{p}_2 \cdots \boldsymbol{p}_{m+1}],$$
hence it only takes **one data pass** to calculate $X\boldsymbol{p}_{m+1}$ and maintain XP:
$$X[\boldsymbol{p}_2 \cdots \boldsymbol{p}_{m+1}] = \big[\underbrace{X[\boldsymbol{p}_2 \cdots \boldsymbol{p}_m]}_{\text{existing}} \quad \underbrace{X\boldsymbol{p}_{m+1}}_{\text{new}}\big].$$
- Our proposed method using (10) (more details in the implementation below) is referred to as "L-Comm," which is an approximation of "L-Comm-Face" (using (7)).
- Comparisons on computational complexity per iteration:

  L-Comm: **3** data passes; L-Comm-Face: $\boldsymbol{2 + m}$ data passes; OWL-QN: **2** data passes.
- If L-Comm gets better directions, the number of iterations may be smaller.

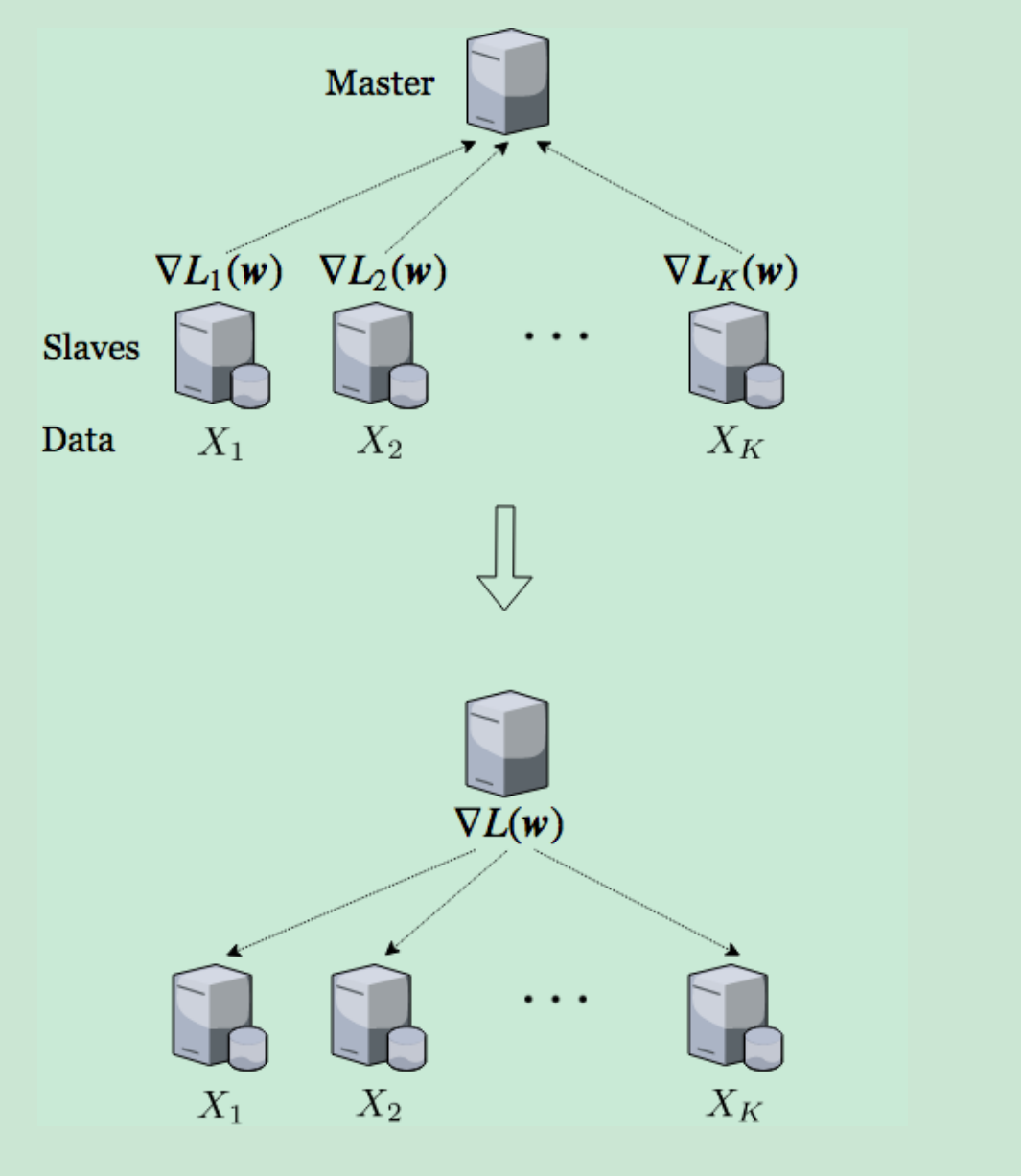
Implementation of our Proposed Method

```
1: Initialize $\boldsymbol{w} \leftarrow \boldsymbol{w}_0$
2: while not optimal do
3:     Compute $\nabla^{\mathrm{P}} f(\boldsymbol{w})$ by (3)
4:     Solve the linear system (10)
5:     Let the direction be $\boldsymbol{d} = P\boldsymbol{t}$
6:     Align $\boldsymbol{d}$ with $-\nabla^{\mathrm{P}} f(\boldsymbol{w})$
7:     Conduct line search on direction $\boldsymbol{d}$ and update $\boldsymbol{w}$
8:     Update $P$ and $XP$
```

