# Learning Theory, SVMs and Using Unlabeled Data

Instructor: Jesse Davis

Slides from: Pedro Domingos, Ray Mooney, David Page, Jude Shavlik

# Announcements

- Homework 3 is due now
- Homework 4 is available
- Homework 2 is graded
- Andrey will be out of town
  - Access to email at funny times
  - Email both of us
- Lecture notes are available online

# Outline

- Homework 2 review
- Computational learning theory
- Support vector machines
- Making use of unlabeled data

# Problem 1: Results

- For M = |V|, P = 1/|V|, Accuracy = 0.902
- Best M ~ 50 |V|, Accuracy = 0.906
- Most common omissions:
  - No code description (5 points)
  - No code comments (3 points)
  - Not reporting best parameter sets (4 points)
  - Reporting precision, recall, TPR, FPR, etc., but not accuracy (no penalty but annoyed me).

# Problem 1: One More Serious Omission

Not using sums of logarithms instead of products of probabilities

# Problem 1: Most Common Mistakes

| Mistake | Accuracy at $M = \|V\|$ $P = 1/\|V\|$ | Penalty | Comments |
|---|---|---|---|
| Ignoring word counts in test emails during classification. | 0.906 | 5 points | Bad because you learned multinomial parameters but are used them in a "binomial" way |
| The above + using $P = 1/\|V_{spam}\|$ or $P = 1 / (\|V_{spam}\|+\|V_{ham}\|)$ when computing P(W\|spam) | Usually 0.908 | 7 points | |
| Implementing binomial Naïve Bayes | 0.913 | 5 points | Not what the assignment asked |

# Problem 1: Good Observations

- True Negative Rate and False Positive Rate are more informative than accuracy in this application.

- Smoothing parameters have little effect in this particular case (don’t generalize it!)

- Cool ideas about additional features (next time)

# Problem 2a: Solution

- Straightforward:
  - Run FOIL
  - Get 10 points

- Learned rules are sometimes counterintuitive or incomplete:
  - Sister(A,B) :- Brother(B,A)

# Problem 2b: Solution

- 12 named predicates + Equals
- 5 variables (A,B,C,E, and X – the new variable)
- For each named predicate:
  - 5*5 - 1 = 24 positive literals resulting from substituting a combination of 2 (not necessarily distinct!) of the above variables, except (X,X).
  - 24 negative literals
- For Equals:
  - 4*4 = 16 positive literals resulting from substituting a combination of 2 (not necessarily distinct!) existing variables. X is not allowed to participate.
  - 16 negative literals
- Thus:
  - 2*(12*24 + 16) = 608 literals.

# Problem 2b: Common Mistakes

- Question was about which literals are generated, not which are valid choices.
  - Can’t exclude literals already in the rule (e.g., Wife(C,A))
  - Can’t exclude predicates already in the rule (e.g., Daughter)
  - Can’t exclude “silly” literals (e.g., Brother(A,A), Equals(B,B))
- Predicates (e.g., Wife, Brother) are not literals (e.g. Wife(A, C), Brother(E, B))

# Problem 3: Solution

- A) $2^d$ or $2^{d-1}$ rules will be created (one for each leaf or one for each positive leaf)
- B) Each rule will have depth of the tree = d preconditions
- C) Number of decisions = #rules * #preconditions = $d*2^d$ or $d*2^{d-1}$
- D) Sequential covering will be more prone to overfitting, because it makes more independent decisions

# Problem 3: Common Mistake

- The number of leaves in a tree of depth d is $2^d$, not $2^{d-1}$
- Just because a decision tree is less robust to noise (mistakes at higher nodes affect these nodes' entire subtrees) doesn't mean it overfits more.
- In fact, it means the opposite – ID3's decisions are less independent, so it's less prone to overfitting

# Problem 4: Solution

- Let r = rabid, d = drool, a = attack
- Given: P(r)= 0.042, P(d|r) = 0.79, P(d|-r) = 0.06, P(a|r) = 0.97, P(a|-r) = 0.02, A and D are independent given Rabid
- A) P(r|d) = P(d|r)P(r) / P(d) = P(d|r)P(r) / (P(d|r)P(r) + P(d|-r)P(-r)) = 0.79*0.042 / (0.79*0.042 + 0.06*0.958) ~ 0.37
- B) P(r|a,d) = P(a,d|r)P(r) / P(a,d) = P(a|r)P(d|r)P(r) / (P(a|r)P(d|r)P(r) + P(a|-r)P(d|-r)P(-r)) ~ 0.97

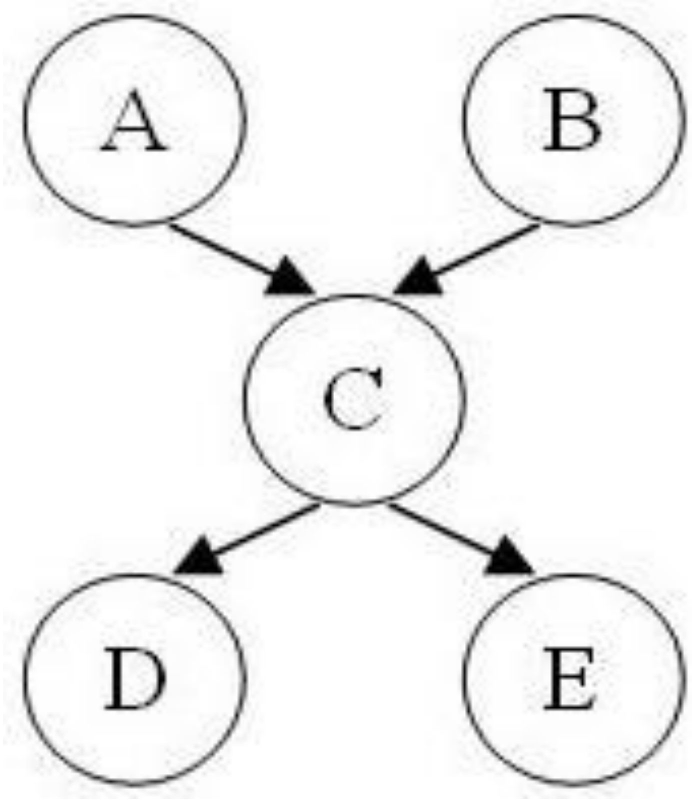
# Problem 4: Common Mistakes

- Attack and Drool are not independent in general – only given Rabid
  - Thus, P(a,d) != P(a)P(d)

- Can't do P(a) = P(a|r) + P(a|-r) – these will generally sum to > 1.

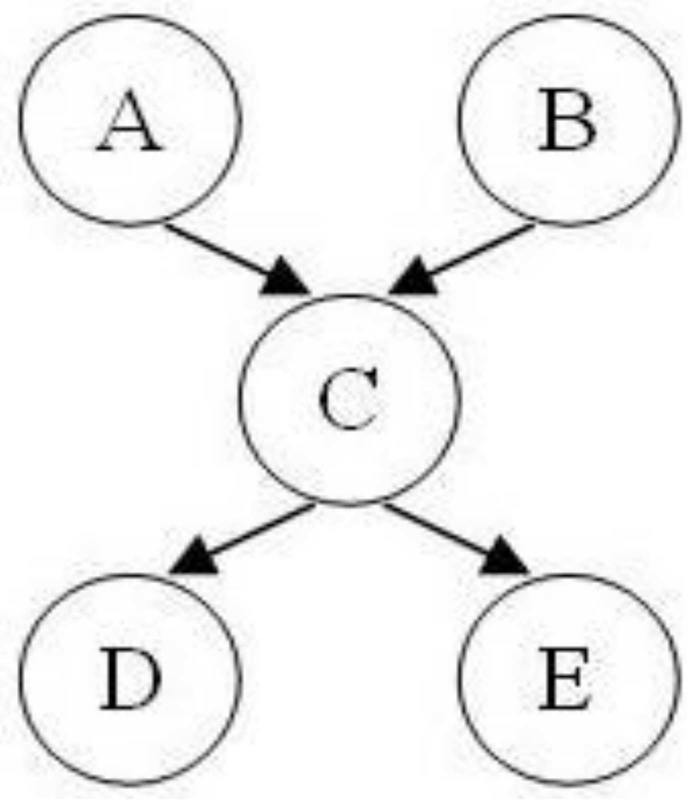
# Problem 5a: Solution

- A) Is D independent of E?
  - No, info flows through C.



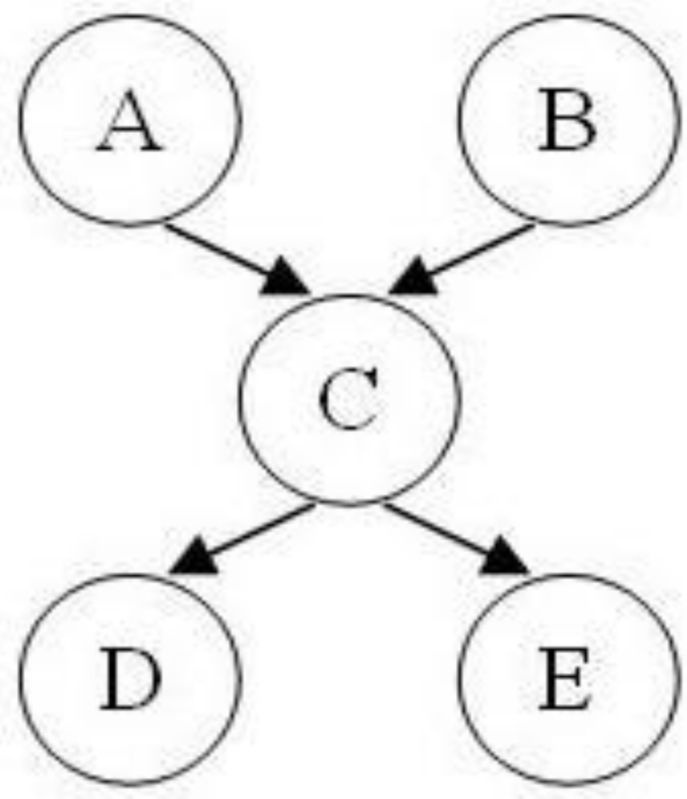
# Problem 5b: Solution

- B) Is A independent of B given C?
  - No, the “explaining away” phenomenon.



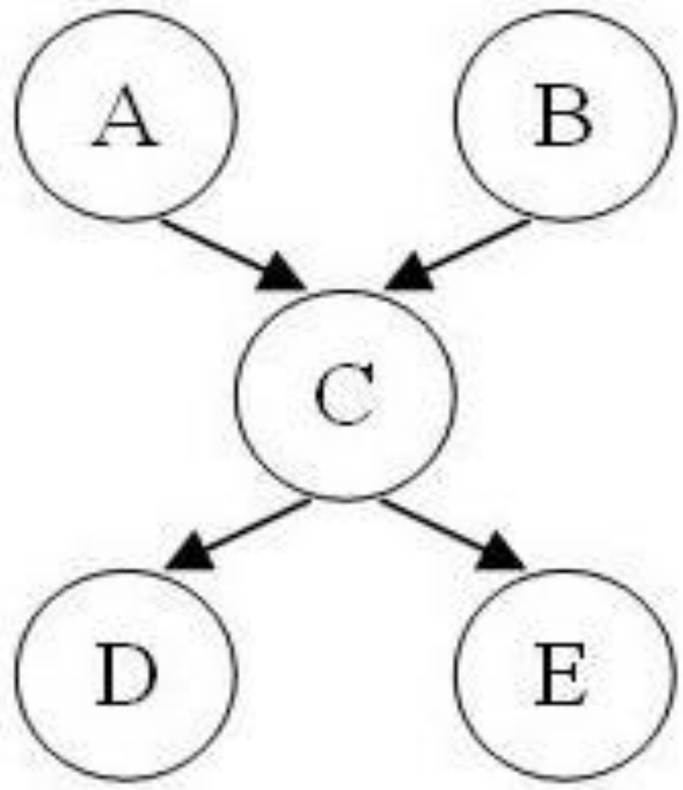
# Problem 5c: Solution

- C) Is E independent of B given C?
  - Yes, C blocks the only information flow path.



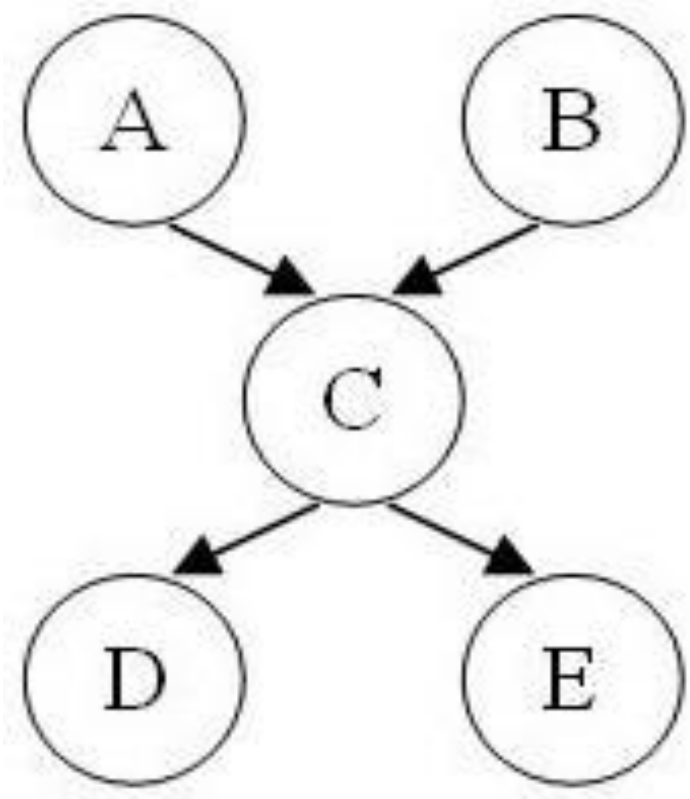
# Problem 5d: Solution

- D) Is A independent of B given D?
  - No, D gives info about C, leading to "explaining away".



# Problem 5e: Solution

- E) Is E independent of D given B?
  - No, info flows through C.



# Outline

- Homework 2 review
- Computational learning theory
- Support vector machines
- Making use of unlabeled data

# Types of Results

- **Learning in the limit**: Is the learner guaranteed to converge to the correct hypothesis in the limit as the number of training examples increases indefinitely?
- **Sample Complexity**: How many training examples are needed for a learner to construct (with high probability) a highly accurate concept?
- **Computational Complexity**: How much computational resources (time and space) are needed for a learner to construct (with high probability) a highly accurate concept?
  - High sample complexity implies high computational complexity, since learner at least needs to read the input data.
- **Mistake Bound**: Learning incrementally, how many training examples will the learner misclassify before constructing a highly accurate concept.

