Distributed Box-Constrained Quadratic Optimization for Dual Linear SVM

Lee, Ching-pei
University of Illinois at Urbana-Champaign

Joint work with Dan Roth
ICML 2015
Outline

Introduction

Algorithm

Experiments

Discussions and Conclusions
Distributed Linear Classification

- Distributed linear classification is the essential technique for dealing with data larger than the capacity of a single machine.
- In distributed training, computation and data I/O time are usually short.
- But **communication and synchronization costs** become the bottleneck.
- We thus need to consider methods requiring fewer rounds of communication. That is, methods that **converge faster in terms of iterations.**
Linear SVM

- We solve dual problems.
- We use the dual problem of linear support vector machines (SVM) by Boser et al. (1992); Vapnik (1995) as an example for illustration.
- The paper also covers squared-hinge loss (L2-loss) SVM (skipped).
- Can also be easily extended to other problems like linear regression, multi-class classification, structured SVM (Lee et al., 2015).
Why Dual?
Why Dual?

- Because primal methods don’t have convergence guarantee

Why Dual?

- Single machine experience: dual methods are sometimes faster, especially when \#features > \#instances.
- Some problems are hard to directly solve the primal.
Dual Problem

- Given the binary-labeled training instances \( \{(x_i, y_i) \in \mathbb{R}^n \times \{-1, 1\}\}^{\ell}_{i=1} \), linear SVM dual problem solves
  \[
  \min_{\alpha} \quad f(\alpha) \equiv \frac{1}{2} \alpha^T Q \alpha - e^T \alpha
  \]
  subject to \( 0 \leq \alpha \leq C e \),

  where \( e \in \mathbb{R}^n \) is the vector of ones, \( Q_{i,j} = y_i y_j x_i^T x_j \), and \( C > 0 \) is a parameter specified by user.

- Convert the dual solution to the primal model \( \mathbf{w} \) by \( \mathbf{w} = \sum_{i=1}^{\ell} y_i x_i \alpha_i \).
We solve SVM dual in a distributed environment, where the training instances are disjointly stored among $K$ machines.

- $X_i$ are data matrices: each row is an instance. $Y_i$ are diagonal matrices representing the labels.
- Dual variables: $\alpha = (\alpha_1, \ldots, \alpha_K)$.
- Then $w = \sum_{i=1}^\ell y_i x_i \alpha_i = \sum_{k=1}^K Y_k X_k \alpha_k$
Challenges of Distributed Dual SVM

- Each node only gets partial information of the data.
- To avoid frequent communication, can only use block-diagonal approximation of $Q$ because data are not available.
- Existing dual methods provide only sub-linear convergence rates.
Our Contributions

▶ We first establish global linear convergence, i.e., $O(\log(1/\epsilon))$ iteration complexity, for distributedly solving non-strongly-convex dual problems like the SVM dual.

▶ The key factor is a communication-efficient line search method to better reduce function values under the same computation and communication cost.

▶ As a result, practical training speed is significantly faster than existing distributed dual methods.
Outline

Introduction

Algorithm

Experiments

Discussions and Conclusions
Algorithm Overview

Let the iterate at iteration $t$ be $\alpha^t$, all iterative algorithms update it by

$$\alpha^{t+1} = \alpha^t + \eta_t \Delta \alpha^t,$$

$\Delta \alpha^t$ is the direction and $\eta_t$ is the step size.

Line search for $\eta_t$ may be expensive, so existing works all use fixed step sizes.

We propose efficient methods for conducting line search.
Algorithm Overview (Cont’d)

▷ First, we obtain $\Delta \alpha^t$ by constructing independent sub-problems that can be solved distributedly without communication.

▷ Machine $k$ computes corresponding $\Delta \alpha^t_k$ only.

▷ Then conduct an $O(n)$ communication to make

$$\Delta w^t = \sum_{k=1}^{K} Y_k X_k \Delta \alpha^t_k$$

available to all machines.

▷ An $O(1)$ communication is then conducted to let each machine obtain the same step size $\eta_t$. 
1. Given $\alpha^t$ and the corresponding $w^t$. Each machine has $\alpha_k^t$ and the whole $w^t$.