Distributed Implementation

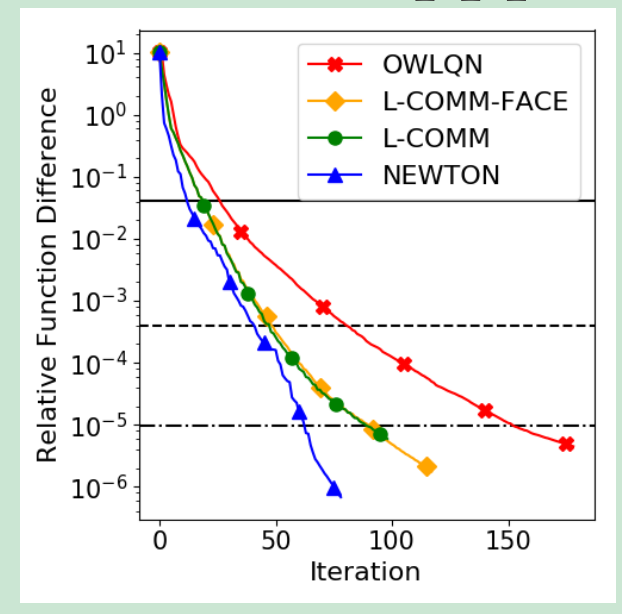
- We consider a master-slave framework, and **Open MPI** [?] is used for communication between machines.
- The data set X is split across K machines in an **instance-wise manner**: J_r, $r = 1, \ldots, K$ partition $\{1, \ldots, l\}$, and the r-th machine stores $X_r \equiv \{(y_i, \boldsymbol{x}_i)\}_{i \in J_r}$.
- The model $\boldsymbol{w}$ and the projected gradient $\nabla^{\mathrm{P}} f(\boldsymbol{w})$ are made available to all K machines.

The main communication costs occur in the following two places:
1. $$\nabla L(\boldsymbol{w}) = \bigoplus_{r=1}^{K} \nabla L_r(\boldsymbol{w}), \quad L_r(\boldsymbol{w}) \equiv C \sum_{i \in J_r} \log(1 + \exp(-y_i \boldsymbol{w}^T \boldsymbol{x}_i)),$$
   where $\bigoplus$ is the ***allreduce*** operation that sums up values from machines and broadcasts the result back (see Figure 1 for an illustration). The communication cost is $\mathcal{O}(n)$.
2. After obtaining $\boldsymbol{d}_{J_r}$ at machine r, we need an ***allgather*** operation to make the whole direction $\boldsymbol{d}$ available at every machine. The communication cost is $\mathcal{O}(n/K)$.

