Chapter 18 – Learning goals

Being familiar with:

- **Motivation** for learning
- **Decision tree** formalism
- Decision tree **learning**
- **Information Gain** for structuring model learning
- **Overfitting** – and what to do to avoid it

Learning

This is the second part of the course:

- We have learned about **representations** for uncertain knowledge
- **Inference** in these representations
- Making **decisions** based on the inferences

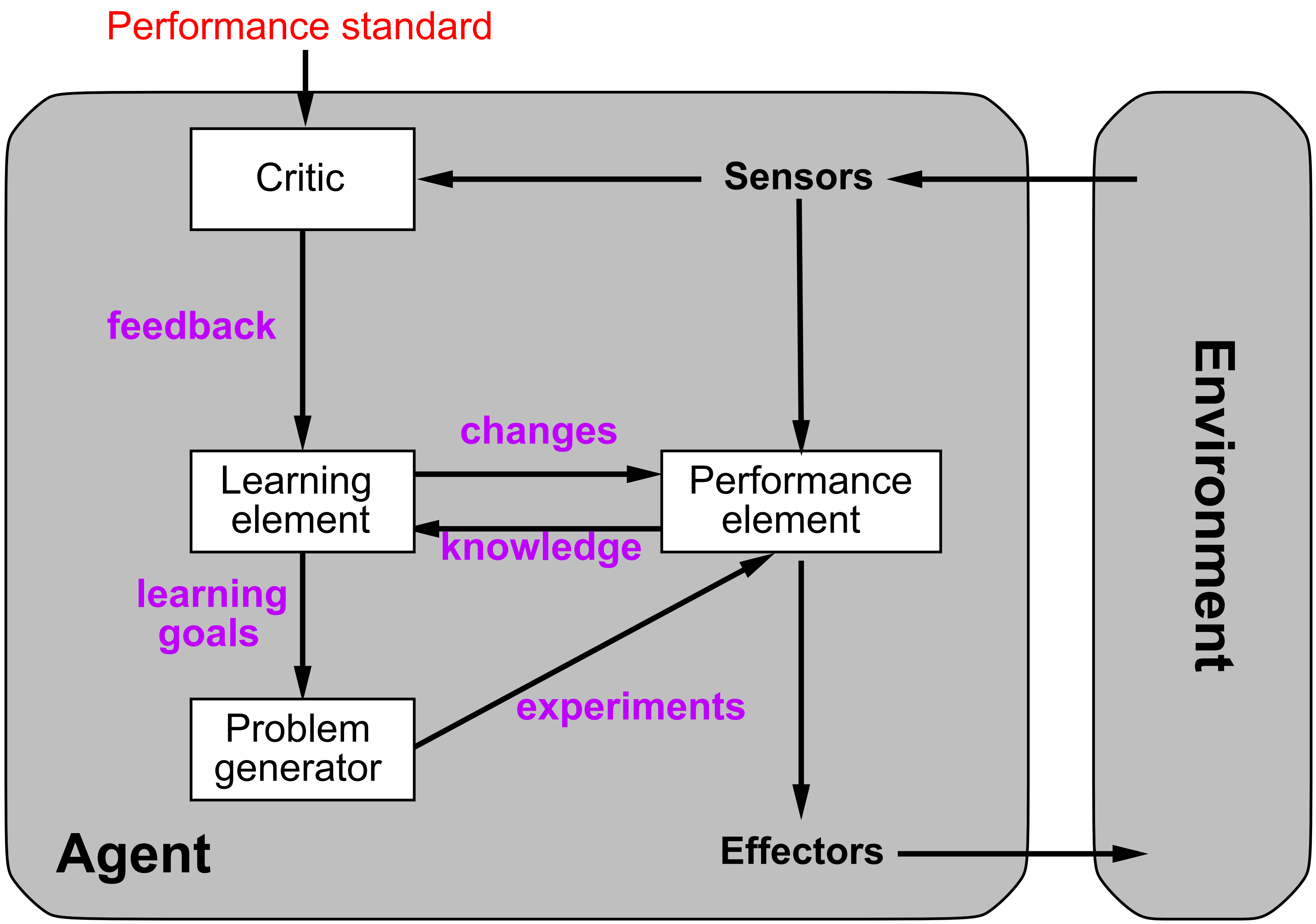
Now we will talk about learning the representations:

- Decision trees
- Instance-based learning/Case-based reasoning
- Artificial Neural Networks
- Reinforcement learning

Why do Learning?

- Learning is essential for **unknown environments**, i.e., when designer lacks omniscience
- Learning is useful as a **system construction** method, i.e., expose the agent to reality rather than trying to write it down
- Learning **modifies the agent's decision mechanisms** to improve performance

Learning agents



Learning element

Design of learning element is dictated by...

- what type of performance element is used
- which functional component is to be learned
- how that functional component is represented
- what kind of feedback is available

Learning element

Design of learning element is dictated by. . .

