INF5390 – Kunstig intelligens

Agents That Learn

Roar Fjellheim

Outline

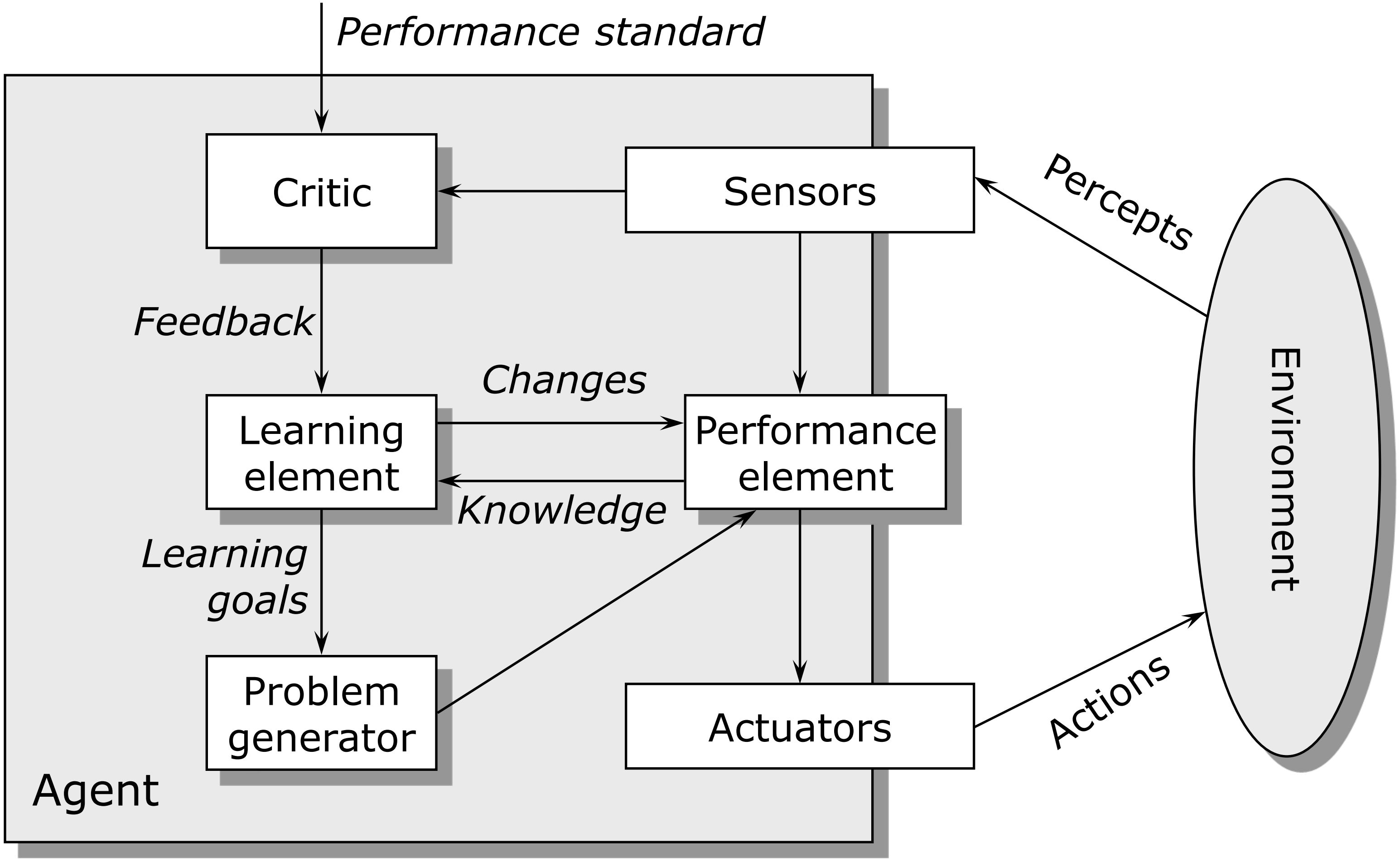
- General model
- Types of learning
- Learning decision trees
- Learning logical descriptions
- Other knowledge-based methods
- Summary

Extracts from
AIMA Chapter 18: Learning From Examples
AIMA Chapter 19: Knowledge in Learning

Why should agents learn?

