Learning

Introduction to Artificial Intelligence

G. Lakemeyer

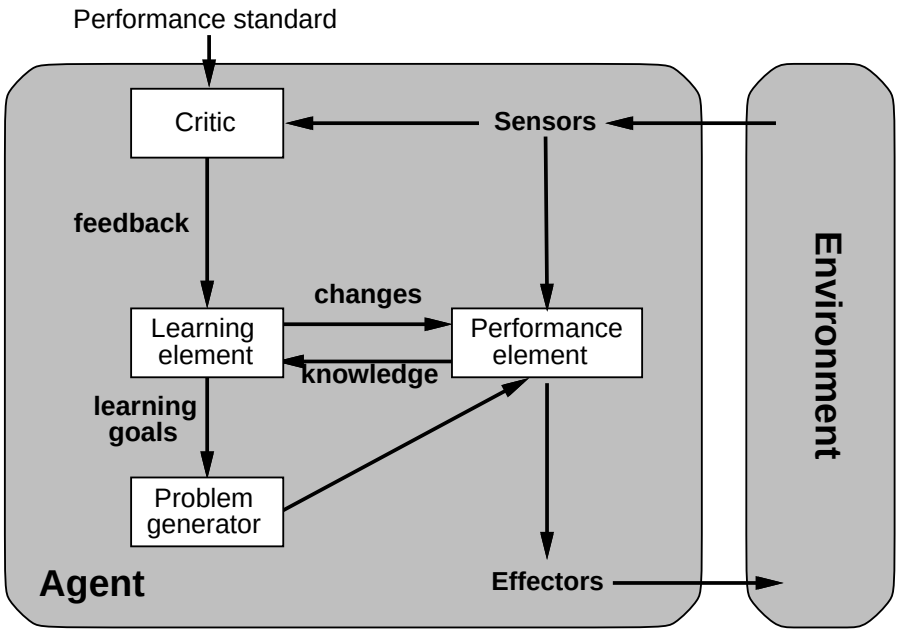
Winter Term 2016/17

Learning

The Goal of Learning

Optimize future behavior on the basis of the history of percepts, actions, and knowledge about the world.

A general architecture of a learning agent:



Performance Element: Agent in the old sense.

Critic: Tells the system how good or bad it is performing.

Learning Element: Improves the system.

Problem Generator: Suggests actions to test how good the system performs.

Kinds of Feedback during Learning

In an abstract sense, an agent is a function from inputs (like percepts) to outputs (actions). We distinguish the actual function from the ideal function, which models optimal behavior. The goal of learning is then to approximate the ideal function as good as possible.

Supervised Learning:	Both the input and the correct output are available to the learner. (There is a teacher (supervisor).)
Reinforcement Learning:	While the correct answer is not available, there is feedback in terms of rewards and punishment.
Unsupervised Learning:	There is no indication of what the correct output is. Can learn structure in the input using supervised learning methods by predicting future inputs on the basis of past inputs. (The system is its own supervisor.)

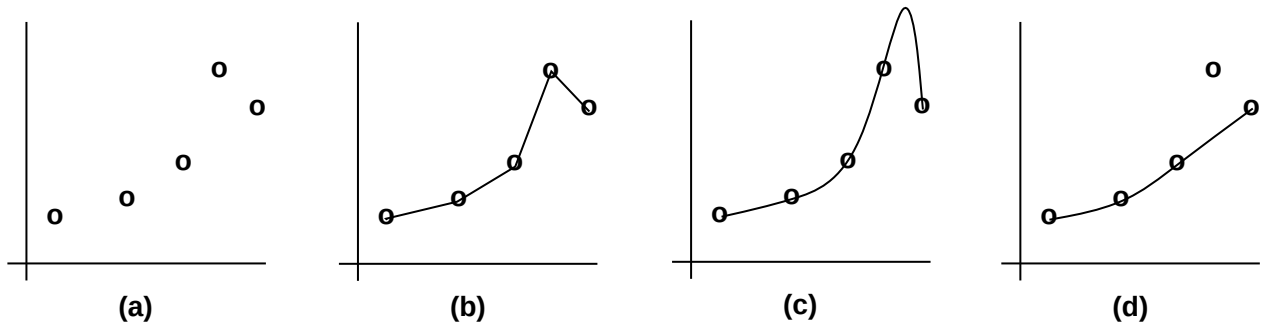
Inductive Learning

Learning the ideal function with supervision.

