



Primal estimated sub-gradient solver for SVM

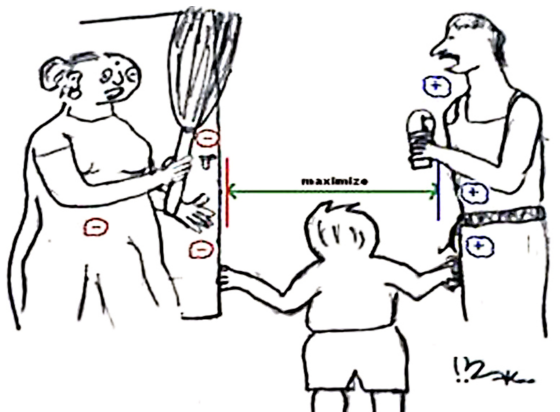
Lei Zhong

Advanced Topics in Machine Learning

Nov. 4, 2014

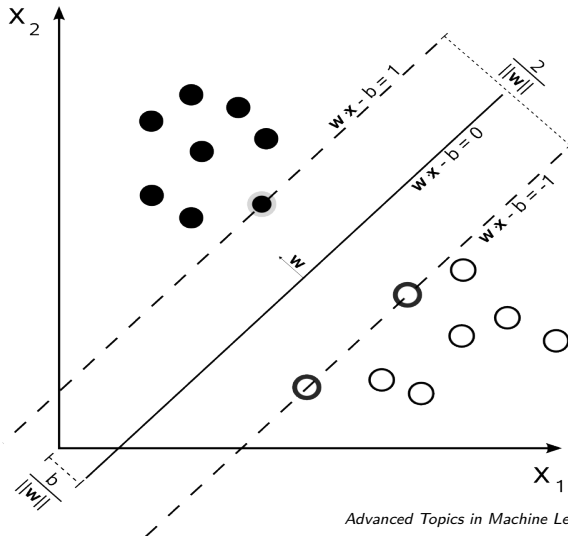
- 1 Introduction
- 2 Analysis - faster convergence rates
- 3 Experiments - outperforms state-of-the-art
- 4 Reference

Family Support Machine



From Peter Richtárik's slides

Motivating example



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 the assumption of $\|l'\| \leq \mathbb{L}$

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 n^3
 - Decomposition
 - Memory: n , time complexity: super-linear in n , slow convergence
- 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 $A_t \subset S = \{1, 2, \dots, n\}$, $|A_t| = k$, 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 $A_t \subset S = \{1, 2, \dots, n\}$, $|A_t| = k$, uniformly at random
 - ② Set stepsize $\eta_t \leftarrow \frac{1}{\lambda t}$
 - ③ Update $\mathbf{w}^{(t+1)} \leftarrow (1 - \eta \lambda) \mathbf{w}^{(t)} + \frac{\eta_t}{k} \sum_{i \in A_t} l'_i \mathbf{x}_i$

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 ϵ 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 λ and ϵ)

How to achieve this?

$$\begin{aligned}\|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|^2 &= \|\mathbf{w}^{(t)} - \eta_t \chi_i^{(t)} - \mathbf{w}^*\|^2 \\ &= \|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 + \eta_t^2 \|\chi_i^{(t)}\|^2 - 2\eta_t \chi_i^{(t)} (\mathbf{w}^{(t)} - \mathbf{w}^*)\end{aligned}$$

Taking the expectation on both sides over the recent step

$$\begin{aligned}\mathbb{E}_{i(t)}[\|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|^2 \mid \mathbf{w}^t] \\ &= \|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 + \eta_t^2 \mathbb{E}_{i(t)} \|\chi_i^{(t)}\|^2 - 2\eta_t \chi_i^{(t)} (\mathbf{w}^{(t)} - \mathbf{w}^*) \\ &\leq \|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 + \eta_t^2 \mathbb{E}_{i(t)} \|\chi_i^{(t)}\|^2 - 2\eta_t [f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*)] + \frac{\lambda}{2} \|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2.\end{aligned}$$

Re-arranging and taking expectation over the whole process again

$$\begin{aligned}\mathbb{E} f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*) &\leq \frac{\eta_t}{2} \mathbb{E} \|\chi_i^{(t)}\|^2 + \frac{1 - \lambda \eta_t}{2\eta_t} \mathbb{E} \|\mathbf{w}^{(t)} - \mathbf{w}^*\|^2 \\ &\quad - \frac{1}{2\eta_t} \mathbb{E} [\|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|^2 \mid \mathbf{w}^t]\end{aligned}$$

Lemma

With the Lipschitz assumption on $l(y, u)$ that $\|l'(y, u)\| \leq \mathbb{L}$, and assuming that $\|\mathbf{x}_i\| \leq R \ \forall i$ where i is picked according to p_i , it holds that

$$\mathbb{E}_{i(t)} \|\boldsymbol{\chi}_i^{(t)}\| \leq 4\mathbb{L}^2 R^2$$

where $l'(y, u)$ denotes any subgradient with respect to the second variable.