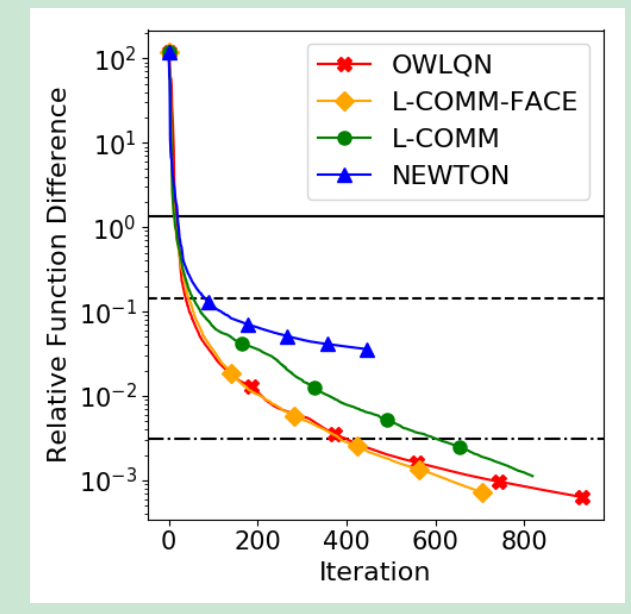

Master
$\nabla L_1(\boldsymbol{w})$
$\nabla L_2(\boldsymbol{w})$
$\nabla L_K(\boldsymbol{w})$
Slaves
Data
X_1
X_2
X_K
$\nabla L(\boldsymbol{w})$
Figure 1: Gradient calculation

Comparisons on the Number of Iterations

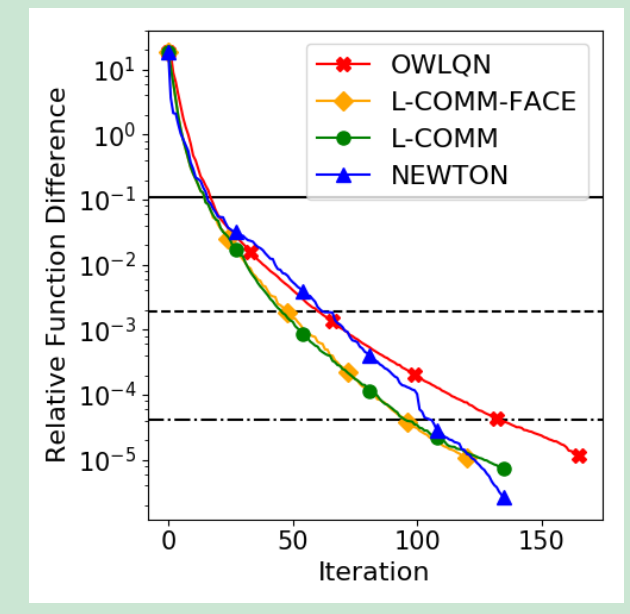
- For OWL-QN, L-COMM, and L-COMM-FACE, we all use information from the **past 10 iterations**. For NEWTON,we run **50 CG steps** at each iteration.
- The figures are plotted with relative function difference versus #iterations. The three horizontal lines indicate when OWL-QN meets the stoppping conditions with different criteria.



