## Outline

- Logistics
- Review
- Machine Learning
  - Induction of Decision Trees (7.2)
  - Version Spaces & Candidate Elimination
  - PAC Learning Theory (7.1)
  - Ensembles of classifiers (8.1)

## Logistics

- Learning Problem Set
- Project Grading
  - Wrappers
  - Project Scope x Execution
  - Writeup

## Course Topics by Week

- Search & Constraint Satisfaction
- Knowledge Representation 1: Propositional Logic
- Autonomous Spacecraft 1: Configuration Mgmt
- Autonomous Spacecraft 2: Reactive Planning
- Information Integration 1: Knowledge Representation
- Information Integration 2: Planning
- Information Integration 3: Execution; Learning 1
- Supervised Learning of Decision Trees ⇦
- PAC Learning; Reinforcement Learning
- Bayes Nets: Inference & Learning; Review

## Learning: Mature Technology

- Many Applications
  - Detect fraudulent credit card transactions
  - Information filtering systems that learn user preferences
  - Autonomous vehicles that drive public highways (ALVINN)
  - Decision trees for diagnosing heart attacks
  - Speech synthesis (correct pronunciation) (NETtalk)
- Datamining: huge datasets, scaling issues

## Defining a Learning Problem

A program is said to **learn** from experience ***E*** with respect to task ***T*** and performance measure ***P***, if it's performance at tasks in T, as measured by P, improves with experience E.

- Experience:
- Task:
- Performance Measure:
- Target Function:
- Representation of Target Function Approximation
- Learning Algorithm

## Choosing the Training Experience

- Credit assignment problem:
  - **Direct** training examples:
    - E.g. individual checker boards + correct move for each
  - **Indirect** training examples :
    - E.g. complete sequence of moves and final result
- Which examples:
  - Random, teacher chooses, learner chooses

Supervised learning ⇐
Reinforcement learning
Unsupervised learning

## Choosing the Target Function

- What type of knowledge will be learned?
- How will the knowledge be used by the performance program?
- E.g. checkers program
  - Assume it knows legal moves
  - Needs to choose best move
  - So learn function: F: Boards -> Moves
    - hard to learn
  - Alternative: F: Boards -> R

## The Ideal Evaluation Function

- V(b) = **100** if b is a final, won board
- V(b) = **-100** if b is a final, lost board
- V(b) = **0** if b is a final, drawn board
- Otherwise, if b is not final

  V(b) = **V(s)** where s is best, reachable final board

Nonoperational…
Want operational approximation of V: $\hat{V}$

## Choosing Repr. of Target Function

- x1 = number of black pieces on the board
- x2 = number of red pieces on the board
- x3 = number of black kings on the board
- x4 = number of red kings on the board
- x5 = number of black pieces threatened by red
- x6 = number of red pieces threatened by black

$$\hat{V}(b) = a + bx1 + cx2 + dx3 + ex4 + fx5 + gx6$$

***Now just need to learn 7 numbers!***

## Example: Checkers

- Task T:
  - ***Playing checkers***
- Performance Measure P:
  - ***Percent of games won against opponents***
- Experience E:
  - ***Playing practice games against itself***
- Target Function
  - ***V: board -> R***
- Target Function representation

$$\hat{V}(b) = a + bx1 + cx2 + dx3 + ex4 + fx5 + gx6$$

## Target Function

- Profound Formulation:

  ***Can express any type of inductive learning as approximating a function***
- E.g., Checkers
  - V: boards -> evaluation
- E.g., Handwriting recognition
  - V: image -> word
- E.g., Mushrooms
  - V: mushroom-attributes -> {E, P}

## Representation

- **Decision Trees**
  - Equivalent to propositional DNF
- **Decision Lists**
  - Order of rules matters
- **Datalog Programs**
- **Version Spaces**
  - More general representation (inefficient)
- **Neural Networks**
  - Arbitrary nonlinear numerical functions
- Many More...