Suppose we are given the input and correct output as a pair $(x, f(x))$.
(f is the ideal yet unknown function)

Wanted: a function (hypothesis) h which approximates f.

Example with 4 different h's.



Note: Since there are many possibilities for h, this works only with additional assumptions which restrict the search space: bias.

In the following: Decision Trees (DT's) as an example of inductive learning.

Decision Trees

Input: Description of a situation using a set of properties (roughly, literals in FOL).

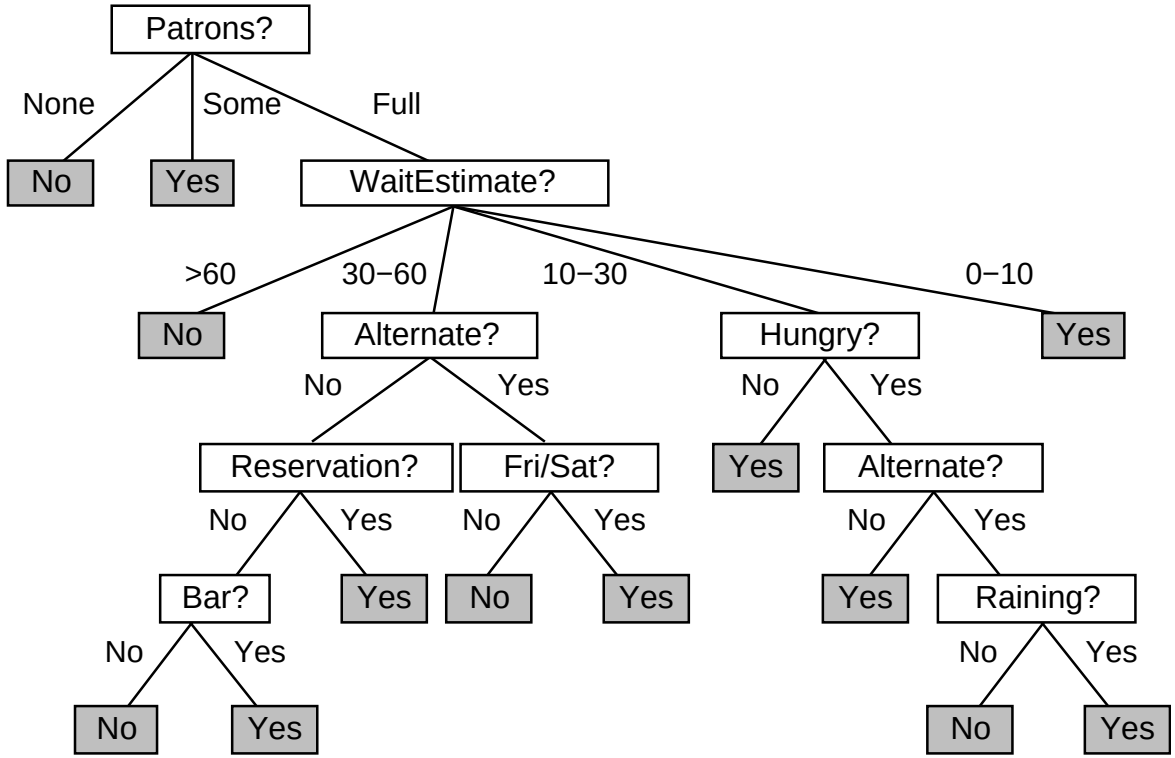
Output: Yes/No decision relative to a goal predicate.

With that decision trees represent Boolean functions.

[Can be easily generalized to many-valued functions.]

We want to learn an ideal Boolean function or a logical formula which represents this function.

