Support vector machines (SVMs)
Lecture 5

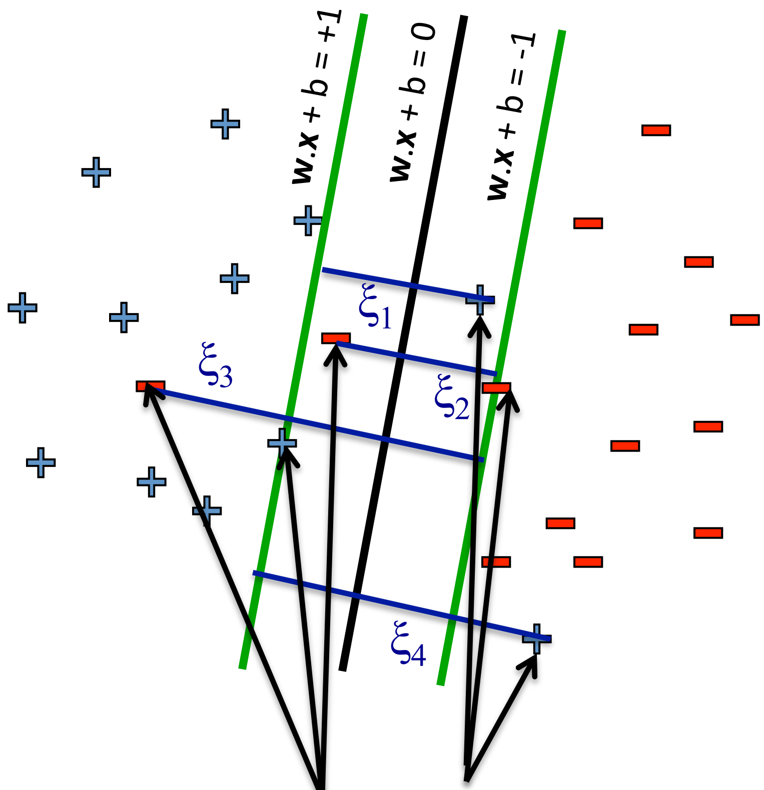
David Sontag
New York University

Soft margin SVM

$$\underset{\mathbf{w},\xi_i\geq 0}{\text{argmin}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i$$

$$\text{s.t.} \quad \forall i,\ y_i\langle \mathbf{w}, \mathbf{x}_i\rangle \geq 1 - \xi_i$$

"slack variables"



Slack penalty $C > 0$:

- $C=\infty$ → minimizes upper bound on 0-1 loss
- $C\approx 0$ → points with $\xi_i=0$ have big margin
- **Select using cross-validation**

Support vectors:

Data points for which the constraints are binding

Soft margin SVM

QP form:

$$\underset{\mathbf{w},\xi_i \geq 0}{\operatorname{argmin}} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i$$

$$\text{s.t.} \quad \forall i, \ y_i\langle \mathbf{w}, \mathbf{x}_i\rangle \geq 1 - \xi_i$$

