Two SVM tutorials linked in class website (please, read both):
- High-level presentation with applications (Hearst 1998)
- Detailed tutorial (Burges 1998)

SVMs, Duality and the Kernel Trick

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

February 27th, 2006
Announcements

- Third homework
  - is out
  - Due March 1st

- Final assigned by registrar:
  - May 12, 1-4p.m Friday
  - Location TBD

- Midterm
  - March 8th, a week from Wednesday
  - Open book, notes, papers, etc. No computers
SVMs reminder

minimize\( w \),

\[
\begin{align*}
\text{subject to} & \quad \begin{array}{c}
   w \cdot w + C \sum_j \xi_j \\
   (w \cdot x_j + b) y_j \geq 1 - \xi_j, \quad \forall j \\
   \xi_j \geq 0, \quad \forall j
\end{array}
\end{align*}
\]
Today’s lecture

- Learn one of the most interesting and exciting recent advancements in machine learning
  - The “kernel trick”
  - High dimensional feature spaces at no extra cost!
- But first, a detour
  - Constrained optimization!
Constrained optimization

\[
\begin{align*}
\min_x & \quad x^2 \\
\text{s.t.} & \quad x \geq b
\end{align*}
\]
Lagrange multipliers – Dual variables

\[ \min_x \ x^2 \]
\[ \text{s.t. } \ x \geq b \]

Moving the constraint to objective function

Lagrangian:
\[ L(x, \alpha) = x^2 - \alpha(x - b) \]
\[ \text{s.t. } \ \alpha \geq 0 \]

Solve:
\[ \min_x \ \max_\alpha \ L(x, \alpha) \]
\[ \text{s.t. } \ \alpha \geq 0 \]

- \( \alpha (x-b) \) \( \Rightarrow \) negative \( \Rightarrow \) max \( \Rightarrow \) \( \alpha = 0 \)
- \( \alpha (x-b) \) \( \Rightarrow \) positive \( \Rightarrow \) max \( \Rightarrow \) \( \alpha = \text{too} \)
- \( \alpha (x-b) \) \( \Rightarrow \) negative \( \Rightarrow \) max \( \Rightarrow \) \( \alpha = 0 \)
- \( \alpha (x-b) \) \( \Rightarrow \) positive \( \Rightarrow \) max \( \Rightarrow \) \( \alpha = \text{too} \)

\( \min_x \Rightarrow \text{only suggest } x \text{ that sat. constraint} \)
Lagrange multipliers – Dual variables

Solving: \( \min_x \max_\alpha \quad x^2 - \alpha(x - b) \)
\[ \text{s.t. } \alpha \geq 0 \]

\( \frac{\partial L}{\partial x} = 2x - \alpha \cdot 0 \Rightarrow x = \frac{\alpha}{2} \]

either \( \frac{\partial L}{\partial x} = 0 \Rightarrow x = 5 \)

or if \( \alpha = 0 \) term \( \alpha(x - b) \) irrelevant

or \( \alpha = 0 \) and constraint is ignored

or \( \alpha \) to constraint plays a role

\( x = 0 \)
Dual SVM derivation (1) – the linearly separable case

\[
\begin{align*}
\text{minimize}_w & \quad \frac{1}{2} w \cdot w \\
(w \cdot x_j + b) y_j & \geq 1, \quad \forall j \in \text{training set}
\end{align*}
\]

\[
L(w, b, x) = \frac{1}{2} w \cdot w - \sum_j \alpha_j [(w \cdot x_j + b) y_j - 1]
\]

\[
\frac{\partial L}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_j \alpha_j x_j y_j
\]
Dual SVM derivation (2) – the linearly separable case

\[ L(w, \alpha) = \frac{1}{2} w . w - \sum_j \alpha_j \left[ (w . x_j^+ + b) y_j - 1 \right] \]

\[ \alpha_i \geq 0, \ \forall j \]

\[ (w . x_j^+ + b) y_j > 1 \]

\[ \Rightarrow \alpha_j = 0 \]

\[ \text{when } (w . x_j^+ + b) y_j = 1 \]

\[ \alpha_j > 0 \]

\[ w = \sum_j \alpha_j y_j x_j \]

\[ \text{minimize}_w \frac{1}{2} w . w \]

\[ (w . x_j + b) y_j \geq 1, \ \forall j \]

\[ b = y_k - w . x_k \]

for any \( k \) where \( \alpha_k > 0 \)
Dual SVM interpretation

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

\[ w = \sum_{i} \alpha_{i} y_{i} x_{i} \]

Weight can be written as a linear combination of input \( y_{i} x_{i} \), but only care about support vectors.
Dual SVM formulation – the linearly separable case

\[
\text{minimize} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j \\
\sum_i \alpha_i y_i = 0 \\
\alpha_i \geq 0
\]

Solve dual program, SVM:

- Solve dual
- Obtain the $\alpha_i$
- Get $w, b$

Object function: dual $\rightarrow$ quadratic $\rightarrow$ dual quadratic program

\[
w = \sum_i \alpha_i y_i x_i \\
b = y_k - w \cdot x_k
\]

for any $k$ where $\alpha_k > 0$
Dual SVM derivation –
the non-separable case

\[
\begin{align*}
\text{minimize}_w & \quad \frac{1}{2} w \cdot w + C \sum_j \xi_j \\
\left( w \cdot x_j + b \right) y_j & \geq 1 - \xi_j, \quad \forall j \quad \alpha_j \\
\xi_j & \geq 0, \quad \forall j \quad \mu_j
\end{align*}
\]

\[
L(w, b, \alpha_j) = \frac{1}{2} w \cdot w + C \sum_j \xi_j - \sum_j \alpha_j \left[ (w \cdot x_j + b) y_j - 1 + \xi_j \right] \\
- \sum_j \mu_j \xi_j
\]
Dual SVM formulation – the non-separable case

\[
\begin{align*}
\text{minimize}_{\alpha} & \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j \\
\text{subject to} & \quad \sum_i \alpha_i y_i = 0 \\
& \quad C \geq \alpha_i \geq 0
\end{align*}
\]

\[
w = \sum_i \alpha_i y_i x_i
\]

\[
b = y_k - w \cdot x_k
\]

for any \( k \) where \( C > \alpha_k > 0 \)
Why did we learn about the dual SVM?

- There are some quadratic programming algorithms that can solve the dual faster than the primal.
- But, more importantly, the “kernel trick”!!!
  - Another little detour…
Reminder from last time: What if the data is not linearly separable?

Use features of features of features of features...

\[ \Phi(x) : \mathbb{R}^m \rightarrow F \]

Feature mapping \( x = (x_1, x_2) \)

\[ \phi(x) = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_p \end{pmatrix} \]

\[ \phi(x) = \begin{pmatrix} x_1 \\ x_1^2 \\ x_1^3 \\ \cdots \\ x_1^{p-1} \\ x_1 x_2 \\ x_1 x_2^2 \\ \cdots \\ x_1 x_2^{p-2} \\ x_2 \end{pmatrix} \]

Feature space can get really large really quickly!
Higher order polynomials

\[ \text{num. terms} = \binom{d + m - 1}{d} = \frac{(d + m - 1)!}{d!(m - 1)!} \]

- \( d \) – degree of polynomial
- \( m \) – input features

The number of monomial terms grows rapidly with increasing degrees of polynomial and input features. For example, with \( d = 6 \) and \( m = 100 \), there are about 1.6 billion terms.
Dual formulation only depends on dot-products, not on $w$!

$$\text{minimize}_{\alpha} \quad \sum \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j$$

$$\sum \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$

Only thing is $x$

$x_i x_j = x_i \cdot x_j$

no $w$!

use features $\phi(x)$

all I need is $\phi(x_j) \cdot \phi(x_i)$

$$K(x_j, x_i) = \phi(x_j) \cdot \phi(x_i)$$

$$\text{minimize}_{\alpha} \quad \sum \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

$$\sum \alpha_i y_i = 0$$

$$C \geq \alpha_i \geq 0$$
Dot-product of polynomials

\[ \Phi(u) \cdot \Phi(v) = \text{polynomials of degree } d \]

Degree 1: \( \Phi(u) = \left( \begin{array}{c} u_1 \\ u_2 \end{array} \right) \quad \Phi(v) = \left( \begin{array}{c} v_1 \\ v_2 \end{array} \right) \quad \Phi(u) \cdot \Phi(v) = u_1 v_1 + u_2 v_2 \]

Degree 2:

\[ \Phi(u) = \left( \begin{array}{c} u_1^2 \\ u_1 u_2 \\ u_2^2 \end{array} \right) \quad \Phi(v) = \left( \begin{array}{c} v_1^2 \\ v_1 v_2 \\ v_2^2 \end{array} \right) \quad \Phi(u) \cdot \Phi(v) = u_1^2 v_1^2 + u_1 u_2 v_1 v_2 + u_2^2 v_2^2 \]

\[ = (u_1 v_1 + u_2 v_2)^2 \leq 3 \text{ multiplies} \]

\[ = (u \cdot v)^2 \]

Polynomials of degree exactly \( d \):

\[ \Phi(u) \cdot \Phi(v) = (u \cdot v)^d \]
Finally: the "kernel trick"!

\[ \text{minimize}_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \]

\[ K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \]

\[ \sum_i \alpha_i y_i = 0 \]

\[ C \geq \alpha_i \geq 0 \]

\[ w = \sum_i \alpha_i y_i \Phi(x_i) \]

\[ b = y_k - w \cdot \Phi(x_k) \]

if using poly. degree exactly d

\[ \forall i,j \text{ compute } x_i \cdot x_j \]

set \[ K(x_i, x_j) = (x_i \cdot x_j)^d \]

solve dual Q.P.

