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

minimize \( w \) \[ w \cdot w + C \sum_j \xi_j \]
\[ - (w \cdot x_j + b) y_j \geq 1 - \xi_j, \forall j \]
\[ \xi_j \geq 0, \forall j \]
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

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

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

The 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

minimize $\alpha_i \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j$

$\sum_i \alpha_i y_i = 0$

$\alpha_i \geq 0$

Dual program, SVM:
- solve dual
- obtain $w, b$

$w = \sum_i \alpha_i y_i x_i$

$b = y_k - w \cdot x_k$

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

Obj function dual = quadratic = dual quadratic program

©2006 Carlos Guestrin
Reminder from last time: What if the data is not linearly separable?

Use features of features of features of features….

$$\Phi(x) : R^m \rightarrow F$$

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

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

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

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

grows fast!
Dual formulation only depends on dot-products, not on $w$!

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

\[
K(x_j, x_i) = \Phi(x_j) \cdot \Phi(x_i)
\]
Finally: the “kernel trick”!

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

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

\[
\sum_i \alpha_i y_i = 0
\]

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

- 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 \( C > \alpha_k > 0 \)
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 → a few support vectors → less overfitting
What about at classification time

- For a new input $x$, if we need to represent $\Phi(x)$, we are in trouble! (if have to write $w, b$, too large)
- Recall classifier: $\text{sign}(w \cdot \Phi(x) + b)$
- Using kernels we are cool!

$$K(u, v) = \Phi(u) \cdot \Phi(v)$$

$$w \cdot \phi(x) = \sum_{i} x_i y_i \phi(x) \cdot \phi(x_i)$$

$$w = \sum_{i} \alpha_i y_i \phi(x_i)$$

$$b = y_k - w \cdot \Phi(x_k)$$

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

new input $x \Rightarrow \Phi(x), \text{sign}(w \cdot \phi(x) + b)$
SVMs with kernels

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

\[
\begin{align*}
\mathbf{w} \cdot \Phi(\mathbf{x}) &= \sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) \\
b &= y_k - \sum_i \alpha_i y_i K(\mathbf{x}_k, \mathbf{x}_i)
\end{align*}
\]

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

Classify as $\text{sign}(\mathbf{w} \cdot \Phi(\mathbf{x}) + b)$
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:
     \[ \text{predict} = \frac{\sum w_i y_i}{\sum w_i} \]
SVMs v. Kernel Regression

SVMs

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

or

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

Kernel Regression

\[ \Pi = \sum_j K(\mathbf{x}, \mathbf{x}_j) \]

\[ \text{sign} \left( \frac{\sum_i y_i K(\mathbf{x}, \mathbf{x}_i)}{\sum_j K(\mathbf{x}, \mathbf{x}_j)} \right) \]

\[ \text{sign} \left( \sum_i \frac{1}{\Pi} y_i K(\mathbf{x}, \mathbf{x}_i) \right) \]
SVMs v. Kernel Regression

**SVMs**

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

or

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

**Differences:**

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

- **KR:**
  - "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>Hinge loss</td>
<td>Log-loss</td>
</tr>
<tr>
<td></td>
<td><img src="image.png" alt="Correct/Incorrect" /></td>
<td></td>
</tr>
<tr>
<td>High dimensional features with kernels</td>
<td>Yes! <strong>Kernel trick!!</strong></td>
<td>No <strong>yes!!</strong></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:
  \[ w = \sum_i \alpha_i \Phi(x_i) \]  
  "Representor Theorem":

- 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 features with kernels</strong></td>
<td>Yes!</td>
<td>Yes!</td>
</tr>
<tr>
<td><strong>Solution sparse</strong></td>
<td>Often yes! because hinge loss</td>
<td>Almost always no! because of log-loss</td>
</tr>
<tr>
<td><strong>Semantics of output</strong></td>
<td>“margin”</td>
<td>“Real” probabilities</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

Playing applet...
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”?  

Learning Theory
A simple setting…

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

$$P(\text{error}_{\text{true}}(h) \geq \varepsilon) \leq \delta$$

E.g., $\delta = 0.01$
How likely is a bad hypothesis to get \( m \) data points right?

- Hypothesis \( h \) that is **consistent** with training data \( \rightarrow \) all \( \) got \( m \) i.i.d. points right.

- Prob. \( h \) with error \( \text{true}(h) \geq \varepsilon \) gets one data point right:
  \[ P(\text{true}(h) \geq \varepsilon, \text{and get one data point right}) \leq 1 - \varepsilon \]

- Prob. \( h \) with error \( \text{true}(h) \geq \varepsilon \) gets \( m \) data points right:
  \[ P(\text{bad h get lucky}) \leq (1 - \varepsilon)^m \]

More data, exponentially less likely a bad hyp. gets all right.
But there are many possible hypothesis that are consistent with training data

The learner picked one arbitrarily, the worst one!!
How likely is learner to pick a bad hypothesis

- Prob. $h$ with error_{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?

  \[ P(\text{at least one of the } k \text{ was bad}) \text{ and it got lucky} \]

  \[ P(\text{one got lucky}) \leq (1-\varepsilon)^m \]
Union bound

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

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

- There are \( k \) hypothesis consistent with data
  - How likely is learner to pick a bad one?

\[
P(h_1 \text{ bad } \text{ & got lucky } \text{ or } h_2 \text{ bad } \text{ & got lucky } \text{ or } h_3 \text{ ...})
\leq P(h_1 \text{ bad } \text{ & lucky}) + P(h_2 \text{ bad } \text{ & lucky}) + P(h_3 \text{ ...}) + ...
\leq (1-\varepsilon)^m
\]

- \( K \leq 1H \) (loose bound!)