- what type of performance element is used
- which functional component is to be learned
- how that functional component is represented
- what kind of feedback is available

Example scenarios:

Performance element	Component	Representation	Feedback
Alpha−beta search	Eval. fn.	Weighted linear function	Win/loss
Logical agent	Transition model	Successor−state axioms	Outcome
Utility−based agent	Transition model	Dynamic Bayes net	Outcome
Simple reflex agent	Percept−action fn	Neural net	Correct action

Supervised learning: correct answers for each instance
Reinforcement learning: occasional rewards

Inductive learning

Simplest form: Learn a function from examples

f is the **target function**

An **example** is a pair $\{x, f(x)\}$, e.g., $\left\{ \begin{array}{c|c|c} \mathrm{O} & \mathrm{O} & \mathrm{X} \\ \hline & \mathrm{X} & \\ \hline \mathrm{X} & & \end{array} , \quad +1 \right\}$

Problem:
Find **hypothesis** $h \in H$ s.t. $h \approx f$ given a **training set** of examples

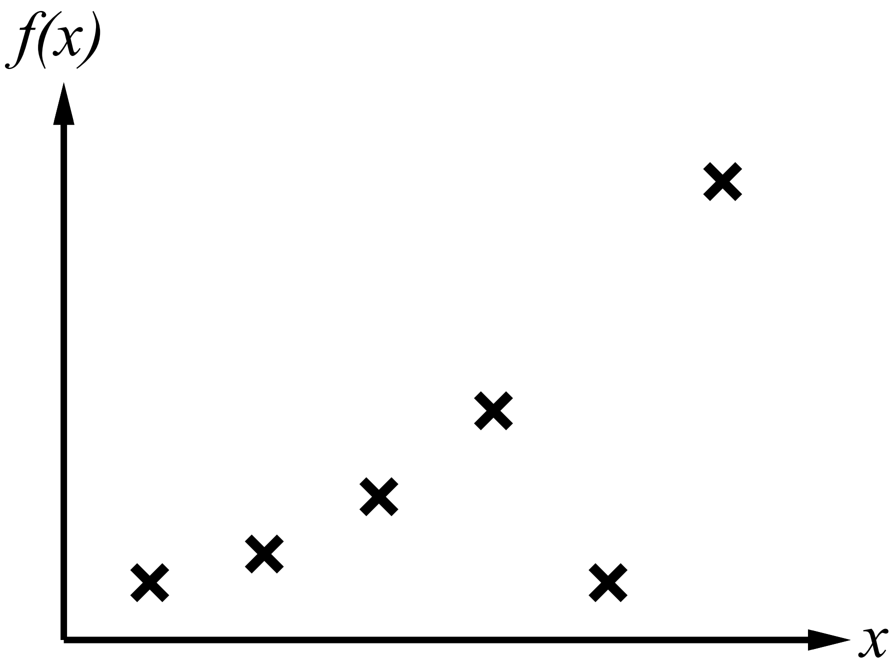
This is a highly simplified model of real learning:

- Ignores **prior knowledge**
- Assumes a **deterministic**, **observable** "environment"
- Assumes examples are **given**

Inductive learning method

- Construct/adjust h to agree with f on training set
- h is **consistent** if it agrees with f on all examples

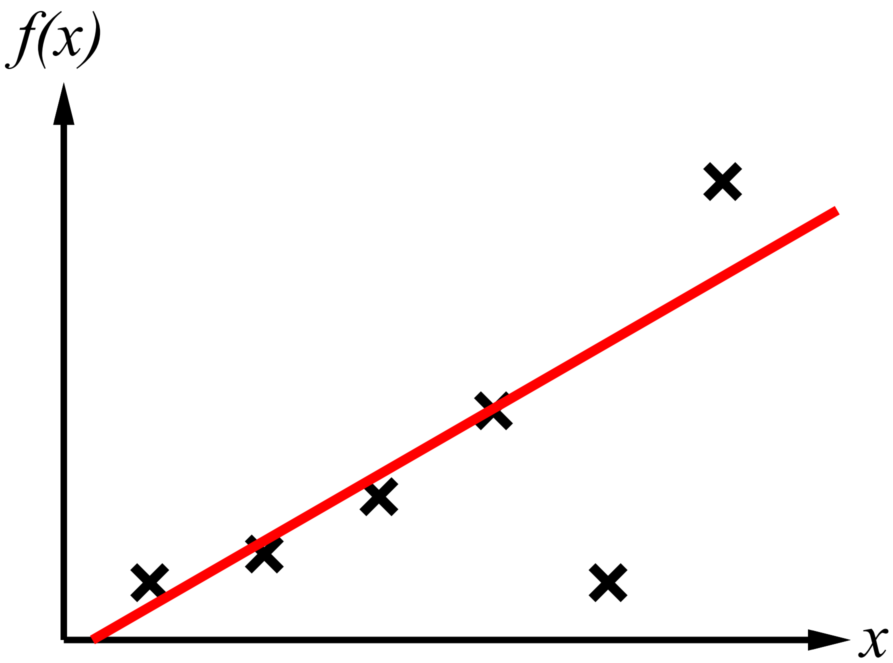
Example – curve fitting:



Inductive learning method

- Construct/adjust h to agree with f on training set
- h is **consistent** if it agrees with f on all examples

