Artificial Intelligence

Topic 6

Agents that Learn

◇ why learn?

◇ general model of learning agents

◇ inductive learning

◇ learning decision trees

Reading: Russell & Norvig, Chapter 18

1. Why Learn?

So far, all intelligence comes from the designer:

- time consuming for designer
- *restricts capabilities of agent*

Learning agents can:

- act autonomously
- deal with unknown environments
- synthesise rules/patterns from large volumes of data
- handle complex data
- improve their own performance

2. General Model of Learning Agents

Idea: percepts used not just for acting, but for improving future performance.

Four basic components. . .

Performance element

— responsible for selecting actions for good agent performance

— agent function considered previously: percepts $\rightarrow$actions

Learning element

— responsible for improving performance element

— requires feedback on how the agent is doing

Critic element

— responsible for providing feedback

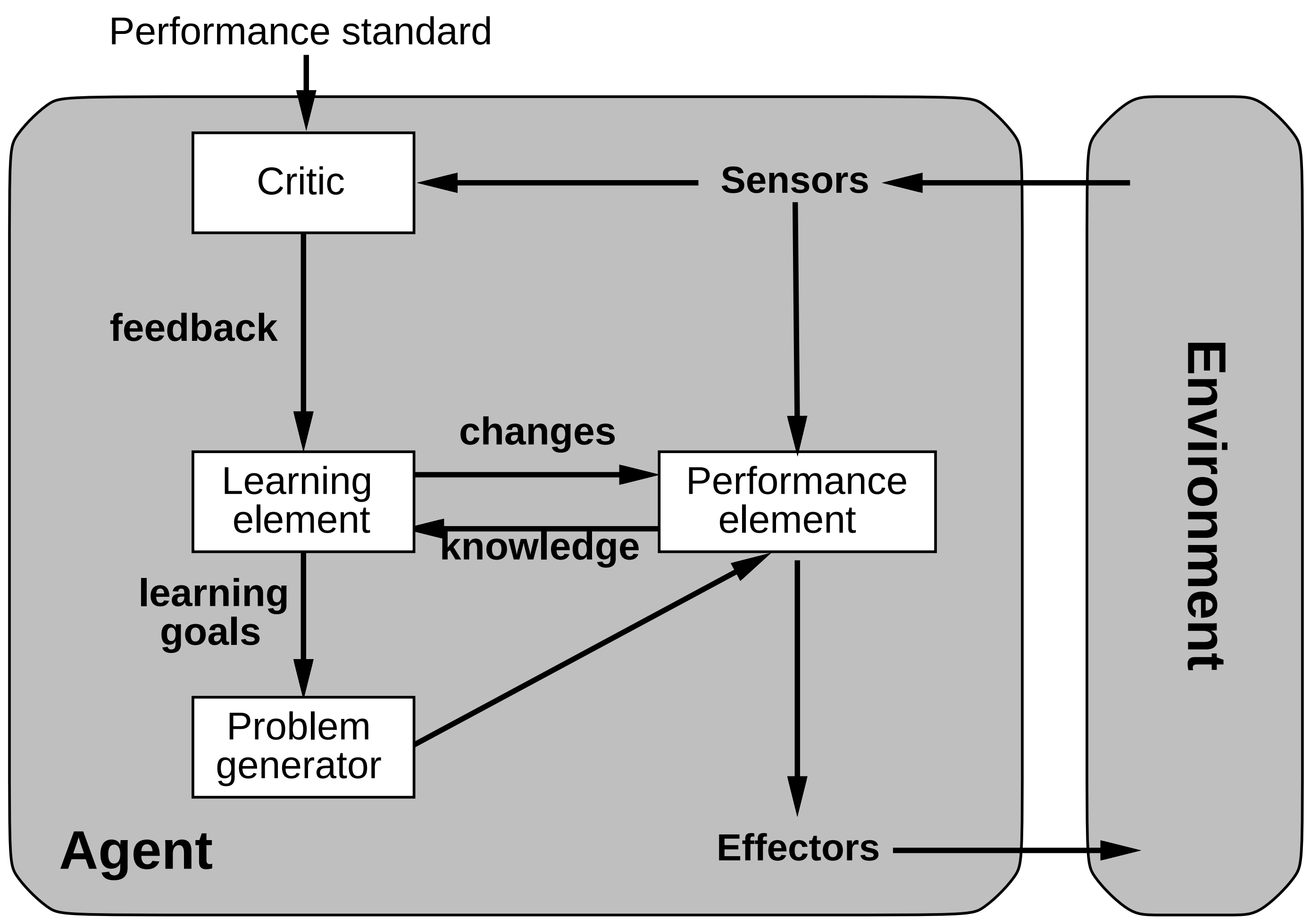
— comparison with objective performance standard (outside agent)

