StreamSVM
Linear SVMs and Logistic Regression When Data Does Not Fit In Memory

S.V.N. (vishy) Vishwanathan

Purdue University and Microsoft
vishy@purdue.edu

October 9, 2012
1. Linear Support Vector Machines
2. StreamSVM
3. Experiments - SVM
4. Logistic Regression
5. Experiments - Logistic Regression
6. Conclusion
Binary Classification

\[ y_i = +1 \]

\[ y_i = -1 \]
Binary Classification

\[ y_i = +1 \]

\[ y_i = -1 \]

\[ \{ x \mid \langle w, x \rangle = 0 \} \]
Linear Support Vector Machines

\[ y_i = +1 \]

\[ \begin{align*}
\langle w, x_1 - x_2 \rangle &= 2 \\
\langle \frac{w}{\|w\|}, x_1 - x_2 \rangle &= \frac{2}{\|w\|}
\end{align*} \]

\[ \{ x \mid \langle w, x \rangle = 1 \} \]

\[ \{ x \mid \langle w, x \rangle = -1 \} \]

\[ \{ x \mid \langle w, x \rangle = 0 \} \]
Linear Support Vector Machines

Optimization Problem

\[
\min_w \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \max(0, 1 - y_i \langle w, x_i \rangle)
\]
The Dual

Objective

\[
\min_{\alpha} \quad D(\alpha) := \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \sum_i \alpha_i \\
\text{s.t.} \quad 0 \leq \alpha_i \leq C
\]

- Convex and quadratic objective
- Linear (box) constraints
- \( w = \sum_i \alpha_i y_i x_i \)
- Each \( \alpha_i \) corresponds to an \( x_i \)
Coordinate Descent
Coordinate Descent
Coordinate Descent

![Graph of Coordinate Descent](image)
Coordinate Descent
Coordinate Descent
Coordinate Descent
Coordinate Descent
Coordinate Descent in the Dual

One dimensional function

\[
\hat{D}(\alpha_t) = \frac{\alpha_t^2}{2} \langle x_t, x_t \rangle + \sum_{i \neq t} \alpha_t \alpha_i y_i y_t \langle x_i, x_t \rangle - \alpha_t + \text{const.}
\]

s.t. \( 0 \leq \alpha_t \leq C \)

Solve

\[
\nabla \hat{D}(\alpha_t) = \alpha_t \langle x_t, x_t \rangle + \sum_{i \neq t} \alpha_i y_i y_t \langle x_i, x_t \rangle - 1 = 0
\]

\[
\alpha_t = \text{median}\left(0, C, \frac{1 - \sum_{i \neq t} \alpha_i y_i y_t \langle x_i, x_t \rangle}{\langle x_t, x_t \rangle}\right)
\]
Outline

1. Linear Support Vector Machines
2. StreamSVM
3. Experiments - SVM
4. Logistic Regression
5. Experiments - Logistic Regression
6. Conclusion
We are Collecting Lots of Data . . .

\[ y_i = +1 \]

\[ y_i = -1 \]
We are Collecting Lots of Data . . .

\[ y_i = +1 \]

\[ y_i = -1 \]
We are Collecting Lots of Data . . .

\[ y_i = +1 \]

\[ y_i = -1 \]
We are Collecting Lots of Data ...
What if Data Does not Fit in Memory?

**Idea 1: Block Minimization [Yu et al., KDD 2010]**
- Split data into blocks $B_1, B_2 \ldots$ such that $B_j$ fits in memory
- Compress and store each block separately
- Load one block of data at a time and optimize only those $\alpha_i$'s

**Idea 2: Selective Block Minimization [Chang and Roth, KDD 2011]**
- Split data into blocks $B_1, B_2 \ldots$ such that $B_j$ fits in memory
- Compress and store each block separately
- Load one block of data at a time and optimize only those $\alpha_i$'s
- Retain *informative samples* from each block in main memory
What are Informative Samples?
Some Observations

SBM and BM are wasteful
- Both split data into blocks and compress the blocks
  - This requires reading the entire data at least once (expensive)
- Both pause optimization while a block is loaded into memory