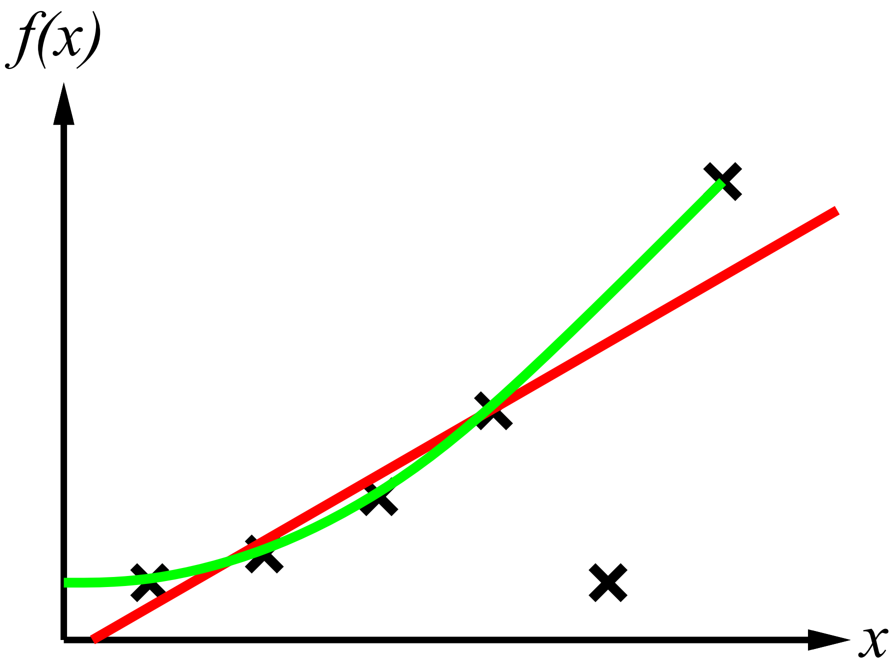
Example – curve fitting:



Inductive learning method

- Construct/adjust h to agree with f on training set
- h is **consistent** if it agrees with f on all examples

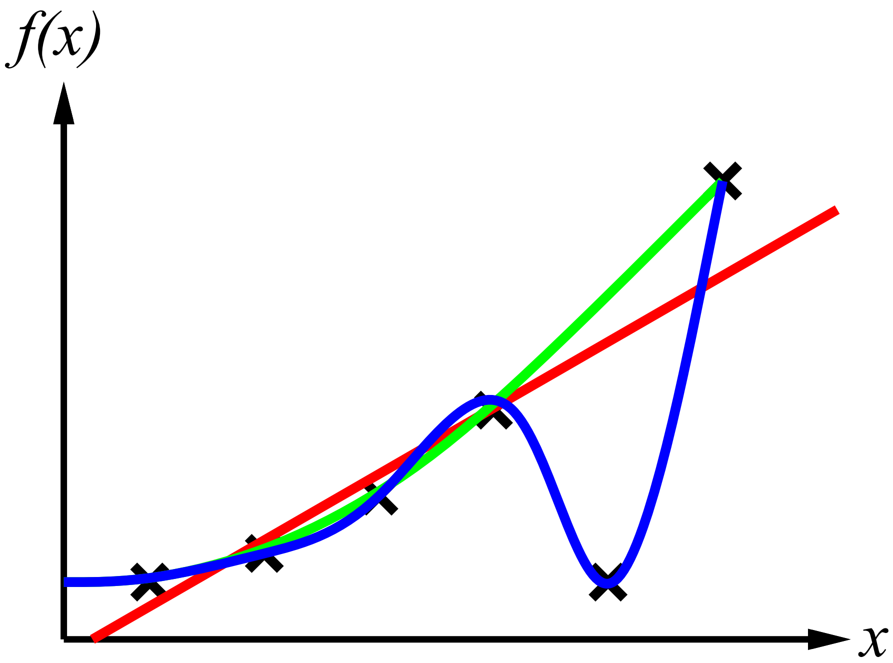
Example – curve fitting:



Inductive learning method

- Construct/adjust h to agree with f on training set
- h is **consistent** if it agrees with f on all examples

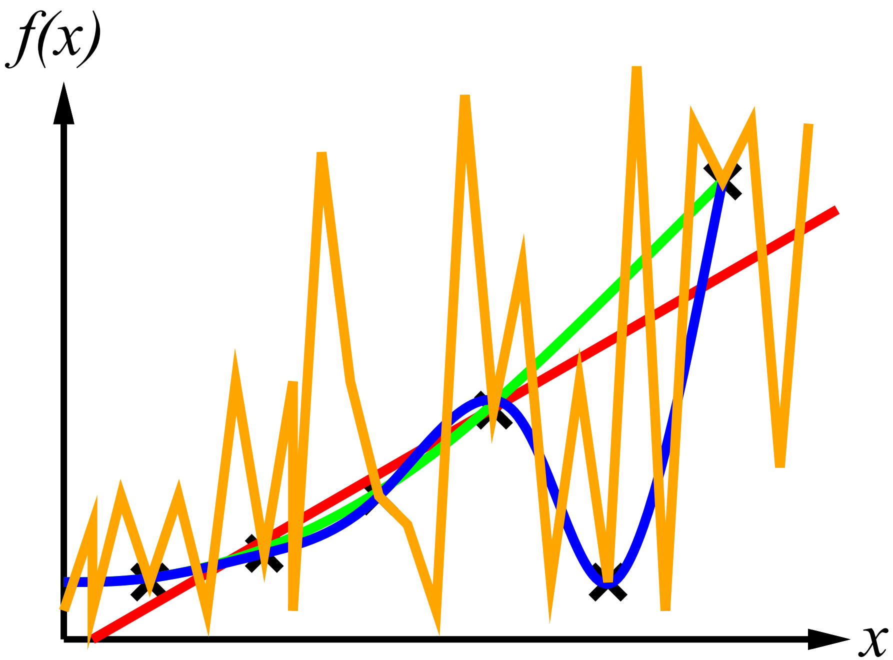
Example – curve fitting:



Inductive learning method

- Construct/adjust h to agree with f on training set
- h is **consistent** if it agrees with f on all examples

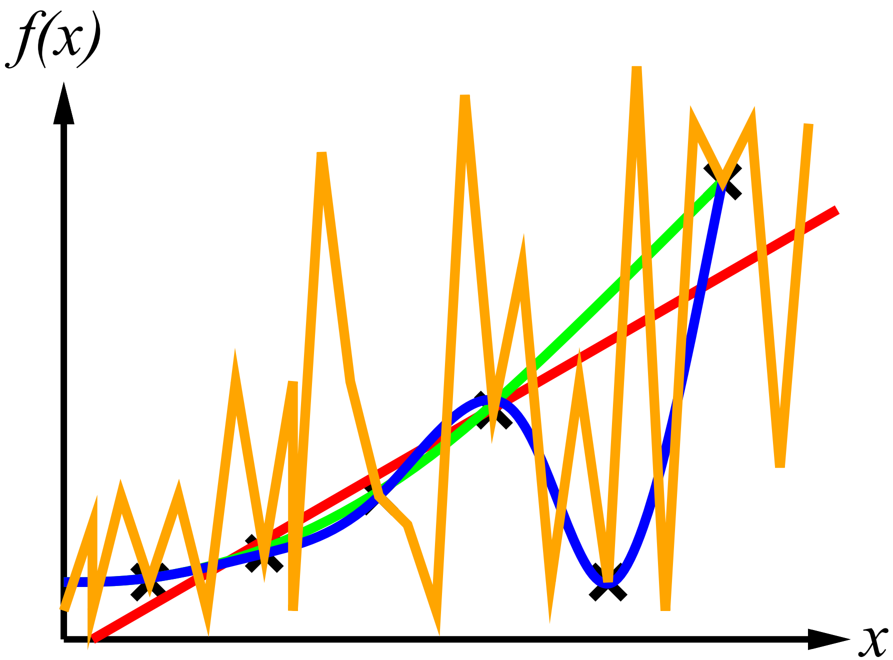
Example – curve fitting:



Inductive learning method

- Construct/adjust h to agree with f on training set
- h is **consistent** if it agrees with f on all examples

Example – curve fitting:



Ockham's razor: maximize consistency and simplicity!

Attribute-based representations

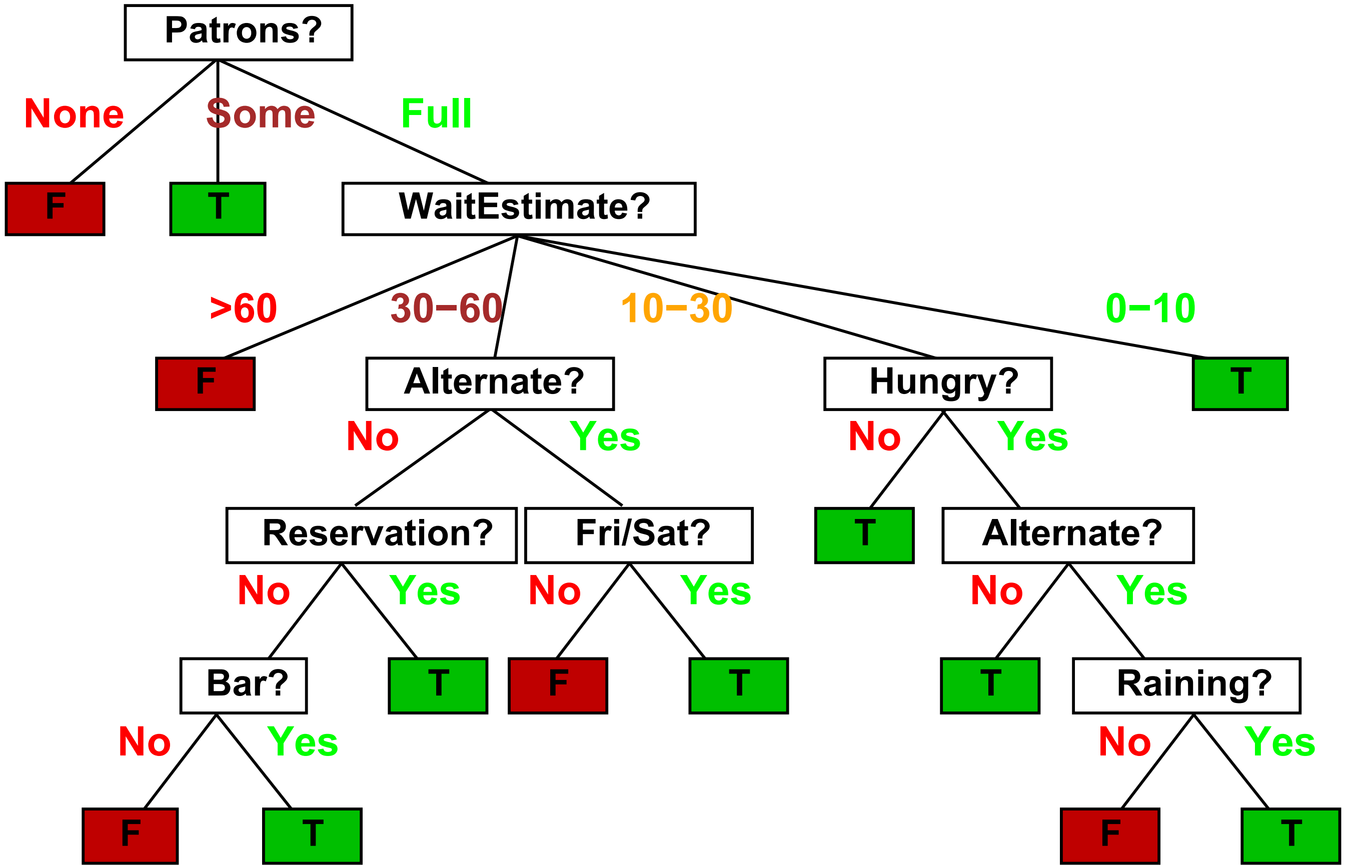
Examples described by **attribute values** (Boolean, discrete, continuous, etc.)
E.g., situations where the authors will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	**WillWait**
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*T*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