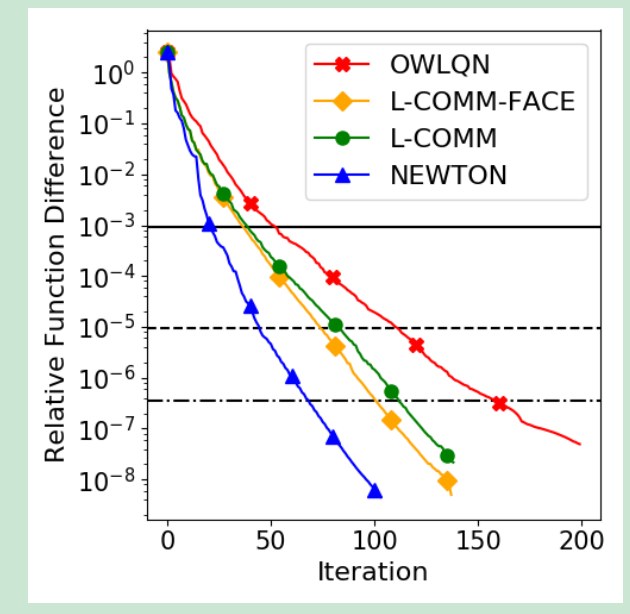
rcv1



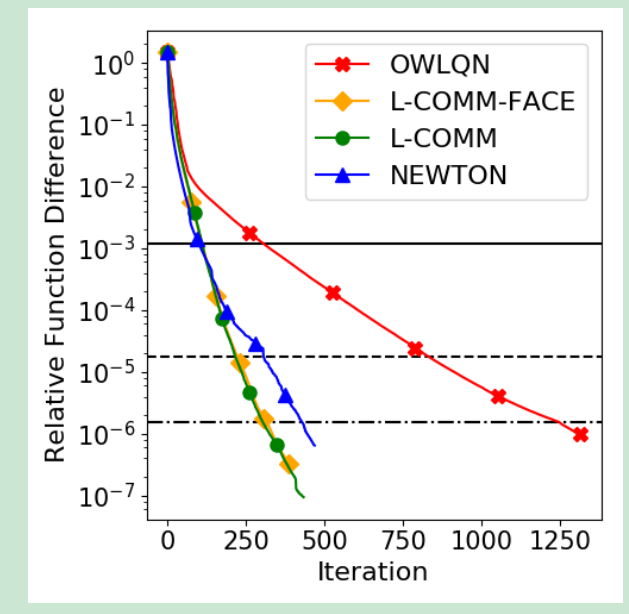
news20



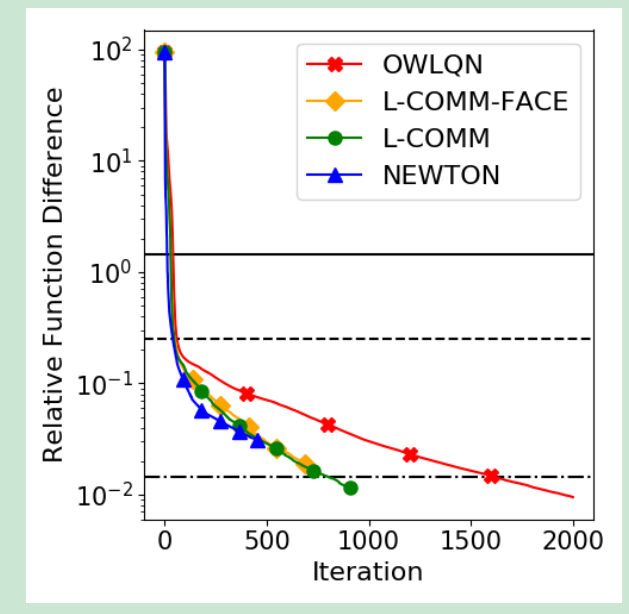
real-sim



yahoojp



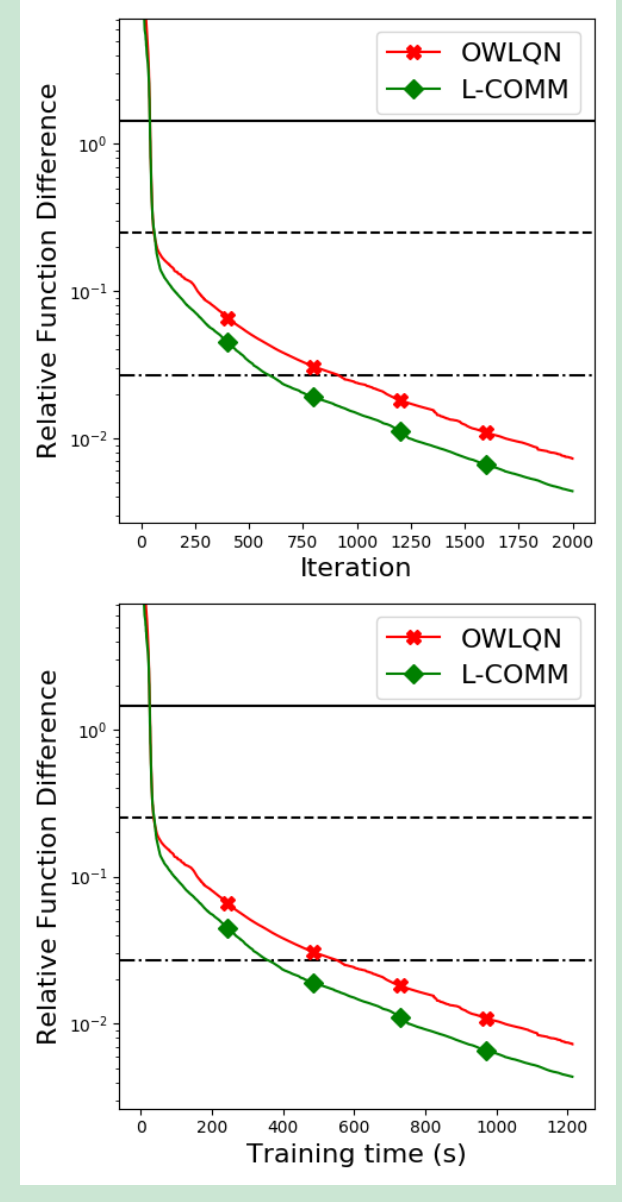
yahookr



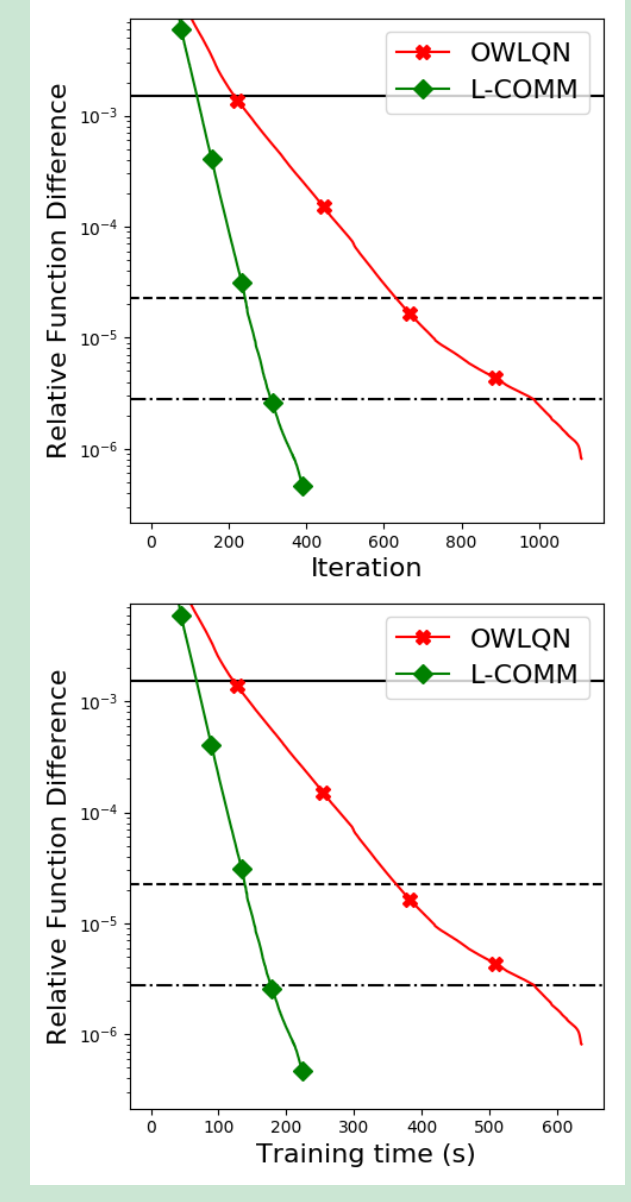
url

Comparisons on the Number of Iterations and Timing in Distributed Environments

