



# Primal estimated sub-gradient solver for SVM

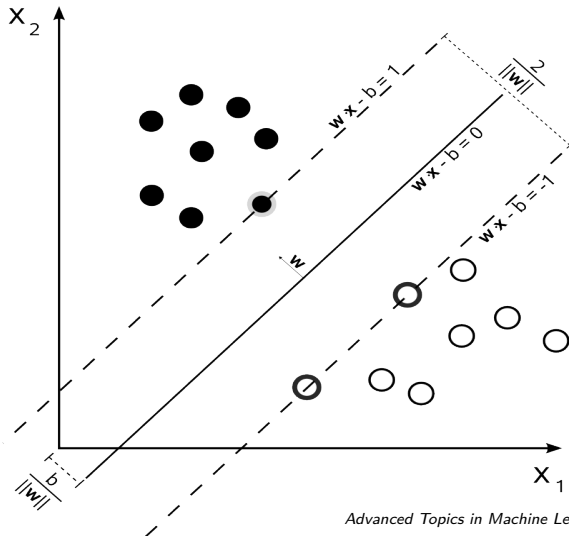
Lei Zhong

Advanced Topics in Machine Learning

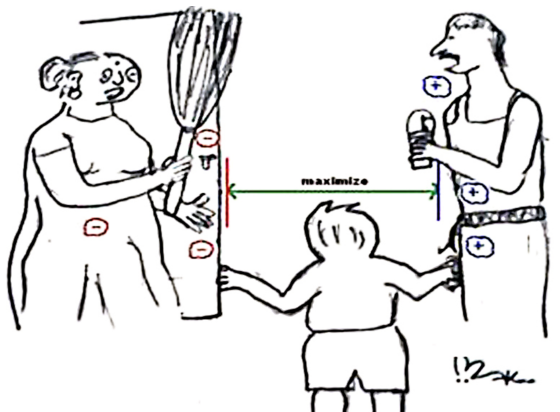
Nov. 4, 2014

- 1 Introduction
- 2 Analysis - faster convergence rates
- 3 Experiments - outperforms state-of-the-art
- 4 Reference

## Motivating example



## Family Support Machine



From Peter Richtárik's slides

## Support Vector Machine: Primal Problem

Data:

$$\{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{+1, -1\} : i \in S \stackrel{\text{def}}{=} \{1, 2, \dots, n\}\}$$

- ▶ Example:  $\mathbf{x}_1, \dots, \mathbf{x}_n$  (assumption:  $\max_i \|\mathbf{x}_i\|_2 \leq R$ )
- ▶ Labels:  $y_i \in \{+1, -1\}$

Optimization formulation of SVM:

$$\min_{\mathbf{w} \in \mathbb{R}^d} \{f(\mathbf{w}) := \hat{L}_S(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|^2\}$$

where

- ▶  $\hat{L}_A(\mathbf{w}) \stackrel{\text{def}}{=} \frac{1}{|A|} \sum_{i \in A} L_i$  (average loss on examples in A)

## Loss Function and Subgradient

### Definition

- Loss:  $L_i := \ell(\langle \mathbf{x}_i, \mathbf{w} \rangle, y_i)$
- Subgradient:  $l'(\langle \mathbf{x}_i, \mathbf{w} \rangle, y_i)$  with  $\|l'\| \leq \mathbb{L}$

Use the notation  $z = \langle \mathbf{w}, \mathbf{x}_i \rangle$ , sample loss functions:

Loss function	Subgradient
$l(z, y_i) = \max\{0, 1 - y_i z\}$	$l' = \begin{cases} -y_i \mathbf{x}_i & \text{if } y_i z < 1 \\ 0 & \text{otherwise} \end{cases}$
$l(z, y_i) = \log(1 + e^{-y_i z})$	$l' = -\frac{y_i}{1 + e^{y_i z}} \mathbf{x}_i$
$l(z, y_i) = \max\{0,  y_i - z  - \epsilon\}$	$l' = \begin{cases} \mathbf{x}_i & \text{if } z - y_i > \epsilon \\ -\mathbf{x}_i & \text{if } y_i - z > \epsilon \\ 0 & \text{otherwise} \end{cases}$

## Previous Work

- Dual-based methods
  - Interior Point
    - Memory:  $n^2$ , time:  $n^3 \log(\log(1/\epsilon))$ , run time per iteration  $\log(\log(1/\epsilon))$
  - Decomposition
    - Memory:  $n$ , time: super-linear in  $n$
- Online learning & Stochastic Gradient
  - Memory:  $O(1)$ , time:  $1/\epsilon^2$  (linear kernel), run-time per iteration:  $O(d)$

