Lecture 32 of 42

Machine Learning: Basic Concepts
Discussion: Inductive Bias, Decision Trees

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course page: http://snipurl.com/v9v3
Course web site: http://www.kddresearch.org/Courses/CIS730
Instructor home page: http://www.cis.ksu.edu/~bhsu

Reading for Next Class:

Section 18.1 – 18.2, Russell and Norvig

Lecture Outline

- Learning Algorithms and Models
  - Models: decision trees, winnow, artificial neural networks, naïve Bayes, genetic algorithms (GAs) and genetic programming (GP), instance-based learning (nearest-neighbor), inductive logic programming (ILP)
  - Algorithms: for decision trees (ID3/C4.5/J48), ANNs (backprop), etc.
  - Methodologies: supervised, unsupervised, reinforcement; knowledge-guided
- Theory of Learning
  - Computational learning theory (COLT): complexity, limitations of learning
  - Probably Approximately Correct (PAC) learning
  - Probabilistic, statistical, information theoretic results
- Multistrategy Learning: Combining Techniques, Knowledge Sources
- Data: Time Series, Very Large Databases (VLDB), Text Corpora
- Applications
  - Performance element: classification, decision support, planning, control
  - Database mining and knowledge discovery in databases (KDD)
  - Computer inference: learning to reason

Why Machine Learning?

- **New Computational Capability**
  - Database mining: converting records into knowledge
  - Self-customizing programs: learning news filters, adaptive monitors
  - Learning to act: robot planning, control optimization, decision support
  - Applications that are hard to program: automated driving, speech recognition
- **Better Understanding of Human Learning and Teaching**
  - Cognitive science: theories of knowledge acquisition (e.g., through practice)
  - Performance elements: reasoning (inference) and *recommender* systems
- **Time is Right**
  - Recent progress in algorithms and theory
  - Rapidly growing volume of online data from various sources
  - Available computational power
  - Growth, interest in learning-based industries (e.g., data mining/KDD)

Rule and Decision Tree Learning

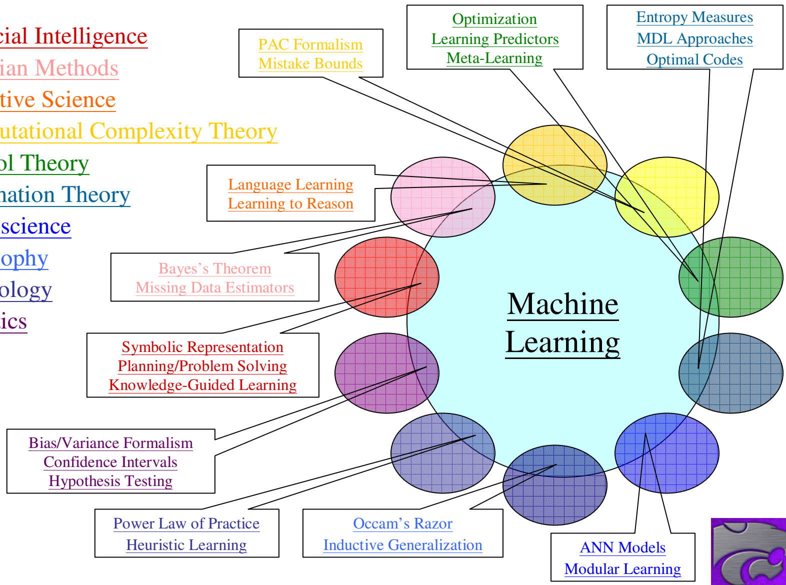
- **Example: Rule Acquisition from Historical Data**
- **Data**
  - Patient 103 (time = 1): Age 23, First-Pregnancy: no, Anemia: no, Diabetes: no, Previous-Premature-Birth: no, Ultrasound: *unknown*, Elective C-Section: *unknown*, Emergency-C-Section: *unknown*
  - Patient 103 (time = 2): Age 23, First-Pregnancy: no, Anemia: no, Diabetes: yes, Previous-Premature-Birth: no, Ultrasound: *unknown*, Elective C-Section: no, Emergency-C-Section: *unknown*
  - Patient 103 (time = n): Age 23, First-Pregnancy: no, Anemia: no, Diabetes: yes, Previous-Premature-Birth: no, Ultrasound: abnormal, Elective C-Section: no, Emergency-C-Section: YES
- **Learned Rule**
  - IF *no previous vaginal delivery*, AND *abnormal 2nd trimester ultrasound*, AND *malpresentation at admission*, AND *no elective C-Section*
    THEN *probability of emergency C-Section is 0.6*
  - Training set: 26/41 = 0.634
  - Test set: 12/20 = 0.600

Neural Network Learning

- Autonomous Learning Vehicle In a Neural Net (ALVINN): Pomerleau *et al.*
  - NAVLAB: http://www.cs.cmu.edu/afs/cs/project/alv/www/index.html
  - Drives 70mph on highways

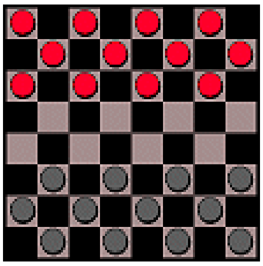
Relevant Disciplines

- Artificial Intelligence
- Bayesian Methods
- Cognitive Science
- Computational Complexity Theory
- Control Theory
- Information Theory
- Neuroscience
- Philosophy
- Psychology
- Statistics



