The Kernelized Stochastic Batch Perceptron

Andrew Cotter\textsuperscript{1}    Shai Shalev-Shwartz\textsuperscript{2}    Nathan Srebro\textsuperscript{1}

\textsuperscript{1}Toyota Technological Institute at Chicago

\textsuperscript{2}Hebrew University of Jerusalem

June 29, 2012
A New Kernel SVM Optimizer

Kernelized SVM optimization

- Data is accessed exclusively via kernel evaluations

We present the Stochastic Batch Perceptron (SBP):

- **Best known learning runtime guarantee (better than previous methods)**
- Performs well in practice
- Efficient, open-source implementation available
The Method

minimize: \( \frac{1}{2} \| w \|_2^2 + C \sum_{i=1}^{n} \max \left( 0, 1 - y_i \langle w, x_i \rangle \right) \)
The Method - Re-parameterization

\[
\text{minimize: } \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^n \max \left(0, 1 - y_i \langle w, x_i \rangle - c_i\right)
\]

Use re-paramaterization of SVM problem due to Hazan et al. (2011)

\[
\text{maximize : } \max_{\xi \in \mathbb{R}^n} \min_{i \in \{1, \ldots, n\}} (\xi_i + c_i)
\]
\[
\text{subject to : } \|w\| \leq 1
\]
\[
: \xi \succeq 0, 1^T \xi \leq n\nu
\]

We refer to this as the “slack-constrained” objective
The Method - Re-parameterization

$$\text{minimize: } \frac{1}{2} \| w \|^2_2 + C \sum_{i=1}^{n} \max \left( 0, 1 - y_i \langle w, x_i \rangle \right)$$

Use re-paramaterization of SVM problem due to Hazan et al. (2011)

$$\text{maximize: } \max_{\xi \in \mathbb{R}^n} \min_{p \in \Delta^n} p^T (\xi + c)$$

subject to: $$\| w \| \leq 1$$

$$\xi \succeq 0, 1^T \xi \leq n\nu$$

We refer to this as the “slack-constrained” objective
The Method - Equivalence of Objectives

Varying $C$ or $\nu$ explores the same Pareto frontier

\[
\text{minimize: } \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^{n} \max (0, 1 - y_i \langle w, x_i \rangle) \\
\text{maximize: } \max_{\xi \in \mathbb{R}^n} \min_{p \in \Delta^n} p^T (\xi + c) \\
\text{subject to: } \|w\| \leq 1 \\
\quad : \xi \succeq 0, 1^T \xi \leq n \nu
\]
maximize \( \max_{\xi \in \mathbb{R}^n} \min_{p \in \Delta^n} p^T (\xi + c) \)

subject to:

\( \|w\| \leq 1 \)

\( \xi \succeq 0, 1^T \xi \leq n \nu \)

Apply stochastic gradient ascent to this re-parameterization

- Different parameterization than Pegasos \( \rightarrow \) different algorithm
- For minimax-optimal \( p^* \), supergradients are \( \sum_{i=1}^n p_i^* y_i x_i \)
- Stochastic supergradients can be found by sampling from \( p^* \)
The Method - Finding a Minimax Optimal $p^*$

Use “water-filling”
- Requires the responses
- $O(n)$ time using a divide-and-conquer algorithm

\[
\begin{align*}
\text{volume } n\nu \\
p_i^* \\
\end{align*}
\]
The Method

Putting it together

At each iteration:

1. Find a minimax-optimal $p^*$
2. Sample $i \sim p^*$
3. Update $w = P(w + \eta_t y_i x_i)$

Separable Case

- $p^*$ supported on $\text{arg min } c_i$
- SBP: update using most violating example at each iteration
- “Batch Perceptron”
The Method

Putting it together

At each iteration:

1. Find a minimax-optimal $p^*$
2. Sample $i \sim p^*$
3. Update $w = P(w + \eta_t y_i x_i)$

Kernelization

Like Pegasos, our algorithm can be kernelized without switching to the dual

- Substitute $w = \sum_{i=1}^{n} \alpha_i y_i x_i$
- Maintain vector of responses $c_i = \sum_{j=1}^{n} \alpha_j y_i y_j K(x_i, x_j)$ throughout
- Cost per iteration is $O(n)$ operations for water-filling, $n$ kernel evaluations for updating $c$
Runtime Analysis

We analyze runtime to ensure generalization error $\mathcal{L}(w^*) + \varepsilon$

- SBP needs $O \left( \left( \frac{\mathcal{L}(w^*) + \varepsilon}{\varepsilon} \right)^2 \|w^*\|^2 \right)$ iterations

- Need $n = O \left( \left( \frac{\mathcal{L}(w^*) + \varepsilon}{\varepsilon} \right) \frac{\|w^*\|^2}{\varepsilon} \right)$ training elements for generalization
Runtime Analysis

We analyze runtime to ensure generalization error $\mathcal{L}(w^*) + \epsilon$

- SBP needs $O\left(\left(\frac{\mathcal{L}(w^*) + \epsilon}{\epsilon}\right)^2 \|w^*\|^2\right)$ iterations
- Need $n = O\left(\left(\frac{\mathcal{L}(w^*) + \epsilon}{\epsilon}\right) \frac{\|w^*\|^2}{\epsilon}\right)$ training elements for generalization

<table>
<thead>
<tr>
<th></th>
<th>Overall Runtime $\epsilon = \Omega(\mathcal{L}(w^*))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBP</td>
<td>$\left(\frac{\mathcal{L}(w^<em>) + \epsilon}{\epsilon}\right)^3 \frac{|w^</em>|^4}{\epsilon}$</td>
</tr>
<tr>
<td>Dual Decomp.</td>
<td></td>
</tr>
<tr>
<td>Pegasos</td>
<td></td>
</tr>
</tbody>
</table>
Runtime Analysis