Better rates for finite dimensional instances (Murata, Bottou)

Typically, online learning algorithms do not converge to the optimal solution of SVM

## Basic Pegasos Algorithm (SGD)

- ① Choose  $\mathbf{w}_1 = \mathbf{0} \in \mathbb{R}^d$
- ② Iterate for  $t = 1, 2, \dots, T$ 
  - ① Choose  $A_t \subset S = \{1, 2, \dots, n\}$ ,  $|A_t| = m$ , uniformly at random
  - ② Set stepsize  $\eta_t \leftarrow \frac{1}{\lambda_t}$
  - ③ Update  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta_t \partial f_{A_t}(\mathbf{w}^{(t)})$

### Theorem

For  $\bar{\mathbf{w}} = \frac{1}{T} \sum_{t=1}^T \mathbf{w}_t$ , we have:

$$\mathbb{E}[f(\bar{\mathbf{w}})] \leq f(\mathbf{w}^*) + c \cdot \frac{1 + \ln(T)}{2\lambda T}$$

where  $c = 4R^2$ .



## Basic Pegasos Algorithm (SGD)

- ① Choose  $\mathbf{w}_1 = \mathbf{0} \in \mathbb{R}^d$
- ② Iterate for  $t = 1, 2, \dots, T$ 
  - ① Choose  $A_t \subset S = \{1, 2, \dots, n\}$ ,  $|A_t| = m$ , uniformly at random
  - ② Set stepsize  $\eta_t \leftarrow \frac{1}{\lambda t}$
  - ③ Update  $\mathbf{w}^{(t+1)} \leftarrow (1 - \eta \lambda) \mathbf{w}^{(t)} + \frac{\eta_t}{m} \sum_{i \in A_t} l'_i \mathbf{x}_i$

### Theorem

For  $\bar{\mathbf{w}} = \frac{1}{T} \sum_{t=1}^T \mathbf{w}_t$ , we have:

$$\mathbb{E}[f(\bar{\mathbf{w}})] \leq f(\mathbf{w}^*) + c \cdot \frac{1 + \ln(T)}{2\lambda T}$$

where  $c = 4R^2$ .

## Run-Time of Pegasos

- Choosing  $|A_t| = 1$   
→ Run-time required for Pegasos to find  $\epsilon$  accurate solution

$$\tilde{O}\left(\frac{d}{\lambda\epsilon}\right)$$

- Run-time does not depends on #examples, suited for learning from large datasets
- Depends on “difficulty” of problem (both  $\lambda$  and  $\epsilon$ )

## Analysis from Lacoste-Julien et.al.[2]

- Classical analysis:  $\eta_t = \frac{1}{\lambda t}$ 
  - $\mathbb{E}f\left(\frac{1}{T} \sum_{t=1}^T \mathbf{w}^{(t)}\right) - f(\mathbf{w}^*) \leq \frac{2\mathbb{L}^2 R^2}{\lambda T} (\ln T + 1)$
  - For Hinge loss  $X = 1$ , the result is same as before.
- New analysis:  $\eta_t = \frac{2}{\lambda(t+1)}$ 
  - $\mathbb{E}f\left(\frac{2}{T(T+1)} \sum_{t=1}^T t\mathbf{w}^{(t)}\right) - f(\mathbf{w}^*) \leq \frac{8\mathbb{L}^2 R^2}{\lambda(T+1)}$
  - $\mathbb{E}_{i(T)} [\|\mathbf{w}^{(T+1)} - \mathbf{w}^*\|^2 \|\mathbf{w}^t\|] \leq \frac{16\mathbb{L}^2 R^2}{\lambda^2(T+1)}$
  - In this case,  $\bar{\mathbf{w}}^{(T)} \doteq \frac{2}{T(T+1)} \sum_{t=1}^T t\mathbf{w}^{(t)}$

# Experiments

- 3 datasets (provided by Joachims)
  - Reuters CCAT (800K examples, 47k features)
  - Covertypes (581k examples, 54 features)
  - Physics ArXiv (62k examples, 100k features)
- 4 competing algorithms
  - SVM-Perf (Joachims'06)
  - SVM-light (Joachims)
  - Norma (Kivinen, Smola, Williamson '02)
  - Zhang'04 (stochastic gradient descent)