## AI = Representation + Search

- Representation
  - How to encode target function
- Search
  - How to construct (find) target function

**Learning = search through the space of possible functional approximations**

## Concept Learning

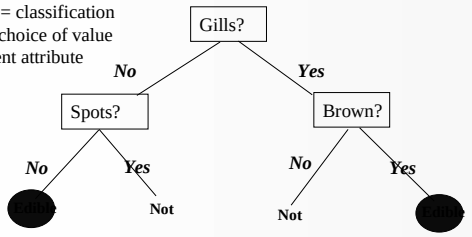
- E.g. Learn concept "Edible mushroom"
  - Target Function has two values: T or F
- Represent concepts as decision trees
- Use ***hill climbing search***
- Thru space of ***decision trees***
  - Start with simple concept
  - Refine it into a complex concept as needed

## Outline

- Logistics
- Review
- Machine Learning
  - Induction of Decision Trees (7.2)
  - Version Spaces & Candidate Elimination
  - PAC Learning Theory (7.1)
  - Ensembles of classifiers (8.1)

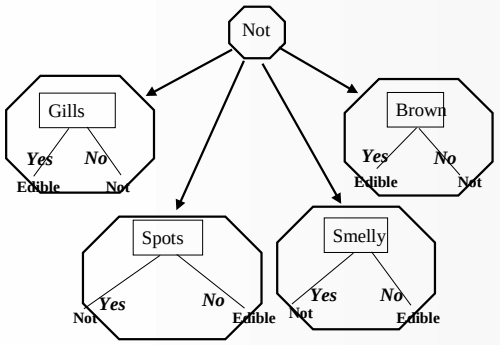
## Decision Tree Representation of ***Edible***

Leaves = classification
Arcs = choice of value for parent attribute




Decision tree is equivalent to logic in disjunctive normal form
Edible $\Leftrightarrow$ $(\neg Gills \wedge \neg Spots) \vee (Gills \wedge Brown)$

## Space of Decision Trees




## Example: "Good day for tennis"

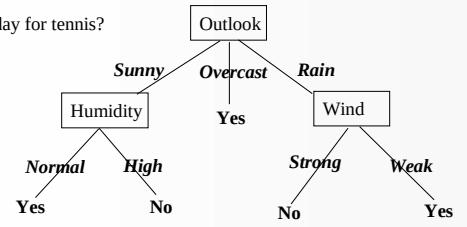
- Attributes of instances
  - Wind
  - Temperature
  - Humidity
  - Outlook
- Feature = attribute with one value
  - E.g. outlook = ***sunny***
- Sample instance
  - wind=***weak***, temp=***hot***, humidity=***high***, outlook=***sunny***

## Experience: "Good day for tennis"

| Day | Outlook | Temp | Humid | Wind | PlayTennis? |
|---|---|---|---|---|---|
| d1 | s | h | h | w | n |
| d2 | s | h | h | s | n |
| d3 | o | h | h | w | y |
| d4 | r | m | h | w | y |
| d5 | r | c | n | w | y |
| d6 | r | c | n | s | y |
| d7 | o | c | n | s | y |
| d8 | s | m | h | w | n |
| d9 | s | c | n | w | y |
| d10 | r | m | n | w | y |
| d11 | s | m | n | s | y |
| d12 | o | m | h | s | y |
| d13 | o | h | n | w | y |
| d14 | r | m | h | s | n |

## Decision Tree Representation

Good day for tennis?




A decision tree
is equivalent to logic in
disjunctive normal form

## DT Learning as Search

- Nodes
  **Decision Trees**
- Operators
  **Tree Refinement: Sprouting the tree**
- Initial node
  **Smallest tree possible: a single leaf**
- Heuristic?
  **Information Gain**
- Goal?
  **Best tree possible (???)**

## Simplest Tree

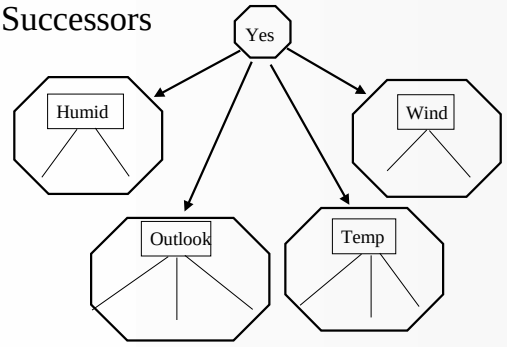
| Day | Outlook | Temp | Humid | Wind | Play? |
|---|---|---|---|---|---|
| d1 | s | h | h | w | n |
| d2 | s | h | h | s | n |
| d3 | o | h | h | w | y |
| d4 | r | m | h | w | y |
| d5 | r | c | n | w | y |
| d6 | r | c | n | s | y |
| d7 | o | c | n | s | y |
| d8 | s | m | h | w | n |
| d9 | s | c | n | w | y |
| d10 | r | m | n | w | y |
| d11 | s | m | n | s | y |
| d12 | o | m | h | s | y |
| d13 | o | h | n | w | y |
| d14 | r | m | h | s | n |

yes

How good?

