Selective Block Minimization for Faster Convergence of Limited Memory Large-Scale Linear Models

Kai-Wei Chang
Department of Computer Science
University of Illinois at Urbana-Champaign

Joint work with Dan Roth
August 21, 2011
Motivation

- The size of data increases steadily:
  - 40MB memory to train data in KDDCup 2004
  - 1.6GB in KDDCup 2009; **10GB** in KDDCup 2010
- Significantly large amounts of data are available now
  - E.g., Spam filtering, web mining, data stream mining
  - Usually more samples and features ⇒ better performance
- **Linear classifiers** are the method of choice
Motivation

- The size of data increases steadily:
  - 40MB memory to train data in KDDCup 2004
  - 1.6GB in KDDCup 2009; **10GB** in KDDCup 2010
- Significantly large amounts of data are available now
  - E.g., Spam filtering, web mining, data stream mining
  - Usually more samples and features ⇒ better performance
- Linear classifiers are the method of choice
Motivation

- The size of data increases steadily:
  - 40MB memory to train data in KDDCup 2004
  - 1.6GB in KDDCup 2009; 10GB in KDDCup 2010
- Significantly large amounts of data are available now
  - E.g., Spam filtering, web mining, data stream mining
  - Usually more samples and features ⇒ better performance
- Linear classifiers are the method of choice
Modeling the Training Time (Yu et al., 2010)

Training time = running time in memory (learning) + accessing data from disk (loading)

- Data can be stored in memory: focus on the first term
  Efficient methods are well-developed
- Data cannot fit in memory:
  Batch learners suffer due to disk swapping
  If load only a portion of data in memory at a time
  ⇒ the cost of disk access may be higher

Our goal: Saving training time by reducing disk access
Modeling the Training Time (Yu et al., 2010)

Training time = running time in memory (learning) + accessing data from disk (loading)

- Data can be stored in memory: focus on the first term
  Efficient methods are well-developed

- Data cannot fit in memory:
  Batch learners suffer due to disk swapping
  If load only a portion of data in memory at a time
  ⇒ the cost of disk access may be higher

Our goal: Saving training time by reducing disk access
Modeling the Training Time (Yu et al., 2010)

Training time = running time in memory (learning) + accessing data from disk (loading)

- Data can be stored in memory: focus on the first term
  Efficient methods are well-developed
- Data cannot fit in memory:
  Batch learners suffer due to disk swapping
  If load only a portion of data in memory at a time
  ⇒ the cost of disk access may be higher

Our goal: Saving training time by reducing disk access
Modeling the Training Time (Yu et al., 2010)

Training time = running time in memory (learning) +
accessing data from disk (loading)

- Data can be stored in memory: focus on the first term
  Efficient methods are well-developed
- Data cannot fit in memory:
  Batch learners suffer due to disk swapping
  If load only a portion of data in memory at a time
  ⇒ the cost of disk access may be higher

Our goal: Saving training time by reducing disk access
Reducing the I/O Overhead

There are two orthogonal directions to reduce the I/O overhead:

- Apply compression to lessen the loading time
- Better utilizing memory in learning
  \[\Rightarrow\] requires less data load to give an accurate model

Our algorithm mainly focus on the second direction:

- If a sample is likely to be important \[\Rightarrow\] cache it for later iterations
- Spend more time and memory on important samples

Details will be shown later
Reducing the I/O Overhead

There are two orthogonal directions to reduce the I/O overhead:

- Apply compression to lessen the loading time
- Better utilizing memory in learning
  ⇒ requires less data load to give an accurate model

Our algorithm mainly focus on the second direction:

- If a sample is likely to be important ⇒ cache it for later iterations
- Spend more time and memory on important samples

Details will be shown later
Reducing the I/O Overhead

There are two orthogonal directions to reduce the I/O overhead:

- Apply compression to lessen the loading time
- Better utilizing memory in learning
  ⇒ requires less data load to give an accurate model

