Large Linear Classification When Data Cannot Fit In Memory

(KDD'10 Best Research Paper Award)

Hsiang-Fu Yu, Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin

National Taiwan University

Motivation

- Existing approaches for large linear classification:
  - Data small than memory:

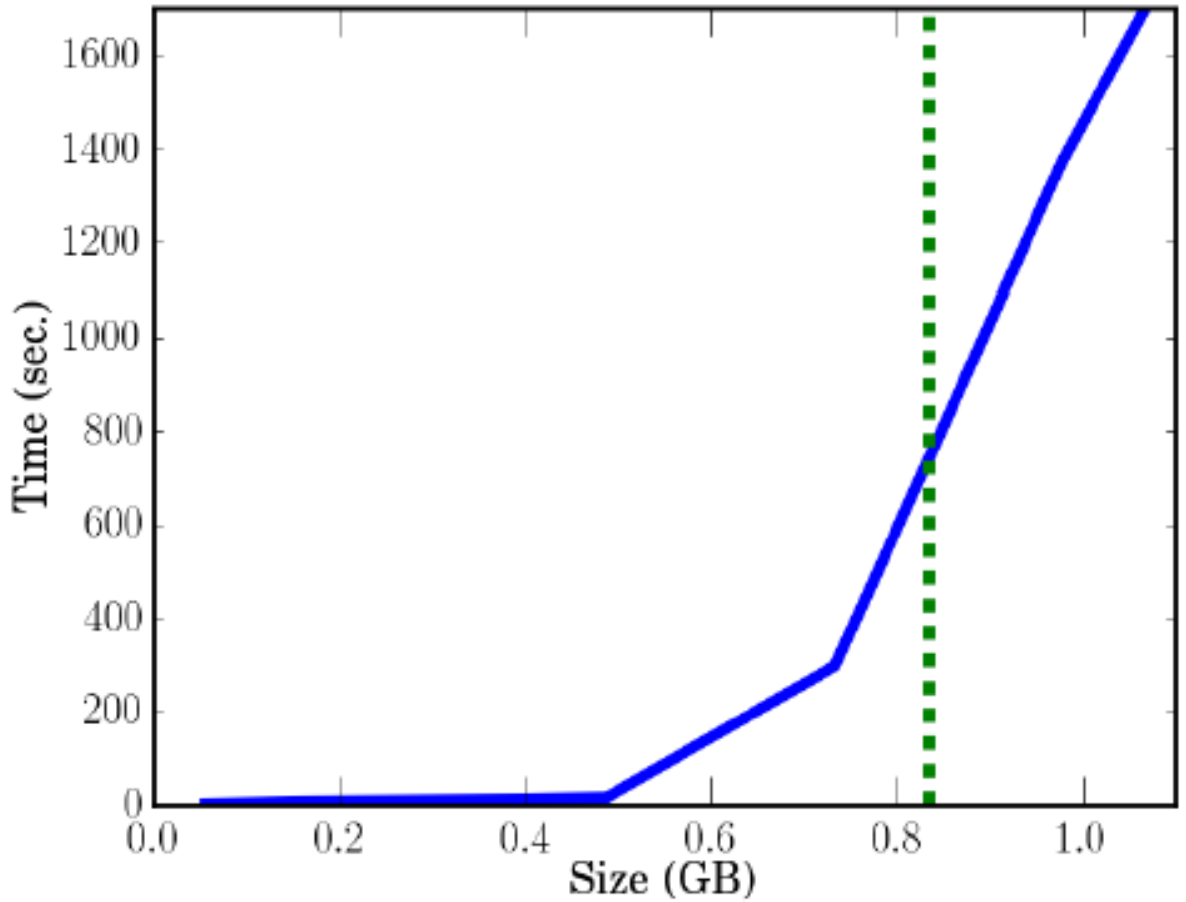
    Efficient methods are well-developed
  - Data beyond disk size:

    Usually handled in a distributed way
- Can we have something in the between?
  - A simple setting

    memory < data < disk

When Data Cannon Fit In Memory

- LIBLINEAR on a machine with 1GB memory:



Disk swap causes lengthy training time

A Block Minimization Framework

1. Split {1,…,m} to B_1,…,B_l such that B_i fits in memory, and store data into l files accordingly.
2. Set initial α and **w**
3. For k = 1,2,…(outer iteration)

   For j = 1,…,l (inner iteration)

   (a) Read $\boldsymbol{x}_r, \forall r \in B\downarrow j$ from disk

   (b) Conduct operations on $\{\boldsymbol{x}r \mid r \in Bj\}$

   (c) Update α and **w**

Here we do not specify operations on each block

Number of Blocks and Block Size

- In the situation, data > memory
  - to avoid random access on the disk
  - $B_1,\ldots,B_l$: fixed partition of $\{1,\ldots,m\}$
- How to decide l (# of blocks)
  - Assume all blocks have similar size $|B|$
  - # blocks: $l=m/|B|$
- Block size
  - Cannot be too large: each B_j must fit in memory
  - Cannot be too small: reading time=initial cost $\times m/|B|$
  - Larger $|B|$ is better (but can't exceed memory)