2. Each machine solves local sub-problem in parallel to obtain $\Delta \alpha^t$.

3. Gather $\Delta w^t = \sum_{k=1}^{K} Y_k X_k \Delta \alpha_k^t$ from local results and broadcast. ($O(n)$ communication)

4. Compute the step size $\eta_t$ using $\alpha^t$, $\Delta \alpha^t$, $w^t$, and $\Delta w^t$ (Key factor, $O(1)$ communication)
   
   4.1 Option 1: Armijo line search (omitted)
   
   4.2 Option 2: exact line search

5. $w^{t+1} = w^t + \eta_t \Delta w^t$, $\alpha_k^{t+1} = \alpha_k^t + \eta_t \Delta \alpha_k^t$.  

---

Lee, Ching-pei and Dan Roth, Distributed Box-constrained training for dual SVM 13/30
1. Given $\alpha^t$ and the corresponding $w^t$. Each machine has $\alpha_k^t$ and the whole $w^t$.

2. Each machine solves local sub-problem in parallel to obtain $\Delta \alpha^t$.

3. Gather $\Delta w^t = \sum_{k=1}^K Y_k X_k \Delta \alpha_k^t$ from local results and broadcast. ($O(n)$ communication)

4. Compute the step size $\eta_t$ using $\alpha^t$, $\Delta \alpha^t$, $w^t$, and $\Delta w^t$ (Key factor, $O(1)$ communication)
   4.1 Option 1: Armijo line search (omitted)
   4.2 Option 2: exact line search

5. $w^{t+1} = w^t + \eta_t \Delta w^t$, $\alpha_k^{t+1} = \alpha_k^t + \eta_t \Delta \alpha_k^t$. 
Update Direction

- Obtain $\Delta \alpha^t$ by solving a quadratic problem.

$$
\Delta \alpha^t = \arg \min_{d: 0 \leq \alpha^t + d \leq C e} \nabla f(\alpha^t)^T d + \frac{1}{2} d^T H d,
$$

where $H$ is some approximation of $Q$.

- Block-diagonal $\Rightarrow$ can be solved locally.

- A simple choice: $H = \tilde{Q} + \epsilon I$ with any $\epsilon > 0$,

$$
\tilde{Q}_{i,j} = \begin{cases} 
Q_{i,j} & \text{if } i, j \text{ are in the same partition.} \\
0 & \text{otherwise.}
\end{cases}
$$

- But can use any positive definite symmetric matrix and still have linear convergence.
1. Given $\alpha^t$ and the corresponding $w^t$. Each machine has $\alpha^t_k$ and the whole $w^t$.

2. Each machine solves local sub-problem in parallel to obtain $\Delta \alpha^t$.

3. Gather $\Delta w^t = \sum_{k=1}^{K} Y_kX_k\Delta \alpha^t_k$ from local results and broadcast. ($O(n)$ communication)

4. Compute the step size $\eta_t$ using $\alpha^t$, $\Delta \alpha^t$, $w^t$, and $\Delta w^t$ (Key factor, $O(1)$ communication)
   
4.1 Option 1: Armijo line search (omitted)

4.2 Option 2: exact line search

5. $w^{t+1} = w^t + \eta_t \Delta w^t$, $\alpha_k^{t+1} = \alpha_k^t + \eta_t \Delta \alpha_k^t$. 

---

Lee, Ching-pei and Dan Roth, Distributed Box-constrained training for dual SVM
1. Given $\alpha^t$ and the corresponding $w^t$. Each machine has $\alpha^t_k$ and the whole $w^t$.

2. Each machine solves local sub-problem in parallel to obtain $\Delta \alpha^t$.

3. Gather $\Delta w^t = \sum_{k=1}^{K} Y_k X_k \Delta \alpha^t_k$ from local results and broadcast. ($O(n)$ communication)

4. Compute the step size $\eta_t$ using $\alpha^t$, $\Delta \alpha^t$, $w^t$, and $\Delta w^t$ (Key factor, $O(1)$ communication)

   4.1 Option 1: Armijo line search (omitted)

   4.2 Option 2: exact line search

5. $w^{t+1} = w^t + \eta_t \Delta w^t$, $\alpha^{t+1}_k = \alpha^t_k + \eta_t \Delta \alpha^t_k$. 
1. Given $\alpha^t$ and the corresponding $w^t$. Each machine has $\alpha^t_k$ and the whole $w^t$.