- Agents in previous lectures have assumed "built-in" knowledge, provided by designers
- In order to handle incomplete knowledge and changing knowledge requirements, agents must *learn*
- Learning is a way of achieving agent *autonomy* and the ability to *improve performance* over time
- The field in AI that deals with learning is called *machine learning*, and is very active

General model of learning agents



Elements of the general model

- Performance element
  - Carries out the task of the agent, i.e. processes percepts and decides on actions
- Learning element
  - Proposes improvements of the performance element, based on previous knowledge and feedback
- Critic
  - Evaluates performance element by comparing results of its actions with imposed performance standards
- Problem generator
  - Proposes exploratory actions to increase knowledge

Aspects of the learning element

- Which *components* of the performance element are to be improved
  - Which parts of the agent's knowledge base is targeted
- What *feedback* is available
  - Supervised, unsupervised or reinforcement learning differ in type of feedback agent receives
- What *representation* is used for the components
  - E.g. logic sentences, belief networks, utility functions, etc.
- What *prior information* (*knowledge*) is available

Performance element components

- Possible components that can be improved
  - Direct mapping from states to actions
  - Means to infer world properties from percept sequences
  - Information about how the world evolves
  - Information about the results of possible actions
  - Utility information about the desirability of world states
  - Desirability of specific actions in specific states
  - Goals describing states that maximize utility
- In each case, learning can be sees as learning an unknown *function* $y = f(x)$

Hypothesis space H

- H: the set of hypothesis functions *h* to be considered in searching for *f(x)*
- *Consistent* hypothesis: Fits with all data
  - If several consistent hypotheses – choose simplest one! (Occam's razor)
- *Realizability* of learning problem:
  - *Realizable* if H contains the ”true” function
  - *Unrealizable* if not
  - We do normally know what the true function is
- Why not choose H as large as possible?
  - May be very inefficient in learning and in applying

Types of learning - Knowledge

- *Inductive* learning
  - √ Given a collection of *examples* (x, $f(x)$)
  - √ Return a function h that approximates f
  - √ Does not rely on prior knowledge ("just data")
- *Deductive* (or analytical) learning
  - √ Going from known general f to a new f' that is logically entailed
  - √ Based on prior knowledge ("data+knowledge")
  - √ Resemble more human learning

Types of learning - Feedback

- *Unsupervised* learning
  - Agent learns patterns in data even though no feedback is given, e.g. via clustering
- *Reinforcement* learning
  - Agent gets reward or punishment at the end, but is not told which particular action led to the result
- *Supervised* learning
  - Agent receives learning examples and is explicitly told what the correct answer is for each case
- Mixed modes, e.g. *semi-supervised* learning
  - Correct answers for some but not all examples