Our algorithm mainly focus on the second direction:
- If a sample is likely to be important ⇒ cache it for later iterations
- Spend more time and memory on important samples

Details will be shown later
Contributions

The properties of **Selective Block Minimization** (SBM):

- **SBM significantly improves the I/O cost:**
  - SBM obtains an accurate model with loading data from disk only once on a spam filtering data
  - SBM saves I/O access by reducing \#required iterations
  - As a result, SBM efficiently gives an accurate model

- **SBM maintains good convergence properties:**
  - SBM caches data selectively and thus treats sample non-uniformly
  - It **converges linearly** to a global optimal solution on the entire data
Contributions

The properties of **Selective Block Minimization** (SBM):

- **SBM significantly improves the I/O cost:**
  - SBM obtains an accurate model with loading data from disk only once on a spam filtering data
  - SBM saves I/O access by reducing #required iterations
  - As a result, SBM efficiently gives an accurate model

- **SBM maintains good convergence properties:**
  - SBM caches data selectively and thus treats sample non-uniformly
  - It converges linearly to a global optimal solution on the entire data
Contributions

The properties of Selective Block Minimization (SBM):

- **SBM significantly improves the I/O cost:**
  - SBM obtains an accurate model with loading data from disk only once on a spam filtering data
  - SBM saves I/O access by reducing #required iterations
  - As a result, SBM efficiently gives an accurate model

- **SBM maintains good convergence properties:**
  - SBM caches data selectively and thus treats sample non-uniformly
  - It converges linearly to a global optimal solution on the entire data
Outline

- Selective Block Minimization Algorithms for Linear SVMs
- Related Methods
- Experiments
- Conclusions
Outline

- Selective Block Minimization Algorithms for Linear SVMs
- Related Methods
- Experiments
- Conclusions
Linear SVM as the Linear Classifier

- Training data \( \{(y_i, x_i)\}_{1}^{l} \), \( x_i \in \mathbb{R}^n \), \( y_i = \pm 1 \)
- \( n \): # of features, \( l \): # of data

Dual SVM

\[
\begin{align*}
\text{min}_{\alpha} & \quad \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \\
\text{subject to} & \quad 0 \leq \alpha_i \leq C, \forall i,
\end{align*}
\]

- \( \alpha \in \mathbb{R}^l \), each \( \alpha_i \) corresponds to \( x_i \)
  \( \Rightarrow \) allow us to put emphasis on informative samples
- \( e = [1, \ldots, 1]^T \), \( Q_{ij} = y_i y_j x_i^T x_j \)
Linear SVM as the Linear Classifier

- Training data \( \{(y_i, x_i)\}_{i=1}^l \), \( x_i \in \mathbb{R}^n \), \( y_i = \pm 1 \)
- \( n \): # of features, \( l \): # of data

Dual SVM

\[
\min_{\alpha} \quad \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \\
\text{subject to} \quad 0 \leq \alpha_i \leq C, \forall i,
\]

- \( \alpha \in \mathbb{R}^l \), each \( \alpha_i \) corresponds to \( x_i \)
  \( \Rightarrow \) allow us to put emphasis on informative samples
- \( e = [1, \ldots, 1]^T \), \( Q_{ij} = y_i y_j x_i^T x_j \)
Linear SVM as the Linear Classifier

- Training data \( \{(y_i, x_i)\}_{i=1}^l \), \( x_i \in \mathbb{R}^n \), \( y_i = \pm 1 \)
- \( n \): # of features, \( l \): # of data

Dual SVM

\[
\begin{align*}
\text{min} & \quad \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \\
\text{subject to} & \quad 0 \leq \alpha_i \leq C, \forall i,
\end{align*}
\]

- \( \alpha \in \mathbb{R}^l \), each \( \alpha_i \) corresponds to \( x_i \)
- \( e = [1, \ldots, 1]^T \), \( Q_{ij} = y_i y_j x_i^T x_j \)