[10+, 4-] ⇐ Means:
correct on 10 examples
incorrect on 4 examples

## Successors




Which attribute should we use to split?

## To be decided:

- How to choose best attribute?
  - Information gain
  - Entropy (disorder)
- When to stop growing tree?

## Intuition: Information Gain

– Suppose N is between 1 and 20

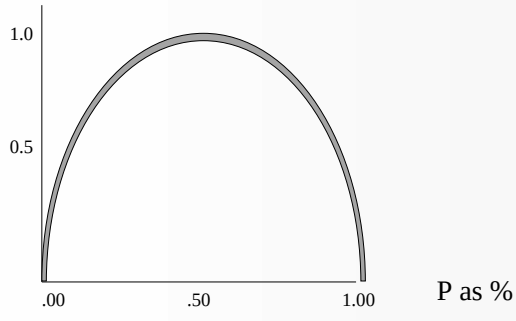
- How many binary questions to determine N?
- What is information gain of being ***told*** N?
- What is information gain of being told N is prime?
  – [7+, 13-]
- What is information gain of being told N is odd?
  – [10+, 10-]
- Which is better first question?

## Entropy (disorder) is bad
## Homogeneity is good

- Let S be a set of examples
- Entropy(S) = -P $\log_2$(P) - N $\log_2$(N)
  – where **P** is proportion of pos example
  – and **N** is proportion of neg examples
  – and 0 log 0 == 0
- Example: S has 9 pos and 5 neg

  Entropy([9+, 5-]) = -(9/14) log2(9/14) - (5/14)log2(5/14)
  = 0.940

## Entropy



## Information Gain

- Measure of expected ***reduction*** in entropy
- Resulting from splitting along an attribute

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} (|S_v| / |S|)\, Entropy(S_v)$$

Where $Entropy(S) = -P \log_2(P) - N \log_2(N)$

## Gain of Splitting on Wind

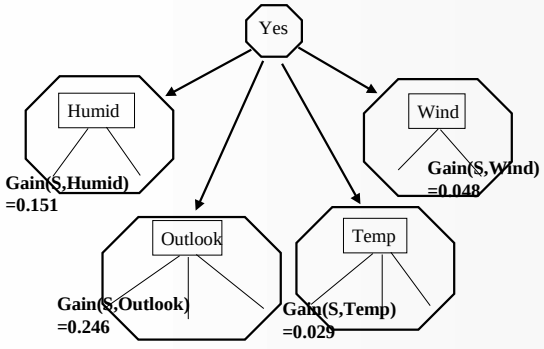
Values(wind)=weak, strong
S = [9+, 5-]
$S_{weak}$ = [6+, 2-]
$S_s$ = [3+, 3-]

Gain(S, wind)
= Entropy(S) - $\sum_{v \in \{weak, s\}} (|S_v| / |S|)\, Entropy(S_v)$

= Entropy(S) - 8/14 Entropy($S_{weak}$)
- 6/14 Entropy($S_s$)
= 0.940 - (8/14) 0.811 - (6/14) 1.00
= 0.048

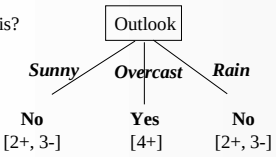
| Day | Wind | Tennis? |
|---|---|---|
| d1 | weak | n |
| d2 | **s** | n |
| d3 | weak | yes |
| d4 | weak | yes |
| d5 | weak | yes |
| d6 | **s** | yes |
| d7 | **s** | yes |
| d8 | weak | n |
| d9 | weak | yes |
| d10 | weak | yes |
| d11 | **s** | yes |
| d12 | **s** | yes |
| d13 | weak | yes |
| d14 | **s** | n |

## Evaluating Attributes

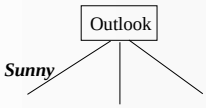


## Resulting Tree ….

Good day for tennis?