Restaurant Example

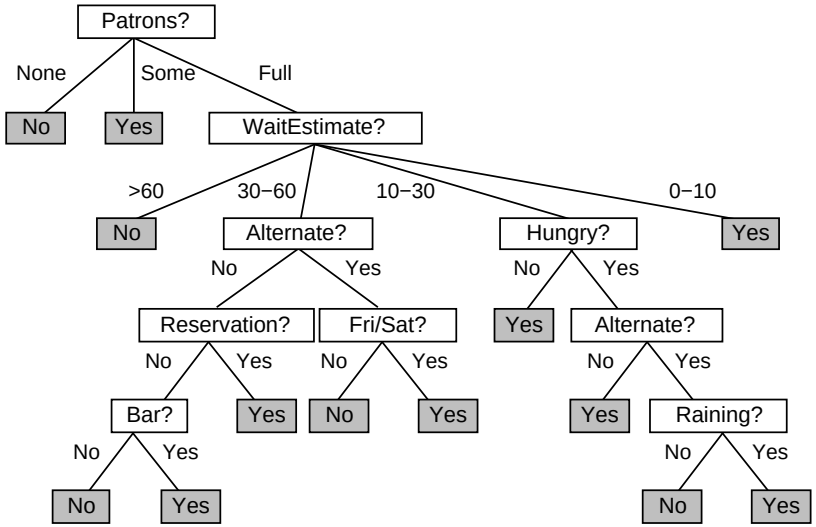


Patrons: how many people?
WaitEstimate: how long to wait?
Alternate: are there alternatives?
Hungry: am I hungry?
Reservation: do I have a reservation?
Bar: is there a bar in the restaurant?

Fri/Sat: is it a Friday or a Saturday?
Raining: is it raining?

Other attributes:
Price: how expensive are the meals?
Type: what type of restaurant is it?

Expressiveness of DT's



Each tree describes a set of implications in FOL:

$$\forall r Patrons(r, Full) \wedge WaitEstimate(r, 10-30) \wedge \neg Hungry(r) \supset WillWait(r).$$

Not all formulas in FOL are representable because the tree only refers to one object (here: the restaurant r).

For example: $\exists r_2 Near(r_2, r) \wedge Price(r, p) \wedge Price(r_2, p_2) \wedge Cheaper(p_2, p)$ is not representable.

Theorem:

Every propositional formula (Boolean function) is representable by a decision tree.

Learning in Decision Trees

Decision trees can trivially represent any Boolean function by having each path represent one valuation of the attributes (atomic formulas). Often, however, there are much more compact representations.

Always?
No! For example, the parity function (answers Yes if an even number of attributes are true).

Learning in decision trees:
Given positive (answer: Yes) and negative (answer: No) examples, find the correct Boolean function represented as a decision tree.

Problem: With n attributes there are 2^{2^n} possible functions. How does one find the right one??

Generating a Decision Tree from Examples

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

Trivial DT: have one path in the tree per example (memorizing).

Instead: find a compact DT which covers all examples.

Problem:

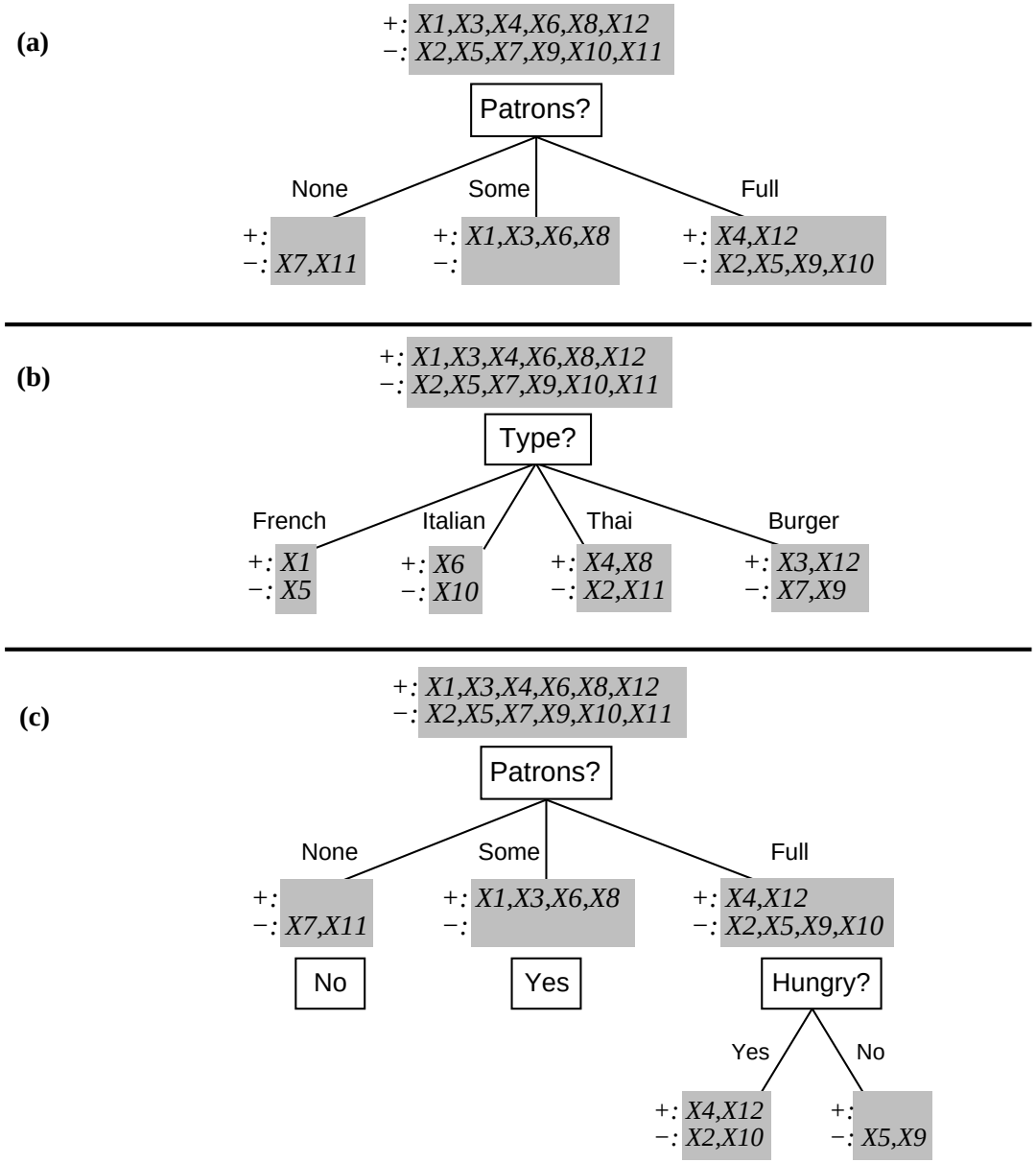
the tree is too big,
no generalization possible.

Idea: Occams Razor

"The most likely hypothesis is the simplest which covers all examples."

$\Rightarrow$ choose an attribute which is most helpful in classifying the examples.

Choosing Attributes/Nodes for the Decision Tree



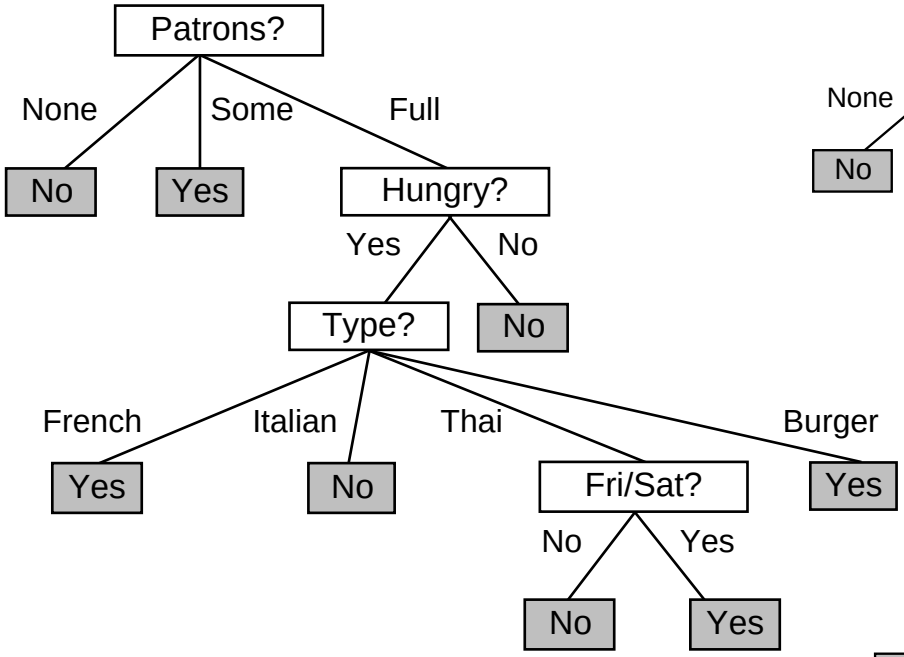
An Algorithm

function DECISION-TREE-LEARNING(*examples*, *attributes*, *default*) **returns** a decision tree
  inputs: *examples*, set of examples
    attributes, set of attributes
    default, default value for the goal predicate