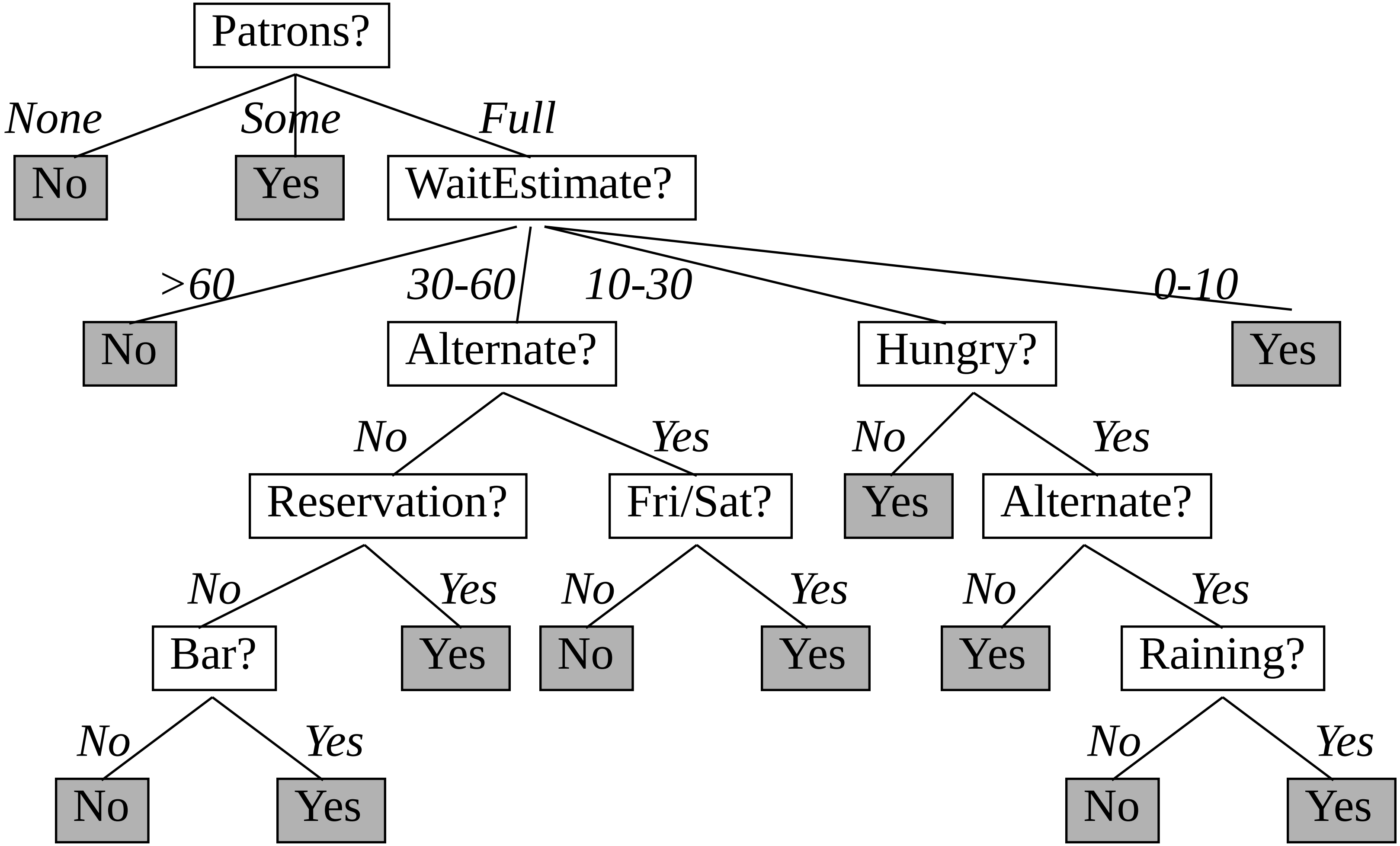
Learning decision trees

- A *decision situation* can be described by
  - A number of *attributes*, each with a set of possible values
  - A *decision* which may be Boolean (yes/no) or multivalued
- A *decision tree* is a tree structure where
  - Each internal node represents a *test* of the value of an attribute, with one branch for each possible attribute value
  - Each leaf node represents the value of the *decision* if that node is reached
- *Decision tree learning* is one of simplest and most successful forms of machine learning
- An example of *inductive* and *supervised* learning

Example: Wait for restaurant table

- Goal predicate: *WillWait* (for restaurant table)
- Domain attributes
    - Alternate (other restaurants nearby)
    - Bar (to wait in)
    - Fri/Sat (day of week)
    - Hungry (yes/no)
    - Patrons (none, some, full)
    - Price (range)
    - Raining (outside)
    - Reservation (made before)
    - Type (French, Italian, ..)
    - WaitEstimate (minutes)

One decision tree for the example



Expressiveness of decision trees

- The tree is equivalent to a conjunction of implications

  $$\forall r \, Patrons(r, Full) \wedge WaitEstimate(r, 10-30) \wedge Hungry(r, No) \Rightarrow WillWait(r)$$