Classification of examples is **positive** (T) or **negative** (F)

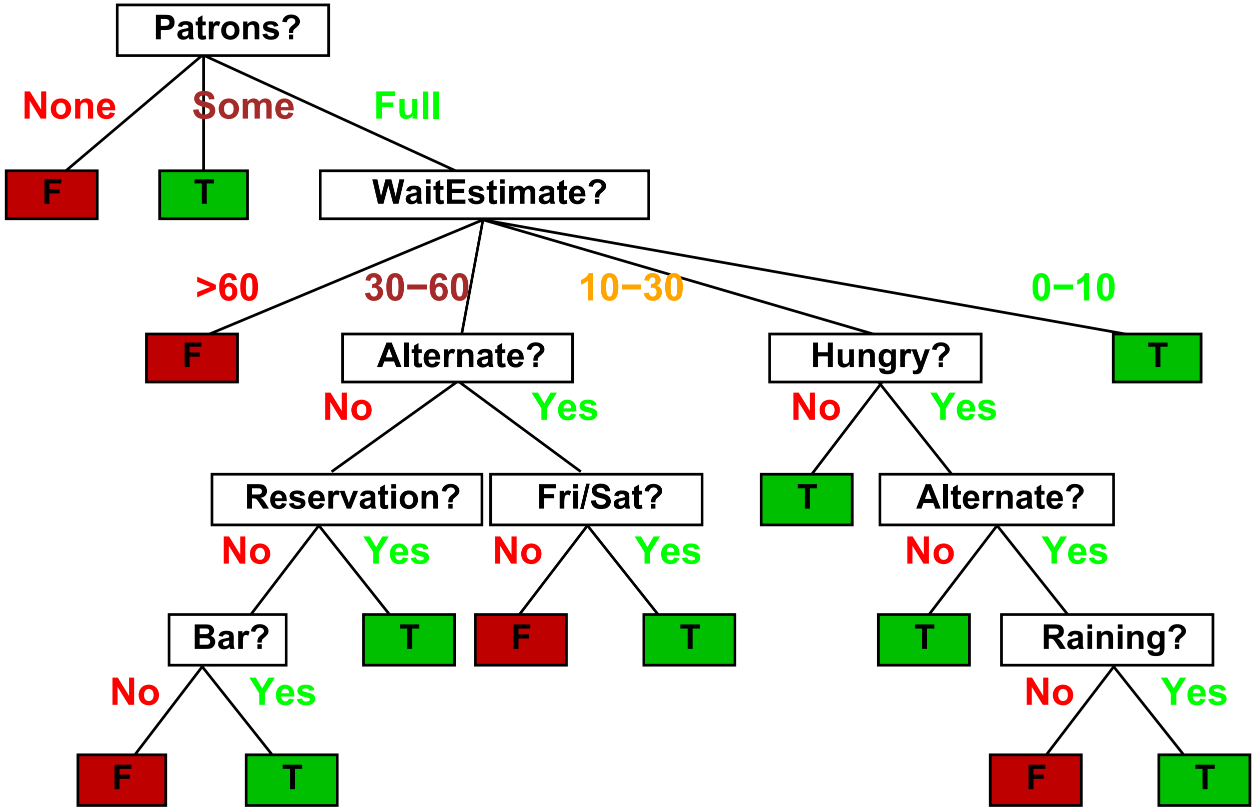
Decision trees

One possible representation for hypotheses: Decision Trees



Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	WillWait
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

Converting a decision-tree to rules



IF $\quad$ (Patrons?=Full) $\wedge$ (WaitEstimate?=0-10)
THEN $\quad$ Wait? = True

IF $\quad$ (Patrons?=Full) $\wedge$ (WaitEstimate?=30-60) $\wedge$ (Alternate?=Yes) $\wedge$ (Fri/Sat?=No)
THEN $\quad$ Wait? = False

...