Problem generator

— responsible for generating new experience

— requires *exploration* — taking suboptimal actions

2. General Model of Learning Agents



E.g. taxi driver agent

performance element — Lets take Winthrop Ave, I know it works.

problem generator — Nah, lets try Mounts Bay Rd for a change. You never know, it may be quicker.

critic — Wow, it was 5 mins quicker, and what a lovely view!

learning element — Yeh, in future we'll take Mounts Bay Rd.

2.1 The Learning Element

Two separate goals:

1. Improve *outcome* of performance element
   — how good is the solution
2. Improve *time performance* of performance element
   — how fast does it reach a solution
   — known as *speedup learning*

Learning systems may work on one or both tasks.

Design of a learning element is affected by four issues:

1. the components of the performance element to be improved
2. representation of those components
3. feedback available
4. prior information available

2.1 The Learning Element

Components of the performance element

Can use:

1. direct mapping from states to actions
2. means to infer properties (model) of the world from percept sequence
3. information on how the world evolves
4. information about how actions change the world
5. utility information about desirability of states
6. action-value information about desirability of actions in states
7. goals whose achievement maximises utility

Each component can be learned, given appropriate feedback.

E.g. our taxi driver agent

- Mounts Bay Rd has a nicer view (5)
- Taking Mounts Bay Rd $\longrightarrow$ arrive more quickly (4,7)
- If travelling from Perth to UWA, always take Mounts Bay Rd (6)

2.1 The Learning Element

Representation of components

Examples:

- utility functions, eg linear weighted polynomials for game-playing agent
- logical sentences for reasoning agent
- probabilistic descriptions, eg belief networks for decision-theoretic agent

Available feedback

supervised learning

— agent provided with both inputs and correct outputs

— usually by a “teacher”

reinforcement learning

— agent chooses actions

— receives some *reward* or *punishment*

unsupervised learning

— no hint about correct outputs

— can only learn relationships between percepts

2.1 The Learning Element

Prior knowledge

- agent begins with a *tabula rasa* (empty slate)
- agent makes use of background knowledge

In practice learning is hard $\Rightarrow$ use background knowledge if available.

Learning = Function approximation

All components of performance element can be described mathematically by a function. eg.

- how the world evolves: f: state $\mapsto$ state
- goal: f: state $\mapsto$ $\{0,1\}$
- utilities: f: state $\mapsto$ $[-\infty, \infty]$
- action values: f: (state,action) $\mapsto$ $[-\infty, \infty]$

all learning can be seen as learning a function

3. Inductive Learning

Assume f is function to be learned.

Learning algorithm supplied with sample inputs and corresponding outputs $\Rightarrow$ supervised learning

Define *example* (or *sample*): pair $(x, f(x))$

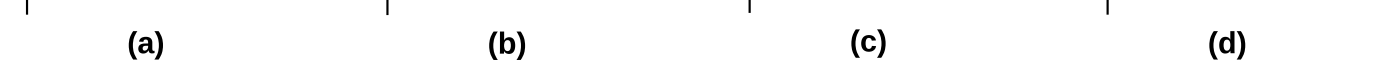
Task:

> given a set of examples of f, return a function h that approximates f.

h is called a *hypothesis*

Task is called *pure inductive inference* or *induction*.

Many choices for h



A preference for one over another is called a *bias*.