\[
\leq K (1-\varepsilon)^m
\leq |H| e^{-\varepsilon m} \leq |H| e^{-\varepsilon} \leq e^{-3} \tag{make eq. simpler}
\]
Review: Generalization error in finite hypothesis spaces

**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:

$$P(\text{error}_\text{true}(h) > \varepsilon) \leq |H|e^{-m\varepsilon} < 0.01$$

Down exponentially fast!
Using a PAC bound

Typically, 2 use cases:

1: Pick $\epsilon$ and $\delta$, give you $m$

2: Pick $m$ and $\delta$, give you $\epsilon$

$$P(\text{error}_{\text{true}}(h) > \epsilon) \leq |H|e^{-me}$$

1. $P \leq |H|e^{-me} \leq \delta$

   $$\ln |H| - me \leq \ln \delta$$

   $$\Rightarrow m \geq \frac{1}{\epsilon} \left( \ln |H| + \ln \frac{1}{\delta} \right)$$

   Smaller $\epsilon$ needs more data

   More data not as much data as you think

   Before you run the algorithm

   True error $\leq m$
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

1. Consistent classifier
   \[ P(\text{error}_2(h) > \epsilon) \leq |H|e^{-m\epsilon} \]
   There may not be such \( h \) in class!

2. Size of hypothesis space
   Bound depends on \( |H| \)
   Is really really large?
   Is infinite?
   How continuous?
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 $\text{error}_{\text{train}}(h)$ in training set?

$h$ is not perfect in training.
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.i.d. coin flips, \( x_1, \ldots, x_m \), where \( x_i \in \{0,1\} \). For \( 0 < \varepsilon < 1 \):

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

\[ P \left( \theta - \frac{1}{m} \sum_{i} x_i > \epsilon \right) \leq e^{-2m\epsilon^2} \]

error\text{\_true}(h) \leftarrow \Theta \quad \text{# times expect to make a mistake}

\[ x_i \leftarrow \text{did I get i\text{\_th} data point wrong} \]

\[ x_i = \mathbb{I}(h(i) \neq t(i)) \]

error\text{\_train} = \frac{1}{n} \sum_{i} x_i = \frac{1}{n} \sum_{i} \mathbb{I}(h(i) \neq t(i))
But we are comparing many hypothesis: **Union bound**

For each hypothesis $h_i$:

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

What if I am comparing two hypothesis, $h_1$ and $h_2$?

\[ P(h_1) \cap P(h_2) \]

Learning is going to compare all of $h_i$'s

$$P(\exists i \text{ error}_{\text{true}}(h_i) - \text{error}_{\text{train}}(h_i) > \epsilon) \leq |H| e^{-2m\epsilon^2}$$

©2006 Carlos Guestrin
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 $h$:

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

\[2m\varepsilon^2 = 20 \quad \text{not as good} !!\]

Side note: Haussler's bound for consistent $h$:

$$P \leq |H|e^{-m\varepsilon}$$

$\varepsilon = 0.1$  $\Rightarrow$  $m, \varepsilon = 166$

$m = 1000$
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-\( \delta \):

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

- "bias"
- \( \uparrow \) larger \( \Downarrow \) smaller \( |H| \) small
- "variance"
- \( \Downarrow \) smaller \( \uparrow \) larger \( |H| \) large

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? \(|H|\)
Boolean formulas with \( n \) binary features

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

Conjunctions:

- \( h_1 = x_1 \land x_2 \land x_7 \)
- \( h_2 = x_2 \land x_5 \land x_8 \)
- \( \ldots \)
- \( n \) attributes
- \( \{\emptyset, 1, 7\}, \{\emptyset, 1, 7\}, \ldots \)

\(|H| = 3^n \) (really large)

\(|\ln |H|| = n \ln 3 \)

"Small"

\(|H| = 2^n \) (really really large)

\(\ln |H| = 2^n \ln 2\)

Too large
Number of decision trees of depth k

Recursive solution
Given $n$ attributes

$H_k =$ Number of decision trees of depth $k$
$H_0 = 2$
$H_{k+1} =$ (#choices of root attribute) *
    (# possible left subtrees) *
    (# possible right subtrees)
    
    $= n \times H_k \times H_k$

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 leaves never more than number data points
Number of decision trees with k leaves

\[ H_k = \text{Number of decision trees with k leaves} \]
\[ H_0 = 2 \]

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

Loose bound:

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

Reminder:

\[ |\text{DTs depth } k| = 2 \ast (2n)^{2^{k-1}} \]
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

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

- 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

$$\text{error}_{true}(h) \leq \text{error}_{train}(h) + \sqrt{\frac{VC(H)}{m} \left( \ln \frac{2m}{VC(H)} + 1 \right) + \ln \frac{4}{\delta}}$$
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.
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, $\text{VC}(H) = d+1$:

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

Doesn’t take margin into account

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

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

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

- H: Class of linear classifiers: $w \cdot \Phi(x)$ (b=0)
  - Canonical form: $\min_j |w \cdot \Phi(x_j)| = 1$

- VC(H) = $R^2 \cdot w \cdot w$
  - Doesn’t depend on number of features!!!
  - $R^2 = \max_j \Phi(x_j).\Phi(x_j)$ – magnitude of data
  - $R^2$ is bounded even for Gaussian kernels $\rightarrow$ bounded VC dimension

- Large margin, low $w \cdot w$, low VC dimension – Very cool!
Applying margin VC to SVMs?

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

- \( VC(H) = R^2 \mathbf{w} \cdot \mathbf{w} \)
  - \( R^2 = \max_j \Phi(x_j) \cdot \Phi(x_j) \) – magnitude of data, doesn’t depend on choice of \( \mathbf{w} \)
- SVMs minimize \( \mathbf{w} \cdot \mathbf{w} \)

- 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

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

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

- 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

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