Specifying A Learning Problem

- Learning = Improving with Experience at Some Task
  - Improve over task *T*,
  - with respect to performance measure *P*,
  - based on experience *E*.
- Example: Learning to Play Checkers
  - *T*: play games of checkers
  - *P*: percent of games won in tournament play
  - *E*: opportunity to play against self
- Refining the Problem Specification: Issues
  - What experience?
  - What *exactly* should be learned?
  - How shall it be *represented*?
  - What specific algorithm to learn it?
- Defining the Problem Milieu
  - Performance element: How shall results of learning be applied?
  - How shall performance element be evaluated? Learning system?



Example: Learning to Play Checkers

- Type of Training Experience
  - Direct or indirect?
  - Teacher or not?
  - Knowledge about the game (e.g., openings/endgames)?
- Problem: Is Training Experience *Representative* (of Performance Goal)?
- Software Design
  - Assumptions of the learning system: *legal* move generator exists
  - Software requirements: generator, evaluator(s), parametric target function
- Choosing a Target Function
  - *ChooseMove: Board* $\rightarrow$ *Move* – action selection function, or *policy*
  - *V: Board* $\rightarrow$ *R* – evaluation function for game tree search (minimax / α–β)
  - Ideal target *V*; approximated target $\hat{V}$
  - Goal of learning process: operational description (approximation) of *V*
- *Chinook*: Checkers Solved by Game Tree Search (July 2007)
- Reference: http://en.wikipedia.org/wiki/Checkers

A Target Function for Learning to Play Checkers

- Possible Definition
  - If b is final board state that is won, then $V(b)$ = +100 (or MAXINT)
  - If b is final board state that is lost, then $V(b)$ = -100 (or MAXINT)
  - If b is final board state that is drawn, then $V(b) = 0$
  - If b is not final board state in the game, then $V(b) = V(b')$ where b' is best final board state that can be achieved starting from b and playing optimally until end
  - *Correct values, but not operational*
- Choosing a Representation for the Target Function
  - Collection of rules?
  - Neural network?
  - Polynomial function (e.g., linear, quadratic combination) of board features?
  - Other?
- A Representation for Learned Function
  - $$\hat{V}(b) = w_0 + w_1 bp(b) + w_2 rp(b) + w_3 bk(b) + w_4 rk(b) + w_5 bt(b) + w_6 rt(b)$$
  - bp/rp = number of black/red pieces; bk/rk = number of black/red kings
  - bt/rt = number of black/red pieces threatened (can be taken on next turn)

A Training Procedure for Learning to Play Checkers

- Obtaining Training Examples
  - $V(b)$ target function
  - $\hat{V}(b)$ learned function
  - $V_{train}(b)$ training value ("signal")
- Rule For Estimating Training Values

$$V_{train}(b) \leftarrow \hat{V}(Successor(b))$$

- Rule for Training (Weight Tuning)
  - Least Mean Square (LMS) weight update rule
  - REPEAT
    - Select training example b at random
    - Compute the $error(b)$ for this training example

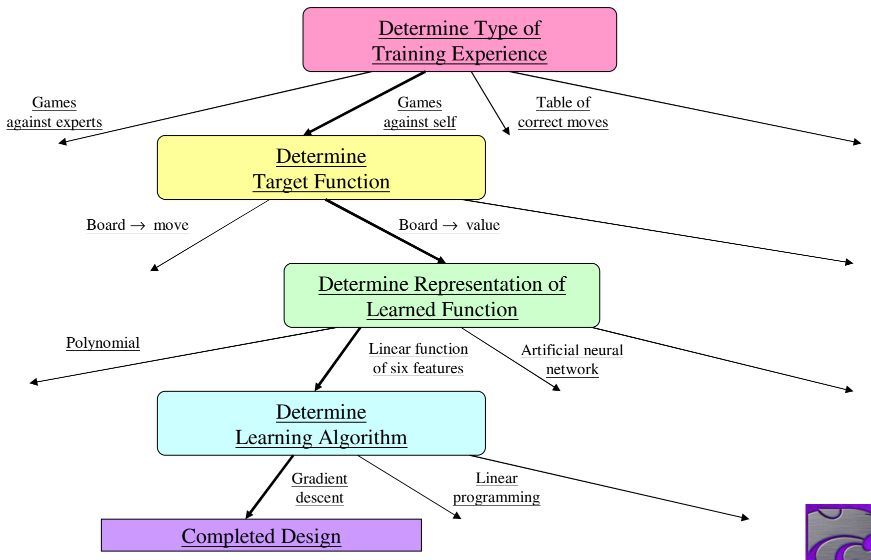
$$error(b) = V_{train}(b) - \hat{V}(b)$$