Hardware 101
- Disk I/O is slower than CPU (sometimes by a factor of 100)
- Random access on HDD is terrible
  - sequential access is reasonably fast (factor of 10)
- Multi-core processors are becoming commonplace
- How can we exploit this?
Our Architecture

Reader

Trainer

HDD
Data

RAM
Working Set

RAM
Weight Vec
Our Philosophy

*Iterate over the data in main memory while streaming data from disk. Evict primarily examples from main memory that are “uninformative”.*
for $k = 1, \ldots, \max\_\text{iter}$ do

for $i = 1, \ldots, n$ do

if $|A| = \Omega$ then

randomly select $i' \in A$

$A = A \setminus \{i'\}$

delete $y_i, Q_{ii}, x_i$ from RAM

end if

read $y_i, x_i$ from Disk

calculate $Q_{ii} = \langle x_i, x_i \rangle$

store $y_i, Q_{ii}, x_i$ in RAM

$A = A \cup \{i\}$

end for

if stopping criterion is met then

exit

end if

end for
\( \alpha^1 = 0, w^1 = 0, \varepsilon = 9, \varepsilon^{\text{new}} = 0, \beta = 0.9 \)

while stopping criterion is not met do

for \( t = 1, \ldots, n \) do

If \( |A| > 0.9 \times \Omega \) then \( \varepsilon = \beta \varepsilon \)
randomly select \( i \in A \) and read \( y_i, Q_{ii}, x_i \) from \( \text{RAM} \)
compute \( \nabla_i D := y_i \langle w^t, x_i \rangle - 1 \)
if \((\alpha^t_i = 0 \text{ and } \nabla_i D > \varepsilon) \text{ or } (\alpha^t_i = C \text{ and } \nabla_i D < -\varepsilon)\) then
\( A = A \setminus \{i\} \) and delete \( y_i, Q_{ii}, x_i \) from \( \text{RAM} \)
continue
end if

\( \alpha^{t+1}_i = \text{median}(0, C, \alpha^t_i - \frac{\nabla_i D}{Q_{ii}}) \), \( w^{t+1} = w^t + (\alpha^{t+1}_i - \alpha^t_i)y_ix_i \)
\( \varepsilon^{\text{new}} = \max(\varepsilon^{\text{new}}, |\nabla_i D|) \)

end for

Update stopping criterion
\( \varepsilon = \varepsilon^{\text{new}} \)

end while
Proof of Convergence (Sketch)

Definition (Luo and Tseng Problem)

\[
\begin{align*}
\text{minimize} & \quad g(E\alpha) + b^\top \alpha \\
\text{subject to} & \quad L_i \leq \alpha_i \leq U_i
\end{align*}
\]

\(\alpha, b \in \mathbb{R}^n, E \in \mathbb{R}^{d \times n}, L_i \in [-\infty, \infty), U_i \in (-\infty, \infty]\)

The above optimization problem is a Luo and Tseng problem if the following conditions hold:

1. \(E\) has no zero columns
2. the set of optimal solutions \(\mathcal{A}\) is non-empty
3. the function \(g\) is strictly convex and twice cont. differentiable
4. for all optimal solutions \(\alpha^* \in \mathcal{A}\), \(\nabla^2 g(E\alpha^*)\) is positive definite.
Proof of Convergence (Sketch)

**Lemma (see Theorem 1 of Hsieh et al., ICML 2008)**

*If no data \( x_i = 0 \), then the SVM dual is a Luo and Tseng problem.*

**Definition (Almost cyclic rule)**

There exists an integer \( B \geq n \) such that every coordinate is iterated upon at least once every \( B \) successive iterations.

**Theorem (Theorem 2.1 of Luo and Tseng, JOTA 1992)**

*Let \( \{\alpha^t\} \) be a sequence of iterates generated by a coordinate descent method using the almost cyclic rule. The \( \alpha^t \) converges linearly to an element of \( \mathcal{A} \).*
Proof of Convergence (Sketch)

**Lemma**

If the trainer accesses the cached training data sequentially, and the following conditions hold:

- The trainer is at most $\kappa \geq 1$ times faster than the reading thread, that is, the trainer performs at most $\kappa$ coordinate updates in the time that it takes the reader to read one training data from disk.

- A point is never evicted from the RAM unless the $\alpha_i$ corresponding to that point has been updated.

Then this confirms to the almost cyclic rule.
Outline

1. Linear Support Vector Machines
2. StreamSVM
3. Experiments - SVM
4. Logistic Regression
5. Experiments - Logistic Regression
6. Conclusion
### Experiments

<table>
<thead>
<tr>
<th>dataset</th>
<th>$n$</th>
<th>$d$</th>
<th>$s(%)$</th>
<th>$n_+ : n_-$</th>
<th>Datasize</th>
</tr>
</thead>
<tbody>
<tr>
<td>ocr</td>
<td>3.5 M</td>
<td>1156</td>
<td>100</td>
<td>0.96</td>
<td>45.28 GB</td>
</tr>
<tr>
<td>dna</td>
<td>50 M</td>
<td>800</td>
<td>25</td>
<td>3e$-$3</td>
<td>63.04 GB</td>
</tr>
<tr>
<td>webspam-t</td>
<td>0.35 M</td>
<td>16.61 M</td>
<td>0.022</td>
<td>1.54</td>
<td>20.03 GB</td>
</tr>
<tr>
<td>kddb</td>
<td>20.01 M</td>
<td>29.89 M</td>
<td>1e$-$4</td>
<td>6.18</td>
<td>4.75 GB</td>
</tr>
</tbody>
</table>
Does Active Eviction Work?

![Graph](image-url)

Wall Clock Time (sec) vs. Relative Function Value Difference for linear SVM on webspam-t with $C = 1.0$. The graph compares the performance of random and active eviction methods.
Comparison with Block Minimization

![Graph showing the comparison between StreamSVM, SBM, and BM with Wall Clock Time (sec) on the x-axis and Relative Function Value Difference on the y-axis. The graph includes a legend indicating the lines for different algorithms and a marker for OCR when C = 1.0.](image)

StreamSVM

SBM

BM
Comparison with Block Minimization

![Graph showing relative function value difference against wall clock time for different methods. The x-axis represents wall clock time in seconds, ranging from 0 to 2 seconds, with a log scale. The y-axis represents the relative function value difference, also on a log scale. The graph includes lines for StreamSVM, SBM, and BM. The inset box shows the data for the webspam-t dataset with $C = 1.0$. The titles for the axes are: Wall Clock Time (sec) and Relative Function Value Difference.]
Comparison with Block Minimization

Experiments - SVM

$kddb \ C = 1.0$

- StreamSVM
- SBM
- BM

Relative Function Value Difference vs. Wall Clock Time (sec)

Logarithmic scale for y-axis (Relative Function Value Difference)

Wall Clock Time (sec)
Comparison with Block Minimization

![Graph showing relative objective function value vs wall clock time for different methods: StreamSVM, SBM, BM. The graph demonstrates the performance of these methods with relative objective function values ranging from $10^{-1}$ to $10^{-11}$ over time from 0 to approximately 40,000 seconds. The legend indicates dna $C = 1.0$.](image-url)
Effect of Varying $C$

The diagram illustrates the relative function value difference over wall clock time for different values of $C$. The y-axis represents the relative function value difference on a logarithmic scale, ranging from $10^{-11}$ to $10^{-1}$. The x-axis represents wall clock time in seconds, ranging from 0 to $1.6 	imes 10^5$ seconds.

The graph shows the performance of different methods: StreamSVM, SBM, and BM, with StreamSVM consistently performing the best across the range of $C$ values tested. The specific value $C = 10.0$ is highlighted in the inset box.
Effect of Varying $C$

![Graph showing the effect of varying $C$ on relative function value difference and wall clock time.]
Effect of Varying $C$

![Graph showing the effect of varying $C$ on the relative objective function value and wall clock time.](image)
Varying Cache Size

Wall Clock Time (sec)

Relative Function Value Difference

$k_{\text{ddb}} C = 1.0$

- 256 MB
- 1 GB
- 4 GB
- 16 GB

S.V. N. Vishwanathan (Purdue, Microsoft)
Expanding Features

![Graph showing relative function value difference against wall clock time for DNA expanded with different memory sizes (16 GB and 32 GB). The graph plots the relative function value difference on a logarithmic scale against wall clock time. The curve for 16 GB shows a steeper decrease than the curve for 32 GB.]
Outline

1. Linear Support Vector Machines
2. StreamSVM
3. Experiments - SVM
4. **Logistic Regression**
5. Experiments - Logistic Regression
6. Conclusion
Optimization Problem

\[
\min_w \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \log (1 + \exp (-y_i \langle w, x_i \rangle))
\]
Logistic Regression

The Dual

**Objective**

\[
\min_{\alpha} D(\alpha) := \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_i \alpha_i \log \alpha_i + (C - \alpha_i) \log(C - \alpha_i)
\]

s.t. \( 0 \leq \alpha_i \leq C \)

- \( w = \sum_i \alpha_i y_i x_i \)
- **Convex** objective (but not quadratic)
- Box constraints
Coordinate Descent

One dimensional function

\[
\hat{D}(\alpha_t) := \frac{\alpha_t^2}{2} \langle x_t, x_t \rangle + \sum_{i \neq t} \alpha_t \alpha_i y_i y_t \langle x_i, x_t \rangle + \alpha_t \log \alpha_t + (C - \alpha_t) \log(C - \alpha_t)
\]

s.t. \(0 \leq \alpha_t \leq C\)

Solve

- No closed form solution
- A modified Newton method does the job!
- See Yu, Huang, Lin 2011 for details.
Determining Importance: SVM case

\[ \alpha_i^t = 0 \text{ and } \nabla_i D(\alpha) > \varepsilon \quad \text{or} \]
\[ \alpha_i^t = C \text{ and } \nabla_i D(\alpha) < -\varepsilon \]
Determining Importance: SVM case

\[ \nabla_w \text{loss}(w, x_i, y_i) = \alpha_i \] (KKT condition)

\[ \nabla_i^\pi D(\alpha) = 0 \]

\[ |\langle w, x_i \rangle - 1| \geq \epsilon \]

Computing \( \epsilon \)

\[ \epsilon = \max_{i \in A} \left| \nabla_i^\pi D(\alpha) \right| \]
Determining Importance: Logistic Regression

\[ \nabla_w \text{loss}(w, x_i, y_i) \approx \alpha_i \quad \text{(KKT condition)} \]
\[ \nabla_i D(\alpha) \approx 0 \]
\[ |\langle w, x_i \rangle| \geq \epsilon \]

Computing \( \epsilon \)

\[ \epsilon = \max_{i \in A} |\nabla_i D(\alpha)| \]
Outline

1. Linear Support Vector Machines
2. StreamSVM
3. Experiments - SVM
4. Logistic Regression
5. Experiments - Logistic Regression
6. Conclusion
Does Active Eviction Work?

Logistic Regression

We consider the problem of active eviction in the context of logistic regression.

We test the effectiveness of active eviction using the web spam dataset (webspam-t).

We vary the value of the regularization parameter $C$ and observe the relative function value difference over time.

The figure shows the relative function value difference for $C = 1.0$.

- **Random** eviction strategy
- **Active** eviction strategy

As the wall clock time increases, the relative function value difference decreases, indicating improved performance with active eviction.
Comparison with Block Minimization

![Graph showing comparison between StreamSVM and Block Minimization (BM) for different relative function value differences and wall clock time. The graph indicates that StreamSVM outperforms BM in terms of relative function value difference and wall clock time.]
Outline

1. Linear Support Vector Machines
2. StreamSVM
3. Experiments - SVM
4. Logistic Regression
5. Experiments - Logistic Regression
6. Conclusion
Things I did not talk about/Work in Progress

- Amortizing the cost of training different models
- Other storage Heirarchies (e.g. Solid State Drives)
- Extensions to other problems
  - Multiclass problems
  - Structured Output Prediction
- Distributed version of the algorithm

Code for StreamSVM available from
http://www.r.dl.itc.u-tokyo.ac.jp/~masin/streamsvm.html
Joint work with

Conclusion