How to prove it?

Minkowski inequality

$$\sqrt{\mathbb{E}(X + Y)^2} \leq \sqrt{\mathbb{E}X^2} + \sqrt{\mathbb{E}Y^2}$$

$$\sqrt{\mathbb{E}_{i(t)} \|\mathbf{x}_i^{(t)}\|^2} \leq \sqrt{\mathbb{E}_{i(t)} \|l' \mathbf{x}_i\|^2} + \lambda \sqrt{\mathbb{E}_{i(1:t-1)} \|\mathbf{w}^{(t)}\|^2}$$

$$\sqrt{\mathbb{E}_{i(t)} \|\mathbf{w}^{(t+1)}\|^2} \leq (1 - \lambda \eta_t) \sqrt{\mathbb{E}_{i(t)} \|\mathbf{w}^{(t)}\|^2} + \eta_t \sqrt{\mathbb{E}_{i(t)} \|l' \mathbf{x}_i\|^2}$$

Why we don't need projection?

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 $\mathbb{L} = 1$, 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, $\bar{\mathbf{w}}^{(T)} \doteq \frac{2}{T(T+1)} \sum_{t=1}^T t\mathbf{w}^{(t)}$

Stochastic Dual Coordinate Ascent

- Dual problem

$$\max_{\alpha \in \mathbb{R}^n, 0 \leq \alpha_i \leq 1} D(\alpha) := \frac{1}{n} \sum_{i=1}^n \alpha_i - \frac{1}{2\lambda n^2} \sum_{i=1}^n \alpha_i^T \mathbf{Q} \alpha_i.$$

where

$$\mathbf{Q} \in \mathbb{R}^{n \times n}, \mathbf{Q}_{i,j} = y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle,$$

- Relationship with primal variable:

$$\mathbf{w}(\alpha) := \frac{1}{\lambda n} \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i.$$

- Traditional SDCA

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} + \delta_i^{(t)} e_i,$$

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)

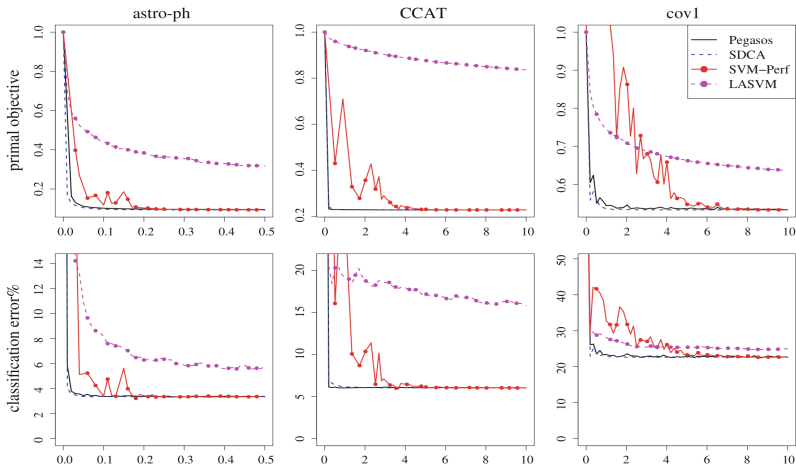
Dataset characteristics

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

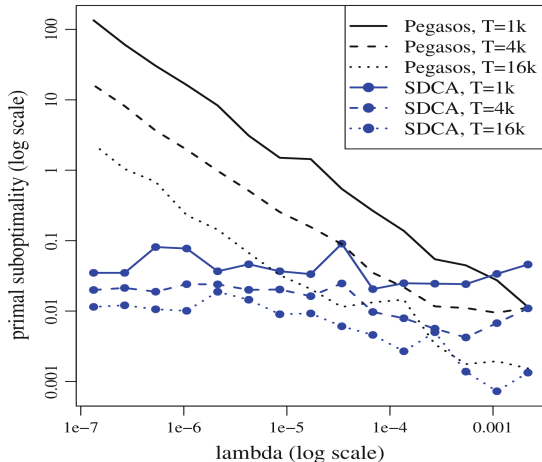
Training runtime and test error

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%)

