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

Wei-Lin Chiang
Department of Computer Science
National Taiwan University

Joint work with Yu-Sheng Li, Ching-Pei Lee, and Chih-Jen Lin

Linear classification

- Nowadays linear models has become a mature technique for classification problems
- However, training large-scale problems (e.g., document classification, click-through-rate prediction) is still challenging

Linear classification

- Nowadays linear models has become a mature technique for classification problems
- However, training 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

Outline

Outline

L1-regularized logistic regression (LR)

- Given l: #instances, n: #features
- Training data $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^{l}$, $y_i \in \{-1, 1\}$, $\boldsymbol{x}_i \in \mathbb{R}^n$
- We consider L1-regularized LR

$$\min_{\boldsymbol{w} \in \mathbb{R}^n} \quad f(\boldsymbol{w}),$$

L1-regularized logistic regression (LR)

- Given l: #instances, n: #features
- Training data $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^{l}$, $y_i \in \{-1, 1\}$, $\boldsymbol{x}_i \in \mathbb{R}^n$
- We consider L1-regularized LR

$$\min_{\boldsymbol{w} \in \mathbb{R}^n} \quad f(\boldsymbol{w}),$$

where

$$f(\boldsymbol{w}) \equiv \|\boldsymbol{w}\|_1 + L(\boldsymbol{w}),$$

$$L(\boldsymbol{w}) \equiv C \sum_{i=1}^{l} \log(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}) \text{ is } C \times \text{losses}$$

L1-regularized logistic regression (LR)

- Given l: #instances, n: #features
- Training data $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^{l}$, $y_i \in \{-1, 1\}$, $\boldsymbol{x}_i \in \mathbb{R}^n$
- We consider L1-regularized LR

$$\min_{\boldsymbol{w} \in \mathbb{R}^n} \quad f(\boldsymbol{w}),$$

where

$$f(\boldsymbol{w}) \equiv \|\boldsymbol{w}\|_1 + L(\boldsymbol{w}),$$

$$L(\boldsymbol{w}) \equiv C \sum_{i=1}^{l} \log(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}) \text{ is } C \times \text{losses}$$

- $\|\boldsymbol{w}\|_1$: avoid overfitting and achieve sparse models
- C: regularization parameter

An example of sparsity through L1-norm

We run LIBLINEAR (Fan et al., 2008), a popular linear-classification package, on a document classification data, **news20**

- L2-regularized LR

  `$./train -s 0 -c 1024 news20.train`

  Nonzeros in model $\boldsymbol{w}$: 93% (1258732/1355191)

An example of sparsity through L1-norm

We run LIBLINEAR (Fan et al., 2008), a popular linear-classification package, on a document classification data, **news20**

- L2-regularized LR

  `$./train -s 0 -c 1024 news20.train`

  Nonzeros in model $\boldsymbol{w}$: 93% (1258732/1355191)

- L1-regularized LR

  `$./train -s 6 -c 1024 news20.train`

  Nonzeros in model $\boldsymbol{w}$: 3.6% (48596/1355191)

- Similar accuracy on test data (96.7% vs. 96.6%)

Challenges

- $\|\boldsymbol{w}\|_1$ is not differentiable, so it is more difficult to develop efficient optimization algorithms
- For distributed training scenario, some state-of-the-art methods on single machine may not be easily deployed

Current status

- Currently, OWL-QN (Andrew and Gao, 2007), an extension of a quasi-Newton method (L-BFGS, Liu and Nocedal 1989), is the most commonly used distributed method
- For example, OWL-QN is the main linear classifier in Spark MLlib (Meng et al., 2016)
- In this work, we investigate why OWL-QN has been successful and whether we can develop a better distributed training method

Outline

1. Introduction
2. Our proposed method
3. Experiments and conclusions

Difference between smooth problems and L1 problems

- For smooth problem f, we have

$$\boldsymbol{w} \text{ is optimal} \Leftrightarrow \nabla f(\boldsymbol{w}) = \mathbf{0}$$