# Learning in the Limit

- Given a continuous stream of examples
  - Learner predicts class for each example then is told the correct answer
  - Does the learner eventually converge to a correct concept?
- No limit on the number of examples required or computational demands
- Must eventually learn the concept exactly
  - Do not need to explicitly recognize this convergence point

# Learning in the Limit

- By simple enumeration, concepts from any known finite hypothesis space are learnable in the limit
  - Know hypothesis space can represent the concept
  - Eliminate hypothesis that are inconsistent with the data

- Typically requires an exponential (or doubly exponential) number of examples and time

# Learning in the Limit vs. PAC Model

- Learning in the limit model is too strong.
  - Requires learning correct exact concept
- Learning in the limit model is too weak
  - Allows unlimited data and computational resources.
- PAC Model
  - Only requires learning a ***Probably Approximately Correct*** Concept: Learn a decent approximation most of the time.
  - Requires polynomial sample complexity and computational complexity.

# PAC Learning

- The only reasonable expectation of a learner is that with ***high probability*** it learns a ***close approximation*** to the target concept.
- In the PAC model, we specify two small parameters, $\varepsilon$ and $\delta$, and require that with probability at least $(1 - \delta)$ a system learn a concept with error at most $\varepsilon$.

# Two Questions

- Overfitting happens because training error is bad estimate of generalization error
  - Can we infer something about generalization error from training error?
- Overfitting happens when learned doesn’t see “enough” examples
  - Can we estimate how many examples are enough?

# Problem Setting

Given

- Set of possible instances X
- Set of possible hypothesis H
- Set of target concepts $c \in C$
- Training instances are generated by an unknown Distribution D over X

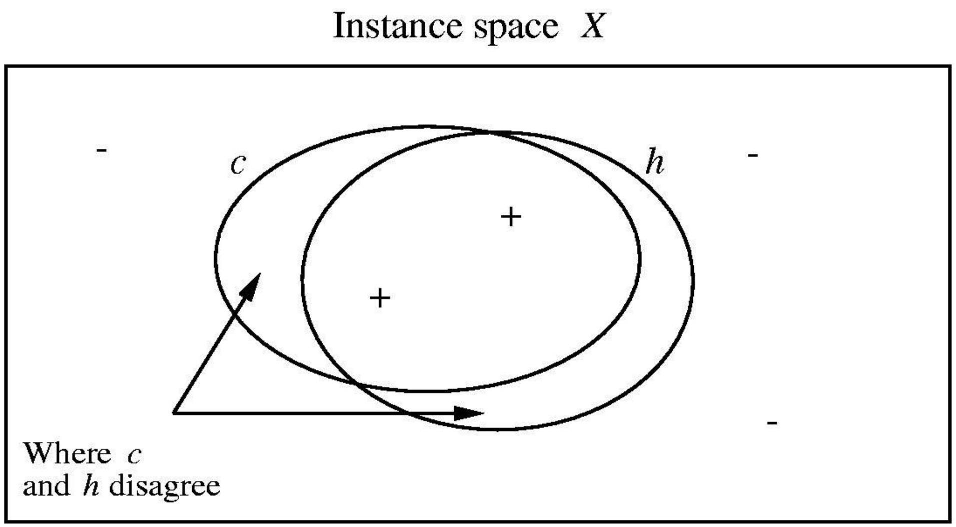
Observe

- some sequence of training data $S = (x_i, c(x_i))$, for some $c \in C$

Do

- Learner outputs some $h \in H$ that approximates c
- Evaluated on future instances drawn from D

# True Error of a Hypothesis




**Definition:** The **true error** (denoted $error_{\mathcal{D}}(h)$) of hypothesis $h$ with respect to target concept $c$ and distribution $\mathcal{D}$ is the probability that $h$ will misclassify an instance drawn at random according to $\mathcal{D}$.

$$error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$$

# Two Notions of Error

**Training error** of hypothesis $h$ with respect to target concept $c$

- How often $h(x) \neq c(x)$ over training instances

**True error** of hypothesis $h$ with respect to $c$

- How often $h(x) \neq c(x)$ over future random instances

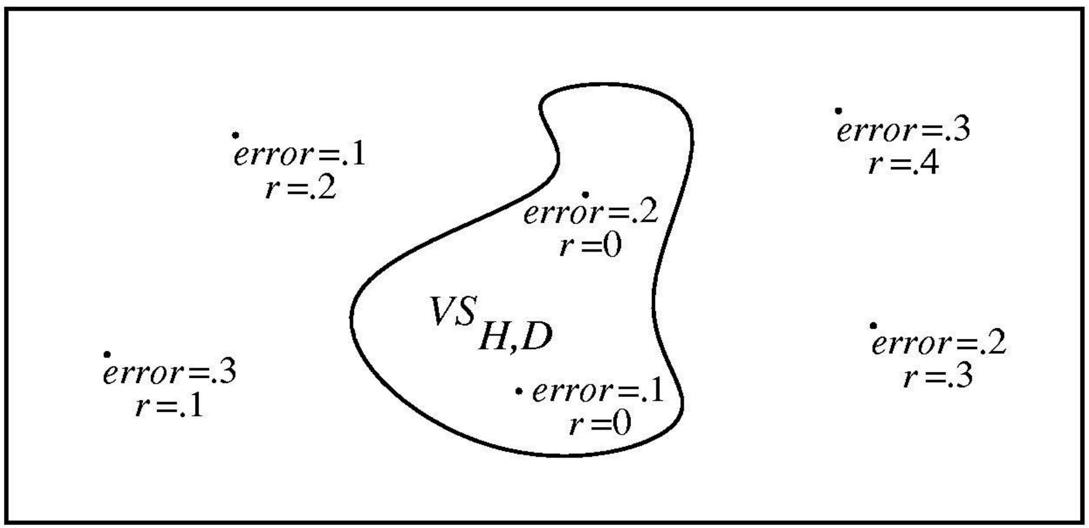
**Our concern:**

- Can we bound the true error of $h$ given the training error of $h$?
- First consider when training error of $h$ is zero

# Version Spaces

**Version Space** $VS_{H,D}$**:**
Subset of hypotheses in $H$ consistent with training data $D$

Hypothesis space $H$




($r$ = training error, $error$ = true error)

# Consistent Learners

- A learner *L* using a hypothesis *H* and training data *D* is said to be a consistent learner if it always outputs a hypothesis with zero error on *D* whenever *H* contains such a hypothesis.
- By definition, a consistent learner must produce a hypothesis in the version space for *H* given *D*.
- Therefore, to bound the number of examples needed by a consistent learner, we just need to bound the number of examples needed to ensure that the version-space contains no hypotheses with unacceptably high error

# ε-Exhausted Version Space

- The version space, $VS_{H,D}$, is said to be ***ε-exhausted*** iff every hypothesis in it has true error less than or equal to ε
- In other words, there are enough training examples to guarantee than any consistent hypothesis has error at most ε
- One can never be sure that the version-space is ε-exhausted, but one can bound the probability that it is not

# How Many Examples Are Enough?

- **Theorem 7.1** (Haussler, 1988): If the hypothesis space *H* is finite, and *D* is a sequence of $m \geq 1$ independent random examples for some target concept *c*, then for any $0 \leq \varepsilon \leq 1$, the probability that the version space $VS_{H,D}$ is ***not*** ε-exhausted is less than or equal to: $|H| e^{-\varepsilon m}$

# Proof

- $H_{bad}=\{h_1,\ldots h_k\}$ is the subset of H w/true error > ε
- The VS is not ε-exhausted if any of these are consistent with all *m* examples
- A single $h_i \in H_{bad}$ is consistent with
  - ***one*** example with probability: $P(\text{consist}(h_i, e_j)) \leq 1 - \varepsilon$
  - ***all*** *m* independent random examples with probability: $P(\text{consist}(h_i, D)) \leq (1 - \varepsilon)^m$
- The probability that ***any*** $h_i \in H_{bad}$ is consistent with all *m* examples is:

$$P(\text{consist}(H_{bad}, D)) = P(\text{consist}(h_1, D) \vee \cdots \vee \text{consist}(h_k, D))$$

# Proof

- Since the probability of a disjunction of events is ***at most*** the sum of the probabilities of the individual events:
  - P(consist($H_{bad}$,D)) $\leq |H_{bad}|(1-\varepsilon)^m$
  - P(consist($H_{bad}$,D)) $\leq |H|e^{-\varepsilon m}$
- Since: $|H_{bad}| \leq |H|$ and $(1-\varepsilon)^m \leq e^{-\varepsilon m}$, $0 \leq \varepsilon \leq 1$, $m \geq 0$

**Q.E.D**

# Sample Complexity Analysis

- Let $\delta$ be an upper bound on the probability of ***not*** exhausting the version space
  - $|H|e^{-\varepsilon m} \leq \delta$
  - $e^{-\varepsilon m} \leq \delta/|H|$
  - $-\varepsilon m \leq \ln(\delta/|H|)$
  - $m \geq -\ln(\delta/|H|)/\varepsilon$
  - $m \geq \ln(|H|/\delta)/\varepsilon$
  - $m \geq [\ln(1/\delta) + \ln|H|]/\varepsilon$

# PAC Learning Definition

- A concept is PAC learnable if:
  - For <u>any</u> target c in C and any distribution D on X
  - <u>Given</u> at least N = poly(|c|,1/$\varepsilon$,1/$\delta$) examples drawn randomly, independently from X
  - <u>Do</u> with probability $1 - \delta$, return an h in C whose accuracy is at least $1 - \varepsilon$
  - In other words, Prob[error(**h**, **c**) > $\varepsilon$] < $\delta$, In time polynomial in |data|

# Sample Complexity Results

- Any consistent learner, given at least $[\ln(1/\delta) + \ln|H|]/\varepsilon$ examples will produce a PAC result
- Just determine the size of a hypothesis space for learning specific classes of concepts.
- This gives a ***sufficient*** number of examples for PAC learning, but ***not*** a ***necessary*** number
- Several approximations like that used to bound the probability of a disjunction make this a gross over-estimate in practice

# Sample Complexity: Conjunctions

- Consider conjunctions over *n* boolean features
  - $3^n$ since each feature can appear positively, appear negatively, or not appear
  - Therefore $|H| = 3^n$,
  - Sufficient number of examples is: $[\ln(1/\delta) + n \ln 3]/\varepsilon$
- Concrete examples:
  - $\delta=\varepsilon=0.05$, $n=10$ gives 280 examples
  - $\delta=0.01$, $\varepsilon=0.05$, $n=10$ gives 312 examples
  - $\delta=\varepsilon=0.01$, $n=10$ gives 1,560 examples
  - $\delta=\varepsilon=0.01$, $n=50$ gives 5,954 examples

# Sample Complexity of Learning Arbitrary Boolean Functions

- Any boolean function over *n* boolean features
  - E.g., DNF or decision trees.
  - There are $2^{2^n}$ of these,
  - Sufficient number of examples is:
    $[\ln(1/\delta) + 2^n \ln 2]/\varepsilon$
- Concrete examples:
  - $\delta=\varepsilon=0.05$, $n=10$ gives 14,256 examples
  - $\delta=\varepsilon=0.05$, $n=20$ gives 14,536,410 examples
  - $\delta=\varepsilon=0.05$, $n=50$ gives $1.561 \times 10^{16}$ examples

# Agnostic Learning

- So far, we assumed that $c \in H$
- Agnostic learning: don't assume that $c \in H$
- What can we say here
  - Assume one hypothesis h, with m independently chosen examples, use Hoeffding bound
  - $P(error_{\mathcal{D}}(h) > P(error_D(h) + \varepsilon) \leq e^{-2m\varepsilon^2}$
- Then for all hypothesis:
  - $P[(h \in H)(error_{\mathcal{D}}(h) > P(error_D(h) + \varepsilon)] \leq |H|e^{-2m\varepsilon^2}$

# Agnostic Learning

- Sample complexity:
  - $m \geq [1/2\varepsilon^2][\ln(1/\delta) + \ln|H|]$

- m depends logarithmically on H and 1/δ

- m grows on the square of 1/ε as opposed to linearly as before

# Handling Infinite Hypothesis Spaces

- The previous analysis was restricted to finite hypothesis spaces
- Some infinite hypothesis spaces (such as those including real-valued thresholds or parameters) are more expressive than others.
  - Rule allowing one threshold on a continuous feature (length<3cm)
  - Rule allowing two thresholds (1cm<length<3cm)
- Need some measure of the expressiveness of infinite hypothesis spaces.

# Handling Infinite Hypothesis Spaces

- The ***Vapnik-Chervonenkis*** (***VC***) ***dimension***, denoted VC(*H*), measures expressivity of infinite hypothesis spaces

- Analagous to ln|*H*|, there are bounds for sample complexity using VC(*H*).

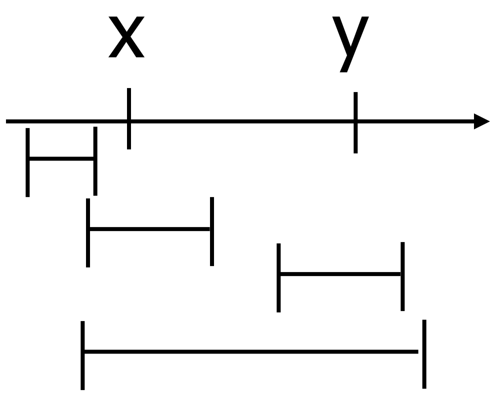
- VC-dim ≡ given a hypothesis space H, the VC-dim is the size of the largest set of examples that can be completely fit by H, no matter how the examples are labeled

# VC-Dimension Impact

- If the number of examples << VC-dim, then memorizing training is trivial and generalization likely to be poor
- If the number of examples >> VC-dim, then the algorithm must generalize to do well on the training set and will likely do well in the future

# Definition: Shattering

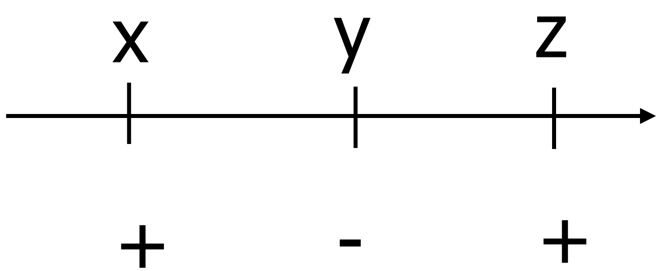
- A hypothesis space is said to shatter a set of instances iff for every partition of the instances into positive and negative, there is a hypothesis that produces that partition
- Example: Consider 2 instances with a single real-valued feature being shattered by intervals




| + | − |
|---|---|
| | x,y |
| x | y |
| y | x |
| x,y | |

# Definition: Shattering

- But 3 instances cannot be shattered by a single interval

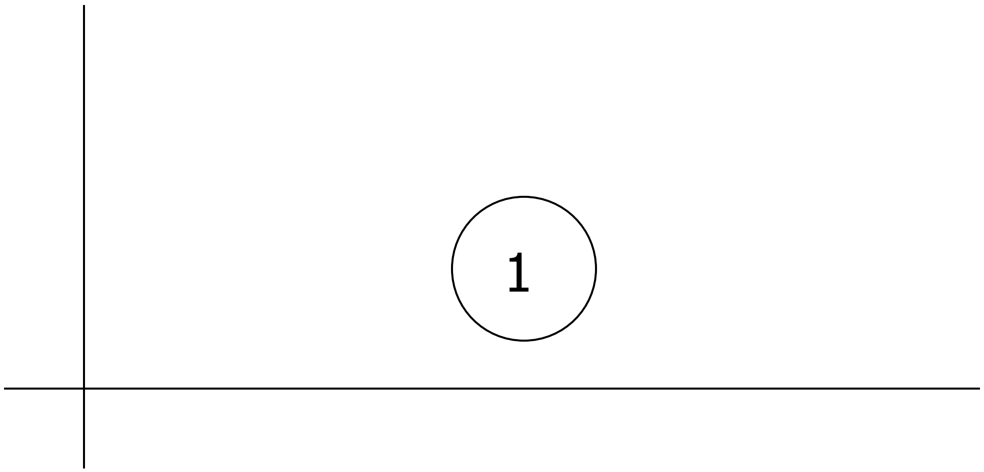



- Since there are $2^m$ partitions of $m$ instances, in order for $H$ to shatter instances: $|H| \geq 2^m$.