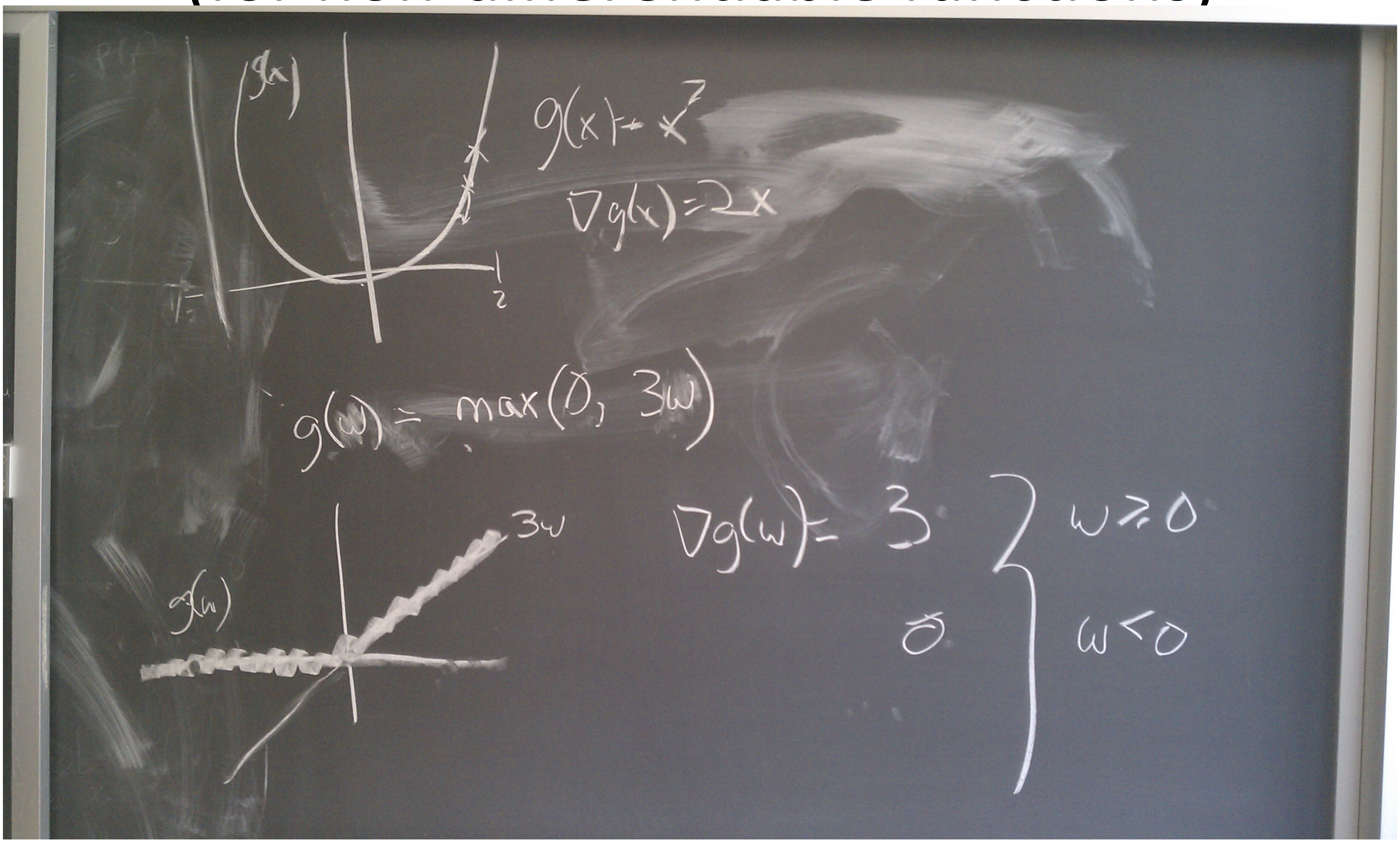
More "natural" form:

Equivalent if $C = \frac{1}{m\lambda}$

$\underset{\mathbf{w}}{\operatorname{argmin}} \ f(\mathbf{w})$ where:

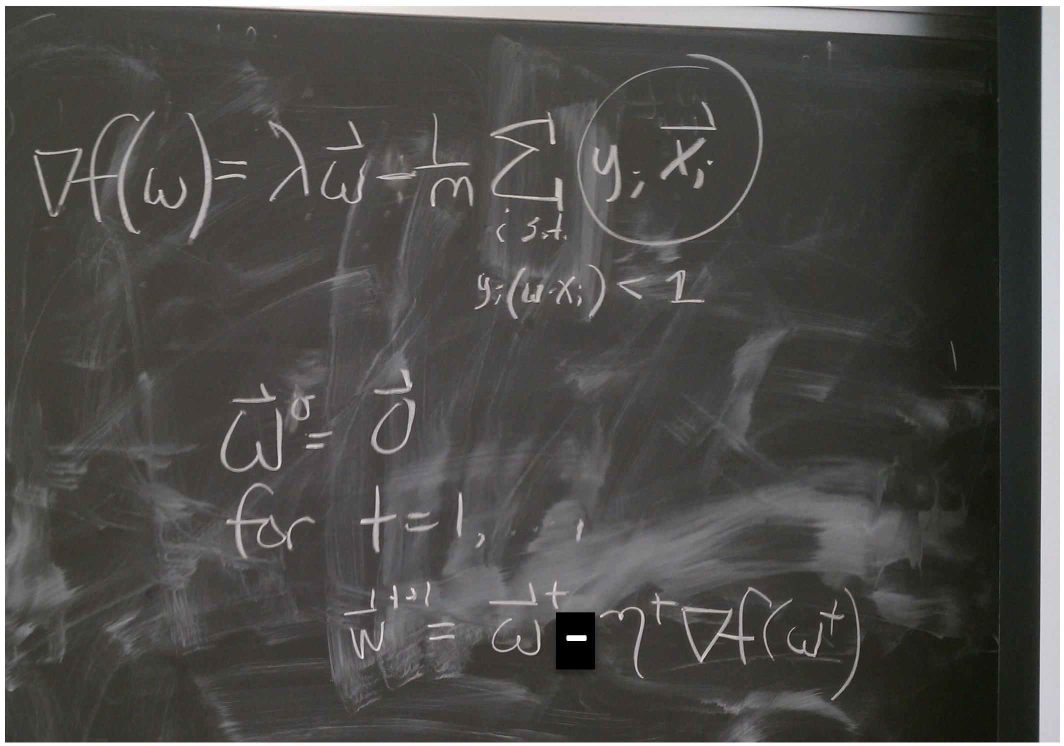
$$f(\mathbf{w}) \stackrel{\text{def}}{=} \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|^2}_{\text{Regularization term}} + \underbrace{\frac{1}{m}\sum_{i=1}^{m}\max\{0, 1 - y_i\langle \mathbf{w}, \mathbf{x}_i\rangle\}}_{\text{Empirical loss}}$$

Subgradient
(for non-differentiable functions)



(Sub)gradient descent of SVM objective

$$f(\mathbf{w}) \stackrel{\text{def}}{=} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i\langle \mathbf{w}, \mathbf{x}_i\rangle\}$$



Step size:

$$\eta_t = \frac{1}{t\lambda}$$

The Pegasos Algorithm

General framework

Initialize: $w_1 = 0$, $t=0$

While not converged

$t = t+1$

Choose a stepsize, η_t

Choose a direction, p_t

Go!

Test for convergence

Output: wt+1

Pegasos Algorithm (from homework)

Initialize: $w_1 = 0$, $t=0$

For iter = 1,2,…,20

For j=1,2,…,|data|

$t = t+1$

$\eta_t = 1/(t\lambda)$

If $y_j(w_t\, x_j) < 1$

$w_{t+1} = (1-\eta_t\lambda)\, w_t + \eta_t\, y_j\, x_j$

Else

$w_{t+1} = (1-\eta_t\lambda)\, w_t$

Output: wt+1

The Pegasos Algorithm

<u>General framework</u>

Initialize: $w_1 = 0$, $t=0$

While not converged

  $t = t+1$

  Choose a stepsize, η_t

  Choose a direction, p_t

  Go!

  Test for convergence

Output: wt+1

<u>Pegasos Algorithm (from homework)</u>

Initialize: $w_1 = 0$, $t=0$

For iter = 1,2,…,20

  For j=1,2,…,|data|

    $t = t+1$

    $\eta_t = 1/(t\lambda)$

    If $y_j(w_t\, x_j) < 1$

      $w_{t+1} = w_t - \eta_t(\lambda w_t - y_j x_j)$

    Else

      $w_{t+1} = w_t - \eta_t \lambda w_t$

Output: wt+1

The Pegasos Algorithm

<u>General framework</u>

Initialize: $w_1 = 0$, $t=0$

While not converged

$t = t+1$

Choose a stepsize, η_t

Choose a direction, p_t

Go!