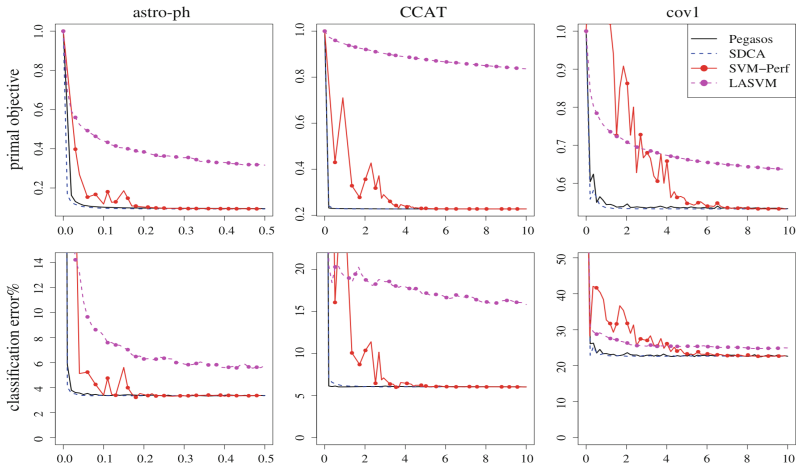
## Linear kernels

Dataset	Training	Testing	Features	Sparsity(%)	$\lambda$
astro-ph	29,882	32,487	99,757	0.08	$5 \times 10^{-5}$
CCAT	781,265	23,149	47,236	0.16	$10^{-4}$
cov1	522,911	58,101	54	22.22	$10^{-6}$

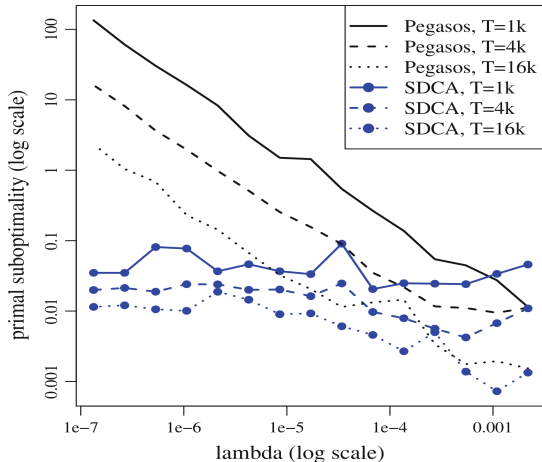
## Linear kernels

Dataset	Pegasos	SDCA	SVM-Perf	LASVM
astro-ph	0.04s(3.56%)	0.03s(3.49%)	0.1s(3.39%)	54s(3.65%)
CCAT	0.16s(6.16%)	0.36s(6.57%)	3.6s(5.93%)	>18000 s
cov1	0.32s(23.2%)	0.20s(22.9%)	4.2s(23.9%)	210s(23.8%)

# Comparison of linear SVM optimizers

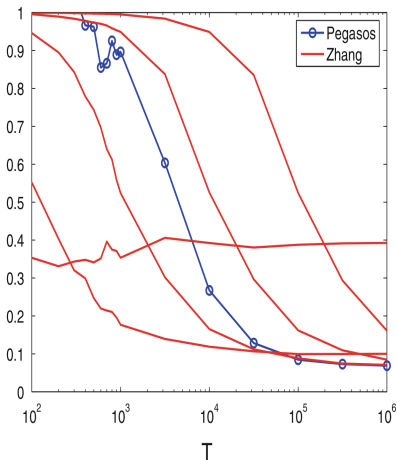
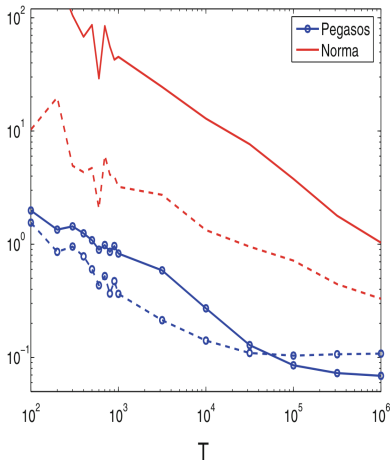