# Shattering: Example

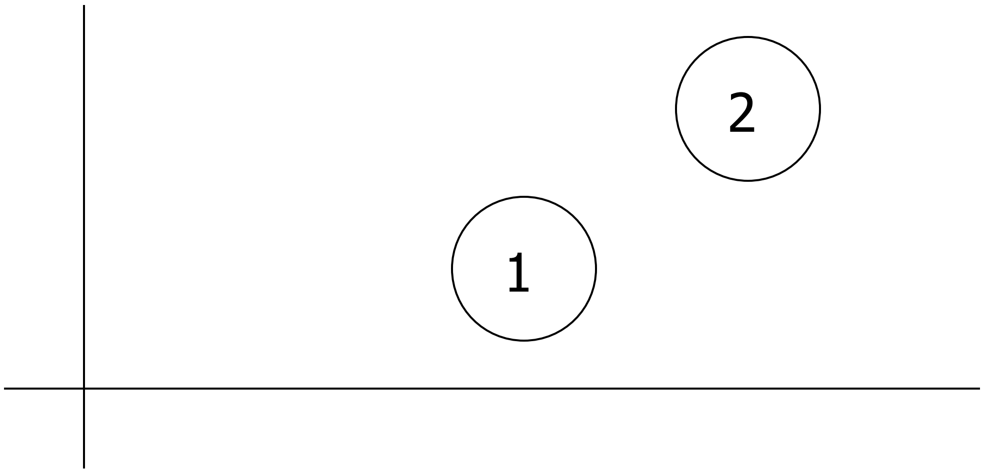
H is set of lines in 2D

Can cover 1 ex no matter how labeled



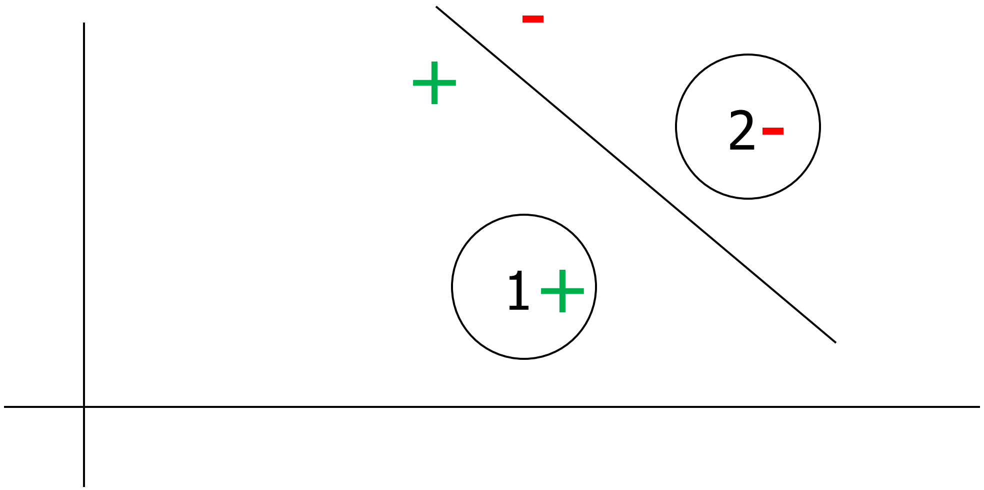
# Shattering: Example

Can cover 2 ex's no matter how labeled



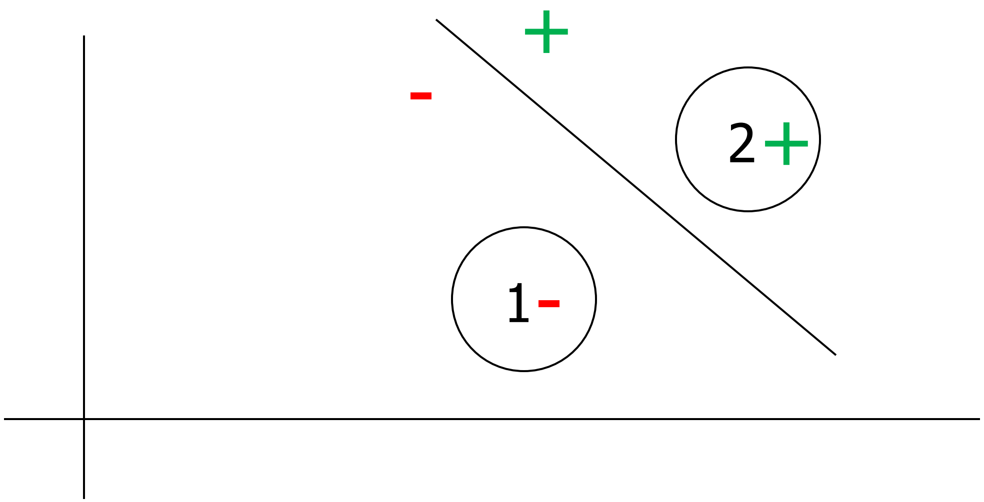
# Shattering: Example

Can cover 2 ex's no matter how labeled



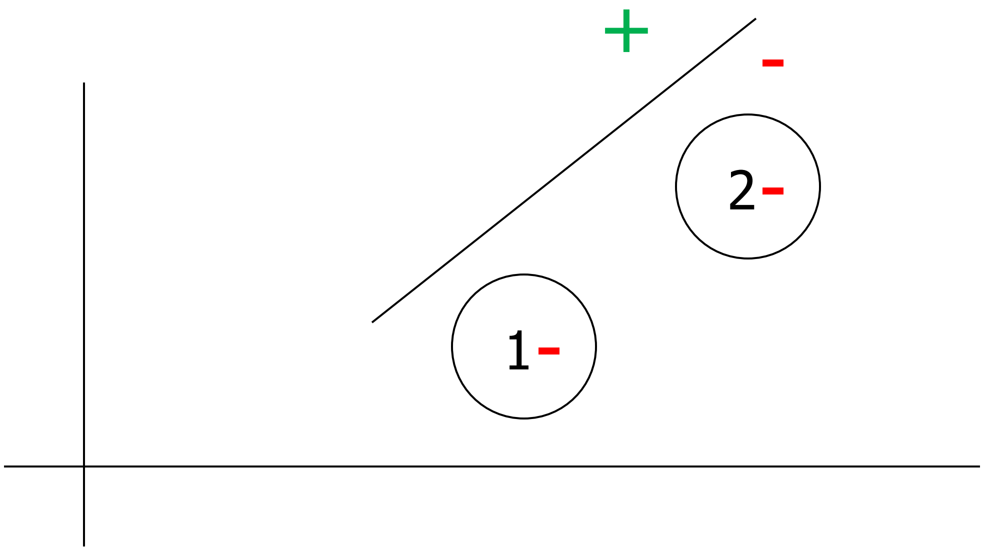
# Shattering: Example

Can cover 2 ex's no matter how labeled



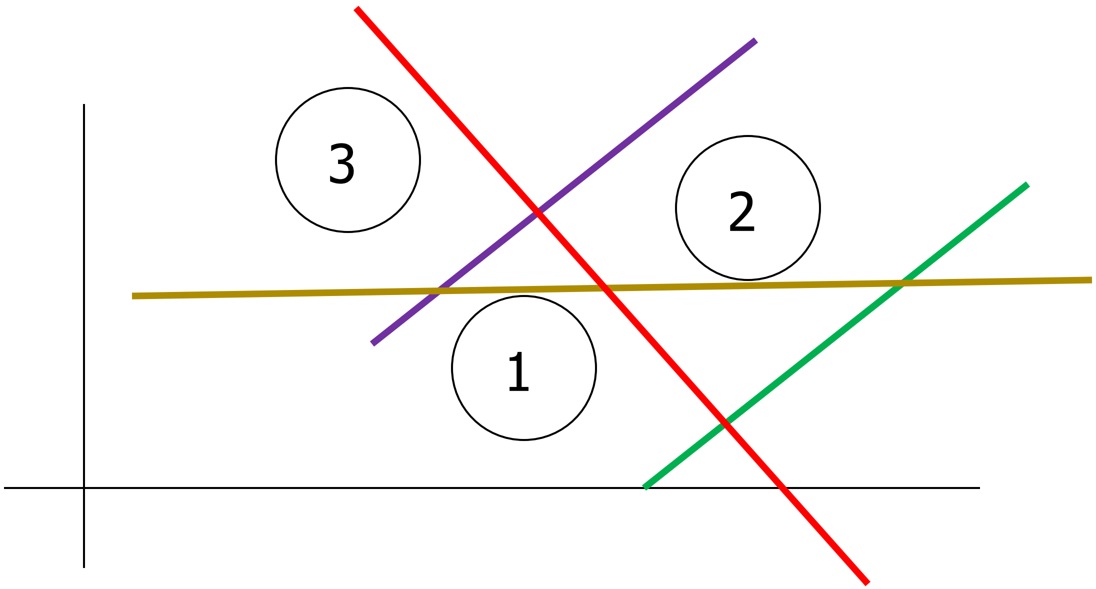
# Shattering: Example

Can cover 2 ex's no matter how labeled



# Shattering: Example

Can cover 3 ex's no matter how labeled




1,2 are same class

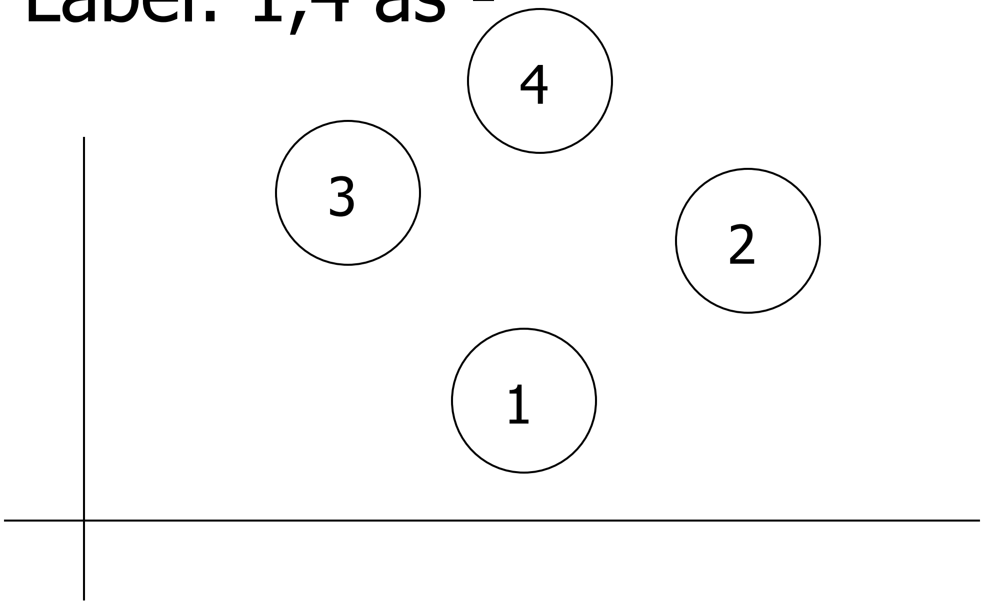
1,2,3 are same class

1,3 are same class

2,3 are same class

# Shattering: Example

Cannot cover <u>4 ex's:</u> XOR!
Label: 2,3 as +
Label: 1,4 as -




Notice |H| = ∞
but VC-dim = 3

For N-dimensions
and N-1 dim
hyperplanes,
VC-dim = N + 1

# More on Shattering

What about collinear points?

If ∃ some set of $d$ examples that $H$ can fully fit ∀ labellings of these $d$ examples then $VC(H) \geq d$

# VC Dimensions

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 subsets of $X$ can be shattered then VC($H$) = $\infty$

# Examples

- An unbiased hypothesis space shatters the entire instance space
- The larger the subset of *X* that can be shattered, the more expressive the hypothesis space is, i.e. the less biased
- If at least one subset of *X* of size *d* exists that can be shattered then $VC(H) \geq d$. If no subset of size *d* can be shattered, then $VC(H) < d$
- Finite hypothesis space: VC-Dim $\leq \log_2|H|$

## Sample Complexity from VC Dimension

How many randomly drawn examples suffice to guarantee error of at most $\epsilon$ with probability at least $(1 - \delta)$?

$$m \geq \frac{1}{\epsilon}(4 \log_2(2/\delta) + 8VC(H) \log_2(13/\epsilon))$$

# Mistake-Bound Model

- Teacher shows input I
- ML algorithm guesses output O
- Teacher shows correct answer
- Can we upper bound the number of errors the learner will make?