Sub-problem for Dual SVM

- Let L(α) be the dual function:

$$L(\alpha) = \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \left\langle x^{(i)}, x^{(j)} \right\rangle - \sum_{i=1}^{m} \alpha_i$$

- Each block of variables corresponds to a block of data:

$$\max_{d_{B_j}} \; L(\alpha + d)$$

$$s.t. \qquad d_{\bar{B}_j} = 0, \bar{B}_j = \{1, \ldots, m\} \setminus B_j$$

$$C \geq \alpha_i + d_i \geq 0, \forall i \in B_j$$

Only $\alpha\downarrow B\downarrow j$ is changed

Solving Sub-problem By LIBLINEAR

- Any bound-constrained method can be used
  - We consider LIBLINEAR: a coordinate descent method
- Sub-problem stopping condition and overall convergence
  - A fixed number of passes to all variables in B_j

    Need to decide the number of passes
  - Gradient-based stopping condition

    Easy to know how accurate the sub-problem's solution is; we use the one in LIBLINEAR

  Can prove that convergence holds for both conditions

An Implementation for Dual SVM

1. Split {1,…,*m*} to $B_1,\ldots,B_l$ such that B_i fits in memory, and store data into *l* files accordingly.
2. Set initial α or **w**
3. For *k* = 1,2,…(outer iteration)

   For *j* = 1,…,*l* (inner iteration)

   (a) Read $\boldsymbol{x}_r, \forall r \in B\downarrow j$ from disk

   (b) Approximately solve the sub-problem to obtain $d\downarrow B\downarrow j$

   (c) Update $\alpha\downarrow B\downarrow j \leftarrow \alpha\downarrow B\downarrow j + d\downarrow B\downarrow j$ and

Block Minimization for Primal SVM

- Let f^P be the dual function:

$$f^P(\boldsymbol{\omega}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - C\sum_{i=1}^{m}\max(0, 1 - y_i\mathbf{w}^T x_i)$$

- A block of primal variable w
  - Corresponds to a subset of features
- Stochastic gradient descent(SGD) approach
  - For each update only a block of data is need
  - Use Pegasos (Shalev-Shwartz et al., 2007)

An Implementation for Pegasos

1. Split {1,…,m} to B_1,…,B_l such that B_i fits in memory, and store data into l files accordingly.
2. Set initial **w**
3. For k = 1,2,…(outer iteration)

   For j = 1,…,l (inner iteration)

   (a) Find a partition of B_j : B^1_j,…, B^r_j

   (b) For r' = 1, …, r

   (b.1) Apply Pegasos update on $B^{r'}_j$

Techniques To Reduce the Training Time

- Data compression for disk reading time
  - Except initial time, reading time $\propto$ data size $|B|$
  - Use zlib for compression, effectively reduces the disk reading time
- Initial Split of Data
  - If original data ordered by labels
    - → a whole block with same label
    - → slow convergence
  - A random split is needed
  - Sequentially read data instances and randomly decide which file it should be saved to

Data and Environment

Data set	l (data)	n (features)	Mem
yahoo-korea	460,554	3,052,939	2.5GB
webspam	350,000	16,609,143	20.8GB
epsilson	500,000	2,000	16.0GB

