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 (cont.)

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

March 1st, 2006
SVMs reminder

$$\text{minimize}_{\mathbf{w}} \quad \mathbf{w} \cdot \mathbf{w} + C \sum_j \xi_j$$

$$- (\mathbf{w} \cdot \mathbf{x}_j + b) y_j \geq 1 - \xi_j, \quad \forall j$$

$$\xi_j \geq 0, \quad \forall j$$
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!
Dual SVM interpretation

\[ w = \sum_{i} \alpha_i y_i x_i \]

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

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

Dual program, SVM:
- solve dual
- obtain the $\alpha_i$
- get $w, b$

Obj function dual = quadratic = dual quadratic program
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

$$x = (x_1, x_2)$$

$$\phi(x) = \begin{pmatrix}
1 \\
x_1 \\
x_1^2 \\
\vdots \\
x_1^p
\end{pmatrix}$$

$$\phi(x) = \begin{pmatrix}
1 \\
x_1 \\
x_2 \\
x_1^2 \\
x_1 x_2 \\
\vdots
\end{pmatrix}$$

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

\(d = 6, m = 100\) about 1.6 billion terms

grows fast!
Dual formulation only depends on dot-products, not on \( \mathbf{w} \! \). 

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

\[
\sum_i \alpha_i y_i = 0
\]

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

only thing is \( x \) 
\( x_i x_j = x_i \cdot x_j \)

no \( \mathbf{w} \)! 

use features \( \phi(x) \)

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

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

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

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

\[
\sum_i \alpha_i y_i = 0
\]

\[
C \geq \alpha_i \geq 0
\]
Finally: the “kernel trick”!

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

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

\[ \sum_i \alpha_i y_i = 0 \]

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

- 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 \( C > \alpha_k > 0 \)
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
  \[ K(u, v) = \exp \left( -\frac{||u - v||}{2\sigma^2} \right) \]
- 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 \( \mathbf{x} \), if we need to represent \( \Phi(\mathbf{x}) \), we are in trouble!  
  
- Recall classifier: \( \text{sign}(\mathbf{w} \cdot \Phi(\mathbf{x}) + b) \)

- Using kernels we are cool!

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

\[
\mathbf{w} \cdot \Phi(x) = \sum_i \alpha_i y_i \Phi(x_i) 
\]

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

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 $\text{sign}(w \cdot \Phi(x) + b)$
Remember kernel regression

1. \( w_i = \exp(-D(x_i, \text{query})^2 / K_w^2) \)
2. How to fit with the local points?
   Predict the weighted average of the outputs:
   \[ \text{predict} = \frac{\Sigma w_i y_i}{\Sigma w_i} \]
SVMs v. Kernel Regression

SVMs

\[ \text{sign} \left( w \cdot \Phi(x) + b \right) \]

or

\[ \text{sign} \left( \sum_i \alpha_i y_i K(x, x_i) + b \right) \]

Kernel Regression

\[ \text{sign} \left( \frac{\sum_i y_i K(x, x_i)}{\sum_j K(x, x_j)} \right) \]
**SVMs v. Kernel Regression**

### SVMs

\[ \text{sign} (w \cdot \Phi(x) + b) \]

- Learn weights \( \alpha_i \) (and bandwidth)
- Often sparse solution

### Differences:

<table>
<thead>
<tr>
<th>SVMs:</th>
<th>Kernel Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>□ Learn weights ( \alpha_i ) (and bandwidth)</td>
<td>[ \text{sign} \left( \frac{\sum_i y_i K(x, x_i)}{\sum_i K(x, x_i)} \right) ]</td>
</tr>
<tr>
<td>□ Often sparse solution</td>
<td></td>
</tr>
</tbody>
</table>

- Fixed “weights”, learn bandwidth
- Solution may not be sparse
- Much simpler to implement
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</td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
<tr>
<td>kernels</td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>

Logistic Regression uses Log-loss, while SVMs use Hinge loss.