# Mistake Bound Model

<u>Example</u>  Learn a conjunct from *N* predicates and their negations

1. Initial $h = p_1 \wedge \neg p_1 \wedge \ldots \wedge p_n \wedge \neg p_n$
2. For each + ex, remove the remaining terms that do not match

# Mistake Bound Model

Worst case # of mistakes?

$$1 + N$$

1. First + ex will remove $N$ terms from $h_{initial}$

2. Each subsequent error on a + will remove at least one more term (never make a mistake on - ex's)

# Outline

- Homework 2 review
- Computational learning theory
- Support vector machines
- Making use of unlabeled data

# What is a Support Vector Machine

- A subset of the examples (the support vectors)
- A vector of weights for them
- A similarity function K(xi,xj) (the kernel)

- Predict: $o_q = \text{sign}( \Sigma_j \, \alpha_j o_j \, K(x_{j,} \, x_q)$

- $o_q = \{-1,+1\}$

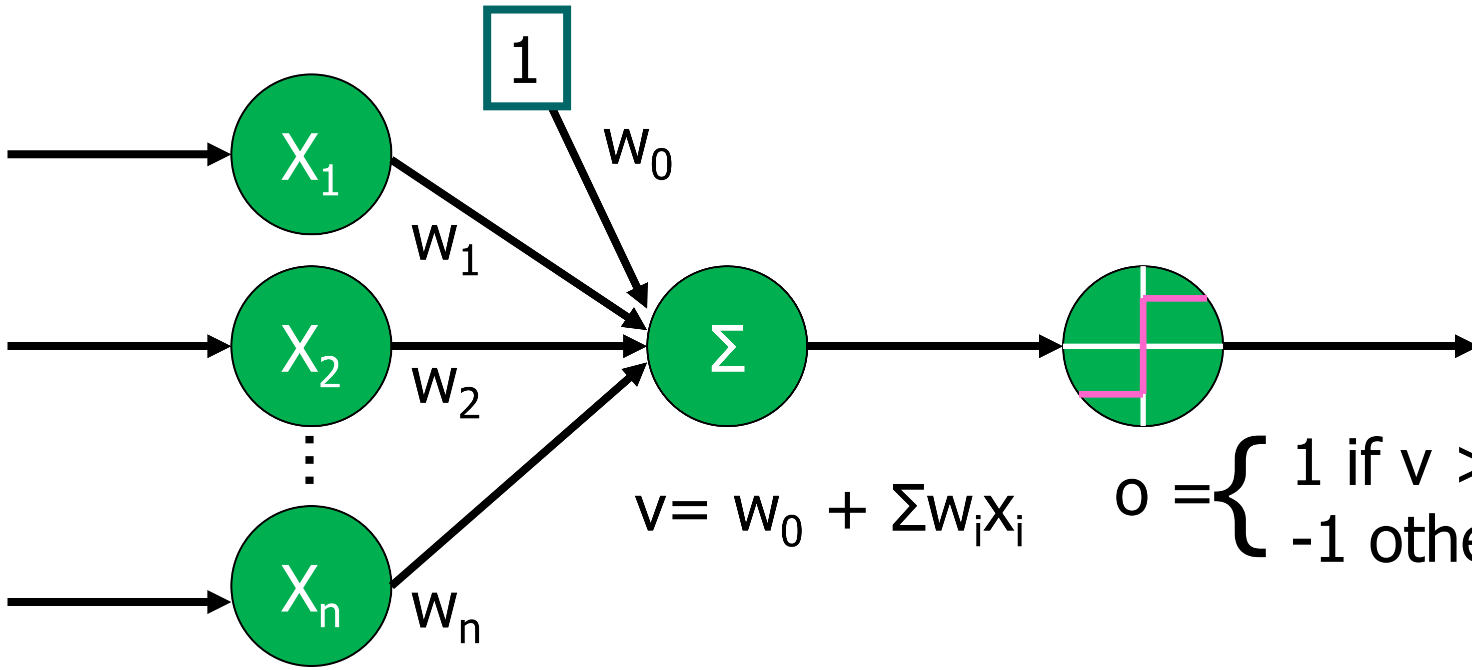
# SVMs and Perceptrons

- So SVMs are a form of instance based learning
- However, SVMs are usually presented as a generalization of a perceptron
- What the relationship between instance-based learning the perceptron?

# Notation

- $<x,w> = \Sigma w_i x_i$
- $<x,w> = <w,x>$
- $r<x,w> = <rw,w>$ [r is a real]
- $<x+y,w> = <x,w> + <y,w>$

# Perceptron Revisited




$$v = w_0 + \Sigma w_i x_i$$

$$o = \begin{cases} 1 & \text{if } v > 0 \\ -1 & \text{otherwise} \end{cases}$$

Vector Notation

$$o(x) = \begin{cases} 1 & \text{if } \langle w\ x \rangle + w_0 > 0 \\ -1 & \text{otherwise} \end{cases}$$

# Perceptron Training Rule

- Assume that $o_j = \{-1,+1\}$
- Weight update rule: $w_i = w_i + \eta(t_j-o_j)x_{j,i}$
  - $\eta = 1/2$
  - If $o_j = +1$ then $w_i = w_i + x_{j,i}$
  - If $o_j = -1$ then $w_i = w_i - x_{j,i}$
- Rewrite as: $w_i = w_i + o_j x_{j,i}$
- $w_i = \Sigma_j \alpha_j o_j x_{j,i}$

# Dual Form of Perceptron

- $w_i = \Sigma_j \alpha_j o_j x_{j,i}$
- Label = $<w,x_q> + w_0$
- Label = $\Sigma_j \alpha_j o_j <x_j,x_q> + w_0$

- Called the dual form because the example appears only within a dot product

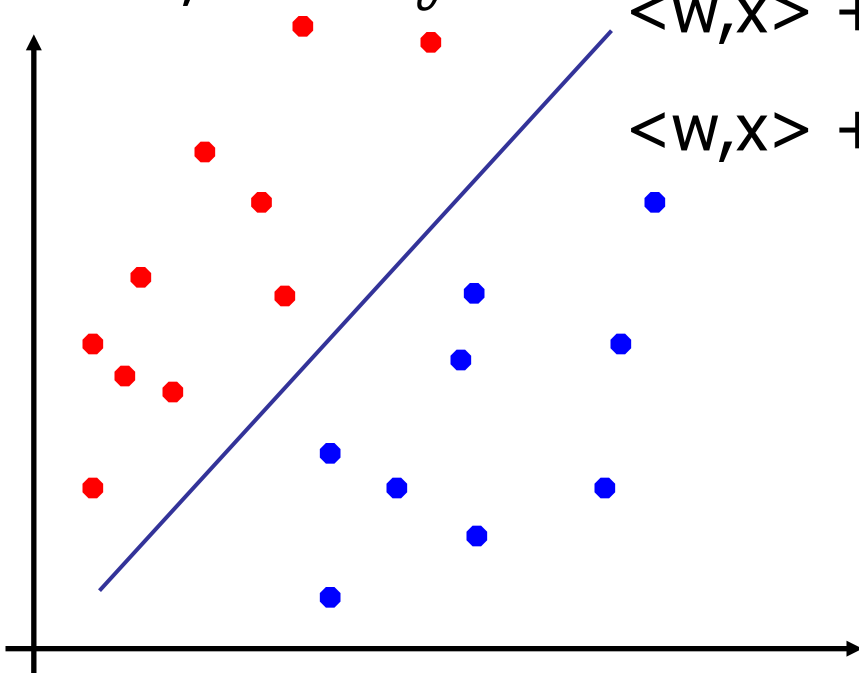
# Perceptron: Linear Separator

- Binary classification can be viewed as the task of separating classes in feature space:

$<w,x> + w_0 > 0$

$<w,x> + w_0 = 0$

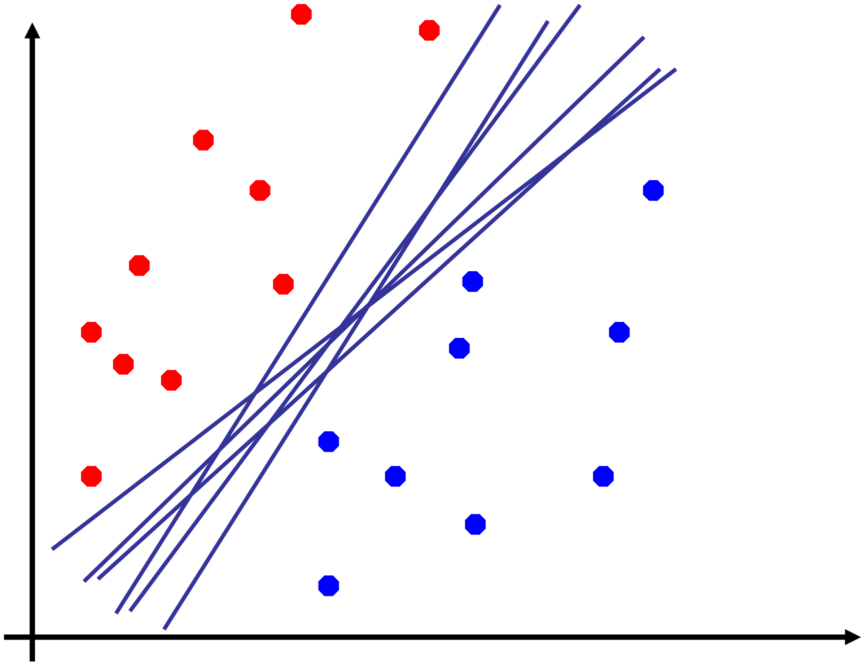
$<w,x> + w_0 < 0$



$f(x) = sign(<w,x> + w_0)$

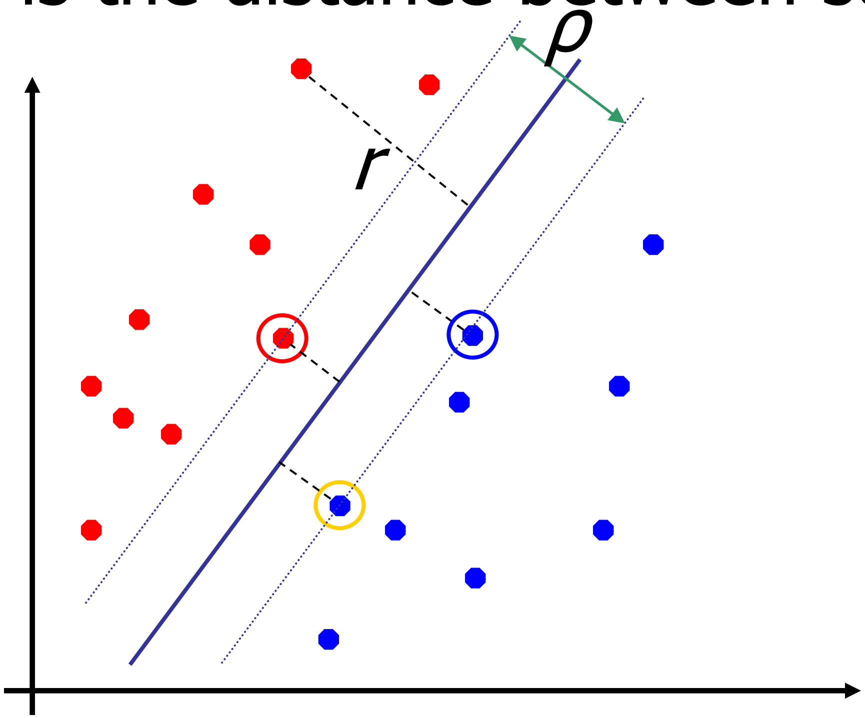
# Linear Separators

- Which of the linear separators is optimal?



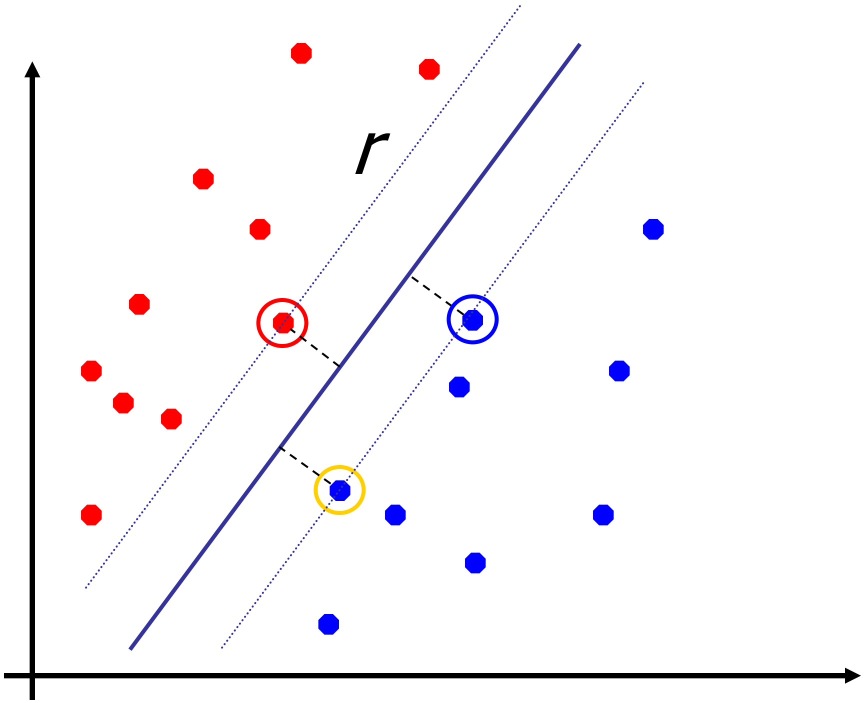
# Idea: Classification Margin

- Support vectors: Examples closest to the hyperplane
- Margin $\rho$ is the distance between support vectors



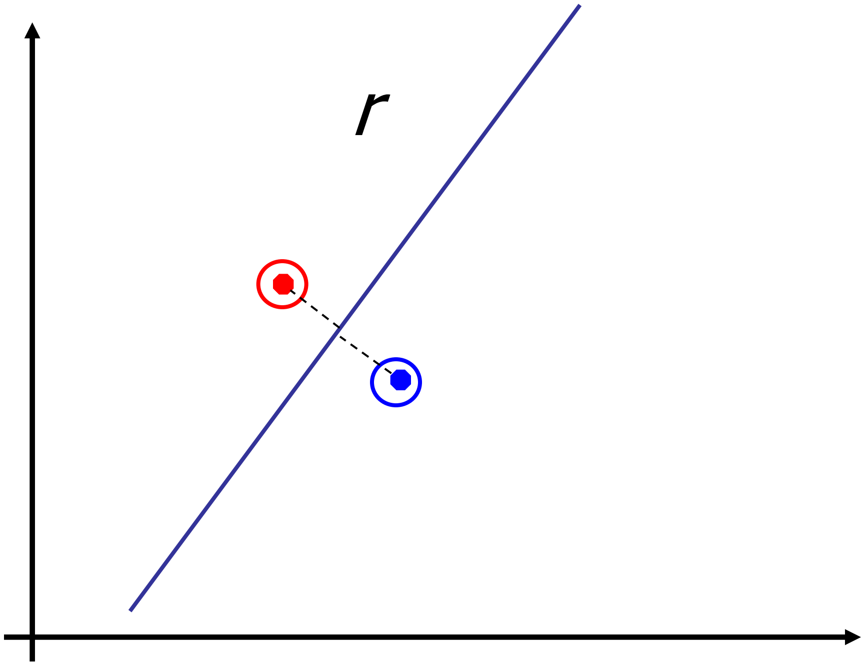
# Maximum Margin Classification

- Intuitive this feels safest
- Implication: Only support vectors matter



# Computing Margin Width

- $<w, x_r> + w_0 = 1$
- $<w, x_b> + w_0 = -1$
- $x_r = x_b + l * w$



# Computing Margin Width

- $\langle w, x_r \rangle + w_0 = 1$
- $\langle w, x_b \rangle + w_0 = -1$
- $x_r = x_b + l * w$
- $|x_r - x_b| = M$

