Support vector machines (SVMs)
Lecture 5

David Sontag
New York University
So5

margin
SVM

\[ w \cdot x + b = +1 \]
\[ w \cdot x + b = -1 \]
\[ w \cdot x + b = 0 \]

Slack penalty \( C > 0 \):
- \( C=\infty \) \( \rightarrow \) minimizes upper bound on 0-1 loss
- \( C\approx0 \) \( \rightarrow \) points with \( \xi_i=0 \) have big margin
- Select using cross-validation

Support vectors:
Data points for which the constraints are binding

\[
\arg\min_{w,\xi_i \geq 0} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \xi_i \\
\text{s.t. } \forall i, \ y_i \langle w, x_i \rangle \geq 1 - \xi_i
\]

“slack variables”
Soft margin SVM

QP form:

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

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

More “natural” form:

$$\arg\min_{\mathbf{w}} f(\mathbf{w}) \quad \text{where:}$$

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

Equivalent if

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

Regularization term

Empirical loss
Subgradient
(for non-differentiable functions)
(Sub)gradient descent of SVM objective

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

\[ \nabla F(w) = \lambda \hat{w} - \frac{1}{m} \sum_{i \in S_t} \frac{y_i}{x_i} \]

\[ \vec{w}^0 = 0 \]

\[ \text{for } t = 1, \ldots, \]

\[ \vec{w}^{t+1} = \vec{w}^t - \eta_t \nabla F(w^t) \]

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, $\eta_t$
- Choose a direction, $p_t$
- Go!
- Test for convergence

**Output:** $w_{t+1}$

**Pegasos Algorithm (from homework)**

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

For iter = 1,2,...,20

For $j=1,2,...,|\text{data}|$
- $t = t+1$
- $\eta_t = 1/(t \lambda)$
- 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:** $w_{t+1}$
The Pegasos Algorithm

General framework

Initialize: \( w_1 = 0, \ t=0 \)

While not converged
  \( t = t+1 \)
  Choose a stepsize, \( \eta_t \)
  Choose a direction, \( p_t \)
  Go!
  Test for convergence

Output: \( w_{t+1} \)

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

Output: \( w_{t+1} \)
The Pegasos Algorithm

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

While not converged
  $t = t+1$
  Choose a stepsize, $\eta_t$
  Choose a direction, $p_t$
  Go!
  Test for convergence

Output: $w_{t+1}$

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: $w_{t+1}$

Convergence choice: Fixed number of iterations
$T=20*|\text{data}|$
The Pegasos Algorithm

General framework

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

While not converged
  
  $t = t+1$
  
  Choose a stepsize, $\eta_t$
  
  Choose a direction, $p_t$
  
  Go!
  
  Test for convergence

Output: $w_{t+1}$

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: $w_{t+1}$

Stepsise 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, \( \eta_t \)
  **Choose a direction,** \( p_t \)
  Go!
  Test for convergence

**Output:** \( w_{t+1} \)

**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:** \( w_{t+1} \)

**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:
\[ \frac{\lambda}{2} ||w|| + \frac{d}{dw} \max\{0, 1 - y_i w \cdot x_i\} \]

(sub)gradient:
\[ \lambda ||w|| + \frac{d}{dw} \max\{0, 1 - y_i w \cdot x_i\} \]

Diagram:
- \( y_i w \cdot x_i < 1 \)
- \( y_i w \cdot x_i = 1 \)
- \( y_i w \cdot x_i > 1 \)
- \( -y_i x_i \)
- \( 0 \)
**Subgradient calculation**