3. Inductive Learning

Example: Simple reflex agent

Sample is a pair (*percept*,*action*)

eg. *percept* — chess board position

action — best move supplied by friendly grandmaster

Algorithm:

```
global examples ← {}
---
function REFLEX-PERFORMANCE-ELEMENT(percept) returns an action
    if (percept, a) in examples then return a
    else
        h ← INDUCE(examples)
        return h(percept)
---
procedure REFLEX-LEARNING-ELEMENT(percept, action)
    inputs: percept, feedback percept
            action, feedback action
    examples ← examples ∪ {(percept, action)}
```

3. Inductive Learning

Many variations possible, eg.

- *incremental learning* — learning element updates h with each new sample
- *reinforcement* — agent receives feedback on quality of action

Many possible representations of h — choice of representation is critical

- type of learning algorithm that can be used
- whether learning problem is feasible
- expressiveness vs tractability

4. Learning Decision Trees

Decision tree

- input — description of a situation
  — abstracted by set of *properties*, *parameters*, *attributes*, *features*,...
- output — boolean (yes/no) decision
  — can also be thought of as defining a *categorisation*, *classification* or *concept*
  $\Rightarrow$ set of situations with a positive response

$$f\colon \text{ situation} \mapsto \{0, 1\}$$

We consider

1. decision trees as performance elements
2. inducing decision trees (learning element)

4.1 Decision Trees as Performance Elements

Example

Problem: decide whether to wait for a table at a restaurant

Aim: provide a definition, expressed as a decision tree, for the goal concept “Will Wait”

First step: identify attributes

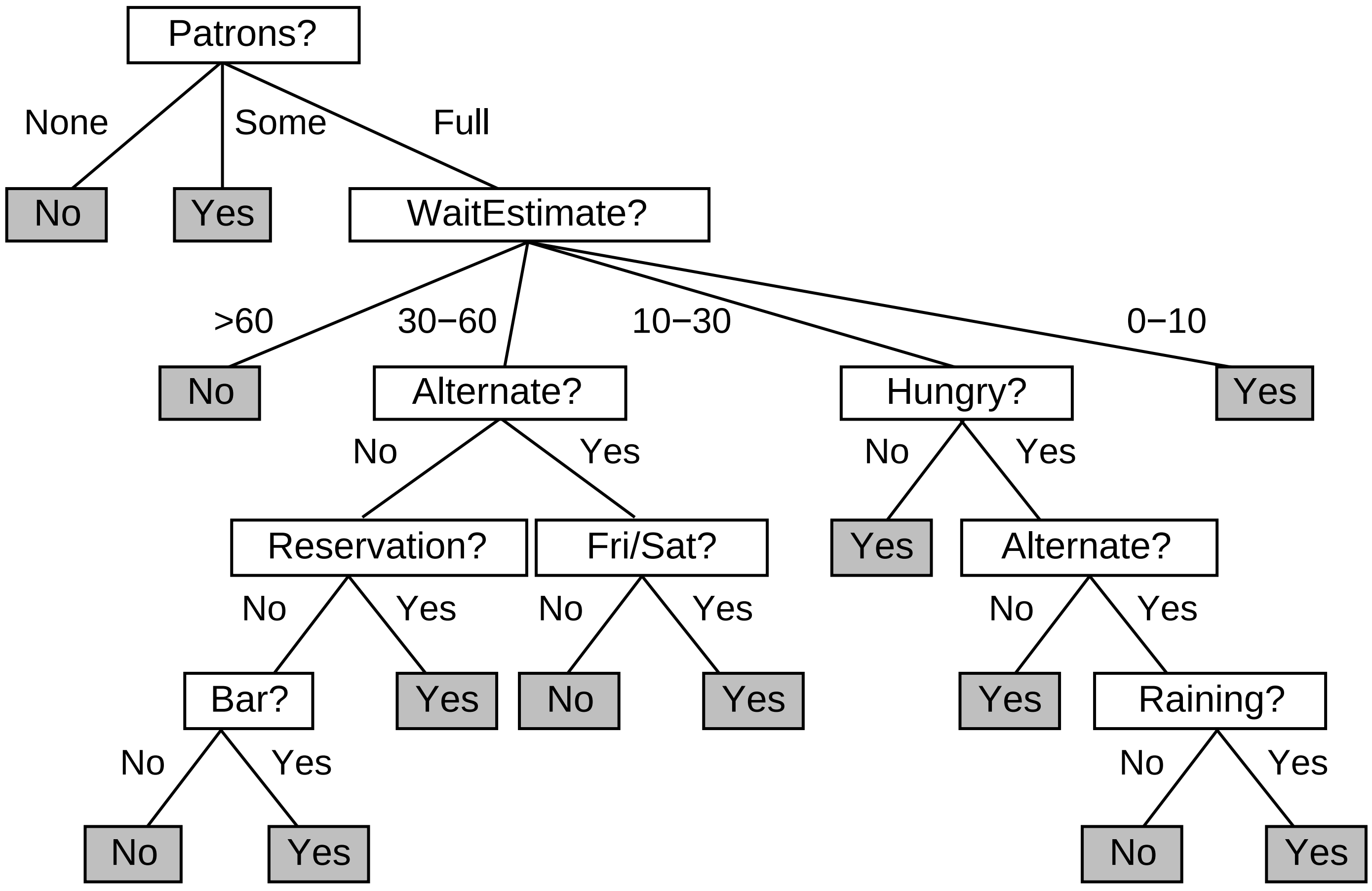
— what factors are necessary to make a rational decision

