The Kernelized Stochastic Batch Perceptron

Andrew Cotter[1] Shai Shalev-Shwartz[2] Nathan Srebro[1]

[1]Toyota Technological Institute at Chicago

[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

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

The Method - Re-parameterization

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

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

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

We refer to this as the "slack-constrained" objective

The Method - Re-parameterization

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

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

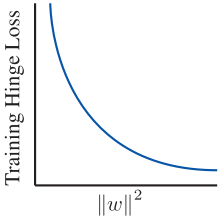
$$\begin{aligned}\text{maximize} : & \max_{\xi\in\mathbb{R}^n}\min_{p\in\Delta^n} p^T(\xi + c)\\ \text{subject to} : & \|w\| \leq 1\\ : & \xi \succeq 0, \mathbf{1}^T\xi \leq n\nu\end{aligned}$$

We refer to this as the “slack-constrained” objective

The Method - Equivalence of Objectives

Varying C or ν explores the same Pareto frontier

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



$$\begin{aligned} \text{maximize} : & \max_{\xi \in \mathbb{R}^n} \min_{p \in \Delta^n} p^T (\xi + c) \\ \text{subject to} : & \|w\| \leq 1 \\ : & \xi \succeq 0, \mathbf{1}^T \xi \leq n\nu \end{aligned}$$

The Method - Stochastic Gradient Ascent

$$\text{maximize}: \max_{\xi \in \mathbb{R}^n} \min_{p \in \Delta^n} p^T (\xi + c)$$
$$\text{subject to}: \|w\| \leq 1$$
$$: \xi \succeq 0, \mathbf{1}^T \xi \leq n\nu$$

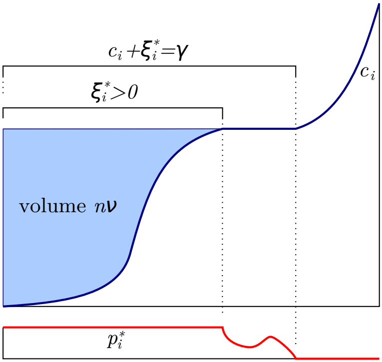
Apply stochastic gradient ascent to this re-parameterization

- 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



The Method

Putting it together

At each iteration:

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

Separable Case

- p^* supported on $\arg\min c_i$
- 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 $w = \mathscr{P}(w + \eta_t y_i x_i)$

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 $\mathscr{L}(w^*) + \varepsilon$

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

Runtime Analysis

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

- 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

	Overall Runtime	$\varepsilon = \Omega(\mathcal{L}(w^*))$
SBP	$\left(\frac{\mathcal{L}(w^*)+\varepsilon}{\varepsilon}\right)^3 \frac{\|w^*\|^4}{\varepsilon}$	
Dual Decomp.		
Pegasos		

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

	Overall Runtime	$\varepsilon = \Omega(\mathcal{L}(w^*))$
SBP	$\left(\frac{\mathcal{L}(w^*)+\varepsilon}{\varepsilon}\right)^3 \frac{\|w^*\|^4}{\varepsilon}$	
Dual Decomp.	$\left(\frac{\mathcal{L}(w^*)+\varepsilon}{\varepsilon}\right)^2 \frac{\|w^*\|^4}{\varepsilon^2}$	
Pegasos	$\left(\frac{\mathcal{L}(w^*)+\varepsilon}{\varepsilon}\right) \frac{\|w^*\|^4}{\varepsilon^3}$	

Runtime Analysis

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

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

	Overall Runtime	$\varepsilon = \Omega(\mathscr{L}(w^*))$
SBP	$\left(\frac{\mathscr{L}(w^*)+\varepsilon}{\varepsilon}\right)^3 \frac{\|w^*\|^4}{\varepsilon}$	$\frac{\|w^*\|^4}{\varepsilon}$
Dual Decomp.	$\left(\frac{\mathscr{L}(w^*)+\varepsilon}{\varepsilon}\right)^2 \frac{\|w^*\|^4}{\varepsilon^2}$	$\frac{\|w^*\|^4}{\varepsilon^2}$
Pegasos	$\left(\frac{\mathscr{L}(w^*)+\varepsilon}{\varepsilon}\right) \frac{\|w^*\|^4}{\varepsilon^3}$	$\frac{\|w^*\|^4}{\varepsilon^3}$