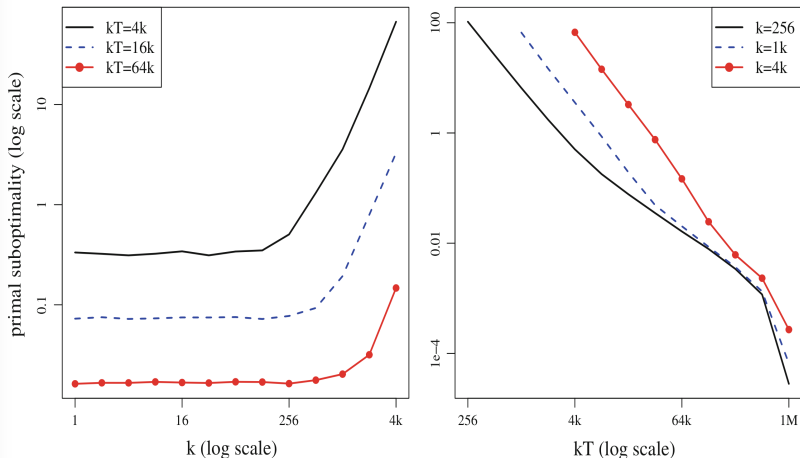
Comparison of linear SVM optimizers



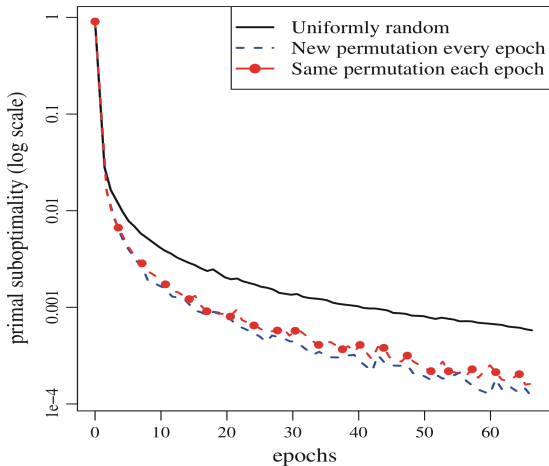
Effect of regularization parameter λ



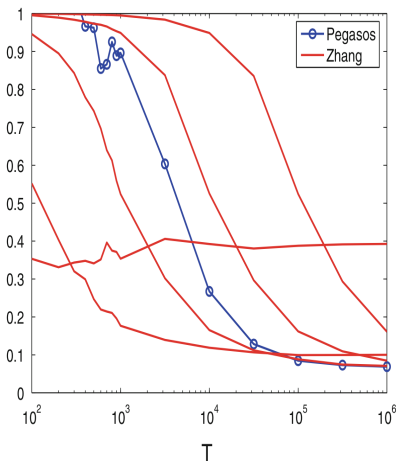
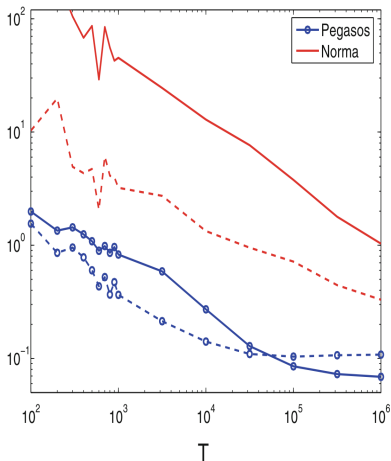
Experiments with the mini-batch variant



Comparison of sampling procedures



Compare to Norma and Zhang (on Physics)



Kernels

The basic Pegasos algorithm can easily be implemented using only kernel evaluations.

- For each t let $\alpha_{t+1} \in R^n$ be the vector such that $\alpha_{t+1}[j]$ counts how many times example j has been selected so far and we had a non-zero loss on it, namely,
$$\alpha_{t+1}[j] = |\{t' \leq t : i_{t'} = j \wedge y_j \langle \mathbf{w}_{t'}, \phi(\mathbf{x}_j) \rangle < 1\}|.$$
- Represent $\mathbf{w}_{t+1} = \frac{1}{\lambda t} \sum_{j=1}^m \alpha_{t+1}[j] y_j \phi(\mathbf{x}_j)$
- Cons: overall runtime $\tilde{O}(nd/(\lambda\epsilon))$

Summary

- Pegasos: Simple & Efficient solver for SVM
- Faster convergence rate
 - Choose of different step size
- Extension
 - Bound on ∇f is enough
 - Non-uniform sampling

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).

Q&A



Thank You!

Acknowledgement:

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