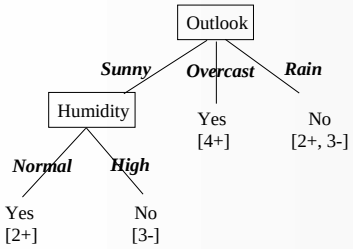



## Recurse!




| Day | Temp | Humid | Wind | Tennis? |
|---|---|---|---|---|
| d1 | h | h | weak | n |
| d2 | h | h | s | n |
| d8 | m | h | weak | n |
| d9 | c | n | weak | yes |
| d11 | m | n | s | yes |

## One Step Later…



## Overfitting…

- DT is **overfit** when exists another DT' and
  - DT has ***smaller*** error on training examples, but
  - DT has ***bigger*** error on test examples
- Causes of overfitting
  - Noisy data, or
  - Training set is too small
- Approaches
  - Stop before perfect tree, or
  - Postpruning

## Summary: Learning = Search

- Target function = concept "edible mushroom"
  - Represent function as decision tree
  - Equivalent to propositional logic in DNF
- Construct approx. to target function via search
  - **Nodes:** decision trees
  - **Arcs:** elaborate a DT (making bigger + better)
  - **Initial State**: simplest possible DT (I.e. a leaf)
  - **Heuristic**: Information gain
  - **Goal**: No improvement possible ...
  - **Search Method**: ***hill climbing***

## Hill Climbing is Incomplete

- Won't necessarily find the best decision tree
  - Local minima
  - Plateau effect
- So…
  - Could search completely…
  - Higher cost…
  - Possibly worth it for data mining
  - Technical problems with over fitting

## Outline

- Logistics
- Review
- Machine Learning
  - Induction of Decision Trees (7.2)
  - Version Spaces & Candidate Elimination
  - PAC Learning Theory (7.1)
  - Ensembles of classifiers (8.1)

## Version Spaces

- Also does concept learning
- Also implemented as search
- Different ***representation*** for the target function
  - No disjunction
- Complete search method
  - Candidate Elimination Algorithm

## Restricted Hypothesis Representation

- Suppose instances have k attributes
- Represent a hypothesis with k ***constraints***

  **?** Means any value is ok

  ∅ Means no value is ok

  A single required **value** is the only acceptable one
- For example

  <?, warm, normal, ?, ?>

  Is consistent with the following examples

| Ex | Sky | AirTemp | Humidity | Wind | Water | Enjoy? |
|---|---|---|---|---|---|---|
| 1 | sunny | warm | normal | strong | cool | yes |
| 2 | cloudy | warm | high | strong | cool | no |
| 3 | sunny | cold | normal | strong | cool | no |
| 4 | cloudy | warm | normal | light | warm | yes |

## Consistency

**Def:** Hypothesis H is ***consistent*** with a set of training examples D iff H(x) = c(x) for each example <x, c(x)> in D

**Def:** The ***version space*** with respect to hypothesis space H and training examples D is the subset of H which is consistent with D

- List-then-enumerate algorithm
  - Let version space := list of all hypotheses in H
  - For each training example <x, c(x)>
    - remove any inconsistent hypothesis from version space
  - Output any hypothesis in the version space

**Stupid….**

**But what if one could represent version space implicitly??**

## General to Specific Ordering

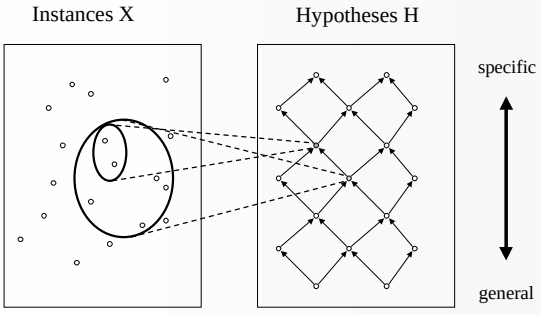
- H1 = <Sunny, ?, ?, Strong, ?, ?>
- H2 = <Sunny, ?, ?, ?, ?, ?>
- H2 is ***more general than*** H1

**Def**: let $H_j$ and $H_k$ be boolean-valued functions defined over X. *(Hj(instance)=1 means instance satisfies hypothesis)*
Then $H_j$ is ***more general than or equal to*** $H_k$ iff

$$\forall x \in X \; [(H_k(x)=1) \Rightarrow (H_j(x)=1)]$$

## Correspondence

A hypothesis = set of instances



## Version Space: Compact Representation

- Defn the ***general boundary*** G with respect to hypothesis space H and training data D is the set of **maximally general** members of H consistent with D
- Defn the ***specific boundary*** S with respect to hypothesis space H and training data D is the set of minimally general (**maximally specific**) members of H consistent with D

## Boundary Sets

S: {<Sunny, Warm, ?, Strong, ?, ?>}

No Need to represent contents of version space — just represent the boundaries

G: {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>}

## Candidate Elimination Algorithm

Initialize G to set of maximally general hypotheses
Initialize S to set of maximally specific hypotheses
For each training example d, do:
  If d is a **positive** example:
    Remove from G any hyp inconsistent with d
    For each hyp in S that is not consistent with d
      Remove s from S
      Add to S all minimal generalizations h of s
        such that consistent(h, d) and
        $\exists$ g$\in$G and g is more general than h
      $\forall$ s$\in$ S, Remove s if s more general than t$\in$ S
  If d is a **negative** example...