# Computing Margin Width

- $<w,x_r> + w_0 = 1$
- $<w,x_b> + w_0 = -1$
- $x_r = x_b + l * w$
- $|x_r - x_b| = M$
- $w <x_b + l*w> + w_0 = 1$

# Computing Margin Width

- $<w,x_r> + w_0 = 1$
- $<w,x_b> + w_0 = -1$
- $x_r = x_b + l * w$
- $|x_r - x_b| = M$
- $w <x_b + l*w> + w_0 = 1$
- $<w,x_b> + w_0 + <w,l*w> = 1$
- $-1 + l<w,w> = 1$

# Computing Margin Width

- $<w,x_r> + w_0 = 1$
- $<w,x_b> + w_0 = -1$
- $x_r = x_b + l * w$
- $|x_r - x_b| = M$
- $w <x_b + l*w> + w_0 = 1$
- $<w,x_b> + w_0 + <w,l*w> = 1$
- $-1 + l<w,w> = 1$
- $l = 2 / <w,w>$

# Linear SVM Mathematically

- Goal: Maximize the margin
- Objective: minimize <**w**,**w**>

- **Quadratic optimization problem:**

Find **w** and *b* such that

$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}\ \mathbf{w}$ is minimized

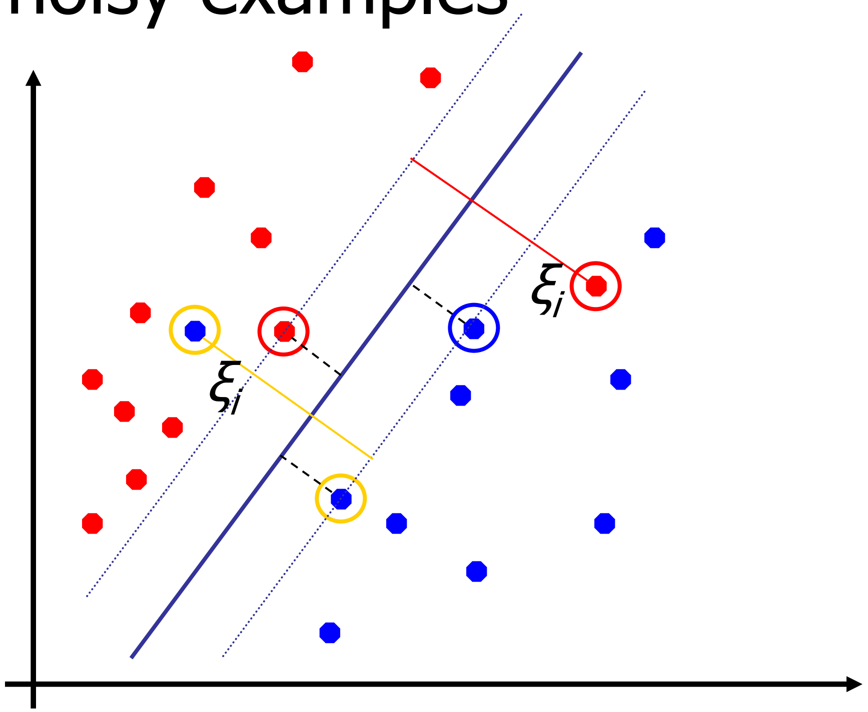
and for all $(\mathbf{x}_i, y_i)$, $i$=1..$n$ : $y_i\ (<\mathbf{w},\mathbf{x}_i> + w_0) \geq 1$

# Solving the Optimization Problem

- Need to optimize a *quadratic* function subject to *linear* constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist
- Not a part of this class

# Soft Margin Classification

- If the training set is not linearly separable?
- *Slack variables* $\xi_i$ allows misclassification of difficult/noisy examples



# Soft Margin Classification Mathematically

- Modified formulation incorporates slack variables:

Find **w** and b such that

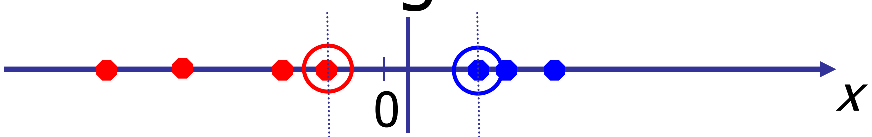
$\mathbf{\Phi}(\mathbf{w}) = \mathbf{w}\,\mathbf{w} + C\sum\xi_i$ is minimized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$ : $y_i\,(<\mathbf{w},\mathbf{x}_i> + w_0) \geq 1 - \xi_i$ , $\xi_i \geq 0$

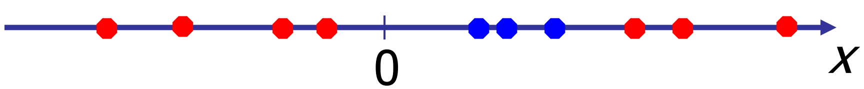
- Parameter $C$ can be viewed as a way to control overfitting: it "trades off" the relative importance of maximizing the margin and fitting the training data.

# Non-Linear SVMs

- Datasets that are linearly separable with some noise work out great:

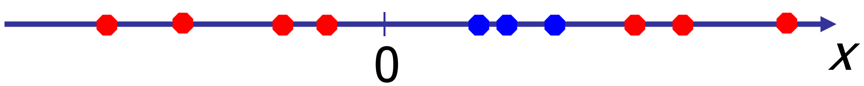


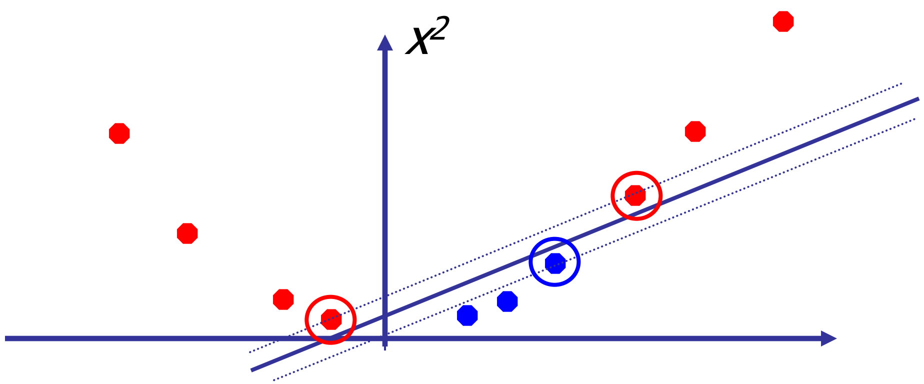

- But what are we going to do if the dataset is just too hard?



# Non-Linear SVMs

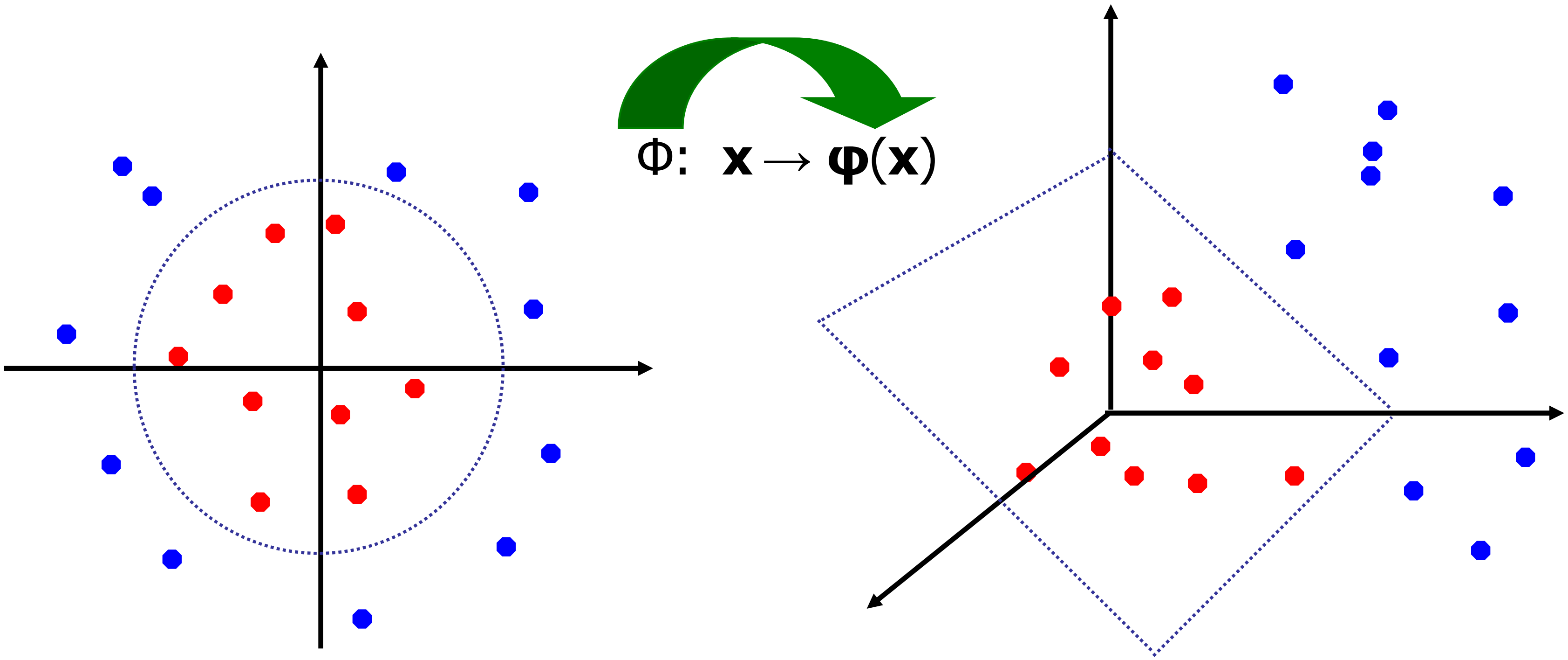
- How about… mapping data to a **higher-dimensional space:**



# Non-linear SVMs:  Feature spaces

- General idea: Original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



# The "Kernel Trick"

- The linear classifier relies on inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$
- If map every datapoint into high-dimensional space via some transformation $\Phi$: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product: $K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\varphi}(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x}_j)$
- A *kernel function* is a function that is equivalent to an inner product in some feature space.
- Kernel function *implicitly* maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\varphi}(\mathbf{x})$ explicitly).

# Another View of SVMs

- Take the perceptron
- Replace dot product with arbitrary similarity function
- Now you have a much more powerful learner
- Kernel matrix: $K(x, x')$ for $x, x' \in$ Data
- If a symmetric matrix $K$ is positive semi-definite (i.e., has non-negative eigenvalues), then $K(x, x')$ is still a dot product, but in a transformed space:

$$K(x, x') = \phi(x) \cdot \phi(x')$$

- Also guarantees convex weight optimization problem
- Very general trick

# Bounds

**Margin bound:**

Bound on VC dimension decreases with margin

**Leave-one-out bound:**

$$E[error_{\mathcal{D}}(h)] \leq \frac{E[\#\text{ support vectors}]}{\#\text{ examples}}$$

# SVM Key Ideas

- Dual problem: Weights on examples (vs. features)
- Maximize the margin
- Kernel trick

# Outline

- Homework 2 review
- Computational learning theory
- Support vector machines
- Making use of unlabeled data

# Using Unlabeled Data

Q: Where does labeled data come from??

- Some tasks, people are willing to label
  - Netflix, amazon, etc.
  - Spam
  - Medical diagnoses
- Often, we have to get people to label data
  - Web ranking
  - Document classification

Problem: Labeling data is expensive!

# Using Unlabeled Data

- Learning methods need labeled data
  - Lots of <x, f(x)> pairs
  - Hard to get… (who wants to label data?)
- But unlabeled data is usually plentiful…Could we use this instead??????
  - Semi-supervised learning
  - Active learning

# Cotraining

- Have ***little*** labeled data + ***lots*** of unlabeled
- Each instance has two parts:

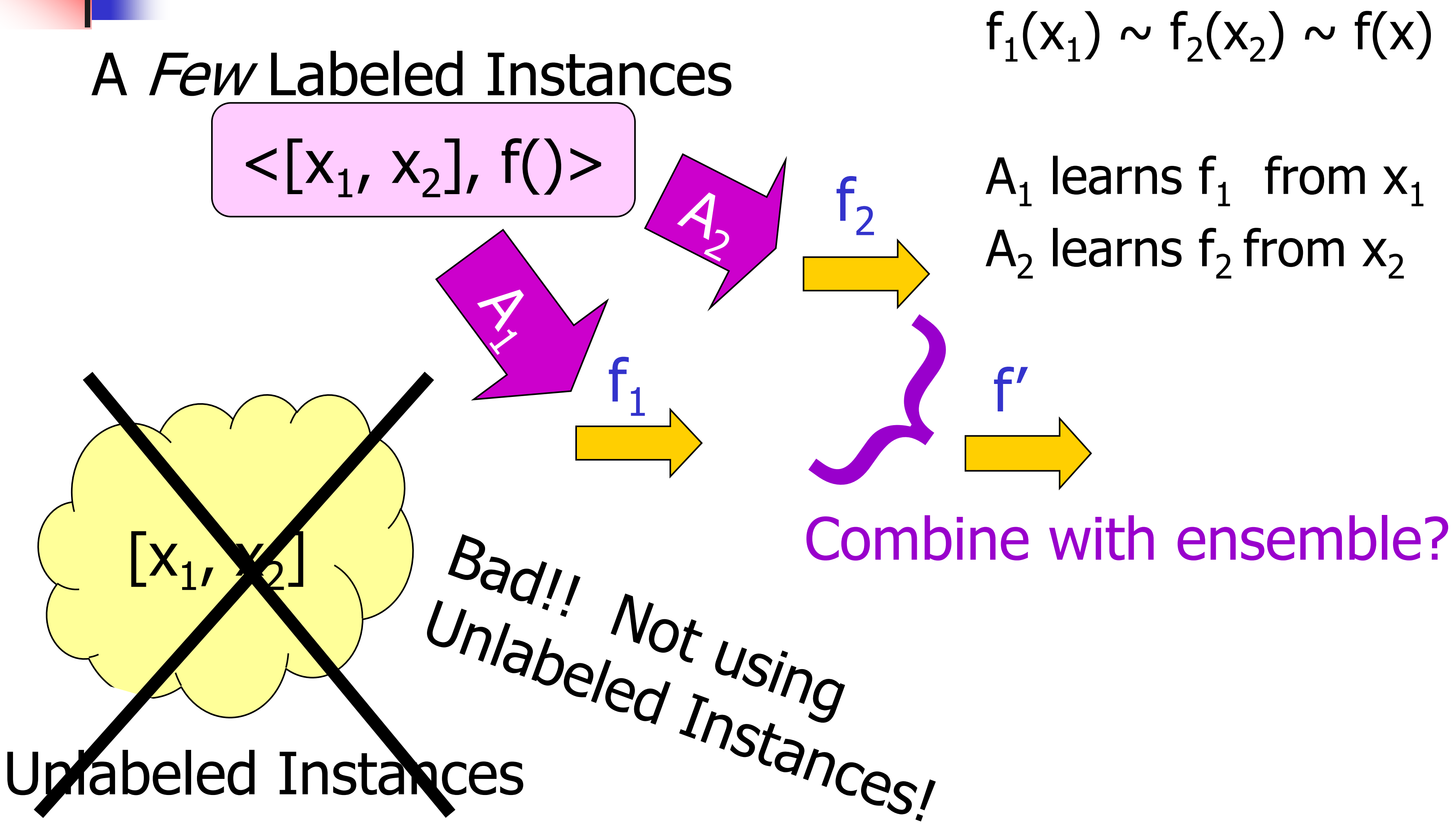
  $x = [x1, x2]$

  x1, x2 conditionally independent given f(x)
- Each half can be used to classify instance