High dimensional features with kernels can be handled by SVMs using kernels.
Define weights in terms of support vectors:

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

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

\[ = \frac{1}{1 + e^{-\left(\sum_i \alpha_i K(x,x_i) + b\right)}} \]

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><strong>Loss function</strong></td>
<td>Hinge loss</td>
<td>Log-loss</td>
</tr>
<tr>
<td><strong>High dimensional</strong></td>
<td>Yes!</td>
<td>Yes!</td>
</tr>
<tr>
<td><strong>features with kernels</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

©2006 Carlos Guestrin
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)
PAC-learning, VC Dimension and Margin-based Bounds

Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

March 1st, 2005
What now…

- We have explored **many** ways of learning from data
- But…
  - How good is our classifier, really?
  - How much data do I need to make it “good enough”?
A simple setting…

- Classification
  - m data points
  - **Finite** number of possible hypothesis (e.g., dec. trees of depth d)
- A learner finds a hypothesis $h$ that is **consistent** with training data
  - Gets zero error in training – $error_{\text{train}}(h) = 0$
- What is the probability that $h$ has more than $\varepsilon$ true error?
  - $error_{\text{true}}(h) \geq \varepsilon$
How likely is a bad hypothesis to get $m$ data points right?

- Hypothesis $h$ that is **consistent** with training data → got $m$ i.i.d. points right
- Prob. $h$ with error$_{true}(h) \geq \varepsilon$ gets one data point right

- Prob. $h$ with error$_{true}(h) \geq \varepsilon$ gets $m$ data points right
But there are many possible hypothesis that are consistent with training data
How likely is learner to pick a bad hypothesis

- Prob. $h$ with $\text{error}_{\text{true}}(h) \geq \varepsilon$ gets $m$ data points right

- There are $k$ hypothesis consistent with data
  - How likely is learner to pick a bad one?
Union bound

- $P(A \text{ or } B \text{ or } C \text{ or } D \text{ or } \ldots)$
How likely is learner to pick a bad hypothesis

- Prob. $h$ with $\text{error}_{\text{true}}(h) \geq \varepsilon$ gets $m$ data points right

- There are $k$ hypothesis consistent with data
  - How likely is learner to pick a bad one?
Review: Generalization error in finite hypothesis spaces [Haussler ’88]

**Theorem:** Hypothesis space $H$ finite, dataset $D$ with $m$ i.i.d. samples, $0 < \varepsilon < 1$: for any learned hypothesis $h$ that is consistent on the training data:

$$P(\text{error}_{\mathcal{X}}(h) > \varepsilon) \leq |H|e^{-m\varepsilon}$$
Using a PAC bound

Typically, 2 use cases:

1: Pick $\varepsilon$ and $\delta$, give you $m$
2: Pick $m$ and $\delta$, give you $\varepsilon$

$P(\text{error}_X(h) > \varepsilon) \leq |H|e^{-m\varepsilon}$
Review: Generalization error in finite hypothesis spaces [Haussler ’88]

**Theorem**: Hypothesis space $H$ finite, dataset $D$ with $m$ i.i.d. samples, $0 < \varepsilon < 1$: for any learned hypothesis $h$ that is consistent on the training data:

$$P(\text{error}_X(h) > \varepsilon) \leq |H|e^{-m\varepsilon}$$

Even if $h$ makes zero errors in training data, may make errors in test.
Limitations of Haussler ‘88 bound

- Consistent classifier
  \[ P(\text{error}_\chi(h) > \epsilon) \leq |H|e^{-m\epsilon} \]

- Size of hypothesis space
What if our classifier does not have zero error on the training data?

- A learner with zero training errors may make mistakes in test set.
- What about a learner with $error_{train}(h)$ in training set?
Simpler question: What’s the expected error of a hypothesis?

- The error of a hypothesis is like estimating the parameter of a coin!

- Chernoff bound: for $m$ i.d.d. coin flips, $x_1, \ldots, x_m$, where $x_i \in \{0,1\}$. For $0 < \epsilon < 1$:

$$P \left( \theta - \frac{1}{m} \sum_{i} x_i > \epsilon \right) \leq e^{-2m\epsilon^2}$$
Using Chernoff bound to estimate error of a single hypothesis

\[ P \left( \theta - \frac{1}{m} \sum_i x_i > \epsilon \right) \leq e^{-2m\epsilon^2} \]
But we are comparing many hypothesis: **Union bound**

For each hypothesis $h_i$:

$$P \left( \text{error}_{true}(h_i) - \text{error}_{train}(h_i) > \epsilon \right) \leq e^{-2m\epsilon^2}$$

What if I am comparing two hypothesis, $h_1$ and $h_2$?
Generalization bound for $|H|$ hypothesis