Test for convergence

Output: wt+1

<u>Pegasos Algorithm (from homework)</u>

Initialize: $w_1 = 0$, $t=0$

For iter = 1,2,…,20

For j=1,2,…,|data|

$t = t+1$

$\eta_t = 1/(t\lambda)$

If $y_j(w_t\, x_j) < 1$

$w_{t+1} = w_t - \eta_t(\lambda w_t - y_j x_j)$

Else

$w_{t+1} = w_t - \eta_t \lambda w_t$

Output: wt+1

Convergence choice : Fixed number of iterations
T=20*|data|

The Pegasos Algorithm

General framework

Initialize: $w_1 = 0$, $t=0$

While not converged

  $t = t+1$

  Choose a stepsize, η_t

  Choose a direction, p_t

  Go!

  Test for convergence

Output: wt+1

Pegasos Algorithm (from homework)

Initialize: $w_1 = 0$, $t=0$

For iter = 1,2,…,20

  For j=1,2,…,|data|

    $t = t+1$

    $\eta_t = 1/(t\lambda)$

    If $y_j(w_t\, x_j) < 1$

      $w_{t+1} = w_t - \eta_t(\lambda w_t - y_j x_j)$

    Else

      $w_{t+1} = w_t - \eta_t \lambda w_t$

Output: wt+1

Stepsize choice: - Initialize with $1/\lambda$

- Decays with $1/t$

The Pegasos Algorithm

General framework

Initialize: $w_1 = 0$, $t=0$

While not converged

  $t = t+1$

  Choose a stepsize, η_t

  Choose a direction, *p_t*

  Go!

  Test for convergence

Output: wt+1

Pegasos Algorithm (from homework)

Initialize: $w_1 = 0$, $t=0$

For iter = 1,2,…,20

  For j=1,2,…,|data|

    $t = t+1$

    $\eta_t = 1/(t\lambda)$

    If $y_j(w_t\, x_j) < 1$

      $w_{t+1} = w_t - \eta_t(\boldsymbol{\lambda w_t - y_j x_j})$

    Else

      $w_{t+1} = w_t - \eta_t \boldsymbol{\lambda w_t}$

Output: wt+1

Direction choice: Stochastic approx to the subgradient

Subgradient calculation

Objective:

$$\frac{\lambda}{2}||w||^2 + \frac{1}{m}\sum_i \max\{0, 1 - y_i w \cdot x_i\}$$

Stochastic Approx:

$$\frac{\lambda}{2}||w||^2 + \max\{0, 1 - y_i w \cdot x_i\}$$

For a randomly chosen data point *i*

(in the assignment the choice of i is ***not random*** *- easier to debug and compare between students).*

Subgradient calculation

Objective:

$$\frac{\lambda}{2}||w||^2 + \frac{1}{m}\sum_i \max\{0, 1 - y_i w \cdot x_i\}$$

Stochastic Approx:

$$\frac{\lambda}{2}||w||^2 + \max\{0, 1 - y_i w \cdot x_i\}$$

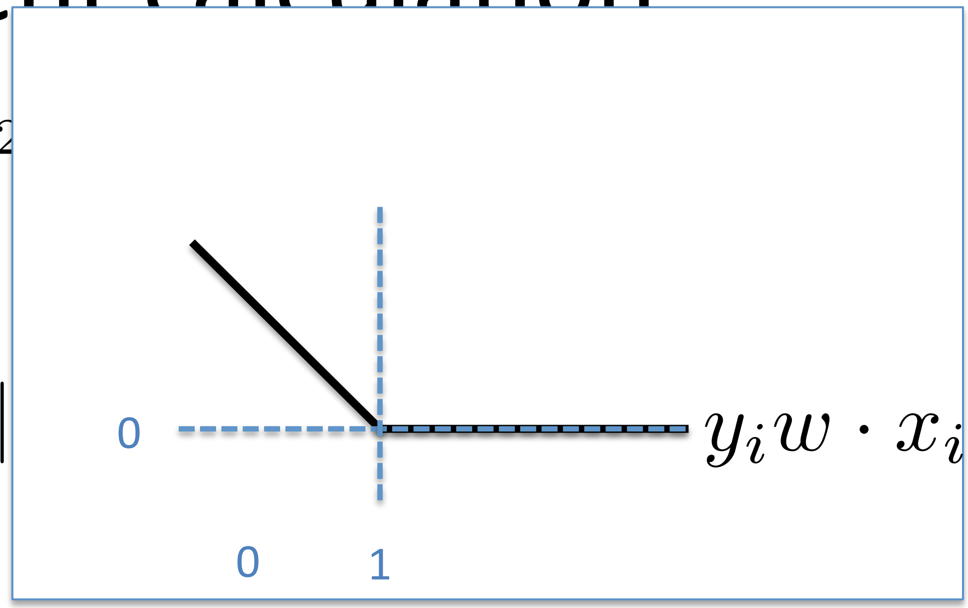
(sub)gradient:

$$\lambda||w|| + \frac{d}{dw}\max\{0, 1 - y_i w \cdot x_i\}$$

Subgradient calculation

Objective: $\frac{\lambda}{2}||w||^2$

Stochastic Approx: $\frac{\lambda}{2}||$



(sub)gradient:

$$\lambda||w|| + \frac{d}{dw}\max\{0, 1 - y_i w \cdot x_i\}$$

$y_i w \cdot x_i$

<1 → $-y_i x_i$

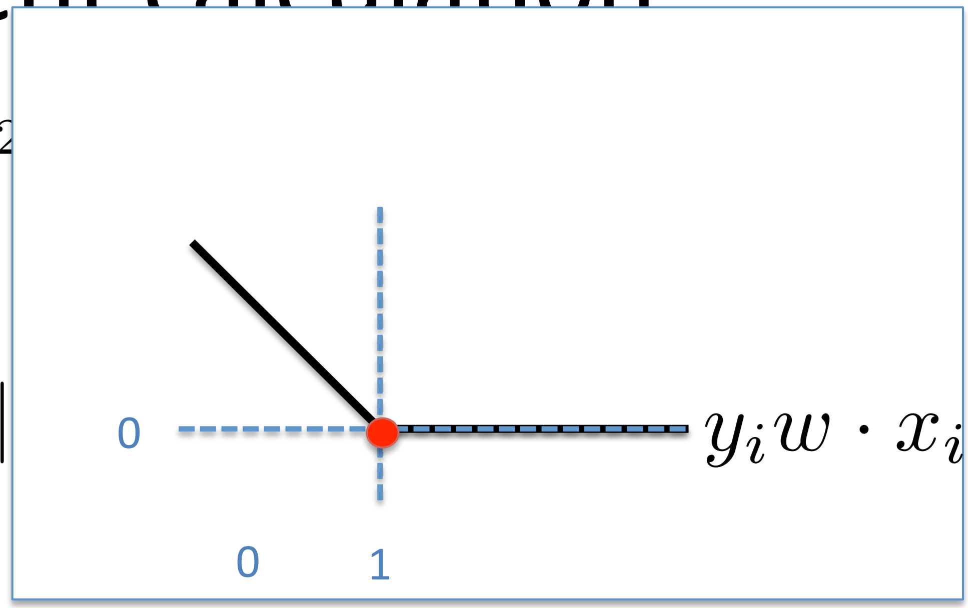
=1

>1 → 0

Subgradient calculation

Objective: $\frac{\lambda}{2}||w||^2$

Stochastic Approx: $\frac{\lambda}{2}||$



(sub)gradient:

$$\lambda||w|| + \frac{d}{dw}\max\{0, 1 - y_i w \cdot x_i\}$$

$y_i w \cdot x_i$

<1 | =1 | >1

$-y_i x_i$ | 0 | 0

Subgradient calculation

Objective:

$$\frac{\lambda}{2}||w||^2 + \frac{1}{m}\sum_i \max\{0, 1 - y_i w \cdot x_i\}$$

Stochastic Approx:

$$\frac{\lambda}{2}||w||^2 + \max\{0, 1 - y_i w \cdot x_i\}$$

(sub)gradient:

$$\text{if } y_i w \cdot x_i < 1 \qquad \lambda w - y_i x_i$$

$$\text{else} \qquad \lambda w + 0$$

The Pegasos Algorithm

General framework

Initialize: $w_1 = 0$, $t=0$

While not converged

- $t = t+1$
- Choose a stepsize, η_t
- **Choose a direction, *p_t***
- Go!
- Test for convergence

Output: wt+1

Pegasos Algorithm (from homework)

Initialize: $w_1 = 0$, $t=0$

For iter = 1,2,…,20

- For **j=1,2,…,|data|**
  - $t = t+1$
  - $\eta_t = 1/(t\lambda)$
  - **If $y_j(w_t\, x_j) < 1$**
    - $w_{t+1} = w_t - \eta_t(\lambda w_t - y_j x_j)$
  - **Else**
    - $w_{t+1} = w_t - \eta_t(\lambda wt + 0)$

Output: wt+1

Direction choice: Stochastic approx to the subgradient

$$\text{if } y_i w \cdot x_i < 1 \qquad \lambda w - y_i x_i$$

$$\text{else} \qquad \lambda w + 0$$

The Pegasos Algorithm

General framework

Initialize: $w_1 = 0$, t=0

While not converged

  $t = t+1$

  Choose a stepsize, η_t

  Choose a direction, p_t

  Go!

  Test for convergence

Output: wt+1

Pegasos Algorithm (from homework)

Initialize: $w_1 = 0$, t=0

For iter = 1,2,…,20

  For j=1,2,…,|data|

    $t = t+1$

    $\eta_t = 1/(t\lambda)$

    If $y_j(w_t\, x_j) < 1$

      $\mathbf{w_{t+1} = w_t - \eta_t(\lambda w_t - y_j x_j)}$

    Else

      $\mathbf{w_{t+1} = w_t - \eta_t \lambda w_t}$

Output: wt+1

Go: update $w_{t+1} = w_t - \eta_t p_t$

Why is this algorithm interesting?

- Simple to implement, state of the art results.
  - Notice similarity to Perceptron algorithm! **Algorithmic differences:** updates if insufficient margin, scales weight vector, and has a learning rate.

- Since based on *stochastic* gradient descent, its running time guarantees are probabilistic.

- Highlights interesting tradeoffs between running time and data.

Much faster than previous methods

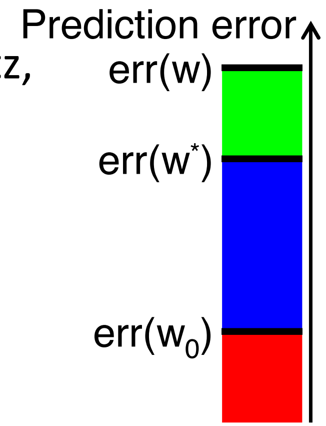
- **3 datasets** (provided by Joachims)
  - Reuters CCAT (800K examples, 47k features)
  - Physics ArXiv (62k examples, 100k features)
  - Covertype (581k examples, 54 features)

Training Time (in seconds):

	Pegasos	SVM-Perf	SVM-Light
Reuters	**2**	77	20,075
Covertype	**6**	85	25,514
Astro-Physics	**2**	5	80

Approximate algorithms

[Shalev Schwartz, Srebro '08]



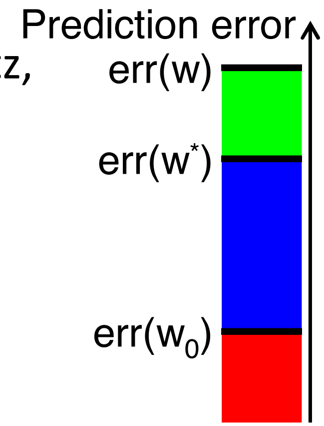
Note: w_0 is redefined in this context (see below) – does *not* refer to initial weight vector