Decision tree representation

Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

Class task: How would we represent these with decision trees?

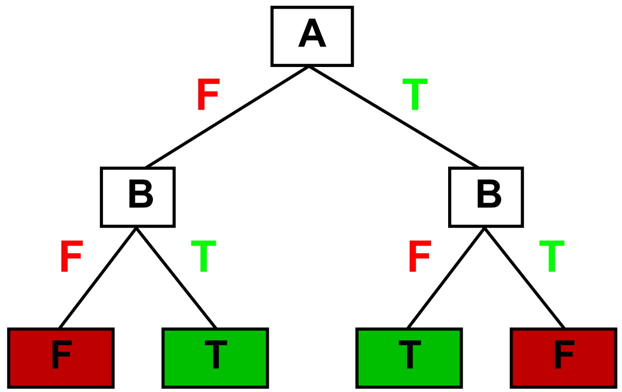
- $A \wedge B$, $A \vee B$, $A \texttt{ XOR } B$.
- $(A \wedge B) \vee (A \wedge \neg B \wedge C)$
- m of n: At least m of $A_1, A_2, \ldots, A_n$ (try $n = 3$, $m = 2$).

Expressiveness

Decision trees can express any function of the input attributes.

E.g., for Boolean functions, truth table row $\longrightarrow$ path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



There is a **consistent** decision tree for any training set w/ one path to leaf for each example (unless f nondeterministic in x)

. . . but it **probably won't generalize to new examples**

Prefer to find more **compact** decision trees

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

E.g., with 6 Boolean attributes, there are
$18,446,744,073,709,551,616$ trees

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions

= number of distinct truth tables with 2^n rows $= 2^{2^n}$

E.g., with 6 Boolean attributes, there are
$18,446,744,073,709,551,616$ trees

How many purely conjunctive hypotheses (e.g., $\texttt{hungry} \land \neg\texttt{rain}$)??

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

E.g., with 6 Boolean attributes, there are
$18,446,744,073,709,551,616$ trees

How many purely conjunctive hypotheses (e.g., $\texttt{hungry} \wedge \neg\texttt{rain}$)??

Each attribute can be in (positive), in (negative), or out

$\Rightarrow$ 3^n distinct conjunctive hypotheses

E.g., with 6 Boolean attributes, there are 729 rules of this kind

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

E.g., with 6 Boolean attributes, there are
$18,446,744,073,709,551,616$ trees

How many purely conjunctive hypotheses (e.g., `hungry` $\wedge \neg$`rain`)??

Each attribute can be in (positive), in (negative), or out

$\Rightarrow$ 3^n distinct conjunctive hypotheses

E.g., with 6 Boolean attributes, there are 729 rules of this kind

More expressive hypothesis space:

- increases chance that target function can be expressed
- increases number of hypotheses consistent w/ training set
  $\Rightarrow$ may get worse predictions

When to Consider Decision Trees

- Instances describable by attribute–value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:

- Equipment or medical diagnosis
- Credit risk analysis
- Classifying email as spam or ham

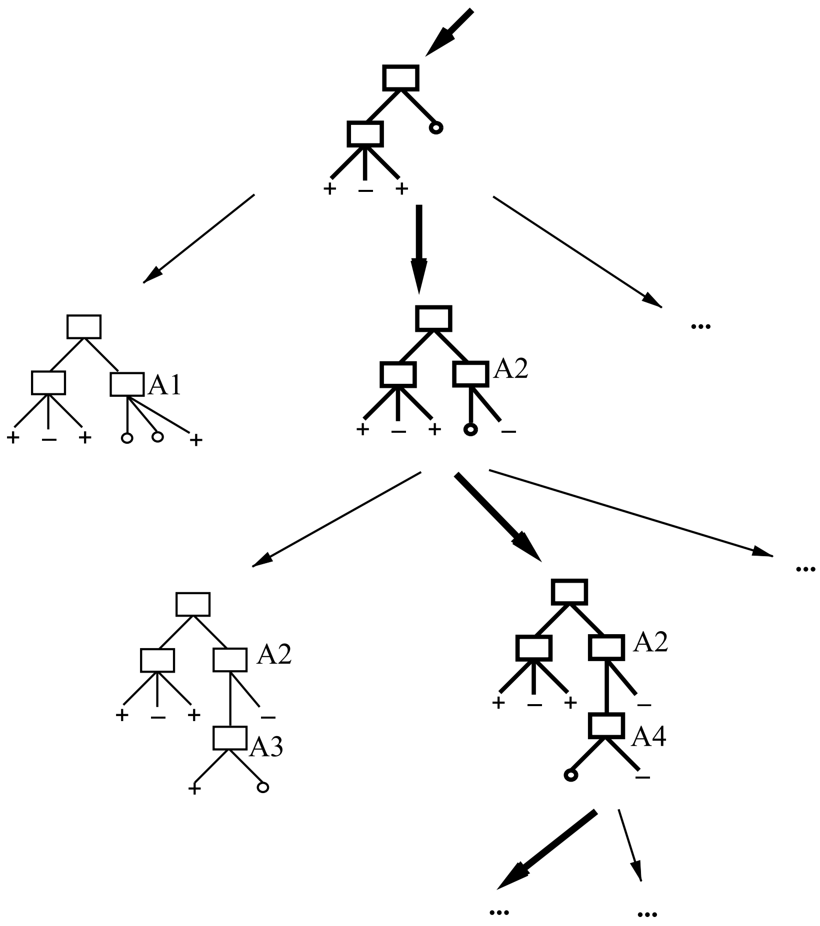
Decision tree learning

Aim: find a small tree consistent with the training examples
Idea: (recursively) choose "best" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a DT