**Theorem:** Hypothesis space $H$ finite, dataset $D$ with $m$ i.i.d. samples, $0 < \varepsilon < 1$ : for any learned hypothesis $h$:

$$P (\text{error}_{true}(h) - \text{error}_{train}(h) > \varepsilon) \leq |H| e^{-2m\varepsilon^2}$$
PAC bound and Bias-Variance tradeoff

\[ P(\text{error}_{\text{true}}(h) - \text{error}_{\text{train}}(h) > \epsilon) \leq |H|e^{-2m\epsilon^2} \]

or, after moving some terms around, with probability at least 1-\(\delta\):
\[ \text{error}_{\text{true}}(h) \leq \text{error}_{\text{train}}(h) + \sqrt{\frac{\ln |H| + \ln \frac{1}{\delta}}{2m}} \]

Important: PAC bound holds for all \(h\), but doesn’t guarantee that algorithm finds best \(h\)!!!
What about the size of the hypothesis space?

\[ m \geq \frac{1}{2\epsilon^2} \left( \ln |H| + \ln \frac{1}{\delta} \right) \]

- How large is the hypothesis space?
Boolean formulas with $n$ binary features

$$m \geq \frac{1}{2\epsilon^2} \left( \ln |H| + \ln \frac{1}{\delta} \right)$$
Number of decision trees of depth $k$ 

Recursive solution

Given $n$ attributes

$H_k =$ Number of decision trees of depth $k$

$H_0 = 2$

$H_{k+1} = (\text{#choices of root attribute}) \times$

\begin{align*}
&\quad (\text{# possible left subtrees}) \times \\
&\quad (\text{# possible right subtrees}) \\
\end{align*}

$= n \times H_k \times H_k$

Write $L_k = \log_2 H_k$

$L_0 = 1$

$L_{k+1} = \log_2 n + 2L_k$

So $L_k = (2^k - 1)(1 + \log_2 n) + 1$
PAC bound for decision trees of depth $k$

\[ m \geq \frac{\ln 2}{2\epsilon^2} \left( (2^k - 1)(1 + \log_2 n) + 1 + \ln \frac{1}{\delta} \right) \]

- Bad!!!
  - Number of points is exponential in depth!

- But, for $m$ data points, decision tree can’t get too big…
Number of decision trees with k leaves

\[ H_k = \text{Number of decision trees with } k \text{ leaves} \]

\[ H_0 = 2 \]

\[ H_{k+1} = n \sum_{i=1}^{k} H_i H_{k+1-i} \]

Loose bound:

\[ H_k = n^{k-1}(k + 1)^{2k-1} \]

Reminder:

\[ |\text{DTs depth } k| = 2 \times (2n)^{2^{k-1}} \]
PAC bound for decision trees with k leaves – Bias-Variance revisited

\[ H_k = n^{k-1}(k + 1)^{2k-1} \]

\[ \text{error}_{\text{true}}(h) \leq \text{error}_{\text{train}}(h) + \sqrt{\frac{\ln |H| + \ln \frac{1}{\delta}}{2m}} \]

\[ \text{error}_{\text{true}}(h) \leq \text{error}_{\text{train}}(h) + \sqrt{\frac{(k - 1) \ln n + (2k - 1) \ln (k + 1) + \ln \frac{1}{\delta}}{2m}} \]
What did we learn from decision trees?

- Bias-Variance tradeoff formalized
  \[
  \text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{(k - 1) \ln n + (2k - 1) \ln (k + 1) + \ln \frac{1}{\delta}}{2m}}
  \]

- Moral of the story:
  Complexity of learning not measured in terms of size hypothesis space, but in maximum number of points that allows consistent classification
  - Complexity \( m \) – no bias, lots of variance
  - Lower than \( m \) – some bias, less variance
What about continuous hypothesis spaces?

\[
\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{\ln |H| + \ln \frac{1}{\delta}}{2m}}
\]

- Continuous hypothesis space:
  - \(|H| = \infty\)
  - Infinite variance???

- As with decision trees, only care about the maximum number of points that can be classified exactly!
How many points can a linear boundary classify exactly? (1-D)
How many points can a linear boundary classify exactly? (2-D)
How many points can a linear boundary classify exactly? (d-D)
PAC bound using VC dimension

- Number of training points that can be classified exactly is VC dimension!!!
  - Measures relevant size of hypothesis space, as with decision trees with k leaves

\[
\text{error}_{\text{true}}(h) \leq \text{error}_{\text{train}}(h) + \sqrt{\frac{VC(H) \left( \ln \frac{2m}{VC(H)} + 1 \right) + \ln \frac{4}{\delta}}{m}}
\]
Shattering a set of points

Definition: a **dichotomy** of a set $S$ is a partition of $S$ into two disjoint subsets.