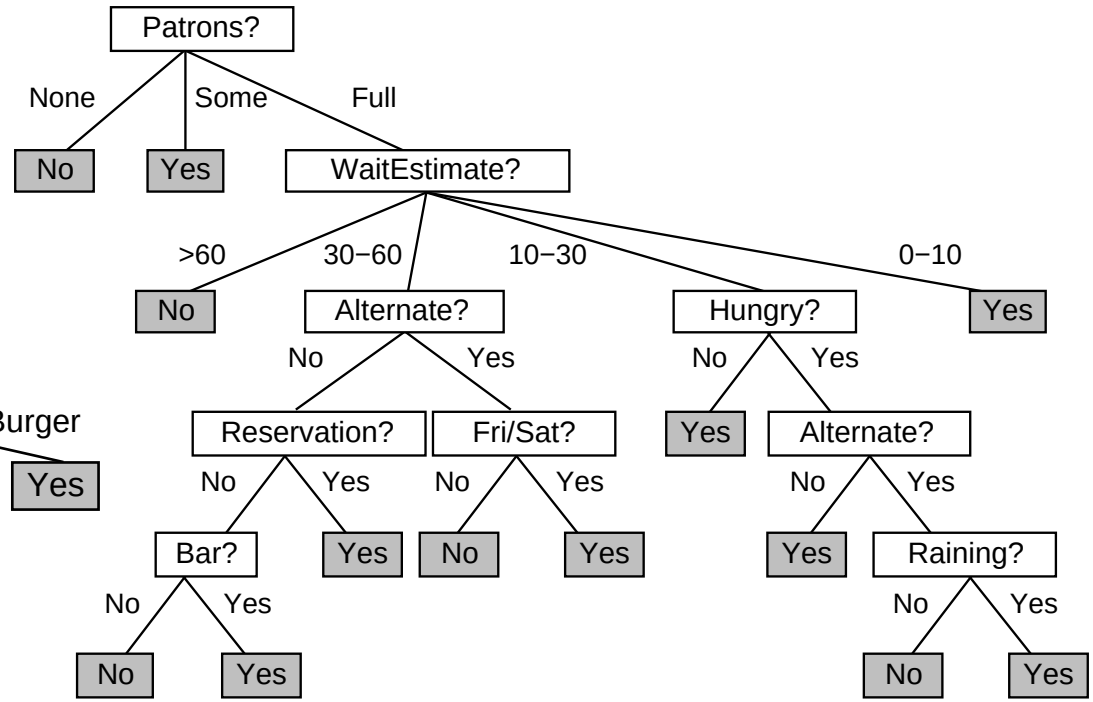
  if *examples* is empty **then return** *default*
  else if all *examples* have the same classification **then return** the classification
  else if *attributes* is empty **then return** MAJORITY-VALUE(*examples*)
  else
    best $\leftarrow$ CHOOSE-ATTRIBUTE(*attributes*, *examples*)
    tree $\leftarrow$ a new decision tree with root test *best*
    for each value v_i of *best* **do**
      $examples_i \leftarrow$ {elements of *examples* with *best* = v_i}
      subtree $\leftarrow$ DECISION-TREE-LEARNING($examples_i$, *attributes* $-$ *best*,
                MAJORITY-VALUE(*examples*))
      add a branch to *tree* with label v_i and subtree *subtree*
    end
    return *tree*

1. If there are only positive or only negative examples, then done. Answer Yes or No, respectively.
2. If there are both positive and negative examples, then choose the best attribute to distinguish between them.
3. If there are no more examples, then there are no examples with these properties. Answer Yes if the majority of the examples at the parent node are positive, otherwise answer No.
4. If there are no more attributes, then there are identical examples with different classifications, that is, there is either an error in the date (noise), or the attributes are insufficient to distinguish between the situations. Answer Yes if the majority of examples are positive, otherwise answer No.

Example



Compared to the DT drawn by hand:



Evaluating a Learning Algorithm

1. Collect a large set of examples.
2. Separate them into disjoint training and test sets.
3. Use the training set to generate a hypothesis H (e.g. a decision tree).
4. Measure the percentage of correctly classified examples of the test set.
5. Repeat steps 1–4 for randomly selected training sets of different size.