    - For each board feature f_i, update weight w_i as follows

$$w_i \leftarrow w_i + c \cdot f_i \cdot error(b)$$

where c is small, constant factor to adjust learning rate

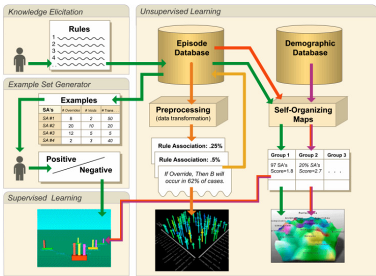
Design Choices for Learning to Play Checkers



Some Issues in Machine Learning

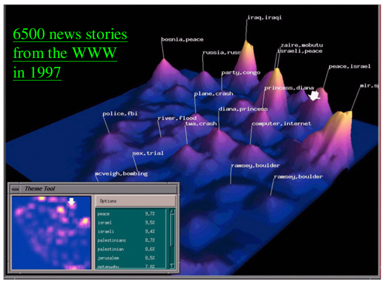
- What Algorithms Can Approximate Functions Well? When?
- How Do Learning System Design Factors Influence Accuracy?
  - Number of training examples
  - Complexity of *hypothesis representation*
- How Do Learning Problem Characteristics Influence Accuracy?
  - Noisy data
  - Multiple data sources
- What Are Theoretical Limits of Learnability?
- How Can Prior Knowledge of Learner Help?
- What Clues Can We Get From Biological Learning Systems?
- How Can Systems Alter Their Own Representation?

Interesting Applications



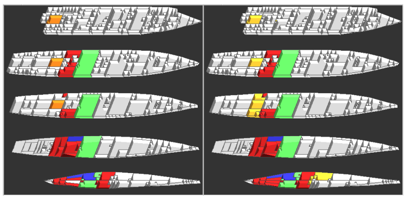
NCSA *D2K* - http://alg.ncsa.uiuc.edu

Database Mining



Clustering (Cartia *ThemeScapes*) - http://snurl.com/1y5l1

Reasoning (Inference, Decision Support)



DC-ARM - http://www.stanford.edu/~dwilkins/members.htm

Normal	Destroyed
Ignited	Extinguished
Engulfed	Fire Alarm
	Flooding

Planning, Control

What to Learn?

- **Classification Functions**
  - **Learning hidden functions: estimating ("fitting") parameters**
  - **Concept learning (e.g., chair, face, game)**
  - **Diagnosis, prognosis: risk assessment, medical monitoring, security, ERP**
- **Models**
  - **Map (for navigation)**
  - **Distribution (query answering, *aka* QA)**
  - **Language model (e.g., automaton/grammar)**
- **Skills**
  - **Playing games**
  - **Planning**
  - **Reasoning (acquiring representation to use in reasoning)**
- **Cluster Definitions for Pattern Recognition**
  - **Shapes of objects**
  - **Functional or taxonomic definition**
- ***Many Problems Can Be Reduced to Classification***

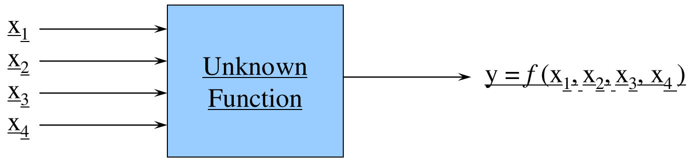
How to Learn It?

- Supervised
  - What is learned? Classification function; other models
  - Inputs and outputs? Learning: examples $\langle x, f(x) \rangle \rightarrow$ approximation $\hat{f}(x)$
  - How is it learned? Presentation of examples to learner (by teacher)
- Unsupervised
  - Cluster definition, or *vector quantization* function (*codebook*)
  - Learning: observations $x \times$ distance metric $d(x_1, x_2) \rightarrow$ discrete codebook $f(x)$
  - Formation, segmentation, labeling of clusters based on observations, metric
- Reinforcement
  - Control policy (function from states of the world to actions)
  - Learning: state/reward sequence $\{\langle s_i, r_i \rangle : 1 \le i \le n\} \rightarrow$ policy $p : s \rightarrow a$
  - (Delayed) feedback of reward values to agent based on actions
  - Model updated based on reward, (partially) observable state

Supervised Inductive Learning: Classification and Regression

- Given: Training Examples <*x*, *f(x)*> of Some Unknown Function *f*
- Find: A Good Approximation to *f*
- Examples (besides Concept Learning)
  - Disease diagnosis
    - *x* = properties of patient (medical history, symptoms, lab tests)
    - *f* = disease (or recommended therapy)
  - Risk assessment
    - *x* = properties of consumer, policyholder (demographics, accident history)
    - *f* = risk level (expected cost)
  - Automatic steering
    - *x* = bitmap picture of road surface in front of vehicle
    - *f* = degrees to turn the steering wheel
  - Part-of-speech tagging
  - Computer security: fraud/intrusion detection, attack graphs
  - Information extraction: clusters of documents
  - Social networks and weblogs: predicting links, sentiment analysis
  - Multisensor integration and prediction

Learning and Types [1]: A Generic Supervised Learning Problem



Example	x_1	x_2	x_3	x_4	y
0	0	1	1	0	0
1	0	0	0	0	0
2	0	0	1	1	1
3	1	0	0	1	1
4	0	1	1	0	0
5	1	1	0	0	0
6	0	1	0	1	0