For example:

1. alternative nearby?
2. bar?
3. Friday/Saturday?
4. hungry?
5. patrons?
6. price?
7. raining?
8. reservation?
9. type of food?
10. estimated wait?

Example of a decision tree $\cdots \leadsto$

4.1 Decision Trees as Performance Elements



Choice of attributes is *critical* $\Rightarrow$ determines whether an appropriate function can be learned ("garbage-in, garbage-out")

No matter how good a learning algorithm is, it will fail if appropriate features cannot be identified (cf. neural nets)

In real world problems feature selection is often the hardest task!

(black art?)

4.1 Decision Trees as Performance Elements

Expressiveness of Decision Trees

- limited to propositional (boolean) problems
  - cannot input arbitrary number of restaurants and ask to choose
  - cannot handle continuous information
- fully expressive within class of propositional problems
- may be exponentially large w.r.t. no. inputs

Number of trees (size of hypothesis space)

$$\begin{array}{c} n \text{ attributes} \\ \Downarrow \\ 2^n \text{ combinations of inputs} \\ \Downarrow \\ 2^{2^n} \text{ different functions} \end{array}$$

eg. 6 attributes, approx. 2×10^{19} functions to choose from

$\Rightarrow$ lots of work for learning element!

4.2 Decision Tree Induction

Terminology...

example — $(\{attributes\}, value)$
positive example — $value = true$
negative example — $value = false$
training set — set of examples used for learning

Example training set:

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*

Goal: find a decision tree that agrees with all the examples in the training set

4.2 Decision Tree Induction

Trivial Solution

Branch on each attribute in turn, until you reach a distinct leaf for each example.

Problems

- tree is bigger than needed — does not find *patterns* that "summarise" or "simplify" information
- cannot answer for examples that haven't been seen ⇒ cannot *generalise*

Two sides of the same coin!

> *Ockham's razor: the most likely hypothesis is the simplest one that is consistent with the examples*

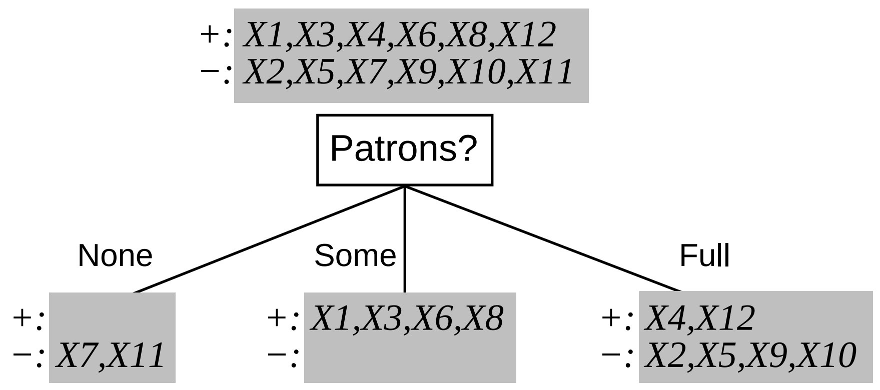
Finding smallest tree is intractable, but can use heuristics to generate reasonably good solution.