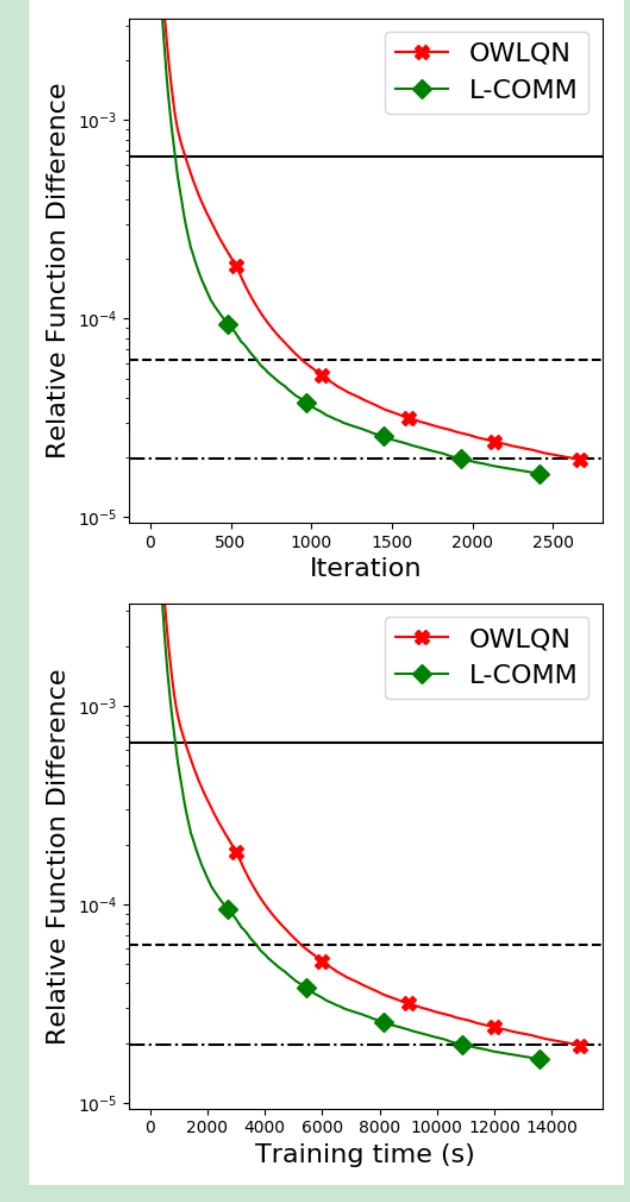
- We compare OWL-QN and L-COMM with **32 machines** on AWS. The figures are plotted with relative function difference versus **#iterations (upper)** and **running time in seconds (lower)**. More larger data sets are considered in this experiment.