- Input x_i: t_i, desired output y: t, "target" function f: $(t_1 \times t_2 \times t_3 \times t_4) \rightarrow t$
- Learning function: Vector $(t_1 \times t_2 \times t_3 \times t_4 \times t) \rightarrow (t_1 \times t_2 \times t_3 \times t_4) \rightarrow t$

Learning and Types [2]: Unrestricted Hypothesis Space

- $| A \rightarrow B | = | B |^{| A |}$ – proof?
- $|H^4 \rightarrow H| = | \{0,1\} \times \{0,1\} \times \{0,1\} \times \{0,1\} \rightarrow \{0,1\} | = 2^{2^4} = 65536$ functions
- Complete Ignorance: Is Learning Possible?
  - Need to see every possible input/output pair
  - After 7 examples, still have $2^9 = 512$ possibilities (out of 65536) for f

Example	x_1	x_2	x_3	x_4	y
0	0	0	0	0	?
1	0	0	0	1	?
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	?
8	1	0	0	0	?
9	1	0	0	1	1
10	1	0	1	0	?
11	1	0	1	1	?
12	1	1	0	0	0
13	1	1	0	1	?
14	1	1	1	0	?
15	1	1	1	1	?

Training Examples for Concept *EnjoySport*

- Specification for Examples
  - Similar to a data type definition
  - 6 attributes: Sky, Temp, Humidity, Wind, Water, Forecast
  - Nominal-valued (symbolic) attributes - enumerative data type
- Binary (Boolean-Valued or H -Valued) Concept
- Supervised Learning Problem: *Describe the General Concept*

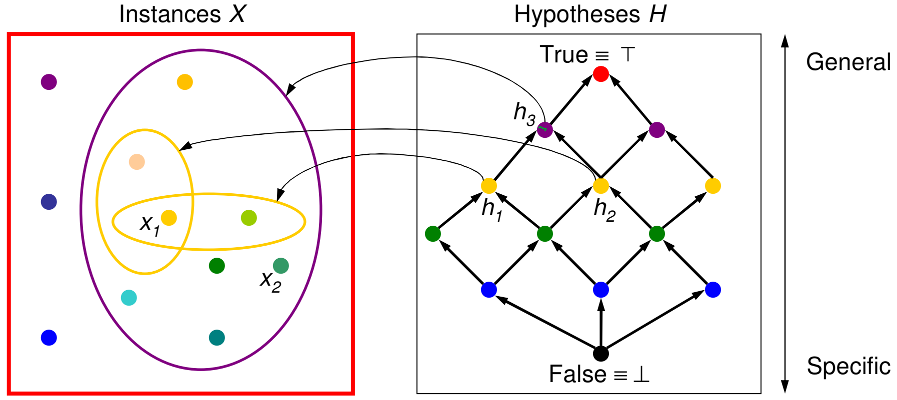
Example	Sky	Air Temp	Humidity	Wind	Water	Forecast	Enjoy Sport
0	Sunny	Warm	Normal	Strong	Warm	Same	**Yes**
1	Sunny	Warm	High	Strong	Warm	Same	**Yes**
2	Rainy	Cold	High	Strong	Warm	Change	**No**
3	Sunny	Warm	High	Strong	Cool	Change	**Yes**

Representing Hypotheses

- Many Possible Representations
- Hypothesis *h*: Conjunction of Constraints on Attributes
- Constraint Values
  - Specific value (e.g., *Water = Warm*)
  - Don’t care (e.g., *“Water = ?”*)
  - No value allowed (e.g., “Water = Ø”)
- Example Hypothesis for *EnjoySport*
  - Sky AirTemp Humidity Wind Water Forecast

    <Sunny ? ? Strong ? Same>
  - Is this consistent with the training examples?
  - What are some hypotheses that are consistent with the examples?

Instances, Hypotheses, and the Partial Ordering *Less-General-Than*



x_1 = *<Sunny, Warm, High, Strong, Cool, Same>*
x_2 = *<Sunny, Warm, High, Light, Warm, Same>*

h_1 = *<Sunny, ?, ?, Strong, ?, ?>*
h_2 = *<Sunny, ?, ?, ?, Cool, ?>*
h_3 = *<Sunny, ?, ?, ?, ?, ?>*

$\leq_P$ ≡ *Less-General-Than*
(corresponding set of instances: *Subset-Of*)

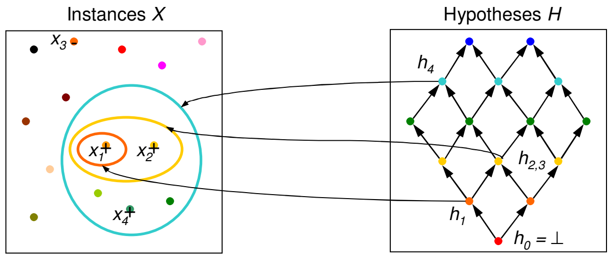
$h_1 \leq_P h_3$
$h_2 \leq_P h_3$

Find-S Algorithm

1. Initialize h to the most specific hypothesis in *H*
   - *H*: the *hypothesis space*
   - (partially ordered set under relation *Less-Specific-Than)*
2. For each positive training instance *x*
   - For each attribute constraint a_i in *h*
     - IF constraint a_i in *h* is satisfied by *x*
     - THEN do nothing
     - ELSE replace a_i in *h* by next more general constraint satisfied by *x*
3. Output hypothesis *h*