Basic idea: recursively select the most "important", or *discriminating*, attribute for successive branch points

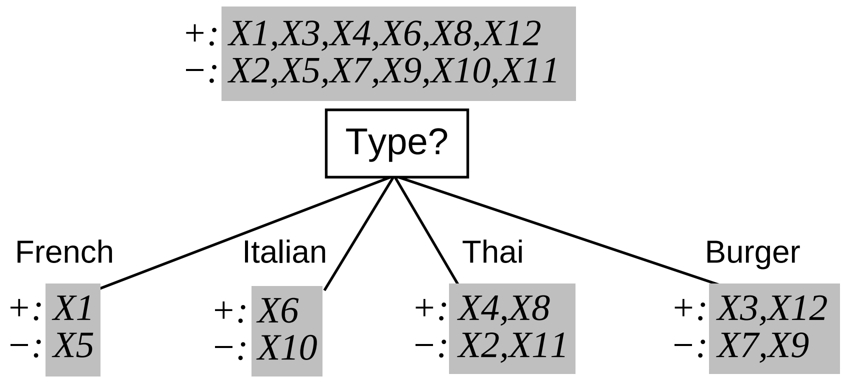
(*discriminating* — formally requires *information theory*, but humans do it intuitively ⇒ sort the "men from the boys", "sheep from the lambs", "oats from the chaff"...)

4.2 Decision Tree Induction

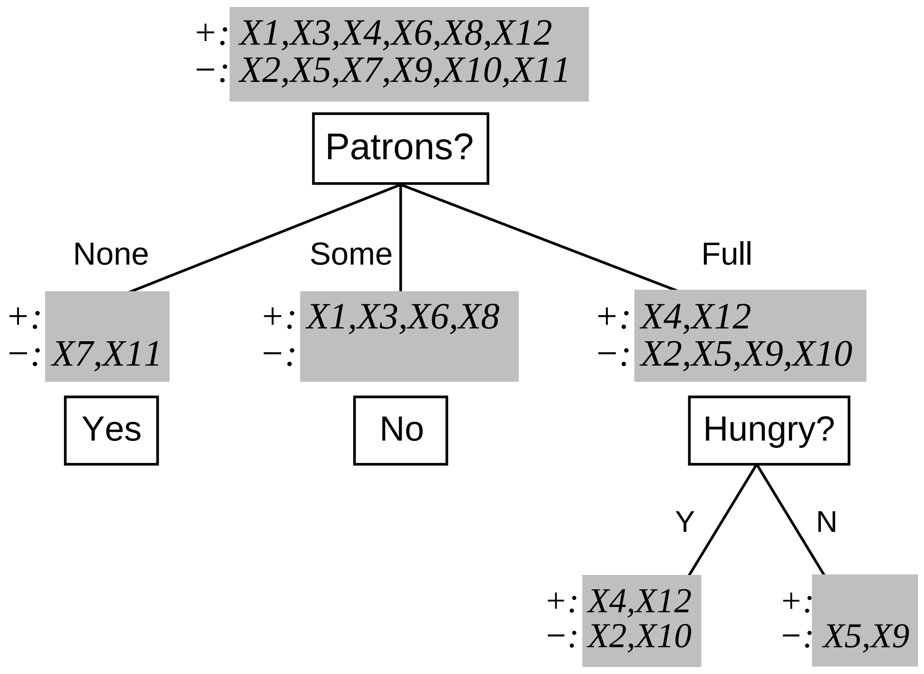
(a)



(b)



(c)



4.2 Decision Tree Induction

Recursive algorithm

Base cases

- remaining examples all +ve or all −ve — stop and label “yes” or “no”
- no examples left — no relevant examples have been seen, return majority in parent node
- no attributes left — problem: either data is inconsistent (“noisy”, or nondeterministic) or attributes chosen were insufficient to adequately discriminate (start again, or use majority vote)

Recursive case

- both +ve and −ve examples — choose next most discriminating attribute and repeat