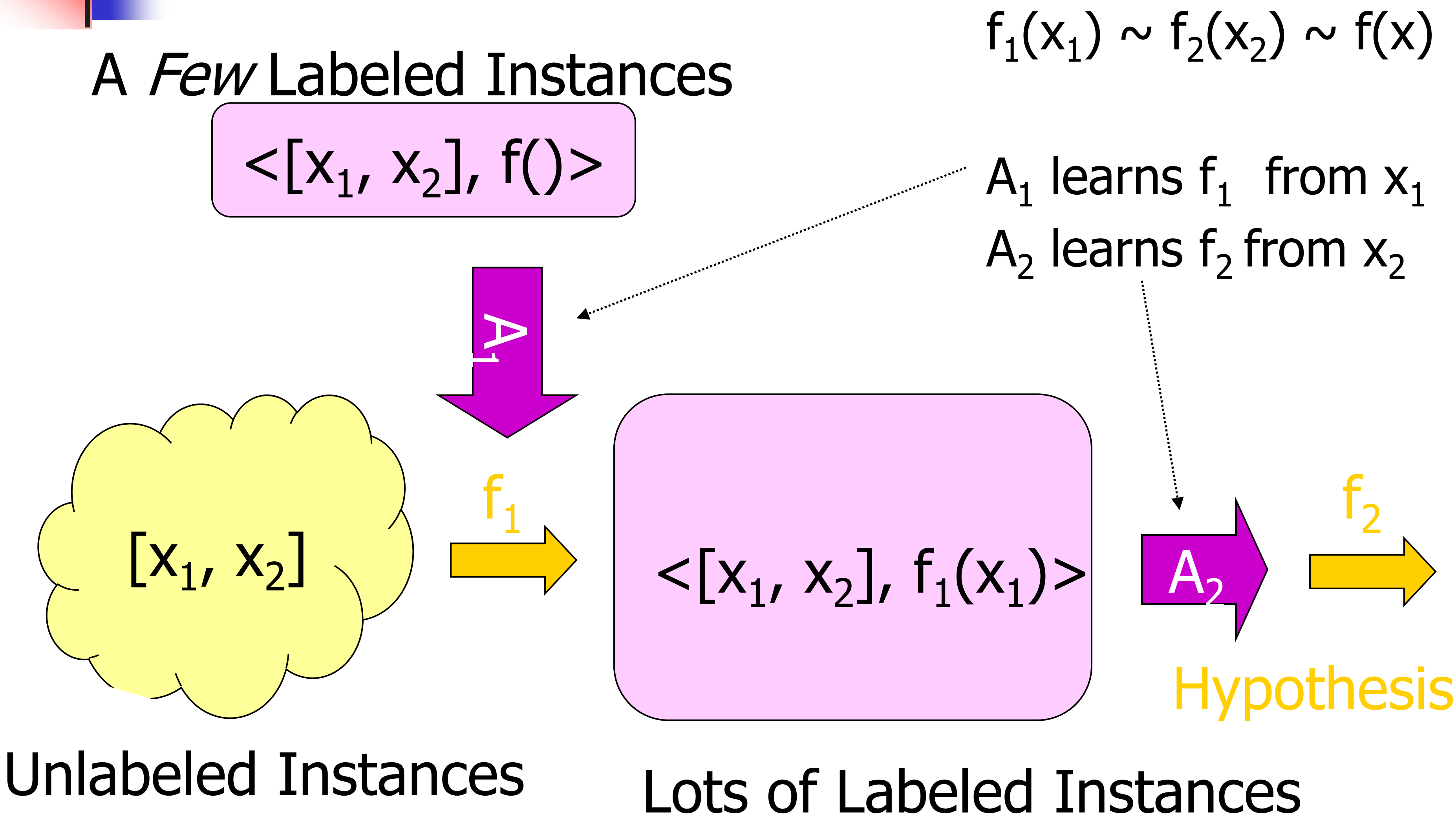
  $\exists f1, f2$ such that $f1(x1) \sim f2(x2) \sim f(x)$
- Both f1, f2 are learnable

  $f1 \in H1, \quad f2 \in H2, \quad \exists$ learning algorithms A1, A2

# Without Co-training



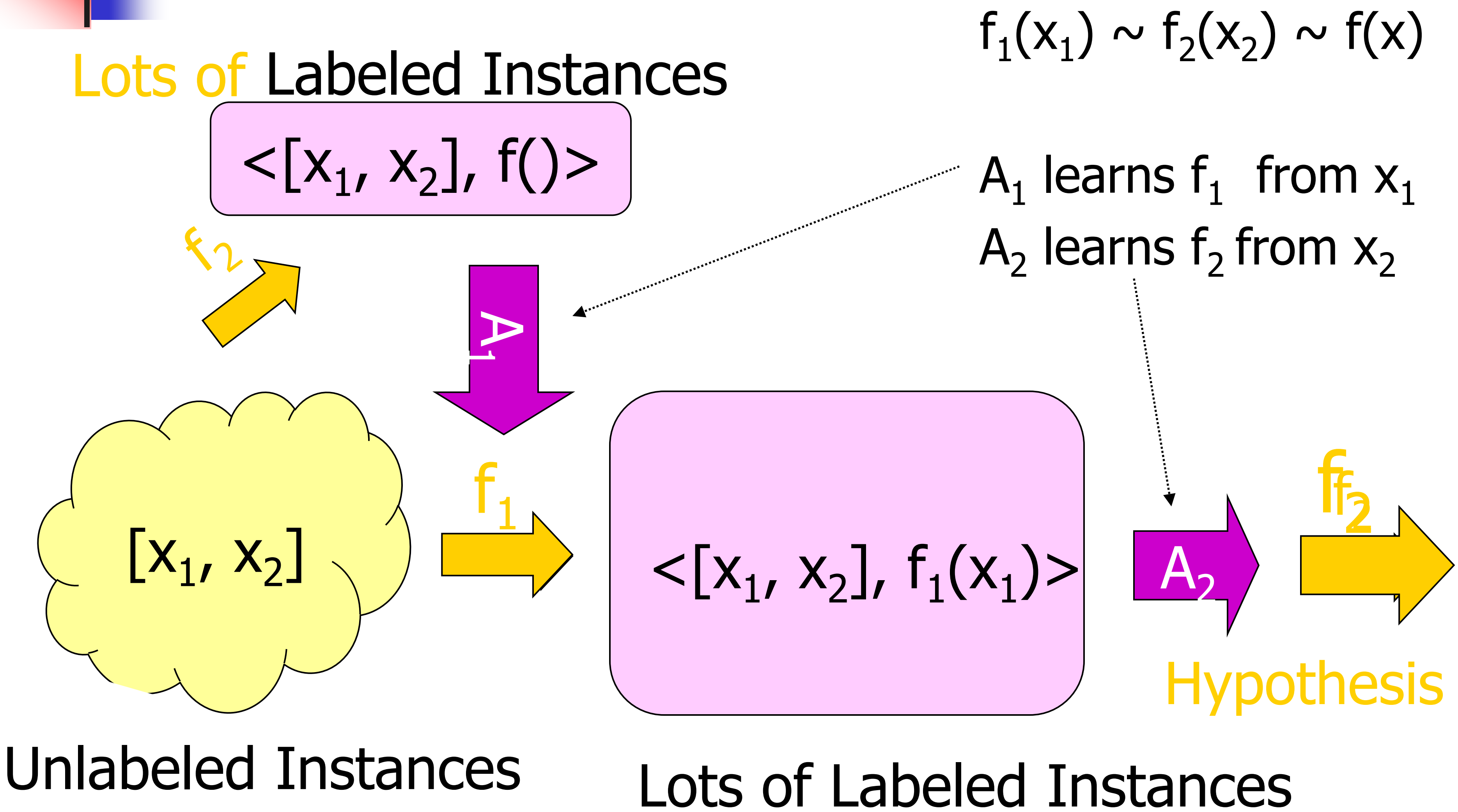
# Cotrainng


$f_1(x_1) \sim f_2(x_2) \sim f(x)$
A *Few* Labeled Instances
$<[x_1, x_2], f()>$
$A_1$ learns $f_1$ from $x_1$
$A_2$ learns $f_2$ from $x_2$
$A_1$
$f_1$
$[x_1, x_2]$
$<[x_1, x_2], f_1(x_1)>$
$A_2$
$f_2$
Hypothesis
Unlabeled Instances
Lots of Labeled Instances

# Observations

- Can apply $A_1$ to generate as much training data as one wants
  - If $x_1$ is conditionally independent of $x_2$ / f(x),
  - then the error in the labels produced by $A_1$
  - ***will look like random noise to $A_2$ !!!***
- Thus ***no limit*** to quality of the hypothesis $A_2$ can make

# Co-training


f1(x1) ~ f2(x2) ~ f(x)
Lots of Labeled Instances
<[x1, x2], f()>
A1 learns f1 from x1
A2 learns f2 from x2
f2
A1
f1
[x1, x2]
<[x1, x2], f1(x1)>
A2
f2
Hypothesis
Unlabeled Instances
Lots of Labeled Instances

# It Really Works!

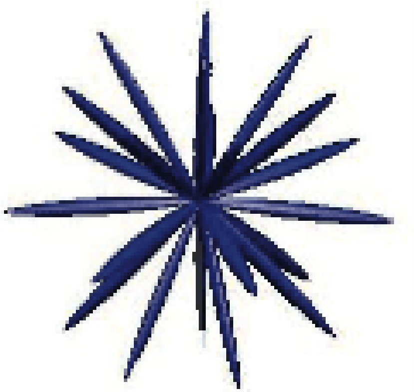
- Learning to classify web pages as course pages
  - x1 = bag of words on a page
  - x2 = bag of words from all anchors pointing to a page
- Naïve Bayes classifiers
  - 12 labeled pages
  - 1039 unlabeled

| | Page-based classifier | Hyperlink-based classifier | Combined classifier |
|---|---|---|---|
| Supervised training | 12.9 | 12.4 | 11.1 |
| Co-training | 6.2 | 11.6 | 5.0 |

Percentage error

# Thought Experiment

- suppose you're the leader of an Earth convoy sent to colonize planet Mars



people who ate the round Martian fruits found them *tasty!*

people who ate the spiked Martian fruits ***died***!

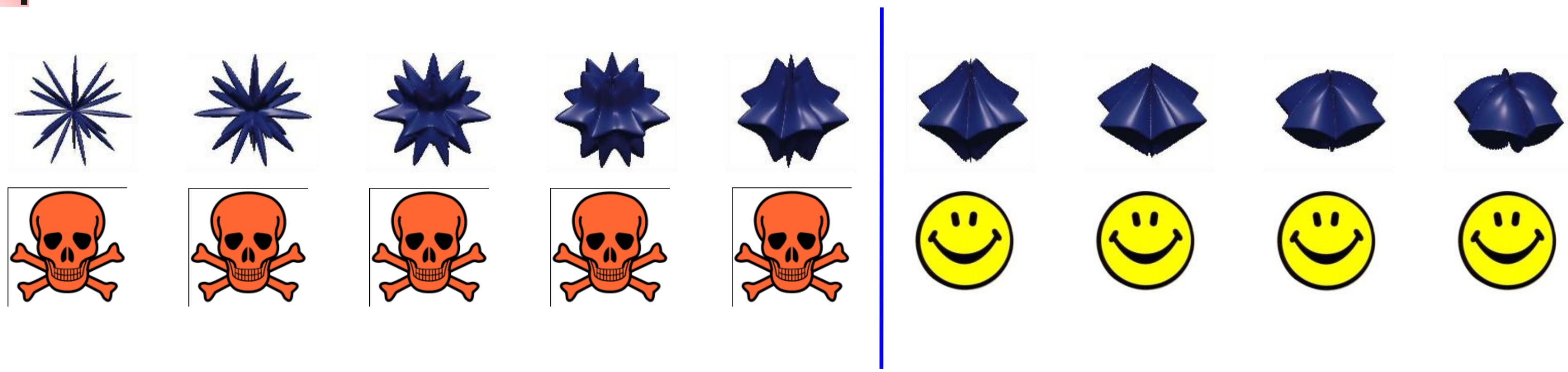
# Poison vs. Yummy Fruits

- *problem*: there's a range of spiky-to-round fruit shapes on Mars:

you need to learn the "threshold" of roundness where the fruits go from poisonous to safe.

and… you need to determine this risking as **few colonists' lives** as possible!