CIS 530 / 730
Artificial Intelligence
Lecture 32 of 42
Computing & Information Sciences
Kansas State University

Hypothesis Space Search by *Find-S*



x_1 = <Sunny, Warm, Normal, Strong, Warm, Same>, +
x_2 = <Sunny, Warm, High, Strong, Warm, Same>, +
x_3 = <Rainy, Cold, High, Strong, Warm, Change>, -
x_4 = <Sunny, Warm, High, Strong, Cool, Change>, +

h_0 = <Ø, Ø, Ø, Ø, Ø, Ø>
h_1 = <Sunny, Warm, Normal, Strong, Warm, Same>
h_2 = <Sunny, Warm, ?, Strong, Warm, Same>
h_3 = <Sunny, Warm, ?, Strong, Warm, Same>
h_4 = <Sunny, Warm, ?, Strong, ?, ?>

- Shortcomings of *Find-S*
  - Can't tell whether it has learned concept
  - Can't tell when training data inconsistent
  - Picks a maximally specific *h* (why?)
  - Depending on *H*, there might be several!

Version Spaces

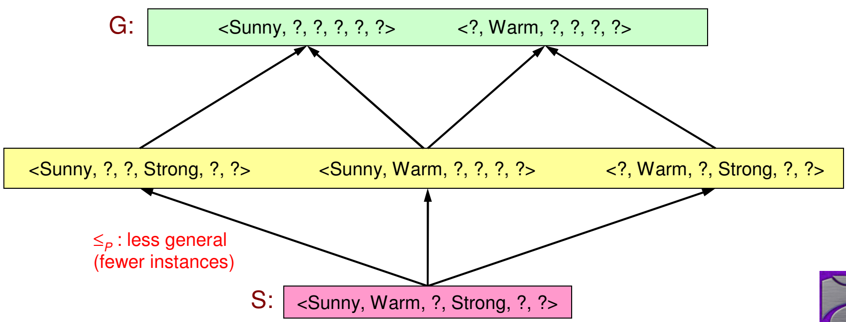
- Definition: Consistent Hypotheses
  - A hypothesis *h* is consistent with a set of training examples *D* of target concept *c* if and only if $h(x) = c(x)$ for each training example $<x, c(x)>$ in *D*.
  - $Consistent(h, D) \equiv \forall <x, c(x)> \in D \,.\, h(x) = c(x)$
- Given
  - Hypothesis space *H*
  - Data set *D*: set of training examples
- Definition
  - Version space $VS_{H,D}$ with respect to *H*, *D*
  - Subset of hypotheses from *H* consistent with all training examples in *D*
  - $VS_{H,D} \equiv \{ h \in H \mid Consistent(h, D) \}$

List-Then-Eliminate Algorithm

1. Initialization: *VersionSpace* ← list containing every hypothesis in *H*
2. For each training example <*x*, *c*(*x*)>

   Remove from *VersionSpace* any hypothesis *h* for which $h(x) \neq c(x)$
3. Output the list of hypotheses in *VersionSpace*

Example Version Space



Hypothesis Spaces As Lattices

- Meet Semilattice
  - Every pair of hypotheses h_i and h_j has *greatest lower bound* (GLB) $h_i \vee h_j$
  - Is *H* meet semilattice?
  - $\perp \equiv$ some ø
- Join Semilattice
  - Every pair of hypotheses h_i and h_j has *least upper bound* (LUB) $h_i \wedge h_j$

  Is *H* join semilattice?
  - $\top \equiv$ all ?
- (Full) Lattice
  - Every pair of hypotheses has GLB $h_i \vee h_j$ and LUB $h_i \wedge h_j$
  - Both meet semilattice and join semilattice
  - Partial ordering Less-General-Than

Representing Version Spaces As Lattices

- Definition: General (Upper) Boundary
  - General boundary G of version space $VS_{H,D}$: set of most general members
  - Most general ≡ *maximal* elements of $VS_{H,D}$ ≡ "set of necessary conditions"
- Definition: Specific (Lower) Boundary
  $\top$
  - Specific boundary S of version space $VS_{H,D}$: set of least general members
  - Most specific ≡ *minimal* elements of $VS_{H,D}$ ≡ "set of sufficient conditions"
- Version Space
  - $VS_{H,D}$ ≡ consistent poset (partially-ordered subset of H)
  - Every member of version space lies between S and G
  - $VS_{H,D} \equiv \{ h \in H \mid \exists\, s \in S,\ g \in G \,.\, s \leq_P h \leq_P g \}$, $\leq_P$ ≡ Less-General-Than

  "Version space is defined as set of hypotheses sandwiched between specific *s* and general *g* (given data)"

Candidate Elimination Algorithm [1]

1. Initialization

   $G_0 \leftarrow \top$ ≡ most general hypothesis in H, denoted {<?, … , ?>}

   $S_0 \leftarrow \bot$ ≡ least general hypotheses in H, denoted {<Ø, … , Ø>}

2. For each training example d

   If d is a positive example (*Update-S*) // generalize

   Remove from G any hypotheses inconsistent with d

   For each hypothesis s in S that is not consistent with d

   Remove s from S // "move S upwards"