We analyze runtime to ensure generalization error $\mathcal{L}(w^*) + \varepsilon$

- SBP needs $O\left(\left(\frac{\mathcal{L}(w^*) + \varepsilon}{\varepsilon}\right)^2 \|w^*\|^2\right)$ iterations
- Need $n = O\left(\left(\frac{\mathcal{L}(w^*) + \varepsilon}{\varepsilon}\right) \frac{\|w^*\|^2}{\varepsilon}\right)$ training elements for generalization

<table>
<thead>
<tr>
<th>Method</th>
<th>Overall Runtime</th>
<th>$\varepsilon = \Omega(\mathcal{L}(w^*))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBP</td>
<td>$\left(\frac{\mathcal{L}(w^<em>) + \varepsilon}{\varepsilon}\right)^3 \frac{|w^</em>|^4}{\varepsilon}$</td>
<td></td>
</tr>
<tr>
<td>Dual Decomp.</td>
<td>$\left(\frac{\mathcal{L}(w^<em>) + \varepsilon}{\varepsilon}\right)^2 \frac{|w^</em>|^4}{\varepsilon^2}$</td>
<td></td>
</tr>
<tr>
<td>Pegasos</td>
<td>$\left(\frac{\mathcal{L}(w^<em>) + \varepsilon}{\varepsilon}\right) \frac{|w^</em>|^4}{\varepsilon^3}$</td>
<td></td>
</tr>
</tbody>
</table>
Runtime Analysis

We analyze runtime to ensure generalization error $\mathcal{L}(w^*) + \varepsilon$

- SBP needs $O\left(\left(\frac{\mathcal{L}(w^*) + \varepsilon}{\varepsilon}\right)^2 \|w^*\|^2\right)$ iterations
- Need $n = O\left(\left(\frac{\mathcal{L}(w^*) + \varepsilon}{\varepsilon}\right) \|w^*\|^2\right)$ training elements for generalization

<table>
<thead>
<tr>
<th>Method</th>
<th>Overall Runtime $\varepsilon = \Omega(\mathcal{L}(w^*))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBP</td>
<td>$\left(\frac{\mathcal{L}(w^<em>) + \varepsilon}{\varepsilon}\right)^3 \frac{|w^</em>|^4}{\varepsilon^3}$</td>
</tr>
<tr>
<td>Dual Decomp.</td>
<td>$\left(\frac{\mathcal{L}(w^<em>) + \varepsilon}{\varepsilon}\right)^2 \frac{|w^</em>|^4}{\varepsilon^2}$</td>
</tr>
<tr>
<td>Pegasos</td>
<td>$\left(\frac{\mathcal{L}(w^<em>) + \varepsilon}{\varepsilon}\right) \frac{|w^</em>|^4}{\varepsilon^3}$</td>
</tr>
</tbody>
</table>
## Runtime Analysis

We analyze runtime to ensure generalization error $\mathcal{L}(w^*) + \varepsilon$ when $\varepsilon = \Omega(\mathcal{L}(w^*))$

<table>
<thead>
<tr>
<th>Kernel Algo.</th>
<th>Iterations</th>
<th>Time per Iteration</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBP</td>
<td>$|w^*|^2$</td>
<td>$n = \frac{|w^*|^2}{\varepsilon}$</td>
<td>$|w^*|^4$</td>
</tr>
<tr>
<td>Dual Decomp.</td>
<td>$\frac{|w^*|^2}{\varepsilon}$</td>
<td>$n = \frac{|w^*|^2}{\varepsilon^2}$</td>
<td>$\frac{|w^*|^4}{\varepsilon^2}$</td>
</tr>
<tr>
<td>Pegasos</td>
<td>$\frac{|w^*|^2}{\varepsilon^2}$</td>
<td>$n = \frac{|w^*|^2}{\varepsilon^3}$</td>
<td>$\frac{|w^*|^4}{\varepsilon^3}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linear Algo.</th>
<th>Iterations</th>
<th>Time per Iteration</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBP</td>
<td>$|w^*|^2$</td>
<td>$dn = \frac{d|w^*|^2}{\varepsilon}$</td>
<td>$\frac{d|w^*|^4}{\varepsilon^2}$</td>
</tr>
<tr>
<td>Dual Decomp.</td>
<td>$\frac{|w^*|^2}{\varepsilon}$</td>
<td>$dn = \frac{d|w^*|^2}{\varepsilon^2}$</td>
<td>$\frac{d|w^*|^4}{\varepsilon^3}$</td>
</tr>
<tr>
<td>Pegasos</td>
<td>$\frac{|w^*|^2}{\varepsilon^2}$</td>
<td>$d$</td>
<td>$\frac{d|w^*|^2}{\varepsilon^2}$</td>
</tr>
</tbody>
</table>
Experiments

- SMO makes little progress until it suddenly enters a regime in which it converges rapidly
- Non-SMO algorithms converge gradually
Summary

We presented the Stochastic Batch Perceptron (SBP)

- Data is accessed via kernel evaluations with an arbitrary kernel
- Can be extended to include an unregularized bias
- Best known learning runtime guarantee
- Performs well in practice
- Efficient, open-source implementation available

ttic.uchicago.edu/~cotter/projects/SBP
- Perceptron performs similarly to SBP, but does not converge “fully” in a single pass
Experiments - Perceptron

- Perceptron performs similarly to SBP, but does not converge “fully” in a single pass
- If we perform multiple passes, Perceptron may overfit