64-bit machine with 1 GB RAM

Data 20 times larger

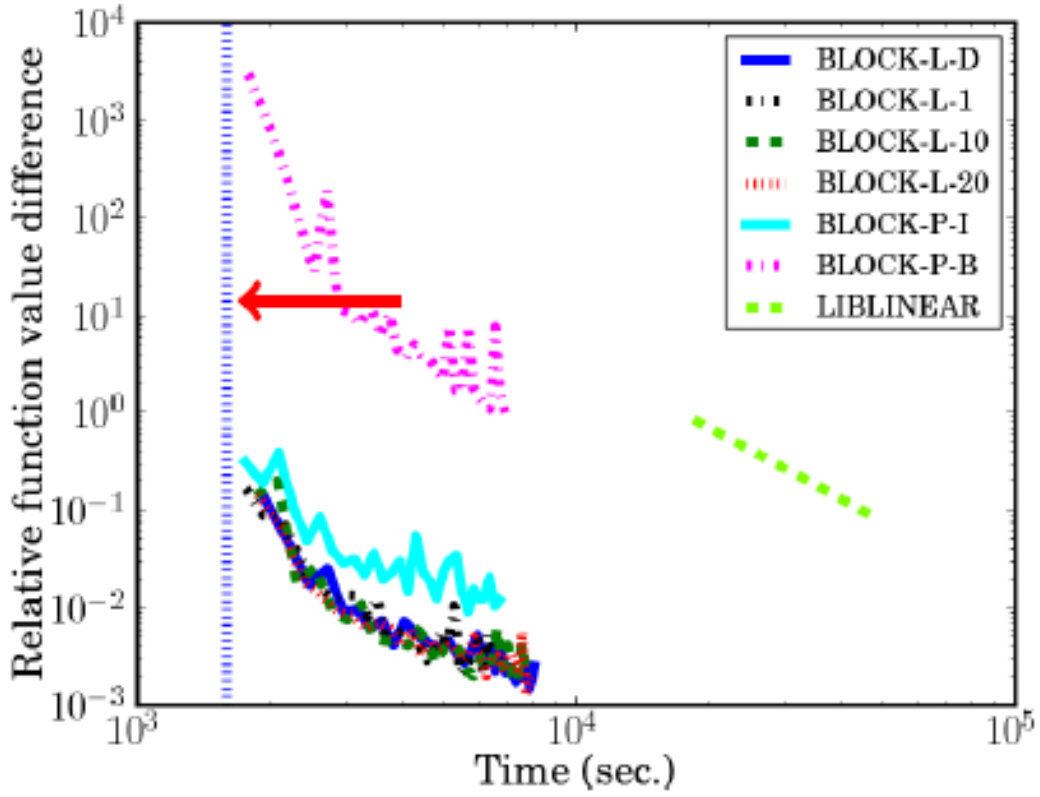
Compared Methods

BLOCK-⋆-⋆: Block minimization methods.

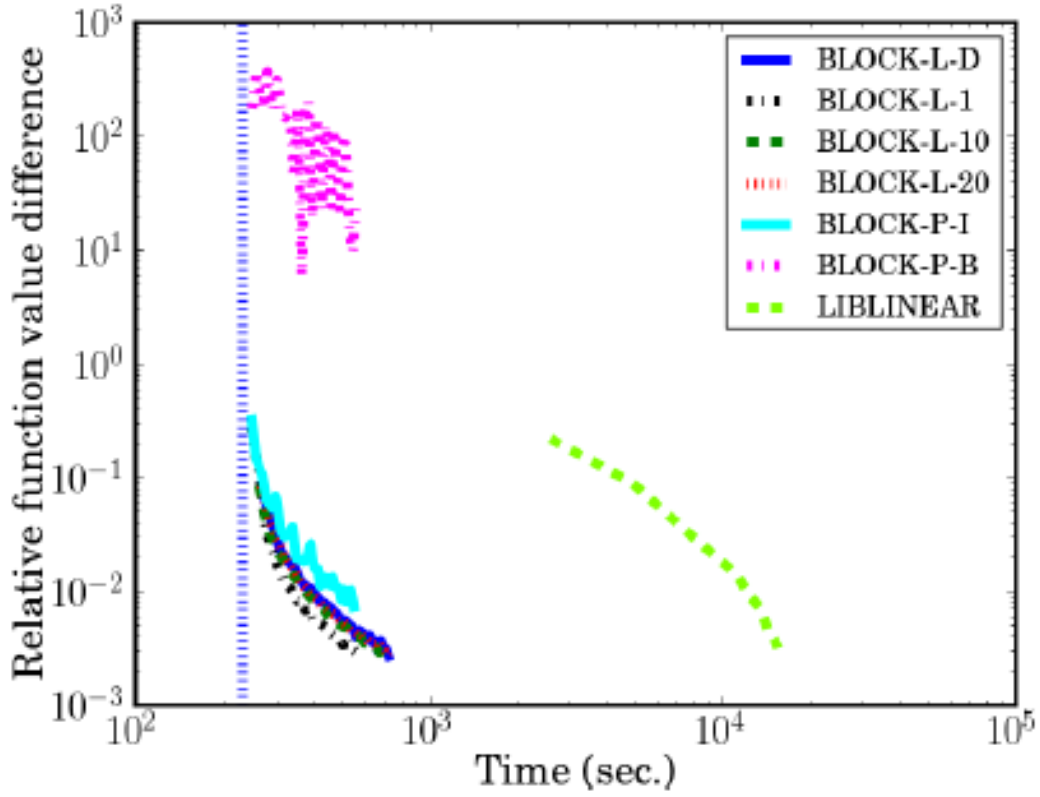
1. BLOCK-L-N: Solving dual. LIBLINEAR goes through each block N rounds; $N = 1, 10, 20$.
2. BLOCK-L-D: Solving dual. LIBLINEAR default stopping condition for each block.
3. BLOCK-P-B: Solving primal. Pegasos on each whole block (one update).
4. BLOCK-P-I: Solving primal. Pegasos on each data instance of the block ($|B|$ updates).
5. LIBLINEAR: The standard LIBLINEAR without any modification.

Experiment Results

webspam



yahoo-korea



Time for initial block split

Thanks!

HP: http://keg.cs.tsinghua.edu.cn/jietang/

Email: jietang@tsinghua.edu.cn