4.2 Decision Tree Induction

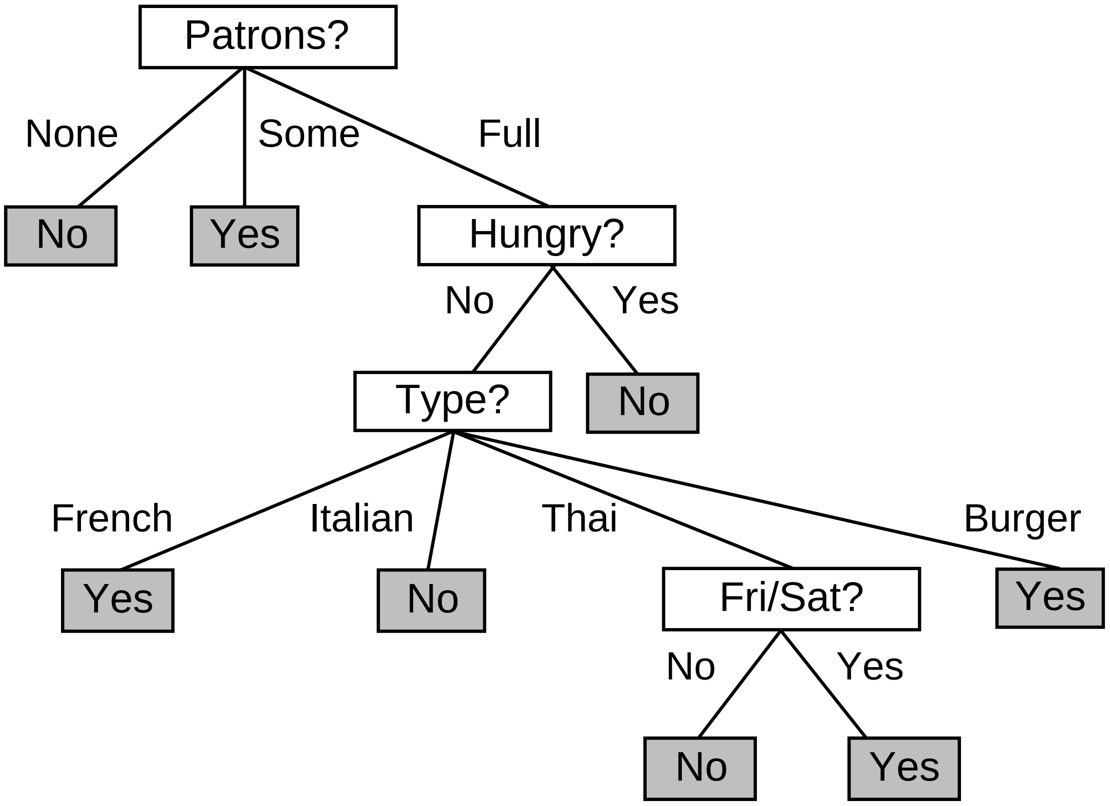
Algorithm...

```
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 ← CHOOSE-ATTRIBUTE(attributes, examples)
          tree ← a new decision tree with root test best
          for each value $v_i$ of best do
                $examples_i$ ← {elements of examples with best = $v_i$}
                subtree ← 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
```

4.2 Decision Tree Induction

Applied to our training set gives:



Note: different to original tree, despite using data generated from that tree. Is the tree (hypothesis) wrong?

- *No — with respect to seen examples!*
  In fact more concise, and highlights new patterns.
- *Probably — w.r.t. unseen examples. . .*

4.3 Assessing Performance

good hypothesis = predicts the classifications of unseen examples

To assess performance we require further examples with known outcomes — *test set*