# Testing Fruit Safety…



this is just a **binary search**, so…

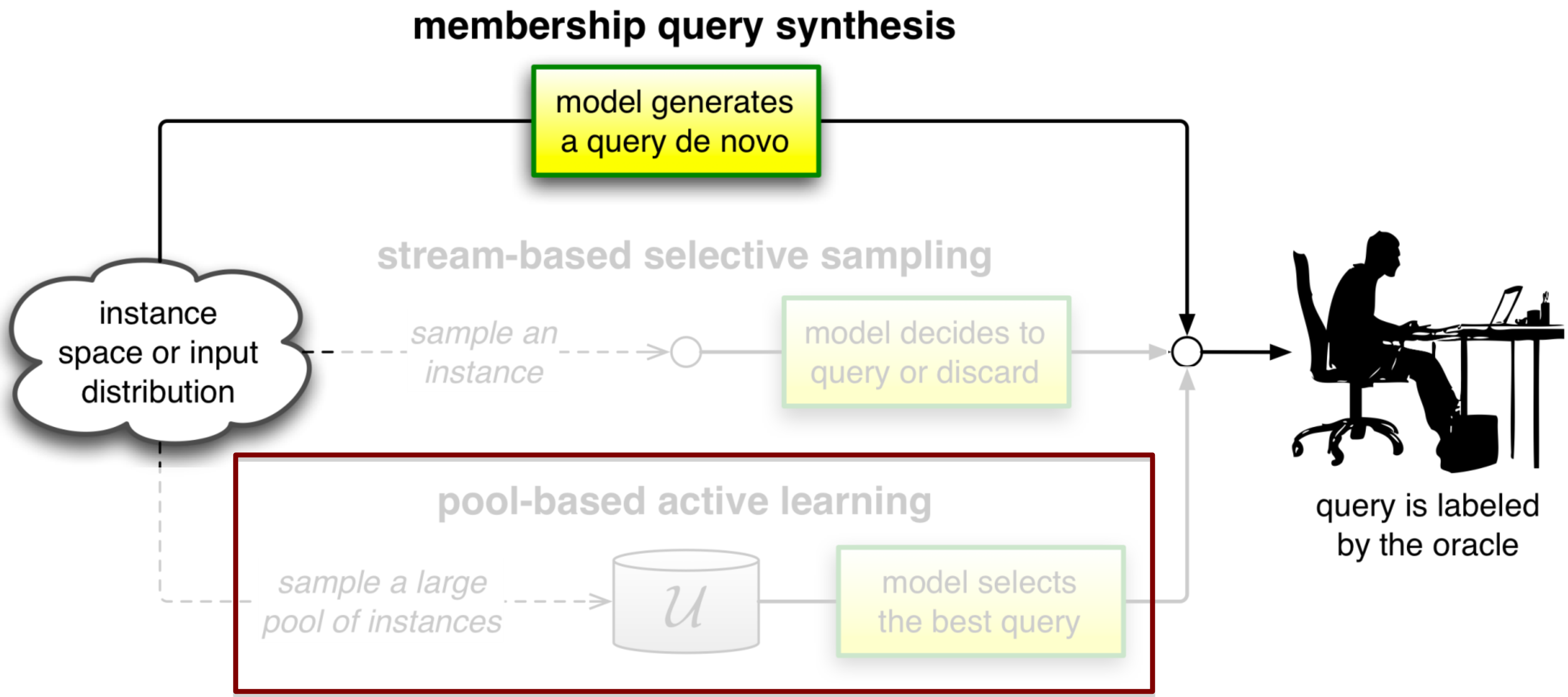
under the PAC model, assume we need $O(1/\varepsilon)$ i.i.d. instances to train a classifier with error $\varepsilon$.

using the binary search approach, we only needed $O(\log_2 1/\varepsilon)$ instances!

# Relationship to Active Learning

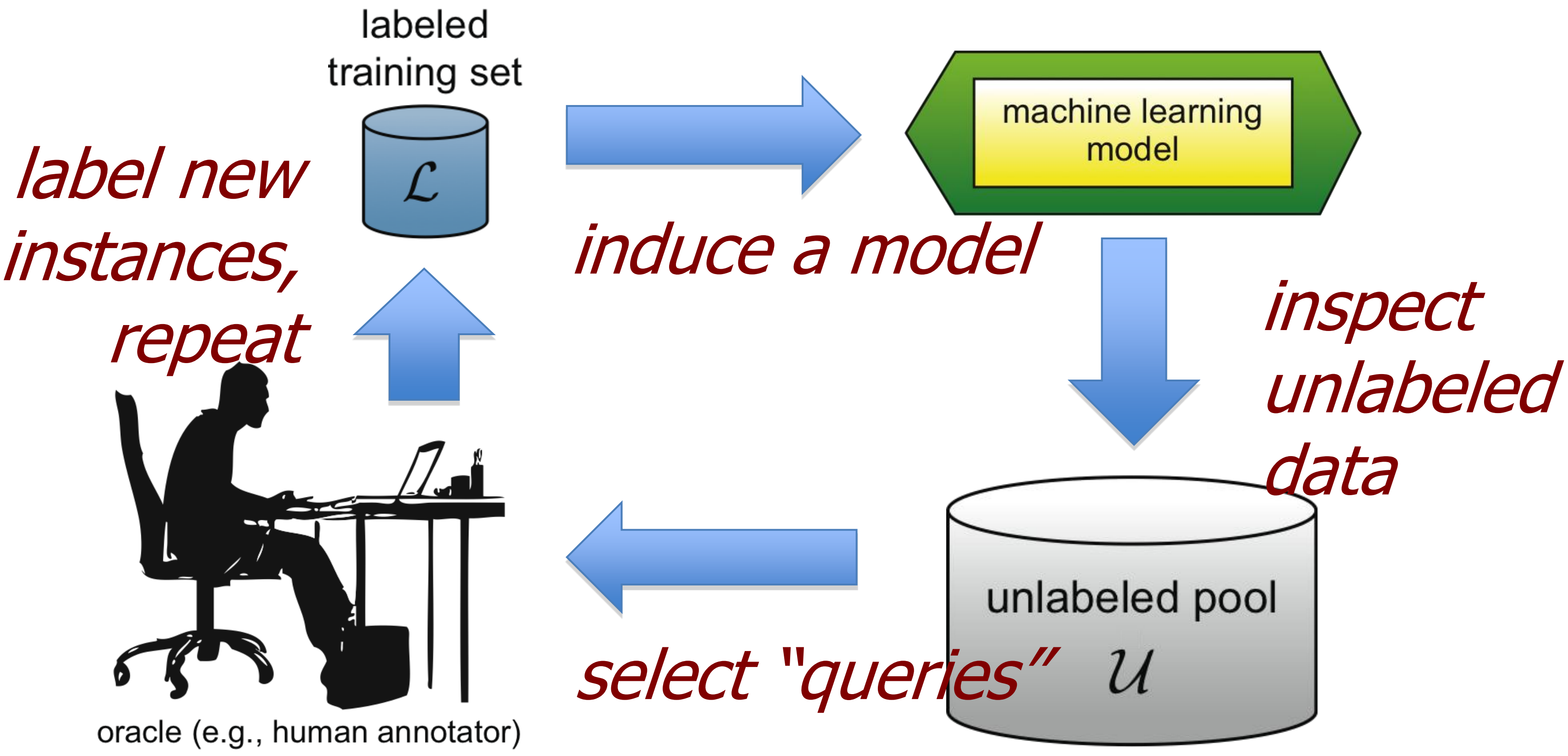
- **key idea:** the learner can choose training data
  - on Mars: whether a fruit was poisonous/safe
  - *in general*: the true label of some instance
- **goal:** reduce the training costs
  - on Mars: the number of "lives at risk"
  - *in general*: the number of "queries"

# Active Learning Scenarios

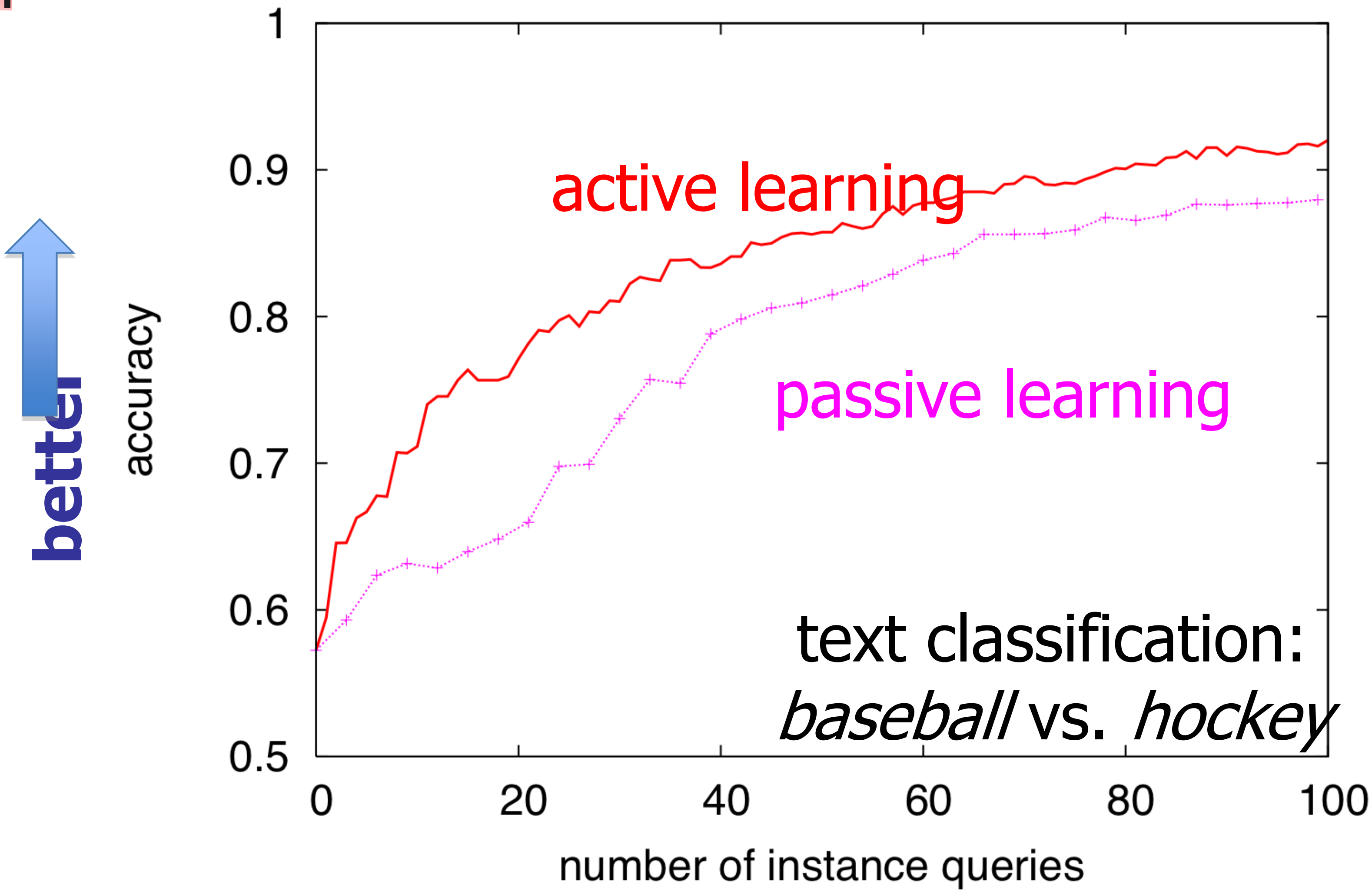



most common in NLP applications

# Pool-Based Active Learning Cycle



# Learning Curves



# Who Uses Active Learning?

Sentiment analysis for blogs; Noisy relabeling
– *Prem Melville*

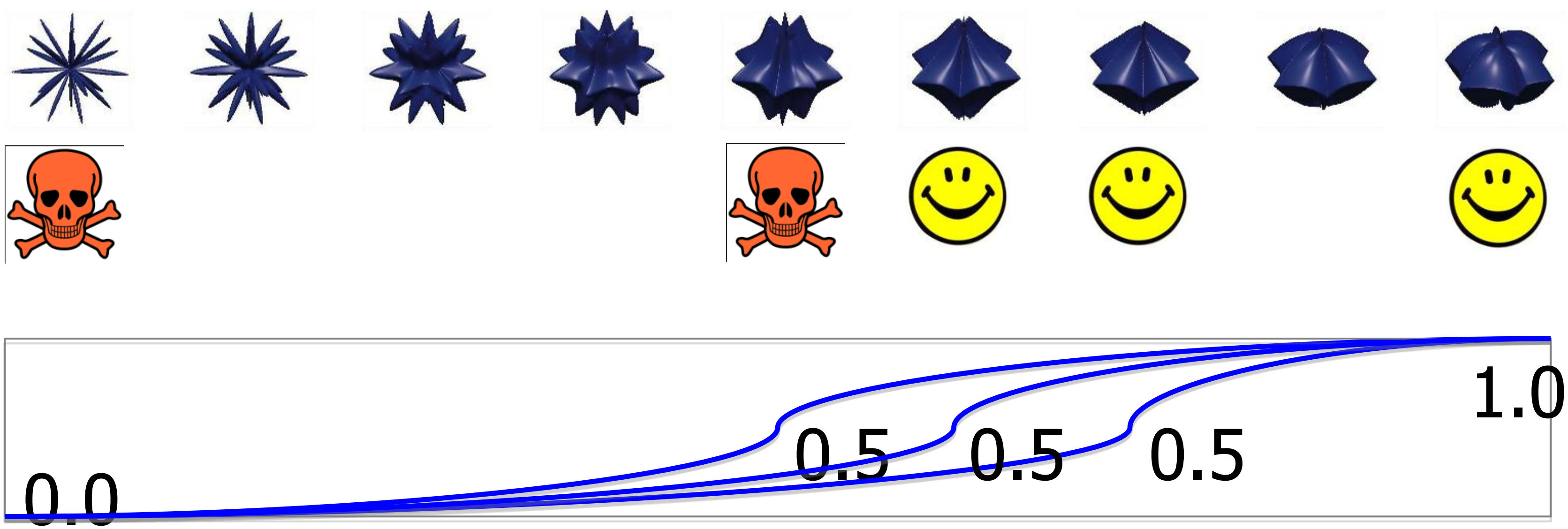
Biomedical NLP & IR; Computer-aided diagnosis
– *Balaji Krishnapuram*

MS Outlook voicemail plug-in [Kapoor et al., IJCAI'07]; "A variety of prototypes that are in use throughout the company." – *Eric Horvitz*

"While I can confirm that we're using active learning in earnest on many problem areas… I really can't provide any more details than that.