Definition: a set of instances $S$ is **shattered** by hypothesis space $H$ if and only if for every dichotomy of $S$ there exists some hypothesis in $H$ consistent with this dichotomy.
VC dimension

Definition: The Vapnik-Chervonenkis dimension, $VC(H)$, of hypothesis space $H$ defined over instance space $X$ is the size of the largest finite subset of $X$ shattered by $H$. If arbitrarily large finite sets of $X$ can be shattered by $H$, then $VC(H) \equiv \infty$. 
Examples of VC dimension

- Linear classifiers:
  - VC(H) = d+1, for d features plus constant term b

- Neural networks
  - VC(H) = #parameters
  - Local minima means NNs will probably not find best parameters

- 1-Nearest neighbor?
PAC bound for SVMs

- SVMs use a linear classifier
  - For $d$ features, $VC(H) = d+1$:

$$
\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{(d+1) \left(\ln \frac{2m}{d+1} + 1\right) + \ln \frac{4}{\delta}}{m}}
$$
VC dimension and SVMs: Problems!!!

- Doesn’t take margin into account

\[
\text{error}_{\text{true}}(h) \leq \text{error}_{\text{train}}(h) + \sqrt{\frac{(d + 1) \left( \ln \frac{2m}{d+1} + 1 \right) + \ln \frac{4}{\delta}}{m}}
\]

- What about kernels?
  - Polynomials: num. features grows really fast = Bad bound
    
    \[
    \text{num. terms} = \binom{p+n-1}{p} = \frac{(p+n-1)!}{p!(n-1)!}
    \]
    
    \(n\) – input features
    \(p\) – degree of polynomial

  - Gaussian kernels can classify any set of points exactly
Margin-based VC dimension

- H: Class of linear classifiers: $\mathbf{w}.\Phi(\mathbf{x})$ (b=0)
  - Canonical form: $\min_j |\mathbf{w}.\Phi(\mathbf{x}_j)| = 1$

- $\text{VC}(H) = R^2 \mathbf{w}.\mathbf{w}$
  - Doesn’t depend on number of features!!!
  - $R^2 = \max_j \Phi(\mathbf{x}_j).\Phi(\mathbf{x}_j)$ – magnitude of data
  - $R^2$ is bounded even for Gaussian kernels → bounded VC dimension

- Large margin, low $\mathbf{w}.\mathbf{w}$, low VC dimension – Very cool!
Applying margin VC to SVMs?

\[
\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{\text{VC}(H) \left( \ln \frac{2m}{\text{VC}(H)} + 1 \right) + \ln \frac{4}{\delta}}{m}}
\]

- **VC**($H$) = $R^2$ $w.w$
  - $R^2 = \max_j \Phi(x_j).\Phi(x_j)$ – magnitude of data, doesn’t depend on choice of $w$

- SVMs minimize $w.w$

- SVMs minimize VC dimension to get best bound?
- **Not quite right:** 😞
  - Bound assumes VC dimension chosen before looking at data
  - Would require union bound over infinite number of possible VC dimensions...
  - But, it can be fixed!
Structural risk minimization theorem

$$\text{error}_{true}(h) \leq \text{error}^\gamma_{train}(h) + C \sqrt{\frac{R^2}{\gamma^2} \ln m + \ln \frac{1}{\delta}}$$

$$\text{error}^\gamma_{\text{train}}(h) = \text{num. points with margin < } \gamma$$

- For a family of hyperplanes with margin $\gamma > 0$
  - $\mathbf{w} \cdot \mathbf{w} \leq 1$
- SVMs maximize margin $\gamma +$ hinge loss
  - Optimize tradeoff training error (bias) versus margin $\gamma$ (variance)
Reality check – Bounds are loose

Bound can be very loose, why should you care?

- There are tighter, albeit more complicated, bounds
- Bounds gives us formal guarantees that empirical studies can’t provide
- Bounds give us intuition about complexity of problems and convergence rate of algorithms

\[ \text{error}_{\text{true}}(h) \leq \text{error}_{\text{train}}(h) + \sqrt{(d + 1) \left( \ln \frac{2m}{d+1} + 1 \right) + \ln \frac{4}{\delta}} \]
What you need to know

- Finite hypothesis space
  - Derive results
  - Counting number of hypothesis
  - Mistakes on Training data

- Complexity of the classifier depends on number of points that can be classified exactly
  - Finite case – decision trees
  - Infinite case – VC dimension

- Bias-Variance tradeoff in learning theory

- Margin-based bound for SVM

- Remember: will your algorithm find best classifier?