⇒ allow us to put emphasis on informative samples
Block Minimization (BM) Algorithms

A block minimization algorithm (Yu et al., 2010):
- Split data into $B_1, \ldots, B_m$ such that $B_j$ fits in memory
- At each time, load and train on a data block

A problem of the BM algorithm
- Split data into blocks randomly
- Informative samples are likely in different blocks
  $\Rightarrow$ waste time and memory on unimportant samples

Intuition: try to only focus on important samples
Block Minimization (BM) Algorithms

A block minimization algorithm (Yu et al., 2010):

- Split data into $B_1, \ldots, B_m$ such that $B_j$ fits in memory
- At each time, load and train on a data block

A problem of the BM algorithm

- Split data into blocks randomly
- Informative samples are likely in different blocks
  $\Rightarrow$ waste time and memory on unimportant samples

Intuition: try to only focus on important samples
Block Minimization (BM) Algorithms

A block minimization algorithm (Yu et al., 2010):

- Split data into $B_1, \ldots, B_m$ such that $B_j$ fits in memory
- At each time, load and train on a data block

A problem of the BM algorithm

- Split data into blocks randomly
- Informative samples are likely in different blocks
  $\Rightarrow$ waste time and memory on unimportant samples

Intuition: try to only focus on important samples
A Selective Block Minimization method

**Intuition:** try to only focus on important samples

- We don’t know which samples are important before training the model
  - ⇒ Need a way to find and cache informative samples during the training process

**Our solution:**
At each step, update the model using data consisting of
- New data loaded from disk
- Informative samples selected from previous steps
Comparisons between SBM and BM

Given a large set of training data
Comparisons between SBM and BM

Find a separator to separate circles from triangles
Comparisons between SBM and BM

Red: informative samples (support vectors)
Comparisons between SBM and BM

First iteration:
Both BM and SBM load a data block from disk
Comparisons between SBM and BM

First iteration:
Update the model using data in memory
Comparisons between SBM and BM

First iteration: SBM selects informative samples and caches them
Comparisons between SBM and BM

Second iteration:
SBM caches informative samples in memory
Comparisons between SBM and BM

Third iteration:
More and more informative samples are cached in memory
Comparisons between SBM and BM

Fourth iteration:
More and more informative samples caches in memory
Comparisons between SBM and BM

Fifth iteration:
More and more informative samples caches in memory
Comparisons between SBM and BM

After several iteration:
Most of cached samples are informative
Comparisons between SBM and BM

After several iterations: SBM converges faster by putting more effort on informative samples
Selective Block Minimization (Key Algorithm)

Algorithm 1

1. Split data into $B_1, \ldots, B_m$ and store them into files
2. Set initial $\alpha$ and cache set $\Omega^{1,1} = \emptyset$
3. For $t = 1, 2, \ldots$ (outer iteration)
   For $j = 1, \ldots, m$ (inner iteration)
   (a) Read $x_r, \forall r \in B_j$ from disk
   (b) Solve a sub-problem using $\{W^{t,j} = B_j \cup \Omega^{t,j}\}$
   (c) Update $\alpha$
   (d) Select cached data $\Omega^{t,j+1}$ from $W^{t,j}$

Details are shown in the paper
Selective Block Minimization (Key Algorithm)

Algorithm 1

1. Split data into $B_1, \ldots, B_m$ and store them into files
2. Set initial $\alpha$ and cache set $\Omega^{1,1} = \emptyset$
3. For $t = 1, 2, \ldots$ (outer iteration)
   For $j = 1, \ldots, m$ (inner iteration)
   (a) Read $x_r, \forall r \in B_j$ from disk
   (b) Solve a sub-problem using $\{W^{t,j} = B_j \cup \Omega^{t,j}\}$
   (c) Update $\alpha$
   (d) Select cached data $\Omega^{t,j+1}$ from $W^{t,j}$