**Objective:**
\[
\frac{\lambda}{2} ||w||^2
\]

**Stochastic Approx:**
\[
\frac{\lambda}{2} ||w||^2 + \frac{d}{dw} \max \{0, 1 - y_i w \cdot x_i\}
\]

**{(sub)gradient}**:
\[
\lambda ||w|| + \frac{d}{dw} \max \{0, 1 - y_i w \cdot x_i\}
\]

- \(y_i w \cdot x_i < 1\):
  - \(-y_i x_i\)
- \(y_i w \cdot x_i = 1\):
  - \(0\)
- \(y_i w \cdot x_i > 1\):
  - \(0\)
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:**

if \( y_i w \cdot x_i < 1 \) \( \lambda w - y_i x_i \)

else \( \lambda w + 0 \)
The Pegasos Algorithm

General framework

**Initialize:** \( w_1 = 0, \ t=0 \)

While not converged
  \( t = t+1 \)
  Choose a stepsize, \( \eta_t \)
  **Choose a direction,** \( p_t \)
  Go!
  Test for convergence

**Output:** \( w_{t+1} \)

Pegasos Algorithm (from homework)

**Initialize:** \( w_1 = 0, \ t=0 \)

**For** iter = 1,2,...,20
  **For** \( j=1,2,\ldots,|\text{data}| \)
    \( t = t+1 \)
    \( \eta_t = 1/(t\lambda) \)
    If \( y_j(w_t \cdot x_j) < 1 \)
      \( w_{t+1} = w_t - \eta_t(\lambda w_t - y_jx_j) \)
    Else
      \( w_{t+1} = w_t - \eta_t(\lambda w_t + 0) \)

**Output:** \( w_{t+1} \)

**Direction choice:** Stochastic approx to the subgradient

\[
\text{if } y_i w \cdot x_i < 1 \quad \lambda w - y_i x_i \\
\text{else} \quad \lambda w + 0
\]
The Pegasos Algorithm

General framework

Initialize: \( w_1 = 0, \ t=0 \)

While not converged
  \( t = t+1 \)
  Choose a stepsize, \( \eta_t \)
  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: \( wt+1 \)

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)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Pegasos</th>
<th>SVM-Perf</th>
<th>SVM-Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reuters</td>
<td>2</td>
<td>77</td>
<td>20,075</td>
</tr>
<tr>
<td>Covertype</td>
<td>6</td>
<td>85</td>
<td>25,514</td>
</tr>
<tr>
<td>Astro-Physics</td>
<td>2</td>
<td>5</td>
<td>80</td>
</tr>
</tbody>
</table>
Approximate algorithms

Approximation error:
- Best error achievable by large-margin predictor
- Error of population minimizer
  \[ w_0 = \text{arg min } E[f(w)] = \text{arg min } \lambda \|w\|^2 + E_{x,y}[\text{loss}(\langle w, x \rangle; y)] \]

Estimation error:
- Extra error due to replacing \( E[\text{loss}] \) with empirical loss
  \[ w^* = \text{arg min } f_n(w) \]

Optimization error:
- Extra error due to only optimizing to within finite precision

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

From ICML’08 presentation (available [here](#))
Approximate algorithms

Approximation error:
- Best error achievable by large-margin predictor
- Error of population minimizer
  \[ w_0 = \text{argmin } E[f(w)] = \text{argmin } \lambda |w|^2 \]

Estimation error:
- Extra error due to replacing \( E[\text{loss}] \) with empirical loss
  \[ w^* = \text{arg min } f_n(w) \]

Optimization error:
- Extra error due to only optimizing to within finite precision

Prediction error

Pegasos Guarantees

After
\[ T = \tilde{O} \left( \frac{1}{\delta \lambda \epsilon} \right) \]
updates:
\[ \text{err}(w_T) < \text{err}(w_0) + \epsilon \]

With probability \( 1 - \tilde{\delta} \)

[Shalev Schwartz, Srebro '08]
Approximate algorithms

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

Prediction error

Pegasos Guarantees

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

\[ \text{err}(w_T) < \text{err}(w_0) + \epsilon \]

With probability \( 1 - \delta \)

Running time does **NOT** depend on:
- # training examples!

It **DOES** depend on:
- Dimensionality \( d \) (why?)
- Approximation \( \epsilon \) and \( \delta \)
- Difficulty of problem \( \lambda \)

[Shalev Schwartz, Srebro ’08]
But how is that possible?

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

[Shalev Schwartz, Srebro ’08]