- For the L1-regularized problem,

$$\boldsymbol{w} \text{ is optimal} \Leftrightarrow \nabla^{\mathrm{P}} f(\boldsymbol{w}) = \mathbf{0},$$

where $\nabla^{\mathrm{P}} f(\boldsymbol{w})$ is the projected gradient

Difference between smooth problems and L1 problems

- For smooth problem f, we have

$$\boldsymbol{w} \text{ is optimal} \Leftrightarrow \nabla f(\boldsymbol{w}) = \mathbf{0}$$

- For the L1-regularized problem,

$$\boldsymbol{w} \text{ is optimal} \Leftrightarrow \nabla^{\mathrm{P}} f(\boldsymbol{w}) = \mathbf{0},$$

  where $\nabla^{\mathrm{P}} f(\boldsymbol{w})$ is the projected gradient
- When not optimal, $\nabla^{\mathrm{P}} f(\boldsymbol{w})$ is used to generate a direction (e.g., projected gradient descent method)

The active set

- Following the projected gradient direction, the set defined by

$$A \equiv \{j \mid \nabla_j^{\mathrm{P}} f(\boldsymbol{w}) \neq 0\} \tag{1}$$

can be seen as the components that are allowed to change

- That is, we want to obtain a direction under the subspace defined by A

The active set

- Following the projected gradient direction, the set defined by

$$A \equiv \{j \mid \nabla_j^{\mathrm{P}} f(\boldsymbol{w}) \neq 0\} \tag{1}$$

  can be seen as the components that are allowed to change
- That is, we want to obtain a direction under the subspace defined by A
- The optimization procedure becomes to iteratively
  - obtain the active set A, and
  - find a direction on the subspace defined by A

Common-directions method

- To obtain a better direction, we consider the second-order Taylor approximation on A

$$f(\boldsymbol{w}+\begin{bmatrix}\boldsymbol{d}_A\\ \boldsymbol{0}\end{bmatrix})-f(\boldsymbol{w}) \approx \nabla^{P}_{A} f(\boldsymbol{w})^T \boldsymbol{d}_A + \frac{1}{2}\boldsymbol{d}_A^T \nabla^2_{AA} L(\boldsymbol{w}) \boldsymbol{d}_A \tag{2}$$

- It is expensive to precisely solve (2)

Common-directions method

- To obtain a better direction, we consider the second-order Taylor approximation on A

$$
\begin{aligned}
& f(\boldsymbol{w} + \begin{bmatrix} \boldsymbol{d}_A \\ \boldsymbol{0} \end{bmatrix}) - f(\boldsymbol{w}) \\
\approx & \nabla_A^{\mathrm{P}} f(\boldsymbol{w})^T \boldsymbol{d}_A + \frac{1}{2} \boldsymbol{d}_A^T \nabla_{AA}^2 L(\boldsymbol{w}) \boldsymbol{d}_A
\end{aligned} \tag{2}
$$

- It is expensive to precisely solve (2)
- We extend a recent work (Lee et al., 2017) for smooth optimization by restricting the direction to be a linear combination of only some vectors

Common-directions method (Cont'd)

- The second-order approximation becomes

$$\min_{\boldsymbol{t}} \nabla^{\mathrm{P}}_{A} f(\boldsymbol{w})^T (P\boldsymbol{t})_A + \frac{1}{2}((P\boldsymbol{t})_A)^T \nabla^2_{AA} L(\boldsymbol{w})(P\boldsymbol{t})_A$$

  $P \in \mathbb{R}^{n \times m}$ is the matrix containing m vectors as its columns (typically $m < 30$)
- Since $\boldsymbol{d} = P\boldsymbol{t}$, we now minimize over $\boldsymbol{t}$, coefficients corresponding to P's columns

Common-directions method (Cont'd)

- The second-order approximation becomes

$$\min_{\boldsymbol{t}} \nabla^{\mathrm{P}}_A f(\boldsymbol{w})^T (P\boldsymbol{t})_A + \frac{1}{2}((P\boldsymbol{t})_A)^T \nabla^2_{AA} L(\boldsymbol{w})(P\boldsymbol{t})_A$$