- Cannot represent tests on two or more objects, restricted to testing attributes of one object
- Fully expressive as propositional language, e.g. any Boolean function can be written as a decision tree
- For some functions, exponentially large decision trees are required
- E.g. decision trees are good for some functions and bad for others

Inducing decision trees from examples

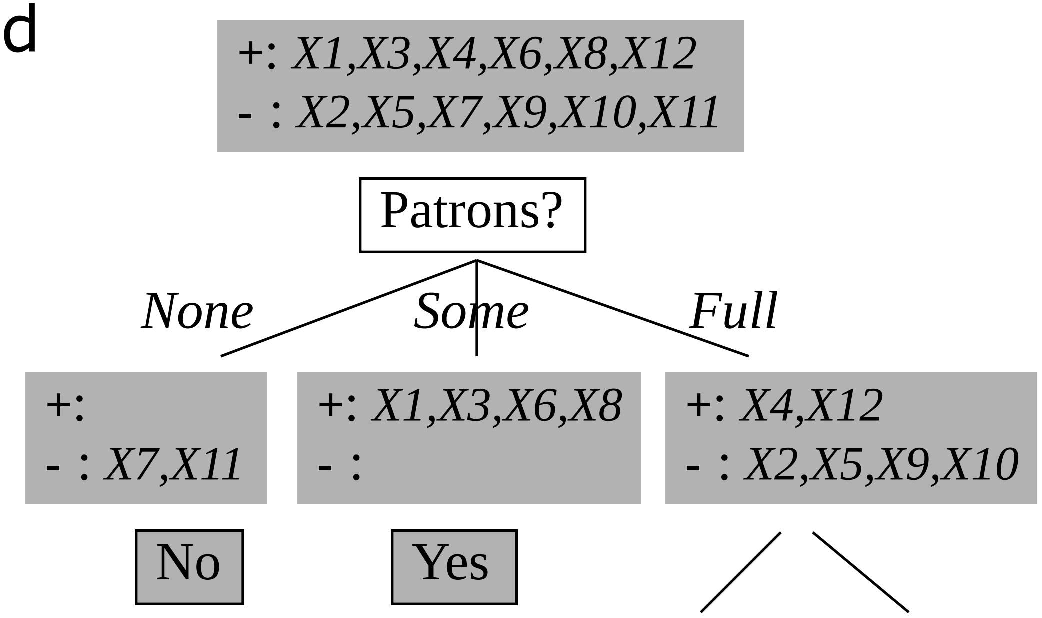
- Terminology
  - *Example* - Specific values for all attributes, plus goal predicate
  - *Classification* - Value of goal predicate of the example
  - *Positive/negative example* - Goal predicate is true/false
  - *Training set* - Complete set of examples
- The task of inducing a decision tree from a training set is to *find the simplest tree that agrees with the examples*
- The resulting tree should be more *compact* and *general* than the training set itself

A training set for the restaurant example

Example	Attributes										Will wait
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	
X1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0-10*	*Yes*
X2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30-60*	*No*
X3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0-10*	*Yes*
X4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*10-30*	*Yes*
X5											
X6											
X7											
X8						ETC.					
X9											
X10											
X11											
X12											

General idea of induction algorithm

- Test the most important attribute first, i.e. the one that makes the most difference to the classification
- *Patrons?* is a good choice for the first attribute, because it allows early decisions
- Apply same principle recursively