   Add to S all minimal generalizations h of s such that

   1. h is consistent with d
   2. Some member of G is more general than h

   (These are least upper bounds, or *joins*, $s \wedge d$, in $VS_{H,D}$)

   Remove from S any hypothesis that is more general than another hypothesis in S (remove any dominating elements)

Candidate Elimination Algorithm [2]

(continued)

If d is a negative example (*Update-G*)   // specialize

Remove from S any hypotheses inconsistent with d

For each hypothesis g in G that is not consistent with d

Remove g from G // "move G downwards"

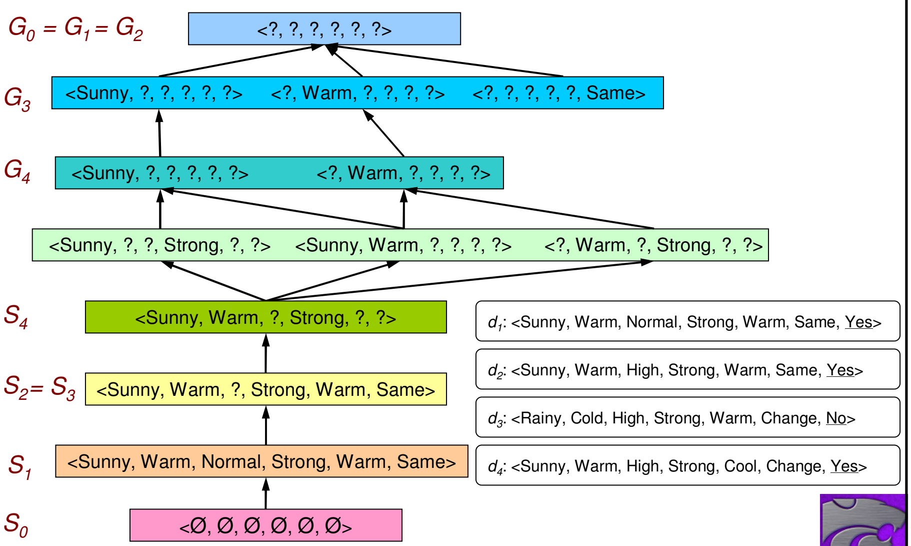
Add to G all minimal specializations h of g such that

1. h is consistent with d
2. Some member of S is less general than h

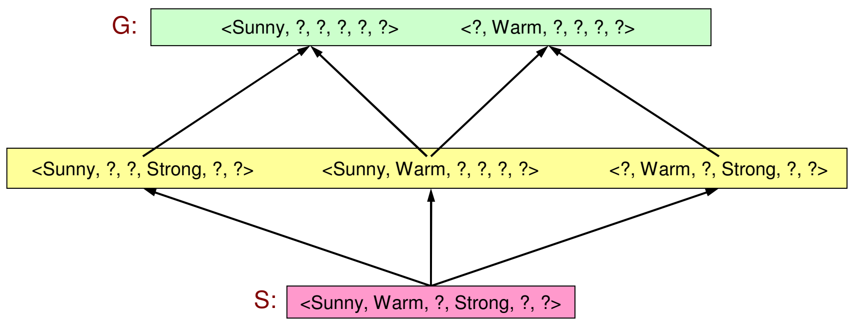
(These are greatest lower bounds, or *meets*, $g \vee d$, in $VS_{H,D}$)

Remove from G any hypothesis that is less general than another hypothesis in G (remove any dominated elements)

Example Trace



What Next Training Example?



- Active Learning: What Query Should The Learner Make Next?
- How Should These Be Classified?
  - <Sunny, Warm, Normal, Strong, Cool, Change>
  - <Rainy, Cold, Normal, Light, Warm, Same>
  - <Sunny, Warm, Normal, Light, Warm, Same>

What Justifies This Inductive Leap?

- Example: Inductive Generalization
  - Positive example: <Sunny, Warm, Normal, Strong, Cool, Change, Yes>
  - Positive example: <Sunny, Warm, Normal, Light, Warm, Same, Yes>
  - Induced *S*: <Sunny, Warm, Normal, ?, ?, ?>
- Why Believe We Can Classify The Unseen?
  - e.g., <Sunny, Warm, Normal, Strong, Warm, Same>
  - When is there enough information (in a new case) to make a prediction?