# Effect of regularization parameter $\lambda$

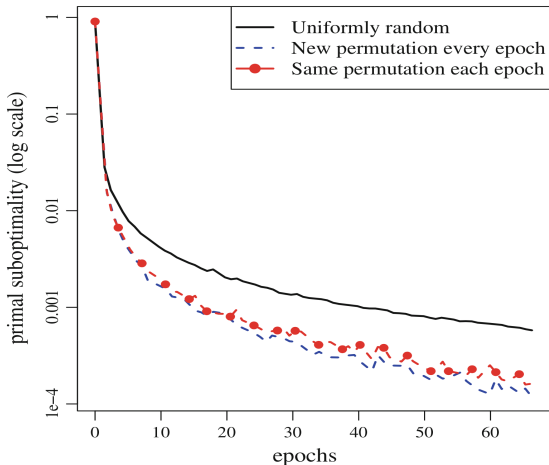




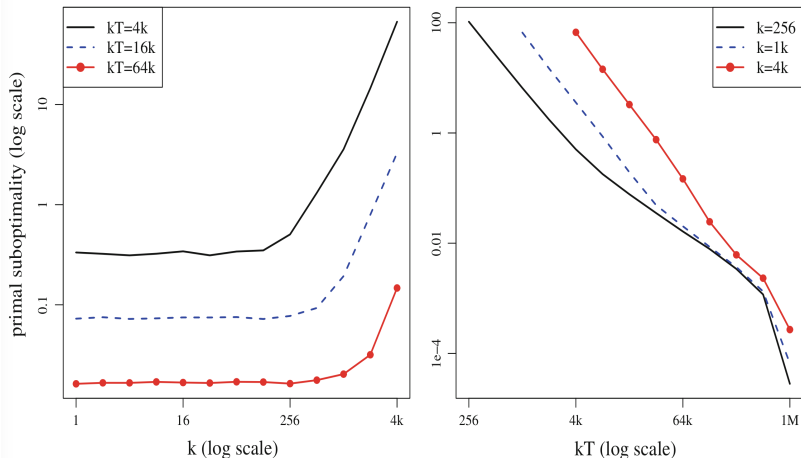
## Experiments with the mini-batch variant



## Comparison of sampling procedures



# Compare to Norma and Zhang (on Physics)



## Bias term

- ① Popular approach: increase dimension of  $\mathbf{x}$   
Cons: “pay” for  $b$  in the regularization term
- ② Define:  $L(\mathbf{w}) = \min_b \sum_{(\mathbf{x}, y) \in S} [1 - y(\langle \mathbf{w}, \mathbf{x} \rangle - b)]_+$
- ③ Rewrite problem:  $\min_{\mathbf{w}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + g(\mathbf{w}; S)$  where  
 $g(\mathbf{w}; S) = \min_b \frac{1}{m} \sum_{(\mathbf{x}, y) \in S} [1 - y(\langle \mathbf{w}, \mathbf{x} \rangle + b)]_+$   
Calculate subgradients w.r.t  $\mathbf{w}$  and w.r.t  $b$ .
- ④ Search  $b$  in an outer loop  
Cons: evaluation time remain same as unbiased

## Discussion

- Pegasos: Simple & Efficient solver for SVM
- Sample vs. computational complexity
  - Sample complexity: How many examples do we need as a function of  $\text{VC-dim}(\lambda)$ ,  $\text{accuracy}(\epsilon)$ , and  $\text{confidence}(\delta)$
  - in Pegasos, we aim at analyzing computational complexity based on  $\lambda$ ,  $\epsilon$ ,  $\delta$  (also in Bottou & Bousquet)
- Finding argmin vs. calculating min: It seems that Pegasos finds the argmin more easily than it requires to calculate the min value

## Q&A



# Thank You!

### Acknowledgement:

Thanks to Martin for helpful discussions, suggestions and chips!!!

## Reference

- [1] Shalev-Shwartz, Shai, et al. "Pegasos: Primal estimated sub-gradient solver for svm." *Mathematical programming* 127.1 (2011): 3-30.
- [2] Lacoste-Julien, Simon, Mark Schmidt, and Francis Bach. "A simpler approach to obtaining an  $o(1/t)$  convergence rate for the projected stochastic subgradient method." *arXiv preprint arXiv:1212.2002* (2012).
- [3] Takáč Martin, et al. "Mini-batch primal and dual methods for SVMs." *arXiv preprint arXiv:1303.2314* (2013).
- [4] Zhao, Peilin, and Tong Zhang. "Stochastic optimization with importance sampling." *arXiv preprint arXiv:1401.2753* (2014).