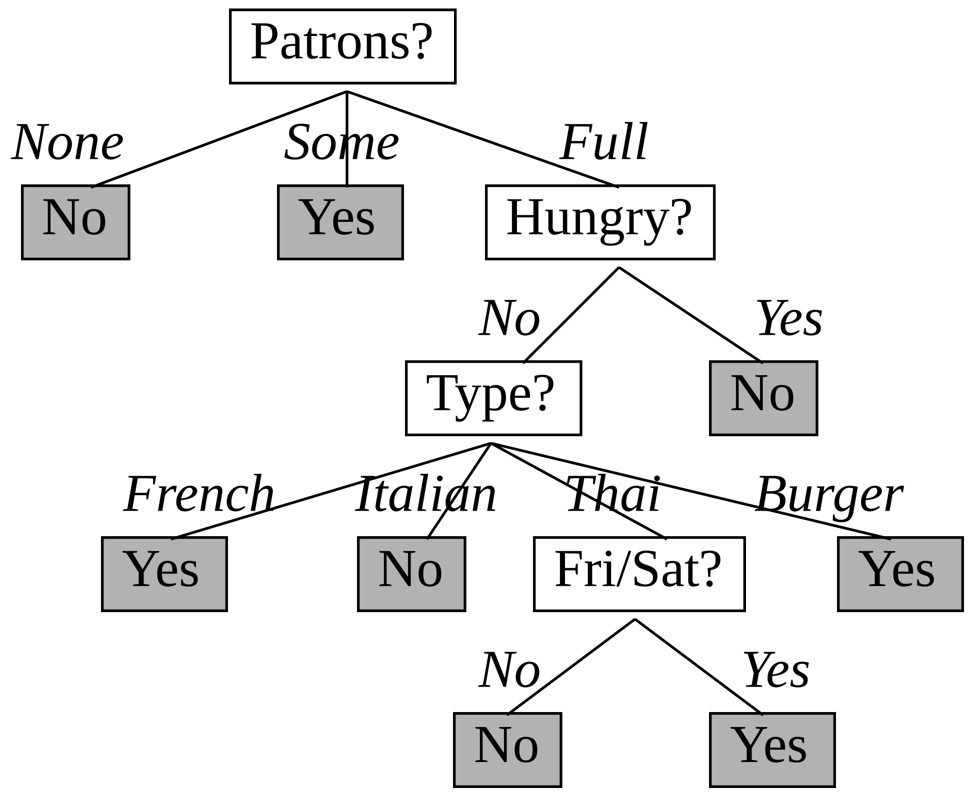


Recursive step of induction algorithm

- The attribute test splits the tree into smaller decision trees, with fewer examples and one attribute less
- Four cases to consider for the smaller trees
  - √ If some positive and some negative examples, choose best attribute to split them
  - √ If examples are all positive (negative), answer *Yes* (*No*)
  - √ If no examples left, return a default value (no example observed for this case)
  - √ If no attributes left, but both positive and negative examples: Problem! (same description, different classifications - *noise*)

Induced tree for the example set

- The induced tree is *simpler* than the original "manual" tree
- It captures some *regularities* that the original creator was unaware of

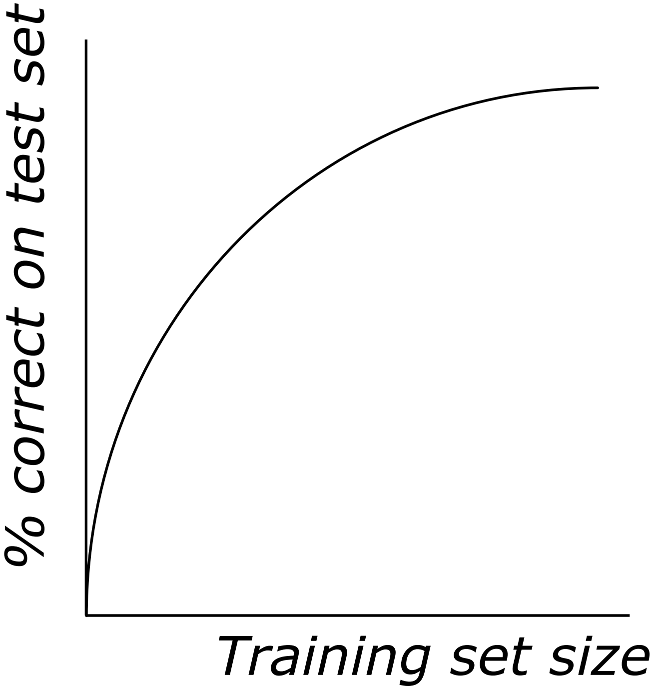


Broaden applicability of decision trees

- Missing data
  - How to handle training samples with partially missing attribute values
- Multi/many-valued attributes
  - How to treat attributes with many possible values
- Continuous or integer-valued input attributes
  - How to branch the decision tree when attribute has a continuous value range
- Continuous-valued output attributes
  - Requires *regression tree* rather than a decision tree, i.e. output value is a linear function of input variables rather than a point value