Note:

- Keeping the training and test sets separate is crucial!
- **Common mistake:** After a round of testing the learning algorithm is modified and then trained and tested with new sets generated from the **same** set of examples as before.
  The problem is that knowledge about the test set is already contained in the algorithm, i.e. training and test sets are no longer independent.

Using Information Theory to Find Next Attribute

Information theory was founded by Shannon and Weaver (1949)

We would like to know: What is the information content of an answer to a Yes/No query?

Analogy to betting:
Information content $\approx$ how much is it worth to me if someone tells me the right answer?

Flipping a coin:

(Heads, Tails). Here the bet is €1,- on Heads.

1. fair coin: $P(\mathrm{H})=P(\mathrm{T})= 0.5$
   I am willing to pay €.99 for the right answer!
2. unfair coin: P(H)=0.99; P(T)=0.01
   How much is the correct answer worth to me now?

Information Theory (2)

The information content is measured in bits.

Let $v_1, \ldots, v_n$ be the possible answers to a question with prob. $P(v_i)$.

Information content:

$$I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) \times log_2 P(v_i).$$

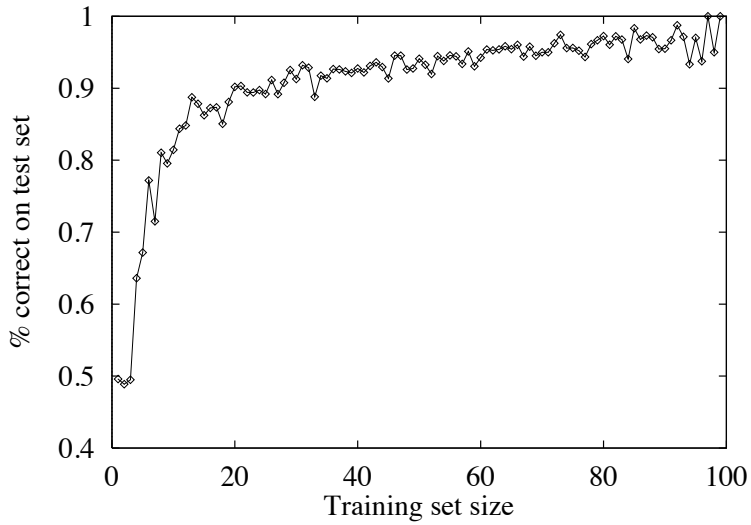
Fair coin:

$$I(\tfrac{1}{2}, \tfrac{1}{2}) = -\tfrac{1}{2} \times log_2 \tfrac{1}{2} - \tfrac{1}{2} \times log_2 \tfrac{1}{2} = 1 \text{ bit.}$$

Unfair coin:

$$I(\tfrac{99}{100}, \tfrac{1}{100}) = 0.08 \text{ bits.}$$

Learning Curve of the Restaurant Example



DT's in practice:

- The GASOIL expert system to separate crude oil from gas. Makes decisions on the basis of attributes like the proportion of oil/gas/water, throughput, pressure, viscosity, temperature etc. The complete system has about 2500 rules (paths in the DT). Is better than most human experts in this area.
- Flight simulator for a Cessna. Data generated by observing 3 test pilots during 30 test flights each. 90000 examples with 20 attributes, uses C4.5 (state-of-the-art DT-Alg.)

Learning general logical descriptions

Goal: Learning a 1-place predicate G [e.g.: $WillWait(r)$].

Hypothesis space: The set of all logical definitions of the goal predicate

$$\forall x G(x) \equiv \alpha(x).$$

Restaurant example:

Hypothesis H_r (corresponds to the previous decision tree):

$$\begin{aligned} \forall r WillWait(r) \quad \equiv \quad & Patrons(r, some) \\ \vee \quad & Patrons(r, full) \wedge Hungry(r) \wedge Type(r, French) \\ \vee \quad & Patrons(r, full) \wedge Hungry(r) \wedge Type(r, Thai) \\ & \wedge Fri/Sat(r) \\ \vee \quad & Patrons(r, full) \wedge Hungry(r) \wedge Type(r, Burger) \end{aligned}$$

False Positive and Negative Examples

Examples are also logical descriptions of the kind:

$$Ex_1 = Alternate(X_1) \wedge \neg Bar(X_1) \wedge \ldots \wedge Patrons(X_1, some) \wedge \ldots \wedge WillWait(X_1)$$

Note that H_r is logically consistent with Ex_1.

