Optimization for Machine Learning
StreamSVM

S.V.N. (vishy) Vishwanathan

Purdue University
vishy@purdue.edu

May 11, 2012
Outline

1. Linear Support Vector Machines
2. StreamSVM
3. Logistic Regression
4. Experiments
5. Conclusion
Linear Support Vector Machines

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

\[ y_i = +1 \]

\[ \{ x \mid \langle w, x \rangle = 0 \} \]

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

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

\[ \langle w, x_1 - x_2 \rangle = 2 \]

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

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

s.t. \hspace{1cm} 0 \leq \alpha_i \leq C

- \hspace{1cm} w = \sum_i \alpha_i y_i x_i
- \hspace{1cm} Convex and quadratic objective
- \hspace{1cm} Linear (box) constraints
Coordinate Descent
Coordinate Descent
Coordinate Descent
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. Logistic Regression
4. Experiments
5. Conclusion
We are Collecting Lots of Data …

\[ y_i = +1 \]

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

<table>
<thead>
<tr>
<th>dataset</th>
<th>$n$</th>
<th>$d$</th>
<th>Datasize</th>
</tr>
</thead>
<tbody>
<tr>
<td>ocr</td>
<td>3.5 M</td>
<td>1156</td>
<td>45.28 GB</td>
</tr>
<tr>
<td>dna</td>
<td>50 M</td>
<td>800</td>
<td>63.04 GB</td>
</tr>
<tr>
<td>webspam-t</td>
<td>0.35 M</td>
<td>16.61 M</td>
<td>20.03 GB</td>
</tr>
<tr>
<td>kddb</td>
<td>20.01 M</td>
<td>29.89 M</td>
<td>4.75 GB</td>
</tr>
</tbody>
</table>
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

RAM

Working Set

Trainer

RAM

Weight Vec

HDD

Data
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 \( \text{RAM} \)

end if

read \( y_i, x_i \) from \( \text{Disk} \)
calculate \( Q_{ii} = \langle x_i, x_i \rangle \)

store \( y_i, Q_{ii}, x_i \) in \( \text{RAM} \)

\( A = A \cup \{i\} \)

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 RAM

compute \(\nabla_i D := y_i \langle w^t, x_i \rangle - 1\)

if \((\alpha_i^t = 0 \text{ and } \nabla_i D > \varepsilon)\) or \((\alpha_i^t = C \text{ and } \nabla_i D < -\varepsilon)\) then

\(A = A \setminus \{i\}\) and delete \(y_i, Q_{ii}, x_i\) from RAM

continue

end if

\[
\alpha_i^{t+1} = \text{median}(0, C, \alpha_i^t - \frac{\nabla_i D}{Q_{ii}})\], 
\[
w^{t+1} = w^t + (\alpha_i^{t+1} - \alpha_i^t)y_ix_i
\]

\(\varepsilon^{\text{new}} = \max(\varepsilon^{\text{new}}, |\nabla_i D|)\)

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 \(A\) is non-empty
3. the function \(g\) is strictly convex and twice cont. differentiable
4. for all optimal solutions \(\alpha^* \in A\), \(\nabla^2 g(E\alpha^*)\) 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 \( A \).
Proof of Convergence (Sketch)

Lemma

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

1. 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.
2. 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.
Optimization Problem

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

Objective

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

s.t. \quad 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

Shrinking

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

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

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

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

**Computing** \( \epsilon \)

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

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

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

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

**Computing** \(\epsilon\)

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

1. Linear Support Vector Machines
2. StreamSVM
3. Logistic Regression
4. Experiments
5. 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 showing the relative function value difference over wall clock time for Linear SVM with webspam-t and various C values. The graph compares random and active eviction strategies.]
Does Active Eviction Work?

- **Logistic Regression**
- `webspam-t C = 1.0`
- Random
- Active

**Wall Clock Time (sec)**

**Relative Function Value Difference**

- Data points show decreasing function value differences over time for both Random and Active eviction methods. The Active method shows a faster convergence rate compared to the Random method.
Comparison with Block Minimization

![Graph showing relative function value difference over wall clock time for different methods.]

- **OCR** $C = 0.0$
- **StreamSVM**
- **SBM**
- **BM**

The graph illustrates the relative function value difference across various methods as a function of wall clock time.
Experiments

Comparison with Block Minimization

Wall Clock Time (sec)

Relative Function Value Difference

webspam-t $C = 1.0$

StreamSVM
SBM
BM
Comparison with Block Minimization

Experiments

Wall Clock Time (sec)

Relative Function Value Difference

- StreamSVM
- SBM
- BM

$kddb C = 1.0$
Experiments

Comparison with Block Minimization

\[
\begin{align*}
\text{dna } C = 1.0 & \\
\text{StreamSVM} & \\
\text{SBM} & \\
\text{BM} &
\end{align*}
\]


Effect of Varying $C$

![Graph showing the effect of varying C]
Experiments

Effect of Varying $C$

- DNA $C = 100.0$
- StreamSVM
- SBM
- BM

Relative Function Value Difference

Wall Clock Time (sec)
Experiments

Effect of Varying $C$

Wall Clock Time (sec)

Relative Objective Function Value

DNA $C = 1000.0$

StreamSVM

SBM

BM
Varying Cache Size

Experiments

Wall Clock Time (sec)

Relative Function Value Difference

kddb $C = 1.0$

- 256 MB
- 1 GB
- 4 GB
- 16 GB
Expanding Features

Wall Clock Time (sec)

Relative Function Value Difference

DNA expanded $C = 1.0$

16 GB

32 GB
Outline

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

**StreamSVM**

- Other storage Heirarchies (e.g. Solid State Drives)
- Extensions to other problems
  - Multiclass problems
  - Structured Output Prediction
References

- Code for StreamSVM coming soon . . . watch my home page
Joint work with