## Initialization

$S_0$ {<∅, ∅, ∅, ∅, ∅, ∅>}

$G_0$ {<?, ?, ?, ?, ?, ?>}

## Training Example 1

**<Sunny, Warm, Normal, Strong, Warm, Same> Good4Tennis=Yes**

$S_0$ {<∅, ∅, ∅, ∅, ∅, ∅>}

↓

$S_1$ {<Sunny, Warm, Normal, Strong, Warm, Same>}

$G_1$, $G_0$ {<?, ?, ?, ?, ?, ?>}

## Training Example 2

**<Sunny, Warm, *High*, Strong, Warm, Same> Good4Tennis=Yes**

$S_1$ {<Sunny, Warm, Normal, Strong, Warm, Same>}

↓

$S_2$ {<Sunny, Warm, **?**, Strong, Warm, Same>}

$G_2$, $G_1$ {<?, ?, ?, ?, ?, ?>}

## Training Example 3

**<Rainy, Cold, High, Strong, Warm, Change> Good4Tennis=No**

$S_3$ $S_2$ {<Sunny, Warm, ?, Strong, Warm, Same>}

$G_3$ {<Sunny,?,?,?,?,?>, <?,Warm,?,?,?,?>, <?,?,?,?,?,Same>}

$G_2$ {<?, ?, ?, ?, ?, ?>}

## A ***Biased*** Hypothesis Space

| Ex | Sky | AirTemp | Humidity | Wind | Water | Enjoy? |
|---|---|---|---|---|---|---|
| 1 | sunny | warm | normal | strong | cool | yes |
| 2 | cloudy | warm | normal | strong | cool | yes |
| 3 | rainy | warm | normal | strong | cool | no |

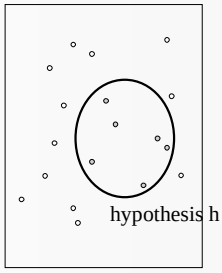
- Candidate elimination algorithm can't learn this concept
- Version space will collapse
- Hypothesis space is biased
  - Not expressive enough to represent disjunctions

## Comparison

- Decision Tree learner searches a ***complete*** hypothesis space (one capable of representing any possible concept), but it uses an incomplete search method (hill climbing)
- Candidate Elimination searches an ***incomplete*** hypothesis space (one capable of representing only a subset of the possible concepts), but it does so completely.

Note: DT learner works better in practice

## An Unbiased Learner




- Hypothesis space =
  - power set of instance space
- For enjoy-sport: |X| = 324
  - 3.147 x 10^70
- Size of version space: 2305
- Might expect: increased size => harder to learn
  - In this case it makes it impossible!
- Some inductive bias is essential

## Two kinds of bias

- Restricted hypothesis space bias
  - shrink the size of the hypothesis space
- Preference bias
  - ordering over hypotheses

## Outline

- Logistics
- Review
- Machine Learning
  - Induction of Decision Trees (7.2)
  - Version Spaces & Candidate Elimination
  - PAC Learning Theory (7.1)
    - Bias
  - Ensembles of classifiers (8.1)

## Formal model of learning

- Suppose examples drawn from X according to some probability distribution: Pr(X)
- Let f be a hypothesis in H
- Let C be the actual concept

$$\text{Error}(f) = \sum_{x \in D} \Pr(x)$$

Where D = set of all examples where f and C disagree

Def: f is ***approximately correct*** (with accuracy e) iff Error(f) ≤ e

## PAC Learning

- A learning program is program is probably approximately correct (with probability d and accuracy e) if given any set of training examples drawn from the distribution Pr, the program outputs a hypothesis f such that
- Pr(Error(f)>e) < d