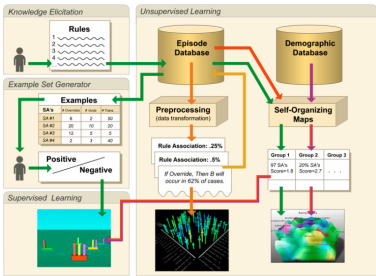
An Unbiased Learner

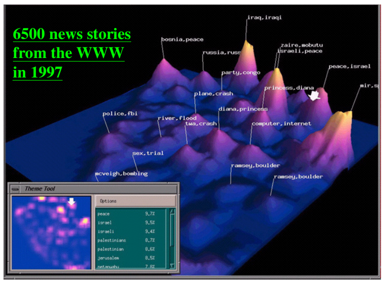
- Inductive Bias
  - Any preference for one hypothesis over another, *besides* consistency
  - Example: $H \equiv$ *conjunctive* concepts with don't cares
  - What concepts can H not express? (Hint: what are its syntactic limitations?)
- Idea
  - Choose unbiased H': expresses every teachable concept (i.e., power set of X)
  - Recall: $| A \rightarrow B | = | B |^{|A|}$ ($A = X$; $B = \{\text{labels}\}$; $H' = A \rightarrow B$)
  - {{Rainy, Sunny, Cloudy} × {Warm, Cold} × {Normal, High} × {None-Mild, Strong} × {Cool, Warm} × {Same, Change}} → {0, 1}
- An Exhaustive Hypothesis Language
  - Consider: H' = disjunctions ($\vee$), conjunctions ($\wedge$), negations ($\neg$) over H
  - $| H' | = 2^{(2 \cdot 2 \cdot 2 \cdot 3 \cdot 2 \cdot 2)} = 2^{96}$; $| H | = 1 + (3 \cdot 3 \cdot 3 \cdot 4 \cdot 3 \cdot 3) = 973$
- What Are S, G For The Hypothesis Language H'?
  - S ← *disjunction of all positive examples*
  - G ← *conjunction of all negated negative examples*

What Justifies This Inductive Leap?

- **Example: Inductive Generalization**
  - **Positive example: <Sunny, Warm, Normal, Strong, Cool, Change, Yes>**
  - **Positive example: <Sunny, Warm, Normal, Light, Warm, Same, Yes>**
  - **Induced *S*: <Sunny, Warm, Normal, ?, ?, ?>**
- **Why Believe We Can Classify The Unseen?**
  - **e.g., <Sunny, Warm, Normal, Strong, Warm, Same>**
  - **When is there enough information (in a new case) to make a prediction?**

Interesting Applications



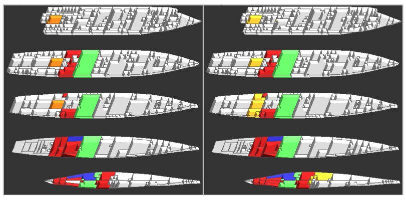


NCSA *D2K* - http://alg.ncsa.uiuc.edu

Cartia *ThemeScapes* - http://www.cartia.com

Database Mining

Reasoning (Inference, Decision Support)



DC-ARM - http://www-kbs.ai.uiuc.edu

Normal
Ignited
Engulfed
Destroyed
Extinguished
Fire Alarm
Flooding

Planning, Control

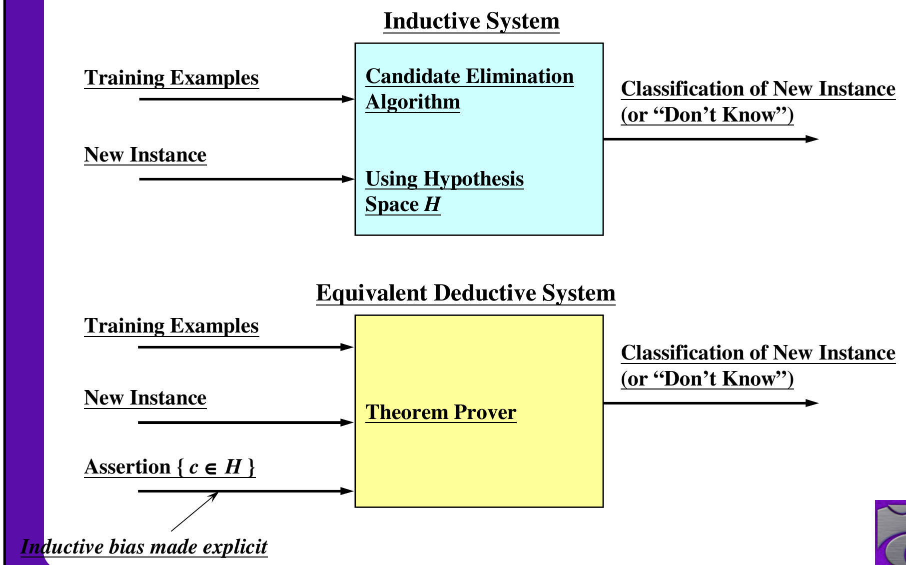
An Unbiased Learner

- **Example of A Biased *H***
  - *Conjunctive* concepts with don't cares
  - What concepts can *H* not express? (Hint: what are its syntactic limitations?)
- **Idea**
  - Choose *H'* that expresses every teachable concept
  - i.e., *H'* is the power set of *X*
  - Recall: $| A \rightarrow B | = | B |^{| A |}$ ($A = X$; B = {labels}; $H' = A \rightarrow B$)
  - {{Rainy, Sunny} × {Warm, Cold} × {Normal, High} × {None, Mild, Strong} × {Cool, Warm} × {Same, Change}} → {0, 1}
- **An Exhaustive Hypothesis Language**
  - Consider: *H'* = disjunctions (∨), conjunctions (∧), negations (¬) over previous *H*
  - $| H' | = 2^{(2 \cdot 2 \cdot 2 \cdot 3 \cdot 2 \cdot 2)} = 2^{96}$; $| H | = 1 + (3 \cdot 3 \cdot 3 \cdot 4 \cdot 3 \cdot 3) = 973$
- **What Are S, G For The Hypothesis Language *H'*?**
  - S ← *disjunction of all positive examples*
  - G ← *conjunction of all negated negative examples*