get \( \alpha \)

It's all pairs of data points including data point with itself.
Finally: the “kernel trick”!

minimize \[ \alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \]

\[ K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \]
\[ \sum_i \alpha_i y_i = 0 \]
\[ C' \geq \alpha_i \geq 0 \]

- Never represent features explicitly
  - Compute dot products in closed form
- Constant-time high-dimensional dot-products for many classes of features
- Very interesting theory – Reproducing Kernel Hilbert Spaces
  - Not covered in detail in 10701/15781, more in 10702

\[ w = \sum_i \alpha_i y_i \Phi(x_i) \]
\[ b = y_k - w \cdot \Phi(x_k) \]
for any \( k \) where \( C' > \alpha_k > 0 \)
Polynomial kernels

- All monomials of degree $d$ in $O(d)$ operations:
  $$\Phi(u) \cdot \Phi(v) = (u \cdot v)^d$$
  polynomials of degree $d$

- How about all monomials of degree up to $d$?
  - Solution 0: $$\Phi(u) \cdot \Phi(v) = \sum_{i=0}^{d} (u \cdot v)^i$$
  - Better solution: $$\Phi(u) \cdot \Phi(v) = (u \cdot v + 1)^d$$
    $O(d^2)$ time
Common kernels

- Polynomials of degree $d$
  \[ K(u, v) = (u \cdot v)^d \]

- Polynomials of degree up to $d$
  \[ K(u, v) = (u \cdot v + 1)^d \]

- Gaussian kernels
  \[ K(u, v) = \exp \left( -\frac{||u - v||^2}{2\sigma^2} \right) \]
  \[ \text{dim} \left[ \phi(x) \right] = \text{infinite} \]

- Sigmoid
  \[ K(u, v) = \tanh(\eta u \cdot v + \nu) \]
Overfitting?

- Huge feature space with kernels, what about overfitting???
  - Maximizing margin leads to sparse set of support vectors
  - Some interesting theory says that SVMs search for simple hypothesis with large margin
  - Often robust to overfitting

Sparse solutions $\rightarrow$ a few support vectors $\rightarrow$ less overfitting
What about at classification time

- For a new input \( x \), if we need to represent \( \Phi(x) \), we are in trouble!

- Recall classifier: \( \text{sign}(w \cdot \Phi(x) + b) \)

- Using kernels we are cool!

\[
K(u, v) = \Phi(u) \cdot \Phi(v)
\]

\[
w \cdot \Phi(x) = \sum_i \alpha_i y_i \Phi(x_i) \quad \text{easy to compute}
\]

\[
b = y_k - w \cdot \Phi(x_k)
\]

for any \( k \) where \( C > \alpha_k > 0 \)
SVMs with kernels

- Choose a set of features and kernel function
- Solve dual problem to obtain support vectors $\alpha_i$
- At classification time, compute:

$$w \cdot \Phi(x) = \sum_i \alpha_i y_i K(x, x_i)$$

$$b = y_k - \sum_i \alpha_i y_i K(x_k, x_i)$$

for any $k$ where $C > \alpha_k > 0$

Classify as $\text{sign}(w \cdot \Phi(x) + b)$
What’s the difference between SVMs and Logistic Regression?

<table>
<thead>
<tr>
<th></th>
<th>SVMs</th>
<th>Logistic Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High dimensional</td>
<td></td>
<td></td>
</tr>
<tr>
<td>features with kernels</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Kernels in logistic regression

\[ P(Y = 1 \mid x, w) = \frac{1}{1 + e^{-(w \cdot \Phi(x) + b)}} \]

- Define weights in terms of support vectors:
  \[ w = \sum_{i} \alpha_i \Phi(x_i) \]

\[ P(Y = 1 \mid x, w) = \frac{1}{1 + e^{-(\sum_{i} \alpha_i \Phi(x_i) \cdot \Phi(x) + b)}} = \frac{1}{1 + e^{-(\sum_{i} \alpha_i K(x,x_i) + b)}} \]

- Derive simple gradient descent rule on \( \alpha_i \)
What’s the difference between SVMs and Logistic Regression? (Revisited)

<table>
<thead>
<tr>
<th></th>
<th>SVMs</th>
<th>Logistic Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss function</td>
<td>Hinge loss</td>
<td>Log-loss</td>
</tr>
<tr>
<td>High dimensional</td>
<td>Yes!</td>
<td>Yes!</td>
</tr>
<tr>
<td>features with kernels</td>
<td>Yes!</td>
<td></td>
</tr>
</tbody>
</table>
What you need to know

- Dual SVM formulation
  - How it’s derived
- The kernel trick
- Derive polynomial kernel
- Common kernels
- Kernelized logistic regression
- Differences between SVMs and logistic regression
Acknowledgment

- SVM applet:
  - [http://www.site.uottawa.ca/~gcaron/applets.htm](http://www.site.uottawa.ca/~gcaron/applets.htm)