  $P \in \mathbb{R}^{n \times m}$ is the matrix containing m vectors as its columns (typically $m < 30$)

- Since $\boldsymbol{d} = P\boldsymbol{t}$, we now minimize over $\boldsymbol{t}$, coefficients corresponding to P's columns
- Columns of P can be chosen as some past projected gradients and steps

$$P = [\nabla^{\mathrm{P}} f(\boldsymbol{w}_k), \nabla^{\mathrm{P}} f(\boldsymbol{w}_{k-1}), \ldots, \\ \boldsymbol{w}_k - \boldsymbol{w}_{k-1}, \boldsymbol{w}_{k-1} - \boldsymbol{w}_{k-2}, \ldots]$$

Common-directions method (Cont'd)

- To obtain $\boldsymbol{t}$, it is equivalent to solving the linear system

$$(P_{A,:})^T \nabla^2_{AA} L(\boldsymbol{w})(P_{A,:})\boldsymbol{t} = -(P_{A,:})^T \nabla^{\mathrm{P}}_{A} f(\boldsymbol{w}) \quad (3)$$

Common-directions method (Cont'd)

- To obtain $\boldsymbol{t}$, it is equivalent to solving the linear system

$$(P_{A,:})^T \nabla^2_{AA} L(\boldsymbol{w})(P_{A,:})\boldsymbol{t} = -(P_{A,:})^T \nabla^{\mathrm{P}}_{A} f(\boldsymbol{w}) \quad (3)$$

- Note that we have $P \in \mathbb{R}^{n \times m}$, where

$$n : \#\text{features} \gg m : \#\text{vectors}$$

- If

$$(P_{A,:})^T \nabla^2_{AA} L(\boldsymbol{w})(P_{A,:}) \in \mathbb{R}^{m \times m} \quad (4)$$

is available, solving (3) is extremely cheap $\mathcal{O}(m^3)$

Common-directions method (Cont'd)

- But the computation cost of (4) is indeed

$$m \times \mathcal{O}(\#\text{nnz of } X),$$

where $X = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_l^T \end{bmatrix}$ is the data matrix

#nnz: number of non-zeros, typically $m < 30$

- It is very expensive. In contrast, it only takes $\mathcal{O}(\#\text{nnz of } X)$ to obtain $\nabla^{\text{P}} f(\boldsymbol{w})$

Common-directions method (Cont'd)

- We want to approximate the left-hand-side matrix

$$(P_{A,:})^T \nabla^2_{AA} L(\boldsymbol{w})(P_{A,:})$$

- Roughly speaking, the idea is to check the active sets during the past m iterations
- Then the cost of calculating (4) can be significantly reduced from

$$\mathcal{O}(m \times \#\text{nnz of } X)$$

to

$$\mathcal{O}(\#\text{nnz of } X)$$

L-Comm: Limited-memory common directions method

- Our method is referred to as "L-Comm" (limited-memory common directions)
- Limited-memory: we use "m" vectors for the direction
- Common directions: we consider the direction to be linear combination of columns of P

Complexity

- In distributed environments,

$$\text{cost per iteration} = \text{computational cost} \\ + \text{communicational cost}$$

- Computational complexity per iteration:

OWL-QN: $2 \times \mathcal{O}(\#\text{nnz of } X)$
L-Comm: $3 \times \mathcal{O}(\#\text{nnz of } X)$

- If L-Comm gets a better direction, the number of iterations may be smaller, which leads to less total training time

Complexity (Cont'd)

- Communicational complexity per iteration:

  OWL-QN: $\mathcal{O}(n)$

  L-Comm: $\mathcal{O}(n)$

- $\mathcal{O}(n)$ comes from, for example, aggregating vectors from all nodes to form $\nabla^{\mathrm{P}} f(\boldsymbol{w}) \in \mathbb{R}^n$
- Their communication costs are similar

Complexity (Cont'd)

- Communicational complexity per iteration:

  OWL-QN: $\mathcal{O}(n)$

  L-Comm: $\mathcal{O}(n)$

- $\mathcal{O}(n)$ comes from, for example, aggregating vectors from all nodes to form $\nabla^{\mathrm{P}} f(\boldsymbol{w}) \in \mathbb{R}^n$
- Their communication costs are similar
- In summary, L-Comm may be faster because of
  - same communicational cost per iteration, and
  - fewer iterations by better directions

Outline

Data sets

We show six data sets, some with large #instances and #features. All of them are sparse matrices (containing lots of zeros).