Let

$$Ex_{13} = Patrons(X_{13}, Full) \wedge Wait(X_{13}, 0\text{-}10) \wedge \neg Hungry(X_{13}) \wedge \ldots \wedge WillWait(X_{13}).$$

Then Ex_{13} is called a false negative example because H_r predicts $\neg WillWait(X_{13})$, yet the example is positive.

Similarly, an example is false positive if H_r says it should be positive, yet in fact it is negative.

Note: False positive and false negative examples are logically inconsistent with a given hypothesis.

Learning as the Elimination of Hypotheses

Suppose we are given n possible hypotheses H_r. Then we can represent the hypothesis space as

$$H_1 \vee H_2 \vee \ldots \vee H_n.$$

Learning can be thought of as a successive reduction of the hypothesis space by eliminating disjuncts for which we have false negative or false positive examples, i.e. examples which are inconsistent with these hypotheses.

Usually not practical since the hypothesis space is too big.

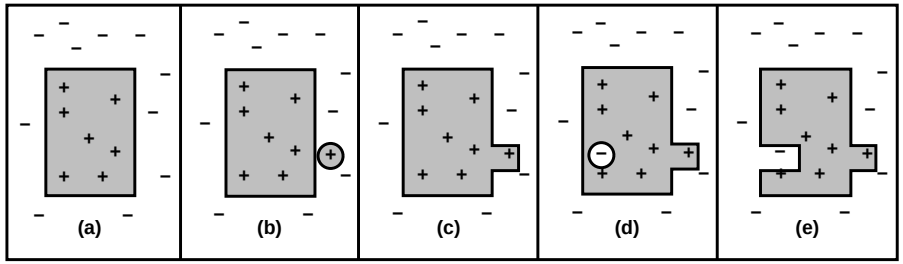
Sometimes it is possible to have compact representations of the hypothesis space (Version Spaces). An alternative is to only consider one hypothesis and modify it when needed.

Strategy of the Current Best Hypothesis

Only consider one hypothesis at a time. If there is a new example which is inconsistent with the hypothesis, then change it in the following way. Let the extension of a hypothesis be the set of objects which satisfy the goal predicate according to the hypothesis.

Generalization: make the extension bigger for a false negative example, (see b+c).

Spezialization: make the extension smaller for a false positive (d+e).



If $H_1 = \forall x G(x) \equiv \alpha(x)$ and $H_2 = \forall x G(x) \equiv \beta(x)$, then H_2 is a generalization of H_1 iff $\forall \alpha(x) \supset \beta(x)$.

A simple kind of generalization is obtained by removing conditions from α.

Example

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

- Example X_1 is positive. Since $Alternate(X_1)$ is true, let H_1: $\forall x WillWait(x) \equiv Alternate(x)$.
- Example X_2 is negative. H_1 predicts it as false positive. H_1 must be specialized. H_2: $\forall x WillWait(x) \equiv Alt.(x) \wedge Patrons(x, some)$
- X_3 is positive, but according to H_2 false negative. Generalization results in H_3: $\forall x WillWait(x) \equiv Patrons(x, some)$
- X_4 is positive, but according to H_3 false negative. Dropping $Patrons(x, some)$ contradicts X_2. Thus add a new disjunct: H_4: $\forall x WillWait(x) \equiv Patrons(x, some) \vee (Patrons(x, full) \wedge Fri/Sat(x))$.

Problems

Some problems with the current-best hypothesis:

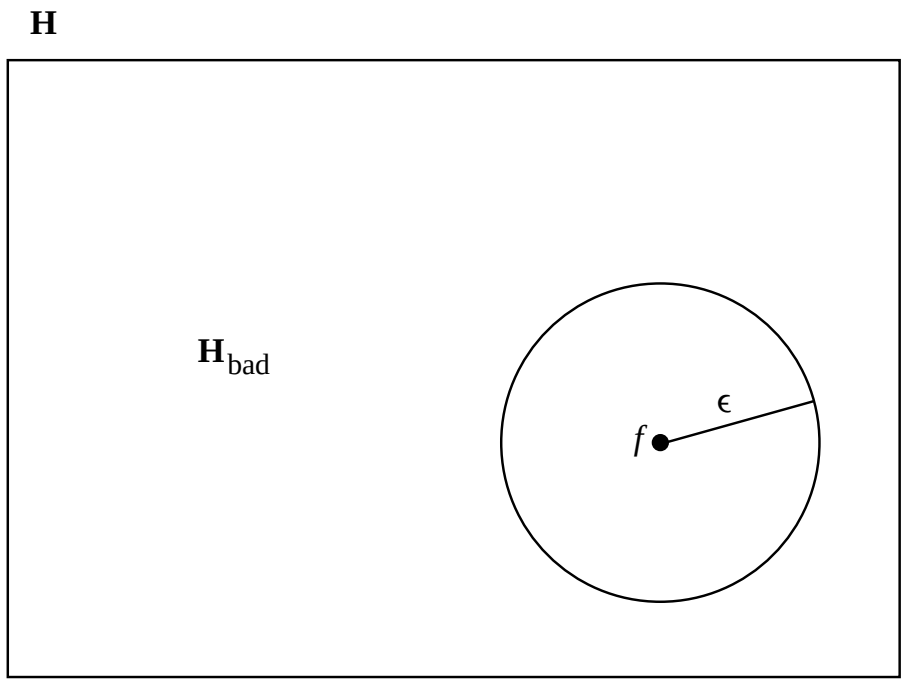
- All previous examples need to be tested again.
- It is difficult to find good heuristics. The search can easily lead to a dead end.
  $\Rightarrow$ uncontrolled backtracking.

PAC-Learning

When (realistically) assuming that the ideal function f to be learned is unknown, how can one ever be certain that the hypothesis h found is close to f?

The PAC-Theorie of Learning gives us criteria when h is Probably Approximately Correct.

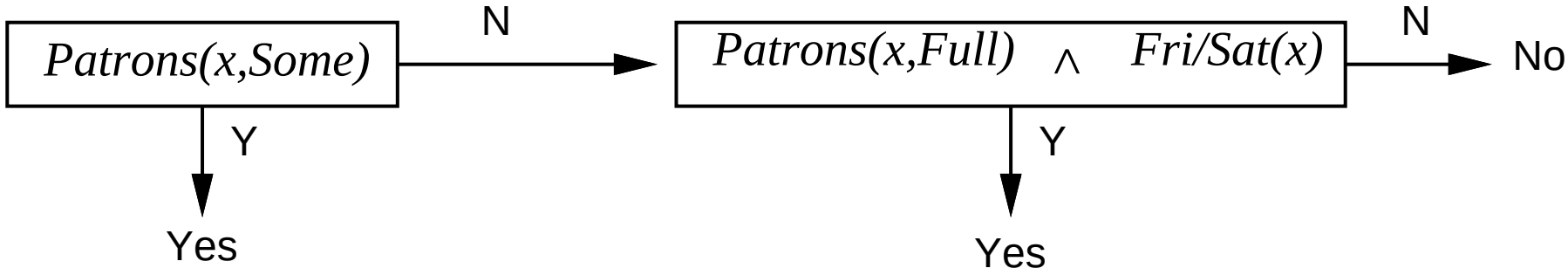
Tells us how many examples one needs to see so that h is within ϵ of f with probability $(1 - \delta)$ for arbitrarily small δ und ϵ ($\neq 0$).



Decision Lists

Decision lists (DL's) consist of a number of tests, which themselves consist of a conjunction of a bounded number of literals. If a test is successful (all the literals are satisfied), then the DL tells us which value to return. Otherwise, the next test is tried.

Example



This corresponds to the hypothesis

$$H_4 : \forall x \, WillWait(x) \equiv Patrons(x, some) \vee (Patrons(x, full) \wedge Fri/Sat(x)).$$

Note:

Decision lists represent only a restricted class of logical formulas.

Algorithm Decision Lists

function DECISION-LIST-LEARNING(*examples*) **returns** a decision list, *No* or failure

if *examples* is empty **then return** the value *No*
t ← a test that matches a nonempty subset $examples_t$ of *examples*
such that the members of $examples_t$ are all positive or all negative
if there is no such *t* **then return** failure
if the examples in $examples_t$ are positive **then** *o* ← *Yes*
else *o* ← *No*
return a decision list with initial test *t* and outcome *o*
and remaining elements given by DECISION-LIST-LEARNING($examples - examples_t$)

Restaurant example: