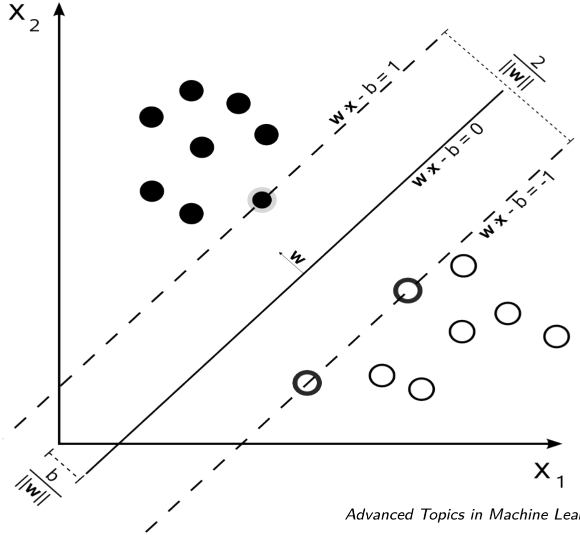
ETH zürich

Primal estimated sub-gradient solver for SVM

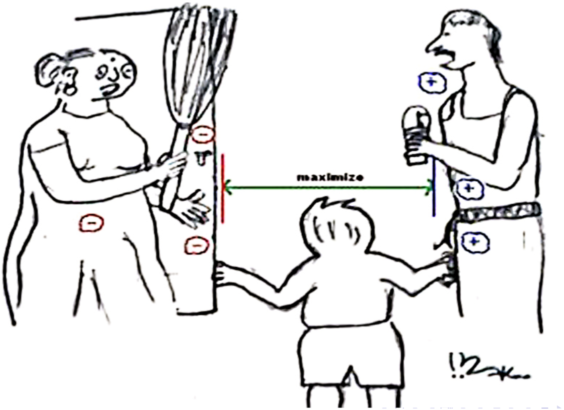
Lei Zhong
Advanced Topics in Machine Learning

Nov. 4, 2014

Motivating example



Family Support Machine



From Peter Richtárik's slides

Support Vector Machine: Primal Problem

Data:

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

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

Optimization formulation of SVM:

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

where

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

Loss Function and Subgradient

Definition

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

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

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

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

Theorem

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

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

where $c = 4R^2$.

Basic Pegasos Algorithm (SGD)

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

Theorem

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

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

where $c = 4R^2$.

Run-Time of Pegasos

- Choosing $|A_t| = 1$

  $\rightarrow$ Run-time required for Pegasos to find ϵ accurate solution

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

- Run-time does not depends on #examples, suited for learning form large datasets
- Depends on "difficulty" of problem (both λ and ϵ)

How to achieve this?

$\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*\|^2 = \|\boldsymbol{w}^{(t)} - \eta_t \boldsymbol{\chi}_i^{(t)} - \boldsymbol{w}^*\|^2 =$
$\|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|^2 + \eta_t^2 \|\boldsymbol{\chi}_i^{(t)}\|^2 - 2\eta_t \boldsymbol{\chi}_i^{(t)}(\boldsymbol{w}^{(t)} - \boldsymbol{w}^*)$
Taking the expectation on both sides

$$\mathbb{E}[\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*\|^2 \mid \boldsymbol{w}^t] = \|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|^2 + \eta_t^2 \mathbb{E}_{i^{(t)}} \|\boldsymbol{\chi}_i^{(t)}\|^2$$
$$-2\eta_t \boldsymbol{\chi}_i^{(t)}(\boldsymbol{w}^{(t)} - \boldsymbol{w}^*)$$
$$\leq \|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|^2 + \eta_t^2 \mathbb{E}_{i^{(t)}} \|\boldsymbol{\chi}_i^{(t)}\|^2 - 2\eta_t [f(\boldsymbol{w}^{(t)}) - f(\boldsymbol{w}^*) + \frac{\lambda}{2} \|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|^2].$$

Re-arranging and taking expectation again

$$\mathbb{E} f(\boldsymbol{w}^{(t)}) - f(\boldsymbol{w}^*) \leq \frac{\eta_t}{2} \mathbb{E} \|\boldsymbol{\chi}_i^{(t)}\|^2 + \frac{1 - \lambda \eta_t}{2\eta_t} \mathbb{E} \|\boldsymbol{w}^{(t)} - \boldsymbol{w}^*\|^2$$
$$-\frac{1}{2\eta_t} \mathbb{E}\left[\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^*\|^2 \mid \boldsymbol{w}^t\right]$$

Lemma

With the Lipschitz assumption on $l(y, u)$ that $\|l'(y, u)\| \leq X$, and assuming that $\|_i\| \leq R \; \forall i$ 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.

Minkowski inequality

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

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

$$\lambda\sqrt{\mathbb{E}_{i(1:t-1)}\|\boldsymbol{w}^{(t)}\|^2}.$$

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

$$\leq (1-\lambda\eta_t)\sqrt{\mathbb{E}_{i(t)}\|\boldsymbol{w}^{(t)}\|^2} + \lambda\eta_t\frac{XR}{\lambda}.$$

Why we don't need projection?

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

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

Stochastic Dual Coordinate Ascent

- Dual problem

$$max_\theta D(\theta) := \frac{1}{n}\sum_{i=1}^{n} -f_i^*(-\theta_i) - \lambda r^*(\frac{1}{\lambda n}\sum_{i=1}^{n}\theta_i).$$

- Relationship with primal variable: $\boldsymbol{w} = \nabla r^*(\boldsymbol{v}(\theta))$, $\boldsymbol{v}(\theta) = \frac{1}{\lambda n}\sum_{i=1}^{n}\theta_i$
- Traditional SDCA

$$\theta_i^t = \theta_i^{t-1} + \Delta\theta_i^{t-1}$$

Experiments

- 3 datasets (provided by Joachims)
  - Reuters CCAT (800K examples, 47k features)
  - Covertype (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)

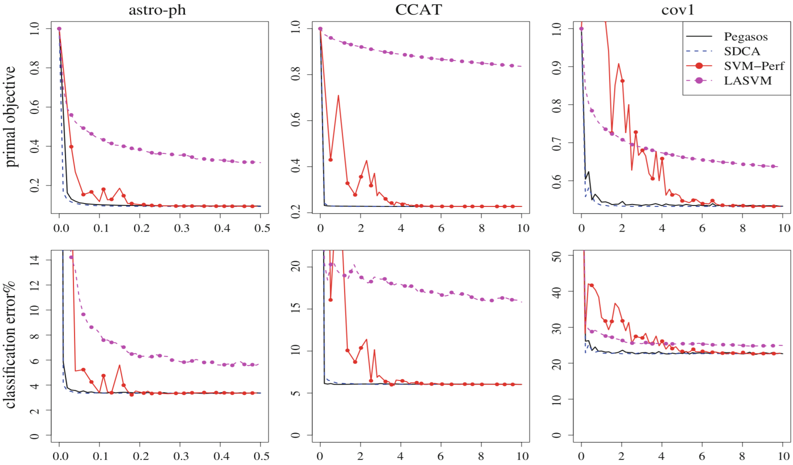
Linear kernels

Dataset	Training	Testing	Features	Sparsity(%)	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}

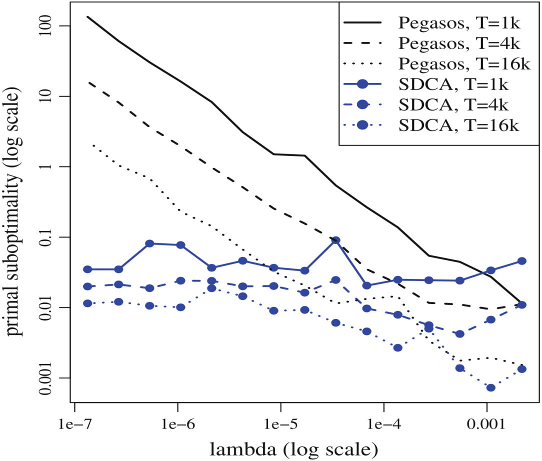
Linear kernels

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%)

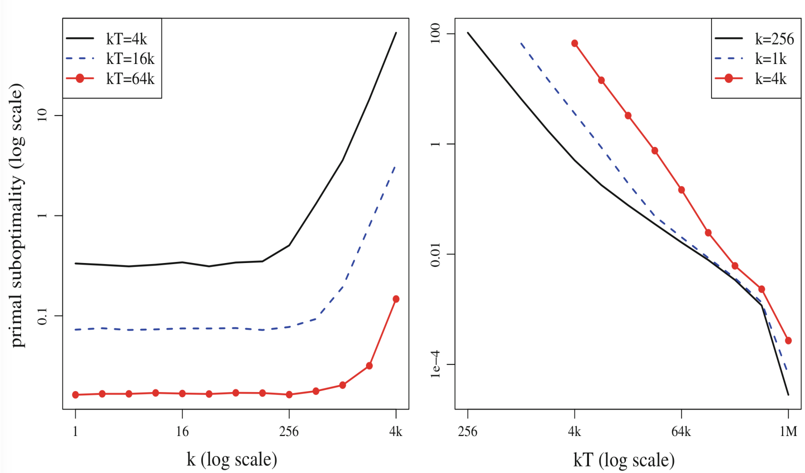
Comparison of linear SVM optimizers



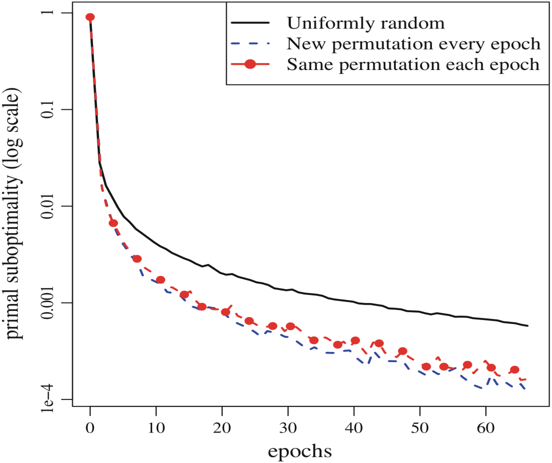
Effect of regularization parameter λ



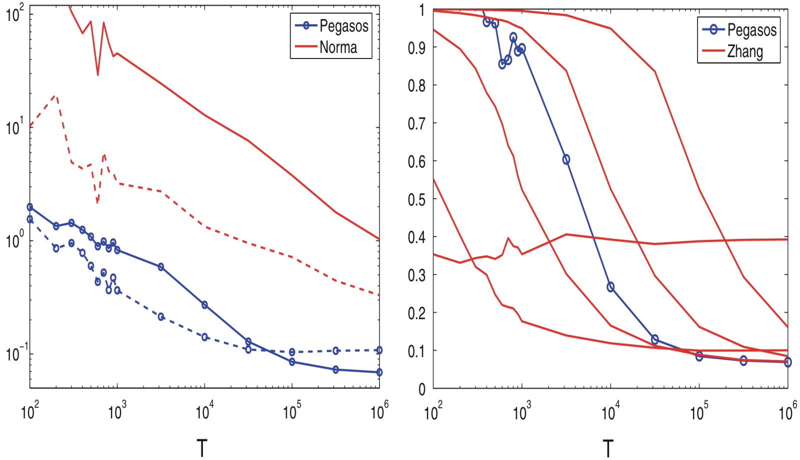
Experiments with the mini-batch variant



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,
  $\alpha_{t+1}[j] = \ |t' \leq t : i_{t'} = j \ \wedge \ y_j \langle \boldsymbol{w}_{t'}, \phi(\boldsymbol{x}_j) \rangle < 1|.$
- Represent $\boldsymbol{w}_{t+1} = \frac{1}{\lambda t} \sum_{j=1}^{m} \alpha_{t+1}[j] y_j \phi(\boldsymbol{x}_j)$
- Cons: overall runtime $\tilde{O}(md/(\lambda\epsilon))$

Discussion

- Pegasos: Simple & Efficient solver for SVM
- Sample vs. computational complexity
  - Sample complexity: How many examples do we need as a function of VC-dim(λ), accuracy(ϵ), and confidence(δ)
  - in Pegasos, we aim at analyzing computational complexity based on λ, ϵ, δ (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).