Assessing learning performance

- Collect large set of examples
- Divide into two disjoint sets, *training set* and *test set*
- Use learning algorithm on training set to generate hypothesis *h*
- Measure percentage of examples in test set that are correctly classified by *h*
- Repeat steps above for differently sized training sets



PAC – The theory of learning

- How can we be sure that the learning algorithm gives a function *h* that predicts correctly?
  - How many learning examples are needed?
  - What hypothesis space *H* should be used?
  - Etc.
- Computational learning theory tries to answer such questions
  - Underlying principle: Any *h* that is consistent with a sufficient large number of examples is *probably approximately correct* (PAC)
- PAC theory can be used to bound hypothesis space and size of example set

A logical formulation of learning

- Inductive learning can be seen as *searching for a good hypothesis* in a large search space
- The *hypothesis space* is defined by a particular representation language, e.g. logic
- Define learning in terms of logical connections between *hypotheses*, *examples*, and *goals*
- This approach enables extensions of simple inductive decision tree learning to applying full *logical inference*

Hypothesis space

- Let Q be a unary *goal predicate*, and C_i a *candidate definition*, i.e. a *hypothesis* H_i for classifying examples x correctly is that

$$\forall x\, Q(x) \Leftrightarrow C_i(x)$$

- Example: Induced decision tree is equivalent to

$$\begin{aligned}\forall r\, WillWait(r) \Leftrightarrow\ & Patrons(r, Some)\\ & \vee Patrons(r, Full) \wedge \neg Hungry(r) \wedge Type(r, French)\\ & \vee Patrons(r, Full) \wedge \neg Hungry(r) \wedge Type(r, Thai) \wedge Fri/Sat(r)\\ & \vee Patrons(r, Full) \wedge \neg Hungry(r) \wedge Type(r, Burger)\end{aligned}$$

- *Hypothesis space* is the set $\{H_1, .., H_n\}$, of which one is believed to be correct: $H_1 \vee H_2 \vee \ldots H_n$

Examples for learning

- An *example* is an object X_i to which the goal concept Q may or may not apply ($Q(X_i)$ or $\neg Q(X_i)$), and which has a logical description $D_i(X_i)$
- E.g. first induction example X_1

$$Alternate(X_1) \wedge \neg Bar(X_1) \wedge \neg Fri/Sat(X_1) \wedge Hungry(X_1) \wedge \ldots$$

  with classification $WillWait(X_1)$
- Complete *training set* is the conjunction of all X_i
- A hypothesis agrees with all examples if and only if it is *logically consistent* with the training set

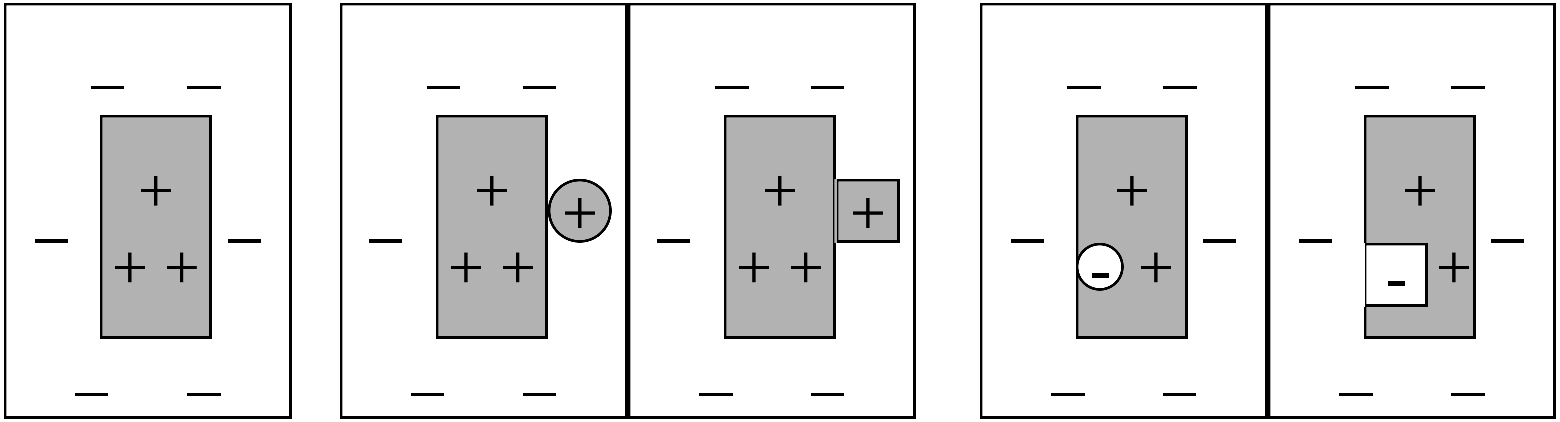
False examples and inductive learning

- If a hypothesis H_i is consistent with the entire training set, it must be consistent with each example
- An example can be a *false negative* for the hypothesis, i.e. H_i says it should be negative but it is positive
- An example can be a *false positive* for the hypothesis, i.e. H_i says it should be positive but it is negative
- If the example is false negative or false positive, the example and hypothesis are *inconsistent*, and the hypothesis can be *ruled out*

> Inductive learning in a logical setting is the process of gradually eliminating hypotheses that are inconsistent with the examples

Current-best-hypothesis search

- *Current-best-hypothesis* search maintains a single hypothesis which is adjusted as new examples arrive to maintain consistency



Consistent hypothesis | False negative | *Generalized* hypothesis | False positive | *Specialized* hypothesis

Generalizing and specializing hypotheses

- If hypothesis H1 with definition C1 is a generalization of H2 with definition C2, then

$$\forall x\ C_2(x) \Rightarrow C_1(x)$$

- *Generalization* of a hypothesis can be achieved by *dropping conditions*

$$C_2(x) = Alternate(x) \wedge Patrons(x, Some) \qquad C_1(x) = Patrons(x, Some)$$

- *Specialization* of a hypothesis can similarly be achieved by *adding conditions*
- Current-best-hypothesis search with generalization and specialization and backtracking has been used in many learning programs, but does not scale well

Least commitment search

- The current-best-hypothesis approach has to backtrack because it is forced to choose *one* hypothesis even if it does not have enough data
- A better approach is to keep all hypotheses consistent with data so far, and gradually remove hypotheses inconsistent with new examples
- Assuming that the right hypothesis is contained in the original set, it will still be in the reduced set (the *version space*)
- The algorithm is *incremental*, need not backtrack

Other knowledge-based learning methods

- EBL – Explanation-based learning
  - √ Extracts general rules from single examples accompanied by an explanation
- RBL – Relevance-based learning
  - √ Uses prior knowledge to identify relevant attributes thereby reducing hypothesis space
- KBIL – Knowledge-based inductive learning
  - √ Uses prior knowledge to find inductive hypotheses that explain sets of observations
- ILP – Inductive logic programming
  - √ Performs KBIL on knowledge expressed in first-order logic, and can learn relational knowledge

Summary

- Learning is an essential capability for agents in unknown or resource-constrained environments
- Learning agents have a *performance* element and a *learning* element
- The learning element tries to improve various parts of the performance element, generally seen as *functions* $y = f(x)$
- Learning can be *inductive* (from examples) or *deductive* (based on knowledge)
- Differ in types of *feedback* to the agent: un-supervised, reinforcement or supervised learning

Summary (cont.)

- Learning a function from examples of inputs and outputs is an example of inductive/supervised learning, of which learning *decision trees* is a simple case
- A logical formulation of learning uses c*urrent-best-hypothesis* approach to maintain a single hypothesis which is updated with new examples
- Other logical or knowledge-based learning methods include EBK, RBIL, KBIL and IPL