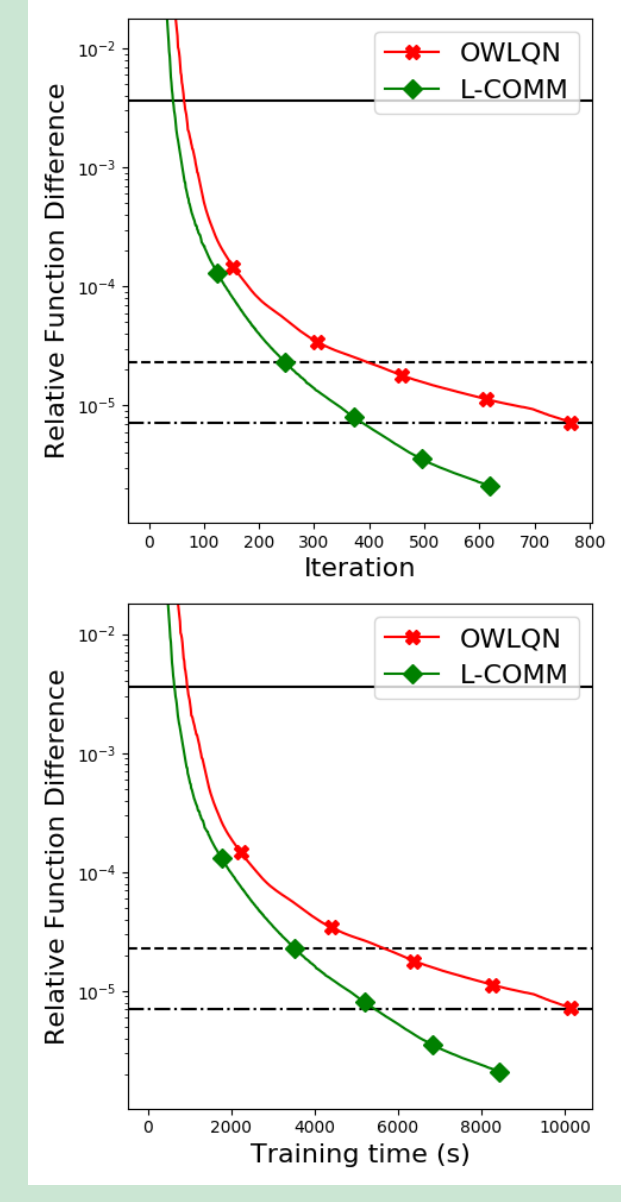
url



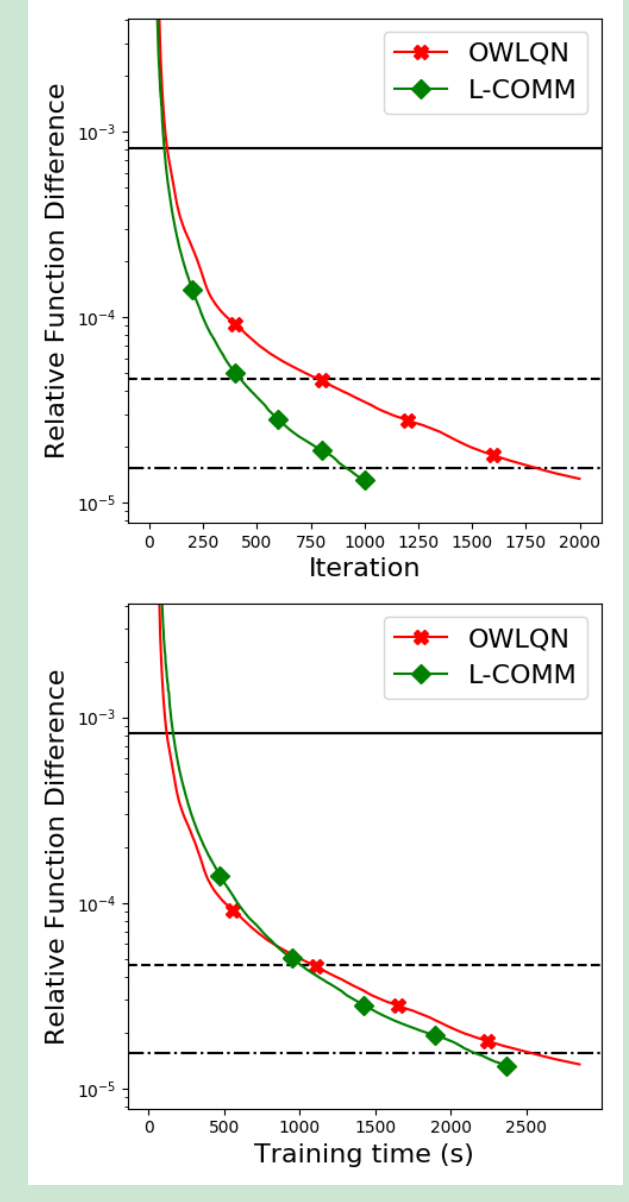
yahookr



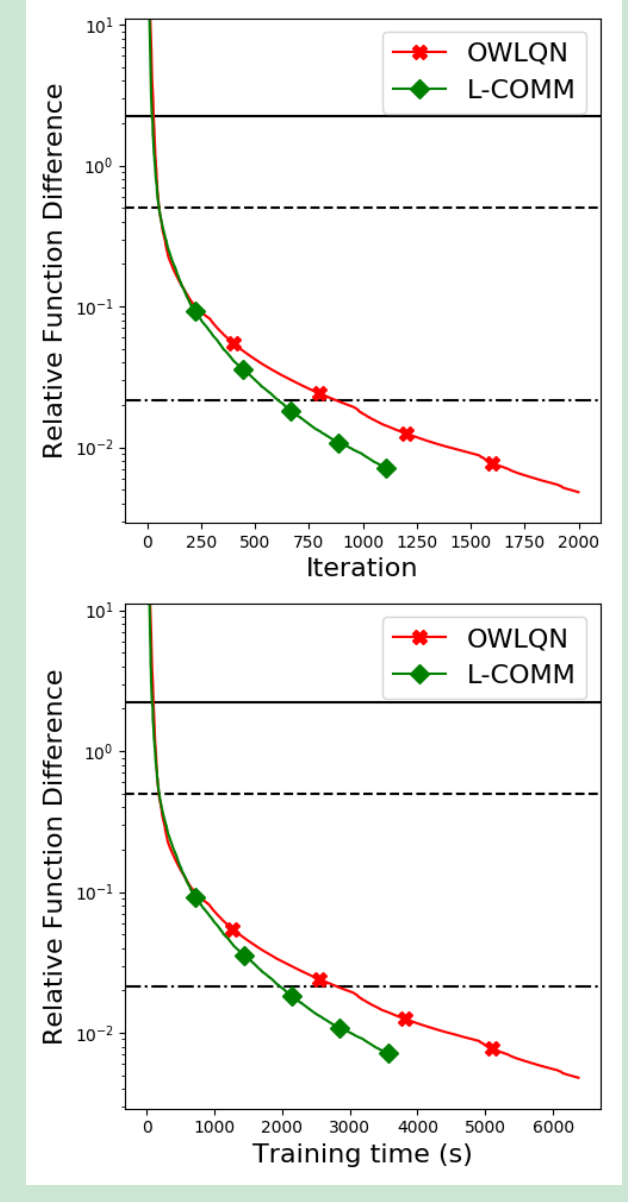
KDD2010-b



kdd2012



criteo



webspam

Conclusions

- In this work, we identify possible issues of OWL-QN, and present a method that improves the search directions. Through experiments, our method is shown to be more efficient than OWL-QN in distributed environments.
- The code is available in **distributed LIBLINEAR**.
  (https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/distributed-liblinear/)