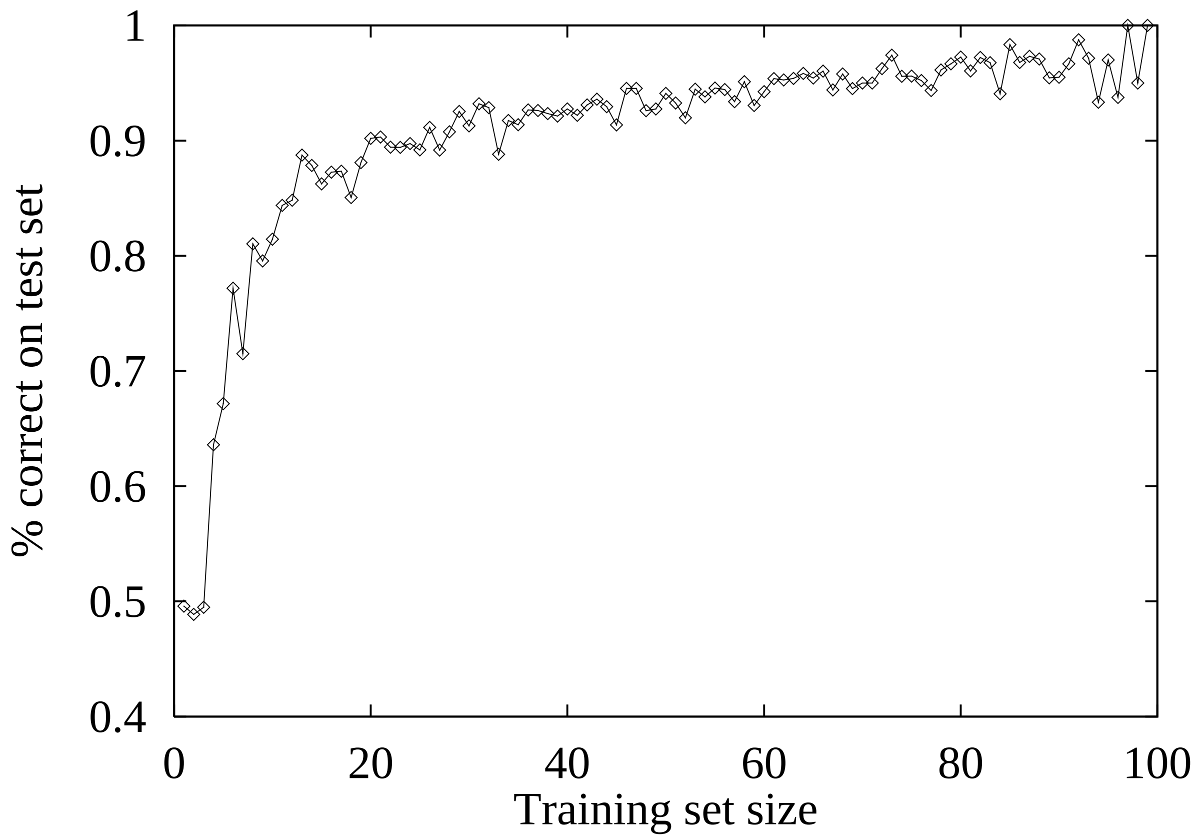
Usual methodology:

1. Collect large set of examples.
2. Divide into two disjoint sets: *training set* and *test set*.
3. Apply learning algorithm to training set to generate hypothesis H.
4. Measure performance of H (% correct) on test set.
5. Repeat for different training sets of different sizes.

Plot prediction quality against training set size. . .

4.3 Assessing Performance

Learning curve (or “Happy Graph”)



4.4 Practical Uses of Decision-Tree Learning

Despite representational limitations, decision-tree learning has been used successfully in a wide variety of applications. eg. . .

Designing Oil Platform Equipment

GASOIL — BP, deployed 1986 (Michie)

— designs complex gas-oil separation systems for offshore oil platforms

— attributes include relative proportions of gas, oil and water, flow rate, pressure, density, viscosity, temperature and susceptibility to waxing

— largest commercial expert system in the world at that time

— approx. 2500 rules

— building by hand would have taken ~ 10 person-years

— descision-tree learning applied to database of existing designs

$\Rightarrow$ developed in 100 person-days

— outperformed human experts

— said to have saved BP millions of dollars!

4.4 Practical Uses of Decision-Tree Learning

Learning to Fly

C4.5 — 1992 (Sammut *et al*)

— one approach — learn correct mapping from state to action

— Cessna on flight simulator

— training: 3 skilled human pilots, assigned flight plan, 30 times each

— training example for each action taken by pilot $\Rightarrow$ 90000 examples

— 20 state variables

— decision tree generated and fed back into simulator to fly plane

— Results: flies *better* than its teachers!

$\Rightarrow$ generalisation process "cleans out" mistakes by teachers

The End