- Approximation error:
  - Best error achievable by large-margin predictor
  - Error of population minimizer
    $w_0 = \text{argmin } E[f(w)] = \text{argmin } \lambda|w|^2 + E_{x,y}[\text{loss}(\langle w,x\rangle;y)]$
- Estimation error:
  - Extra error due to replacing E[loss] with empirical loss
    $w^* = \text{arg min } f_n(w)$
- Optimization error:
  - Extra error due to only optimizing to within finite precision

From ICML'08 presentation (available here)

Approximate algorithms

[Shalev Schwartz, Srebro '08]



- Approximation error:
  - Best error achievable by large-marg
  - Error of population minimizer
    w_0 = argmin E[f(w)] = argmin $\lambda|w|^2$
- Estimation error:
  - Extra error due to replacing E[loss]
    w^* = arg min $f_n(w)$
- Optimization error:
  - Extra error due to only optimizing to

Pegasos Guarantees

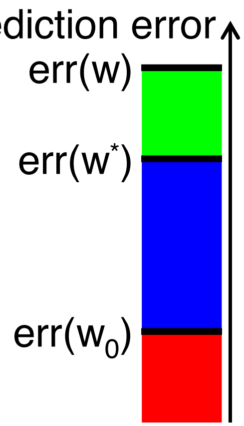
After $T = \tilde{O}\left(\frac{1}{\delta\lambda\epsilon}\right)$ updates:

$$\mathrm{err}(w_T) < \mathrm{err}(w_0) + \epsilon$$

With probability 1- δ

Approximate algorithms

[Shalev Schwartz, Srebro '08]



Running time does **NOT** depend on:

-# training examples!

It **DOES** depend on:

- Dimensionality *d* (why?)
- Approximation ϵ and δ
- Difficulty of problem λ

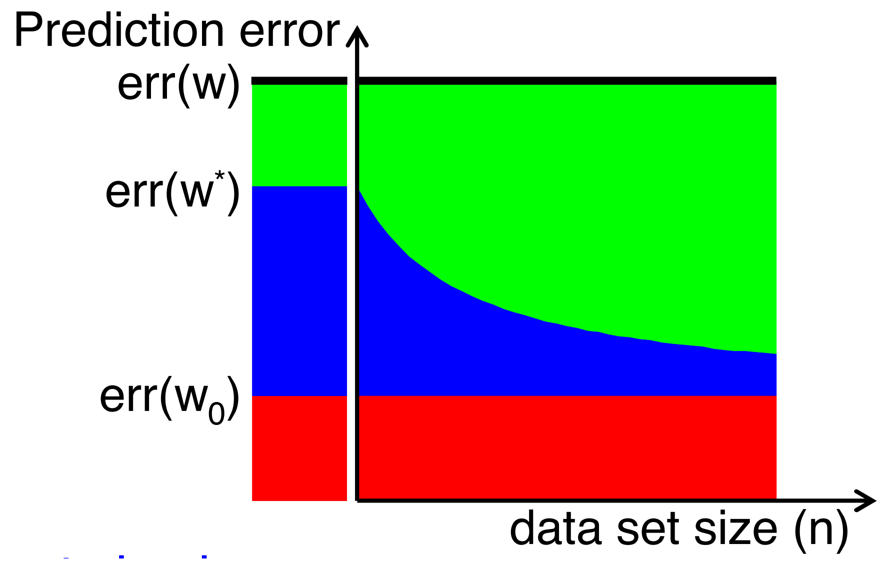
Pegasos Guarantees

After $T = \tilde{O}\left(\frac{1}{\delta\lambda\epsilon}\right)$ updates:

$$err(w_T) < err(w_0) + \epsilon$$

With probability 1- δ

But how is that possible?



[Shalev Schwartz, Srebro ’08]

As the dataset grows, our approximations can be worse to get the same error!