Data set	#instances	#features	sparsity
yahookr	460,554	3,052,939	0.0001113
avazu-site	25,832,830	999,962	0.0000150
kdd2010-b	19,264,097	29,890,096	0.0000010
criteo	45,840,617	1,000,000	0.0000390
kdd2012	149,639,105	54,686,452	0.0000002
webspam	350,000	16,609,143	0.0002244

Experimental settings

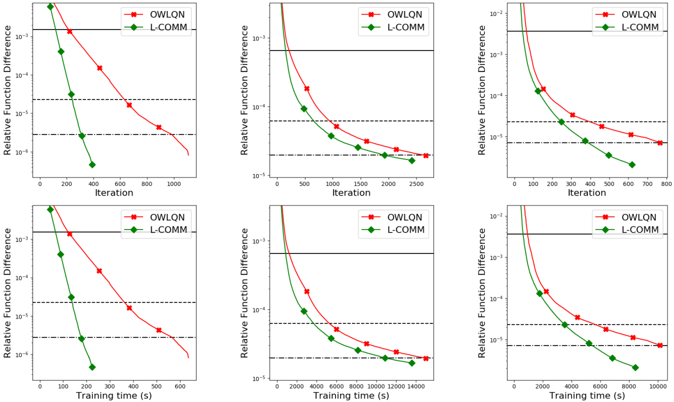
- We compare OWL-QN and L-Comm by using 32 machines on AWS
- Open MPI (Gabriel et al., 2004) is used for the communication between machines

Experimental settings

- We compare OWL-QN and L-Comm by using 32 machines on AWS
- Open MPI (Gabriel et al., 2004) is used for the communication between machines
- Note that for machine learning, it is not necessary to solve the optimization problem too accurately
- So we also show some horizontal lines to indicate different stopping conditions

Results

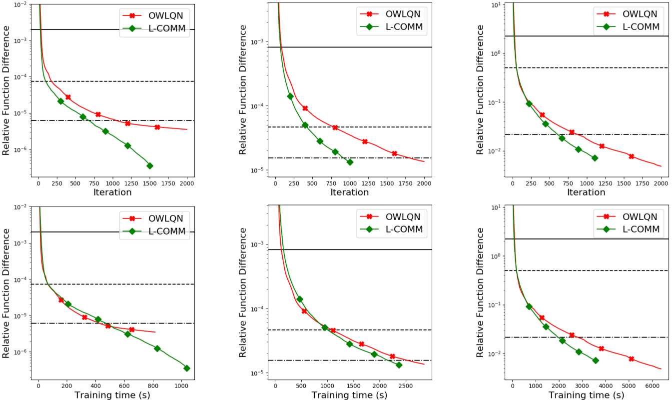
y-axis: relative distance to the optimal value (log-scale)
x-axis: #iterations (upper), training time (lower)



(a) yahookr (b) kdd2010-b (c) kdd2012

Results (Cont'd)

y-axis: relative distance to the optimal value (log-scale)
x-axis: #iterations (upper), training time (lower)



(a) avazu-site (b) criteo (c) webspam

Observations

- L-Comm always needs fewer iterations
- Although being more expensive per iteration, L-Comm is generally faster due to a smaller number of iterations
- L-Comm is not significantly superior to OWL-QN on criteo and avazu-site. Their #features are relatively small, so computational cost is more dominant

Conclusions

- In this work, we study L1-regularized linear classification in distributed environments
- By improving the search direction, our proposed method is shown to be faster than OWL-QN in distributed environments
- The code is available in distributed LIBLINEAR (`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/distributed-liblinear/`)