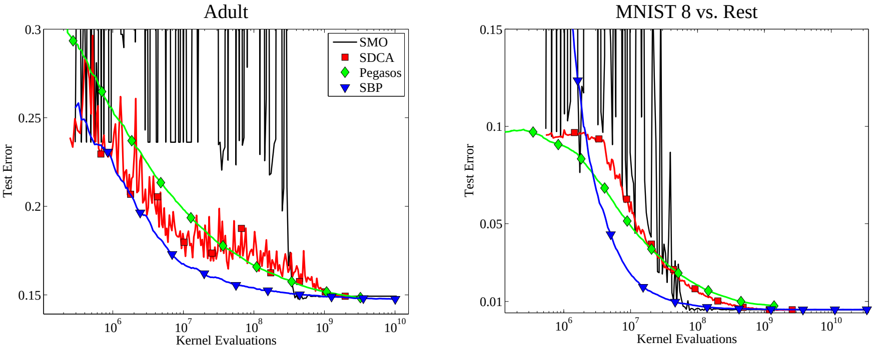
Runtime Analysis

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

Kernel Algo.	Iterations	Time per Iteration	Runtime
SBP	$\|w^*\|^2$	$n = \frac{\|w^*\|^2}{\varepsilon}$	$\frac{\|w^*\|^4}{\varepsilon}$
Dual Decomp.	$\frac{\|w^*\|^2}{\varepsilon}$	$n = \frac{\|w^*\|^2}{\varepsilon}$	$\frac{\|w^*\|^4}{\varepsilon^2}$
Pegasos	$\frac{\|w^*\|^2}{\varepsilon^2}$	$n = \frac{\|w^*\|^2}{\varepsilon}$	$\frac{\|w^*\|^4}{\varepsilon^3}$

Linear Algo.	Iterations	Time per Iteration	Runtime
SBP	$\|w^*\|^2$	$dn = \frac{d\|w^*\|^2}{\varepsilon}$	$\frac{d\|w^*\|^4}{\varepsilon}$
Dual Decomp.	$\frac{\|w^*\|^2}{\varepsilon}$	$dn = \frac{d\|w^*\|^2}{\varepsilon}$	$\frac{d\|w^*\|^4}{\varepsilon^2}$
Pegasos	$\frac{\|w^*\|^2}{\varepsilon^2}$	d	$\frac{d\|w^*\|^2}{\varepsilon^2}$

Experiments



- SMO makes little progress until it suddently eners a regime in which it converges rapidly
- Non-SMO algorithms converge gradually
- SMO: Platt (1998). SDCA: Hsieh et al. (2008). Pegasos: Shalev-Shwartz et al. (2007)

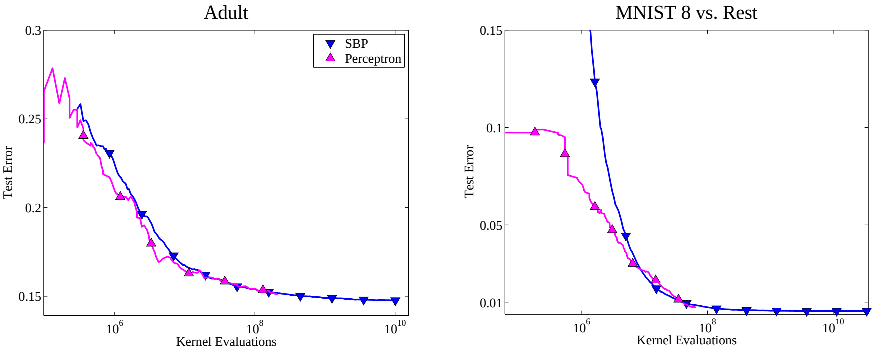
Summary

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

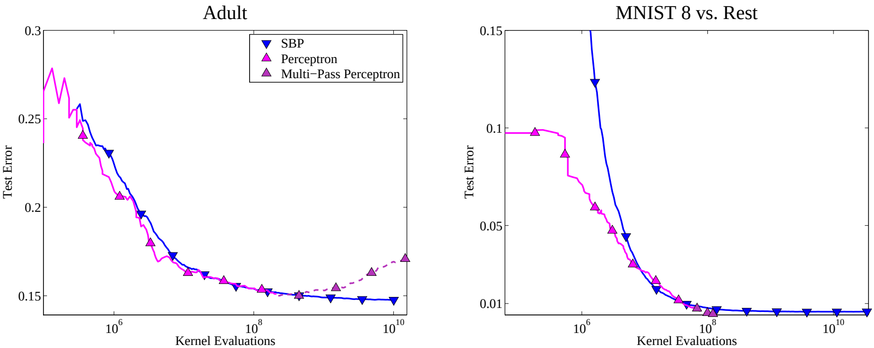
ttic.uchicago.edu/~cotter/projects/SBP

Experiments - Perceptron



- 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