2. Each machine solves local sub-problem in parallel to obtain $\Delta \alpha^t$.

3. Gather $\Delta w^t = \sum_{k=1}^{K} Y_k X_k \Delta \alpha^t_k$ from local results and broadcast. ($O(n)$ communication)

4. Compute the step size $\eta_t$ using $\alpha^t$, $\Delta \alpha^t$, $w^t$, and $\Delta w^t$ (Key factor, $O(1)$ communication)

   4.1 Option 1: Armijo line search (omitted)

   4.2 Option 2: exact line search

5. $w^{t+1} = w^t + \eta_t \Delta w^t$, $\alpha^{t+1}_k = \alpha^t_k + \eta_t \Delta \alpha^t_k$. 
Exact Line Search

- Observe $f(\alpha^t + \eta_t \Delta \alpha^t)$ is a convex quadratic function of $\eta_t$, so we can simply solve

$$\frac{\partial f(\alpha + \eta_t \Delta \alpha)}{\partial \eta_t} = 0$$

$$(\Delta \alpha_t)^T \Delta \alpha_t$$ can be obtained because $\Delta w_t$ is available.
Exact Line Search

- Observe \( f(\alpha^t + \eta_t \Delta \alpha^t) \) is a convex quadratic function of \( \eta_t \), so we can simply solve

\[
\frac{\partial f(\alpha + \eta_t \Delta \alpha)}{\partial \eta_t} = 0 \quad \Rightarrow \quad \eta^*_t = \frac{-\nabla f(\alpha^t)^T \Delta \alpha^t}{(\Delta \alpha^t)^T Q \Delta \alpha^t}.
\]
Exact Line Search

- Observe $f(\alpha^t + \eta_t \Delta \alpha^t)$ is a convex quadratic function of $\eta_t$, so we can simply solve

$$\frac{\partial f(\alpha + \eta_t \Delta \alpha)}{\partial \eta_t} = 0 \Rightarrow \eta_t^* = \frac{-\nabla f(\alpha^t)^T \Delta \alpha^t}{(\Delta \alpha^t)^T Q \Delta \alpha^t}.$$

- $(\Delta \alpha^t)^T Q \Delta \alpha^t = (\Delta w^t)^T \Delta w^t$ can be obtained because $\Delta w^t$ is available.

- To ensure $0 \leq \alpha^t + \eta_t \Delta \alpha^t \leq C e$, take

$$\eta_t = \min\{\eta^*, \max\{\eta \mid 0 \leq \alpha^t + \eta \Delta \alpha^t \leq C e\}\}.$$
Convergence

Theorem

*Our algorithm with either line search approach has global linear convergence.* Therefore this approach requires \( O(\log(1/\epsilon)) \) iterations to obtain an \( \epsilon \)-accurate solution.
Pocket Approach

- Dual solvers do not guarantee decreasing primal objective
- But we want $w$ that minimizes the primal problem
- Maintain $w^k$ that has the smallest primal objective as the current output model
Outline

Introduction

Algorithm

Experiments

Discussions and Conclusions
## Solvers

<table>
<thead>
<tr>
<th>Method</th>
<th>$H$</th>
<th>$\eta_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSVM-AVE (Pechyony et al., 2011)</td>
<td>$\tilde{Q}$</td>
<td>$1/K$</td>
</tr>
<tr>
<td>Jaggi et al. (2014)</td>
<td>$\tilde{Q}$</td>
<td>$1/K$</td>
</tr>
<tr>
<td>DisDCA (Yang, 2013), practical variant</td>
<td>$K\tilde{Q}$</td>
<td>1</td>
</tr>
<tr>
<td>Ma et al. (2015)</td>
<td>$K\tilde{Q}^*$</td>
<td>1</td>
</tr>
</tbody>
</table>

*:Other choices require solving an eigenvalue problem

- $O(1/\epsilon)$ convergence.
- Distributed Newton method (Zhuang et al., 2015): solves the primal, requires differentiability. Compare in L2-loss SVM.
- **BQO-E**: ours: exact line search ($H = \tilde{Q} + \epsilon I$)
- **BQO-A**: Armijo line search ($H = \tilde{Q} + \epsilon I$)