Details are shown in the paper
Selective Block Minimization (Key Algorithm)

Algorithm 1

1. Split data into $B_1, \ldots, B_m$ and store them into files
2. Set initial $\alpha$ and cache set $\Omega^{1,1} = \emptyset$
3. For $t = 1, 2, \ldots$ (outer iteration)
   For $j = 1, \ldots, m$ (inner iteration)
   (a) Read $x_r, \forall r \in B_j$ from disk
   (b) Solve a sub-problem using $\{W^{t,j} = B_j \cup \Omega^{t,j}\}$
   (c) Update $\alpha$
   (d) Select cached data $\Omega^{t,j+1}$ from $W^{t,j}$

Details are shown in the paper
Selective Block Minimization (Key Algorithm)

Algorithm 1

1. Split data into $B_1, \ldots, B_m$ and store them into files
2. Set initial $\alpha$ and cache set $\Omega^{1,1} = \emptyset$
3. For $t = 1, 2, \ldots$ (outer iteration)
   For $j = 1, \ldots, m$ (inner iteration)
   (a) Read $x_r, \forall r \in B_j$ from disk
   (b) Solve a sub-problem using $\{W^{t,j} = B_j \cup \Omega^{t,j}\}$
   (c) Update $\alpha$
   (d) Select cached data $\Omega^{t,j+1}$ from $W^{t,j}$

Details are shown in the paper
Selective Block Minimization (Key Algorithm)

Algorithm 1

1. Split data into $B_1, \ldots, B_m$ and store them into files
2. Set initial $\alpha$ and cache set $\Omega^{1,1} = \emptyset$
3. For $t = 1, 2, \ldots$ (outer iteration)
   For $j = 1, \ldots, m$ (inner iteration)
   (a) Read $x_r, \forall r \in B_j$ from disk
   (b) Solve a sub-problem using $\{W^{t,j} = B_j \cup \Omega^{t,j}\}$
   (c) Update $\alpha$
   (d) Select cached data $\Omega^{t,j+1}$ from $W^{t,j}$

Details are shown in the paper
Solving Sub-problem by LIBLINEAR

- Any bound-constrained method can be used
- We consider LIBLINEAR: a coordinate descent method
- Convergence holds regardless of the cache selection strategy used
Solving Sub-problem by LIBLINEAR

- Any bound-constrained method can be used
- We consider LIBLINEAR: a coordinate descent method
- Convergence holds regardless of the cache selection strategy used
Selecting Cached Data

- Related to selective sampling and shrinking techniques
- The samples that are close to the current separator are usually important ⇒ cache these samples
- Define a scoring function and keep the samples with higher scores in cache

Implementation issues are discussed in the paper
Selecting Cached Data

- Related to selective sampling and shrinking techniques
- The samples that are close to the current separator are usually important ⇒ cache these samples
- Define a scoring function and keep the samples with higher scores in cache

Implementation issues are discussed in the paper
Outline

- Selective Block Minimization Algorithms for Linear SVMs
- Related Methods
- Experiments
- Conclusions
Relations with Selective Sampling

Selective sampling (e.g., Schohn and Cohn (2000))

Start from a subset of the data set, iteratively do

1. Select samples that are close to the current margin
2. Update the model on selected samples

Comparisons between SBM and selective sampling

- SBM loads a data block not only for selection but also for training \(\Rightarrow\) The overhead of selecting samples is small
- SBM has convergence guarantees

Also related to active set methods for non-linear model
Relations with Selective Sampling

Selective sampling (e.g., Schohn and Cohn (2000))

Start from a subset of the data set, iteratively do

1. Select samples that are close to the current margin
2. Update the model on selected samples

Comparisons between SBM and selective sampling

- SBM loads a data block not only for selection but also for training ⇒ The overhead of selecting samples is small
- SBM has convergence guarantees

Also related to active set methods for non-linear model
Relations with Online Learning Algorithms