# How to Select Queries?

- let's try generalizing our binary search method using a *probabilistic* classifier:




$P(Y = 😊 | X)$

# Uncertainty Sampling

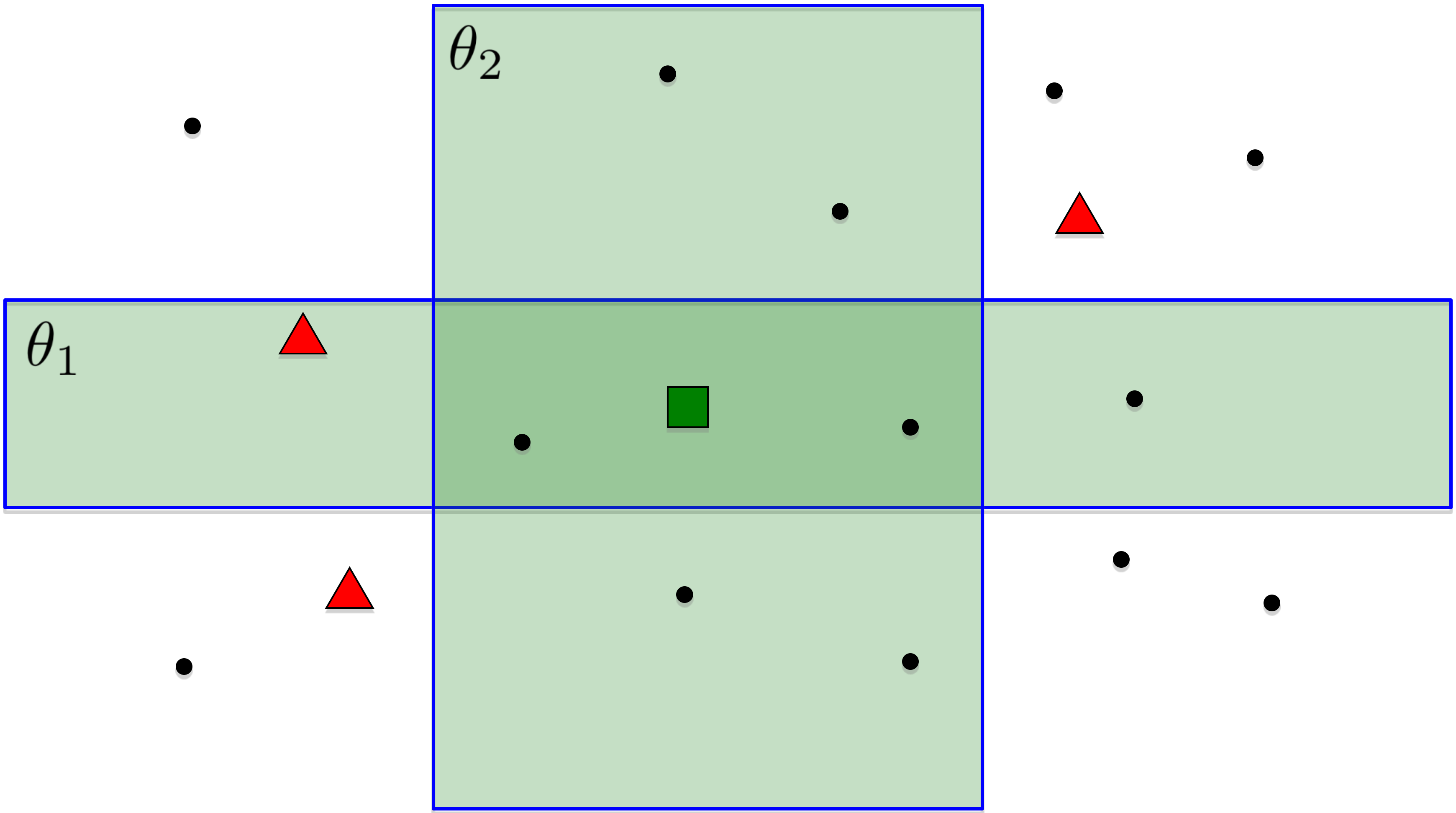
- Query examples learner is most uncertain about
  - Closest to 0.5 prob
  - Closest to decision surface

[Seung et al., COLT'92]

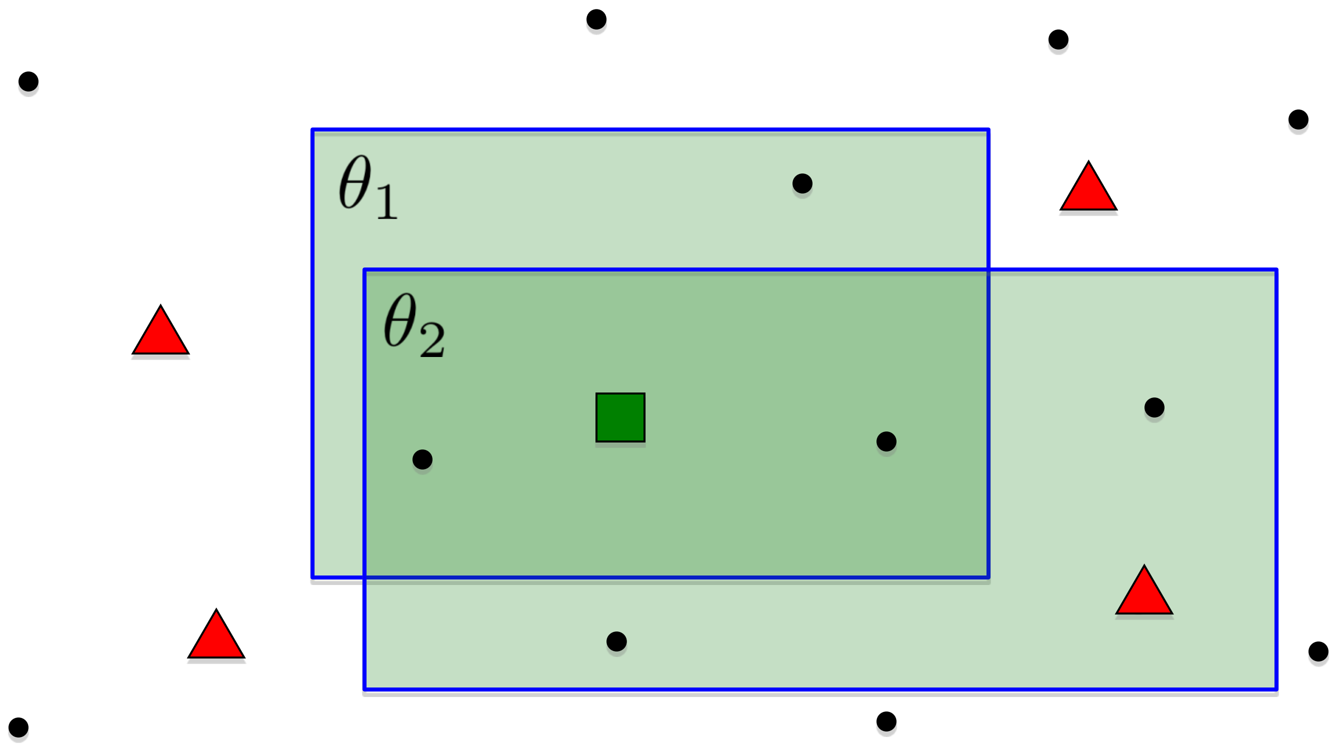
# Query-By-Committee (QBC)

- train a committee $C = \{\theta_1, \theta_2, ..., \theta_C\}$ of classifiers on the labeled data in L
- query instances in U for which the committee is in most *disagreement*
- **key idea:** reduce the model *version space*
  - expedites search for a model during training

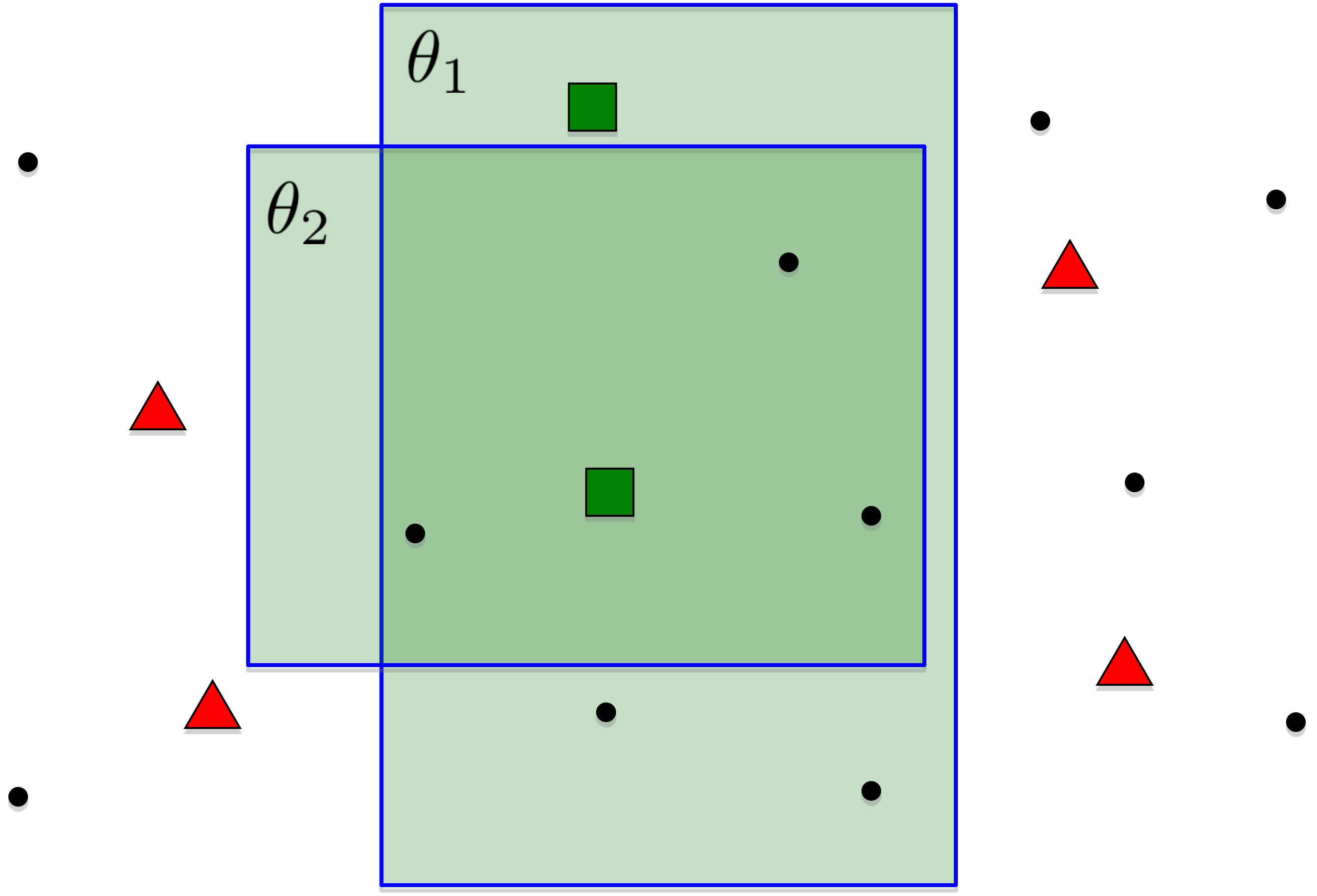
# QBC Example



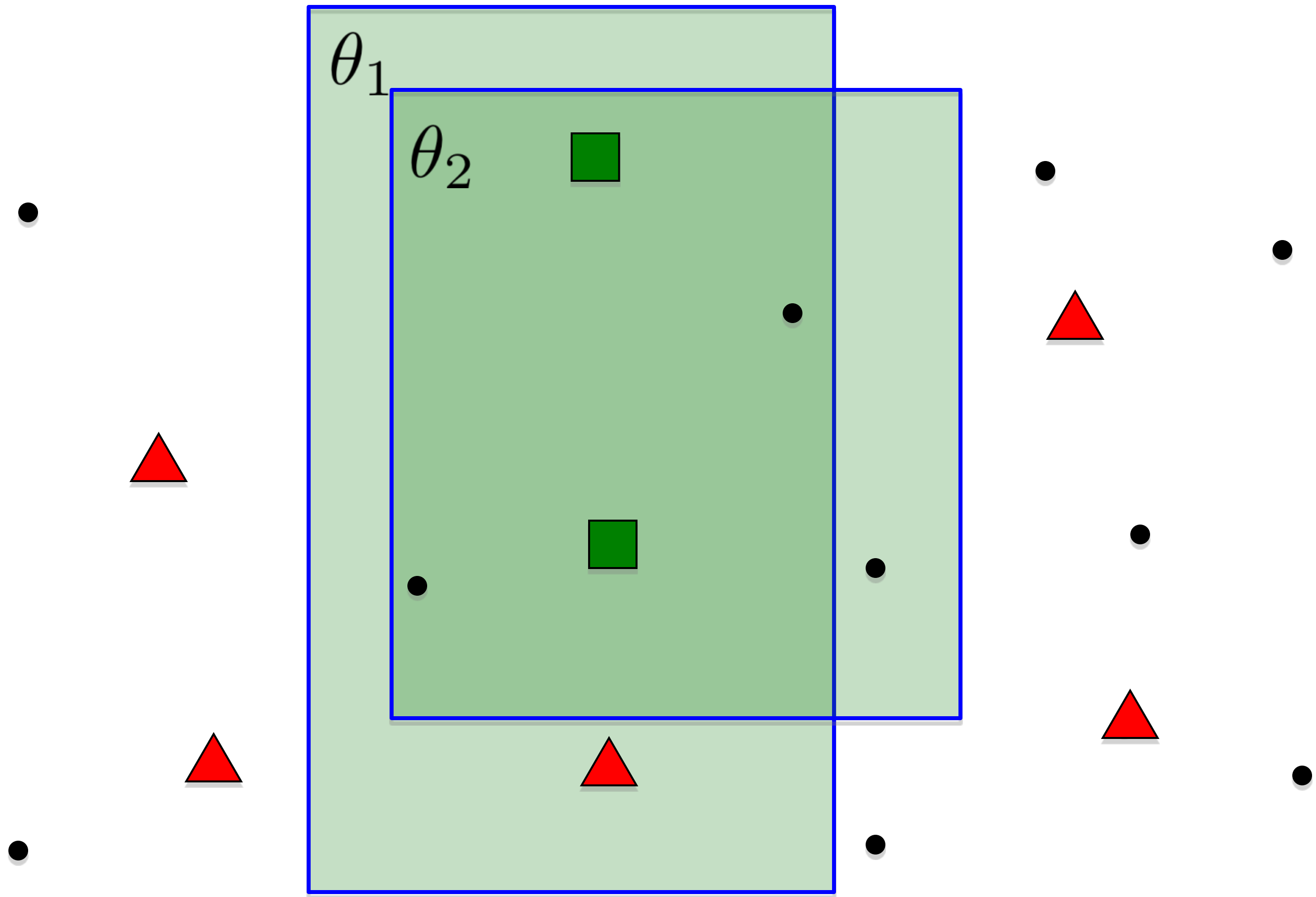
# QBC Example



# QBC Example



# QBC Example



# QBC: Design Decisions

- How to build a committee:
  - "sample" models from $P(\theta|L)$ [Dagan & Engelson, ICML'95; McCallum & Nigam, ICML'98]
  - standard ensembles (e.g., bagging, boosting) [Abe & Mamitsuka, ICML'98]
- How to measure disagreement:
  - "XOR" committee classifications
  - view vote distribution as probabilities, use uncertainty measures (e.g., entropy)

# Alternative Query Types

- so far, we assumed queries are *instances*
  - e.g., for document classification the learner queries *documents*

- can the learner do better by asking different types of questions?
  - *multiple-instance* active learning
  - *feature* active learning

# Feature Active Learning

- in NLP tasks, we can often intuitively label *features*
  - the feature word "*puck*" indicates the class **hockey**
  - the feature word "*strike*" indicates the class **baseball**

- **tandem learning** exploits this by asking both instance-label and feature-relevance queries [Raghavan et al., JMLR'06]
  - e.g., "is *puck* an important discriminative feature?"

# Next Class

- Clustering

# Summary

- Learning theory:
  - Several ways to analyze a problem’s complexity
  - Bounds on generalization error
- SVMs:
  - Maximum the margin
  - Kernel trick
- Unlabeled data:
  - Semi-supervised learning
  - Active learning

# Questions?