- Key points:
  - Double hedge
  - Same distribution for training & testing

## Example of a PAC learner

- Candidate elimination
  - Algo returns f which is consistent with examples
- Suppose H is finite
- PAC if number of training examples is
  > ln(d/|H|) / ln(1-e)
- Distribution free learning

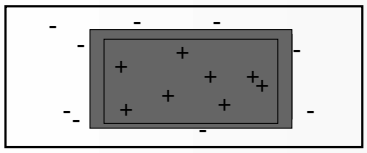
## Sample complexity

- As a function of 1/d and 1/e
- How fast does ln(d /|H|) / ln(1-e) grow?

| d | e | \|H\| | n |
|---|---|---|---|
| .1 | .9 | 100 | 70 |
| .1 | .9 | 1000 | 90 |
| .1 | .9 | 10000 | 110 |
| .01 | .99 | 100 | 700 |
| .01 | .99 | 1000 | 900 |

## Infinite Hypothesis Spaces

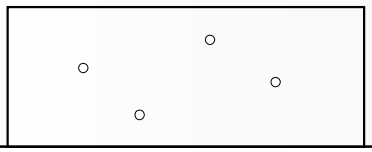
- Sample complexity = ln(d /|H|) / ln(1-e)
- Assumes |H| is finite
- Consider
  - Hypothesis represented as a **rectangle**



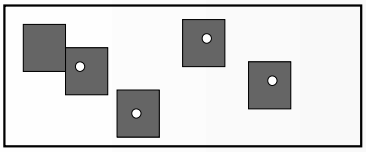

|H| is infinite, but ***expressiveness*** is not! ∴ bias!

## Vapnik-Chervonenkis Dimension

- A set of instances S is ***shattered*** by hypothesis space H iff ∀dichotomy of S ∃ some hypothesis in H consistent with the dichotomy
- VC(H) is the size of the largest finite subset of examples shattered by H
- VC(rectangles) = 4

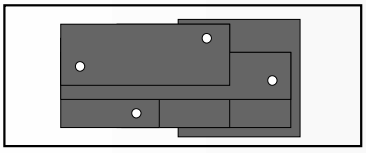


## Dichotomies of size 0 and 1



**Space of Instances X**

## Dichotomies of size 2

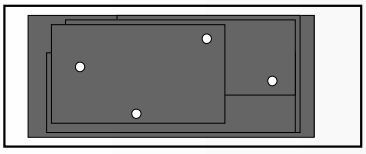


**Space of Instances X**

## Dichotomies of size 3 and 4

**So VD(rectangles) ≥ 4**
**Exercise: there is no set of size 5 which is shattered**

**Sample complexity:** $m = \frac{1}{e}(4\log_2\frac{2}{d}) + 8\mathrm{VC}(H)\log_2(\frac{13}{e})$



**Space of Instances X**

## Outline

- Logistics
- Review
- Machine Learning
  - Induction of Decision Trees (7.2)
  - Version Spaces & Candidate Elimination
  - PAC Learning Theory (7.1)
  - Ensembles of classifiers (8.1)

## Ensembles of Classifiers

- Idea: instead of training one classifier (dec. tree)
- Train *k* classifiers and let them vote
  - Only helps if classifiers disagree with each other
  - Trained on different data
  - Use different learning methods
- Amazing fact: can help a lot!

## How voting helps

- Assume errors are independent
- Assume majority vote
- Prob. majority is wrong = area under biomial dist

- If individual area is 0.3
- **Area under curve for ≥11 wrong is 0.026**
- Order of magnitude improvement!

## Constructing Ensembles

- Bagging
  - Run classifier *k* times on m examples drawn randomly with replacement from the original set of m examples
  - Training sets correspond to 63.2% of original (+ duplicates)
- Cross-validated committees
  - Divide examples into *k* disjoint sets
  - Train on *k* sets corresponding to original *minus* 1/k th
- Boosting
  - Maintain a probability distribution over set of training ex
  - On each iteration, use distribution to sample
  - Use error rate to modify distribution
    - Create harder and harder learning problems...

## Review: Learning

- *Learning as **Search***
  - Search in the space of hypotheses
  - Hill climbing in space of decision trees
  - Complete search in conjunctive hypothesis representation
- *Notion of **Bias***
  - Restricted set of hypotheses
  - Small H means can jump to conclusion
- ***Tradeoff**: Expressiveness / Tractability*
  - Big H => harder to learn
  - *PAC Definition*
- Ensembles of classifiers:
  - Bagging, Boosting, Cross validated committees