- Popular approach for learning from huge data sets
- Usually perform a simple update on a single instance
- Require large I/O due to the large \#iterations required
- Some online methods are related to SBM, for examples
  - Pegasos can solve a block of data at a time but cannot do multiple updates on one block otherwise it has no convergence guarantees
  - Collins and Roark (2004) using a caching heuristic but the method has no convergence guarantees
Relations with Online Learning Algorithms

- Popular approach for learning from huge data sets
- Usually perform a simple update on a single instance
- Require large I/O due to the large #iterations required
- Some online methods are related to SBM, for examples
  - Pegasos can solve a block of data at a time but cannot do multiple updates on one block otherwise it has no convergence guarantees
  - Collins and Roark (2004) using a caching heuristic but the method has no convergence guarantees
Relations with Online Learning Algorithms

- Popular approach for learning from huge data sets
- Usually perform a simple update on a single instance
- Require large I/O due to the large \#iterations required
- Some online methods are related to SBM, for examples
  - Pegasos can solve a block of data at a time but cannot do multiple updates on one block otherwise it has no convergence guarantees
  - Collins and Roark (2004) using a caching heuristic but the method has no convergence guarantees
Outline

- Selective Block Minimization Algorithms for Linear SVMs
- Related Methods
- Experiments
- Conclusions
## Data and Environment

<table>
<thead>
<tr>
<th>Data set</th>
<th>(l) (data)</th>
<th>(n) (features)</th>
<th>Mem</th>
<th>#class</th>
</tr>
</thead>
<tbody>
<tr>
<td>webspam</td>
<td>350,000</td>
<td>16,609,143</td>
<td>16.7GB</td>
<td>2</td>
</tr>
<tr>
<td>kddcup10</td>
<td>19,264,093</td>
<td>29,890,095</td>
<td>9.0GB</td>
<td>2</td>
</tr>
<tr>
<td>prep</td>
<td>60,000,000</td>
<td>11,148,531</td>
<td>14.7GB</td>
<td>10</td>
</tr>
</tbody>
</table>

- 64-bit machine with memory restriction: 2 GB RAM
Compared Methods

**Update the model using a data block at at time**

1. **SBM**: proposed method; set $|B_j| = |\Omega| = \frac{1}{2} |\text{Mem}|$
2. **Rand-Cache-BM**: cached samples are picked randomly
3. **BMD**: block-minimization method solving in dual form Yu et al. (2010)

**Update the model using one sample at at time**

4. **BM-Pegasos**: a stochastic gradient decent method under block minimization framework
5. **Vowpal Wabbit**: a well-known online learning package
Comparisons on a Spam Filtering Data (webspam)

Convergence to optimum

Test performance

**y-axis**: relative dual function value difference to optimum
Rand-Cache-BM is faster than BMD, even though it picks cached samples randomly
**Comparisons on a Spam Filtering Data (webspam)**

**Convergence to optimum**

**y-axis**: relative dual function value difference to optimum

With a cache selection strategy, SBM **converges** even faster
Comparisons on a Spam Filtering Data (webspam)

Convergence to optimum

Test performance

y-axis: difference to the best accuracy (lower is better)

Methods that update the model on one sample at a time require a large number of iterations to converge.
Comparisons on a Spam Filtering Data (webspam)

Convergence to optimum

Test performance

**y-axis**: difference to the best accuracy (lower is better)
BMD updates the model using a data block at a time, and it is faster.
Comparisons on a Spam Filtering Data (webspam)

Convergence to optimum

Test performance

y-axis: difference to the best accuracy (lower is better)

SBM are faster than existing approaches to obtain an accurate model
Learning from Streaming Data

- Data is too large ⇒ difficult to process it multiple passes
- Treat it as a steam and process it in an online manner
- Existing methods usually use a simple update rule ⇒ single pass over the sample is typically not enough
- We can apply SBM in training streaming data