---

Lee, Ching-pei and Dan Roth, Distributed Box-constrained training for dual SVM
### Data Sets and Experiment Setting

<table>
<thead>
<tr>
<th>Data set</th>
<th>#instances</th>
<th>#features</th>
<th>#nonzeros</th>
</tr>
</thead>
<tbody>
<tr>
<td>webspam</td>
<td>280,000</td>
<td>16,609,143</td>
<td>1,044,393,506</td>
</tr>
<tr>
<td>url</td>
<td>1,916,904</td>
<td>3,231,961</td>
<td>221,663,296</td>
</tr>
<tr>
<td>epsilon</td>
<td>400,000</td>
<td>2,000</td>
<td>800,000,000</td>
</tr>
</tbody>
</table>

- 16 machines in a local cluster.
- $C = 1$.
- All methods implemented in MPI/C++.
- Every dual solver: one pass through data (sample without replacement) by dual coordinate descent (Hsieh et al., 2008) and then communication.
- Pocket approach for all dual solvers.
Your framework requires obtaining min of the problem

$$\Delta \alpha^t = \arg \min_{d: 0 \leq \alpha^t + d \leq C} \nabla f(\alpha^t)^T d + \frac{1}{2} d^T H d,$$

but one pass apparently is not the min.

One pass will generate a descent direction with enough decrease (Wang and Lin, 2014), and we can find an $H$ such that this direction is its minimizer.

Still global linear convergence.
Results

Training time of L1-loss SVM for reaching

\[ \left| \frac{f(\alpha) - f(\alpha^*)}{f(\alpha^*)} \right| \leq 0.01. \]

<table>
<thead>
<tr>
<th>Data set</th>
<th>BQO-E</th>
<th>BQO-A</th>
<th>DisDCA</th>
<th>DSVM-AVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>webspam</td>
<td>21.7</td>
<td>30.6</td>
<td>54.7</td>
<td>79.5</td>
</tr>
<tr>
<td>url</td>
<td>1,092.9</td>
<td>1,801.2</td>
<td>3,524.2</td>
<td>4,343.4</td>
</tr>
<tr>
<td>epsilon</td>
<td>2.8</td>
<td>3.9</td>
<td>8.0</td>
<td>13.2</td>
</tr>
</tbody>
</table>

▶ **BQO-E** is 2.56 - 4.76 times faster than existing methods
Results (Cont’d)

L2-loss SVM: test accuracy

webspam

url

BQO-E achieves stable test accuracies the fastest

Lee, Ching-pei and Dan Roth, Distributed Box-constrained training for dual SVM
Speedup

L2-loss SVM: speedup of different #machines

epsilon

webspam

Training

Training + IO
Outline

Introduction

Algorithm

Experiments

Discussions and Conclusions
Speedup

- All dual methods have bad speedup in comparison to Newton because of the block-diagonal approximation.
- But running time might still be good enough for moderate number of machines if we also consider I/O time.
- Still important in problems in which directly solving the primal is not feasible, like multi-class SVM and structured SVM.
Why not Spark?

- Apache Spark is emerging as a new distributed platform, and we had successful experiences in developing packages on it (Lin et al., 2014).
- But dual methods do not work well on Spark because of higher overheads.
Why not Spark?

- Apache Spark is emerging as a new distributed platform, and we had successful experiences in developing packages on it (Lin et al., 2014).
- But dual methods do not work well on Spark because of higher overheads.
- L2-loss SVM, $C = 1$, 16 machines.

*Spark-dual: L2-loss modified from http://github.com/gingsmith/cocoa/*.
When is Dual Method not Suitable?

- When $\ell \gg n$, usually dual methods are worse.
When is Dual Method not Suitable?

- When $\ell \gg n$, usually dual methods are worse.
- Example (from the previous page): covtype: $\ell = 581,012$, $n = 54$. 

![Graph showing relative objective value over training time for different methods.](image)
Conclusions

- An efficient method for distributedly training dual problems.
- The algorithm is superior to state of the art distributed linear SVM solvers both theoretically and practically.