Inductive Bias

- **Components of An Inductive Bias Definition**
  - Concept learning algorithm L
  - Instances X, target concept c
  - Training examples $D_c = \{<x, c(x)>\}$
  - $L(x_i, D_c)$ = classification assigned to instance x_i by L after training on D_c
- **Definition**
  - The inductive bias of L is any minimal set of assertions B such that, for any target concept c and corresponding training examples D_c,
    $\forall\, x_i \in X \,.\, [(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$ where A $\vdash$ B means A *logically entails* B
  - Informal idea: preference for (i.e., restriction to) certain hypotheses by structural (syntactic) means
- **Rationale**
  - Prior assumptions regarding target concept
  - Basis for inductive generalization

Inductive Systems and Equivalent Deductive Systems



Three Learners with Different Biases

- **Rote Learner**
  - Weakest bias: anything seen before, i.e., no bias
  - Store examples
  - Classify x *if and only if* it matches previously observed example
- **Version Space Candidate Elimination Algorithm**
  - Stronger bias: concepts belonging to conjunctive H
  - Store extremal generalizations and specializations
  - Classify x *if and only if* it "falls within" S and G boundaries (all members agree)
- **Find-S**
  - Even stronger bias: most specific hypothesis
  - Prior assumption: any instance *not observed to be positive* is negative
  - Classify x based on S set

Views of Learning

- **Removal of (Remaining) Uncertainty**
  - Suppose unknown function was *known* to be *m-of-n* Boolean function
  - Could use training data to infer the function
- **Learning and Hypothesis Languages**
  - Possible approach to *guess a good, small hypothesis language*:
    - Start with a very small language
    - Enlarge until it contains a hypothesis that fits the data
  - Inductive bias
    - Preference for certain languages
    - Analogous to data compression (removal of redundancy)
    - Later: coding the "model" versus coding the "uncertainty" (error)
- **We Could Be Wrong!**
  - Prior knowledge could be wrong (e.g., $y = x_4 \wedge$ one-of (x_1, x_3) also consistent)
  - If guessed language was wrong, errors will occur on new cases

Two Strategies for Machine Learning

- Develop Ways to Express Prior Knowledge
  - Role of prior knowledge: guides search for hypotheses / hypothesis languages
  - Expression languages for prior knowledge
    - Rule grammars; stochastic models; etc.
    - Restrictions on computational models; other (formal) specification methods
- Develop Flexible Hypothesis Spaces
  - Structured collections of hypotheses
    - Agglomeration: nested collections (hierarchies)
    - Partitioning: decision trees, lists, rules
    - Neural networks; cases, etc.
  - Hypothesis spaces of adaptive size
- Either Case: Develop Algorithms for Finding A Hypothesis That Fits Well
  - Ideally, will generalize well
- Later: Bias *Optimization* (Meta-Learning, Wrappers)

Computational Learning Theory

- What General Laws Constrain Inductive Learning?
- What Learning Problems Can Be Solved?
- When Can We Trust The Output of A Learning Algorithm?
- We Seek Theory To Relate:
  - Probability of successful learning
  - Number of training examples
  - Complexity of hypothesis space
  - Accuracy to which target concept is approximated
  - Manner in which training examples are presented

Summary

- Reading: Chapters 1-2, Mitchell
- Suggested Exercises: 2.2, 2.3, 2.4, 2.6
- Taxonomy of Learning Systems
- Today: Overview, Learning from Examples
  - (Supervised) concept learning framework
  - Simple approach: assumes no noise; illustrates key concepts
- Today & Next Class: Hypothesis Learning and Inductive Bias
  - Sources: Mitchell (1997) – online notes and handout
  - Wednesday: inductive learning, version space
  - Friday: candidate elimination algorithm, active learning, inductive bias
  - Background concepts: partially-ordered set (poset) formalism
- Next Class: Decision Trees, Intro Computational Learning Theory (COLT)
  - First paper review due Wed 30 Jan 2008
  - Sources: Kearns & Vazirani (1994), Han & Kamber 2nd edition (2006)

Terminology

- Learning: Improving at Task given Performance Measure, Experience
- Performance Element: Part of System that Applies Result of Learning
- Types of Learning
  - Supervised: with "teacher" (often, classification from labeled examples)
  - Unsupervised: from data, using similarity measure (unlabeled instances)
  - Reinforcement: "by doing", with reward/penalty signal
- Supervised Learning: Target Functions
  - Target function – function *c* or *f* to be learned
  - Target – desired value y to be predicted (sometimes "target function")
  - Example / labeled instance – tuples of the form $<x, f(x)>$
  - Classification function, classifier – nominal-valued *f* (enumerated return type)
- Clustering: Application of Unsupervised Learning
- Concepts and Hypotheses
  - Concept – function *c* from observations to TRUE or FALSE (membership)
  - Class label – output of classification function
  - Hypothesis – proposed function *h* believed to be similar to *c* (or *f*)