Advantages

- Can fully utilize its available resources:
  - E.g., memory capacity, sub-problem learning time
- The performance is more stable:
  - update on both incoming samples and cached samples
Learning from Steaming Data

- Data is too large ⇒ difficult to process it multiple passes
- Treat it as a **steam** and process it in an online manner
- Existing methods usually use a simple update rule
  ⇒ single pass over the sample is **typically not enough**
- We can apply SBM in training streaming data

**Advantages**

- Can fully utilize its available resources:
  E.g., memory capacity, sub-problem learning time
- The performance is more stable:
  update on both incoming samples and cached samples
Learning from Steaming Data

- Data is too large ⇒ difficult to process it multiple passes
- Treat it as a steam and process it in an online manner
- Existing methods usually use a simple update rule
  ⇒ single pass over the sample is typically not enough
- We can apply SBM in training streaming data

Advantages

- Can fully utilize its available resources:
  E.g., memory capacity, sub-problem learning time
- The performance is more stable:
  update on both incoming samples and cached samples
Learning from Steaming Data

- Data is too large ⇒ difficult to process it multiple passes
- Treat it as a steam and process it in an online manner
- Existing methods usually use a simple update rule
  ⇒ single pass over the sample is typically not enough
- We can apply SBM in training streaming data

Advantages

- Can fully utilize its available resources:
  E.g., memory capacity, sub-problem learning time
- The performance is more stable:
  update on both incoming samples and cached samples
Learning from Streaming Data

- Data is too large \(\Rightarrow\) difficult to process it multiple passes
- Treat it as a **steam** and process it in an online manner
- Existing methods usually use a simple update rule
  \(\Rightarrow\) single pass over the sample is **typically not enough**
- We can apply SBM in training streaming data

**Advantages**

- Can fully utilize its available resources:
  - E.g., memory capacity, sub-problem learning time
- **The performance is more stable:**
  - update on both incoming samples and cached samples
### Experiments on Streaming Setting

<table>
<thead>
<tr>
<th></th>
<th>Training Time (sec.)</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I/O</td>
<td>Learning</td>
</tr>
<tr>
<td>VW</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BM-Pegasos</td>
<td>470</td>
<td>9</td>
</tr>
<tr>
<td>SBM</td>
<td>484</td>
<td>81</td>
</tr>
<tr>
<td>Reference</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **I/O**: the time to load data from an ASCII file
- **Learning**: the time to learn model using data in memory
- **Reference**: solving linear SVM until it converges
Experiments on Streaming Setting

<table>
<thead>
<tr>
<th></th>
<th>Training Time (sec.)</th>
<th></th>
<th></th>
<th></th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I/O</td>
<td>Learning</td>
<td>Total</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VW</td>
<td>-</td>
<td>-</td>
<td>507</td>
<td></td>
<td>98.42</td>
</tr>
<tr>
<td>BM-Pegasos</td>
<td>470</td>
<td>9</td>
<td>479</td>
<td></td>
<td>97.87</td>
</tr>
<tr>
<td>SBM</td>
<td>484</td>
<td>81</td>
<td>565</td>
<td></td>
<td>99.55</td>
</tr>
<tr>
<td>Reference</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>99.55</td>
</tr>
</tbody>
</table>

- **I/O**: the time to load data from an ASCII file
- **Learning**: the time to learn model using data in memory
- **Reference**: solving linear SVM until it converges
## Experiments on Streaming Setting

<table>
<thead>
<tr>
<th>Method</th>
<th>I/O (sec.)</th>
<th>Learning (sec.)</th>
<th>Total (sec.)</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>VW</td>
<td>-</td>
<td>-</td>
<td>507</td>
<td>98.42</td>
</tr>
<tr>
<td>BM-Pegasos</td>
<td>470</td>
<td>9</td>
<td>479</td>
<td>97.87</td>
</tr>
<tr>
<td>SBM</td>
<td>484</td>
<td>81</td>
<td>565</td>
<td>99.55</td>
</tr>
<tr>
<td>Reference</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>99.55</td>
</tr>
</tbody>
</table>

- **I/O**: the time to load data from an ASCII file
- **Learning**: the time to learn model using data in memory
- **Reference**: solving linear SVM until it converges
## Experiments on Streaming Setting

<table>
<thead>
<tr>
<th></th>
<th>Training Time (sec.)</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I/O + Learning = Total</td>
<td></td>
</tr>
<tr>
<td>VW</td>
<td>-</td>
<td>507</td>
</tr>
<tr>
<td>BM-Pegasos</td>
<td>470</td>
<td>9</td>
</tr>
<tr>
<td>SBM</td>
<td>484</td>
<td>81</td>
</tr>
<tr>
<td>Reference</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **I/O**: the time to load data from an ASCII file
- **Learning**: the time to learn model using data in memory
- **Reference**: solving linear SVM until it converges
## Experiments on Streaming Setting

<table>
<thead>
<tr>
<th></th>
<th>Training Time (sec.)</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I/O</td>
<td>Learning</td>
</tr>
<tr>
<td>VW</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BM-Pegasos</td>
<td>470</td>
<td>9</td>
</tr>
<tr>
<td>SBM</td>
<td>484</td>
<td>81</td>
</tr>
<tr>
<td>Reference</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **I/O**: the time to load data from an ASCII file
- **Learning**: the time to learn model using data in memory
- **Reference**: solving linear SVM until it converges
## Experiments on Streaming Setting

<table>
<thead>
<tr>
<th></th>
<th>Training Time (sec.)</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I/O</td>
<td>Learning</td>
</tr>
<tr>
<td>VW</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BM-Pegasos</td>
<td>470</td>
<td>9</td>
</tr>
<tr>
<td>SBM</td>
<td>484</td>
<td>81</td>
</tr>
<tr>
<td>Reference</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **I/O**: the time to load data from an ASCII file
- **Learning**: the time to learn model using data in memory
- **Reference**: solving linear SVM until it converges
Experiments on Streaming Setting

<table>
<thead>
<tr>
<th></th>
<th>Training Time (sec.)</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I/O</td>
<td>Learning</td>
</tr>
<tr>
<td>VW</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BM-Pegasos</td>
<td>470</td>
<td>9</td>
</tr>
<tr>
<td>SBM</td>
<td>484</td>
<td>81</td>
</tr>
<tr>
<td>Reference</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- I/O time takes a large fraction of total training time
- SBM is able to utilize the resources to achieve an accurate model
### Experiments on Streaming Setting

<table>
<thead>
<tr>
<th></th>
<th>Training Time (sec.)</th>
<th></th>
<th></th>
<th></th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I/O</td>
<td>Learning</td>
<td>Total</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VW</td>
<td>-</td>
<td>-</td>
<td>507</td>
<td></td>
<td>98.42</td>
</tr>
<tr>
<td>BM-Pegasos</td>
<td>470</td>
<td>9</td>
<td>479</td>
<td></td>
<td>97.87</td>
</tr>
<tr>
<td>SBM</td>
<td>484</td>
<td>81</td>
<td>565</td>
<td></td>
<td><strong>99.55</strong></td>
</tr>
<tr>
<td>Reference</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>99.55</td>
</tr>
</tbody>
</table>

- I/O time takes a large fraction of total training time
- SBM is able to utilize the resources to achieve an accurate model
Outline

- Selective Block Minimization Algorithms for Linear SVMs
- Related Methods
- Experiments
- Conclusions
Conclusions

- We have presented a selective block minimization (SBM) method for training large-scale linear classifiers.
- Can be extended to solve the logistic regression and the Crammer & Singer’s multi-class formulation.
- The experimental code are available at:
  http://cogcomp.cs.illinois.edu/page/publication_view/660
- SBM has been implemented in a branch of LIBLINEAR:
  http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/#large_linear_classification_when_data_cannot_fit_in_memory