    if examples is empty then return default
    else if all examples have same class then return class
    else if attributes is empty then return Mode(examples)
    else
        best ← Choose-Attribute(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            ex_i ← {elements of examples with best = v_i}
            subtree ← DTL(ex_i, attributes − best, Mode(examples))
            add a branch to tree with label v_i and subtree subtree
    return tree
```

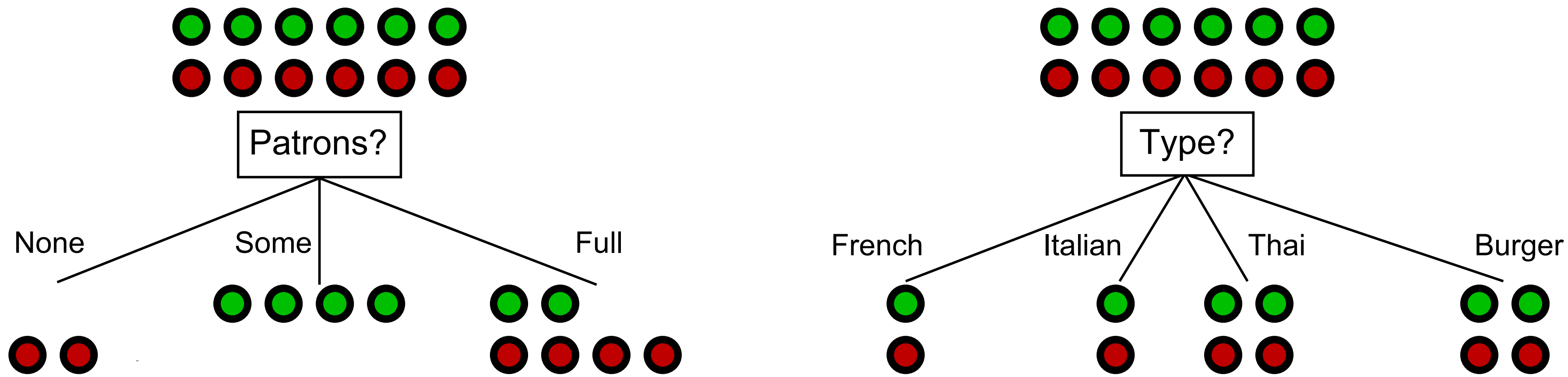
Search in the hypothesis space



DEMO: Random selection

Choosing an attribute

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



`Patrons?` is a better choice – gives **information** about the classification

Information

Information **answers questions**!

The more clueless we are about the answer initially, the more information is contained in the answer:

Scale: 1 bit = answer to Boolean question with prior $\langle 0.5, 0.5 \rangle$

Information in an answer when prior is $\langle P_1, \ldots, P_n \rangle$ is

$$H(\langle P_1, \ldots, P_n \rangle) = \Sigma_{i=1}^{n} - P_i \log_2 P_i$$

(also called **entropy** of the prior $\langle P_1, \ldots, P_n \rangle$)

Information contd.

Suppose we have p positive and n negative examples at root

- $H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify new example
  E.g., for 12 restaurant examples, $p=n=6$ so we need 1 bit

Information contd.

Suppose we have p positive and n negative examples at root

- $H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify new example
  E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit

An attribute splits the examples E into subsets E_i, we hope each needs less information to classify...

Let E_i have p_i positive and n_i negative examples:

- $H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify
  $\Rightarrow$ **expected** number of bits per example over all branches is

$$\sum_i \frac{p_i + n_i}{p + n} H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$$

For `Patrons?`, this is 0.459 bits, for `Type` this is (still) 1 bit

Information contd.

Suppose we have p positive and n negative examples at root

- $H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify new example
  E.g., for 12 restaurant examples, $p=n=6$ so we need 1 bit

An attribute splits the examples E into subsets E_i, we hope each needs less information to classify...

Let E_i have p_i positive and n_i negative examples:

- $H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify
  $\Rightarrow$ **expected** number of bits per example over all branches is

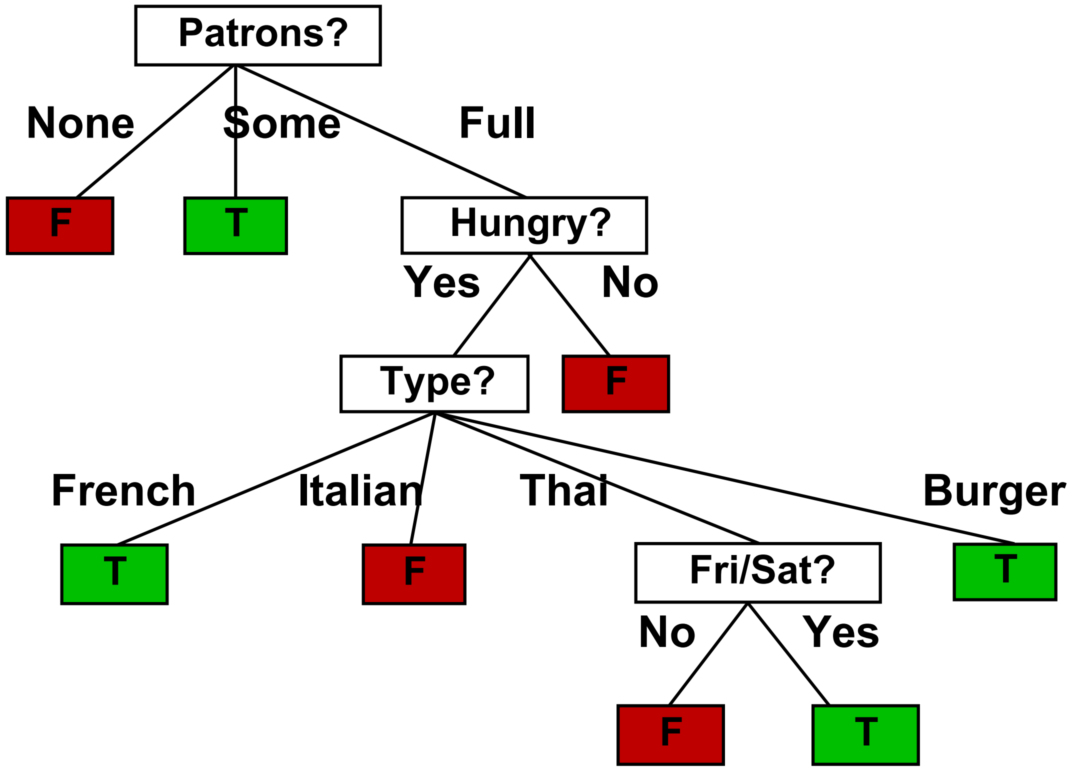
$$\sum_i \frac{p_i+n_i}{p+n} H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$$

For `Patrons?`, this is 0.459 bits, for `Type` this is (still) 1 bit

Heuristic: Choose the attribute that minimizes the **remaining** information needed to classify new example

Example contd.

Decision tree learned from the 12 examples:



Substantially **simpler than "true" tree** – a more complex hypothesis isn't justified by **small amount of data**

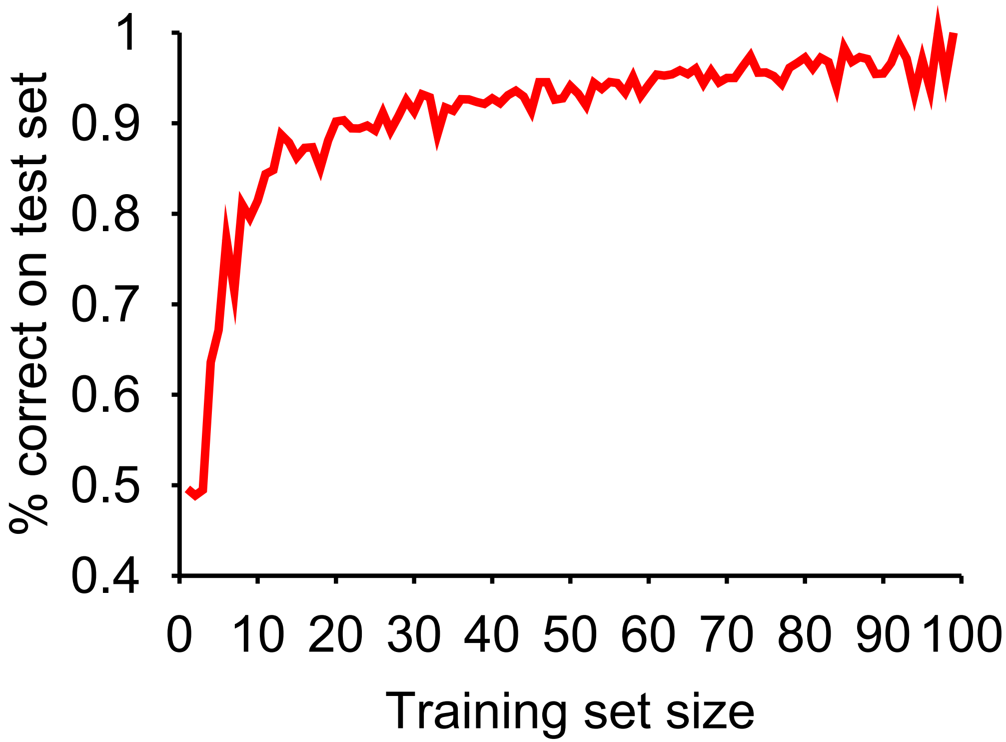
DEMO: Gain selection

Performance measurement

How do we know that $h \approx f$?

Try h on a new **test set** of examples
(use **same distribution over example space** as training set)

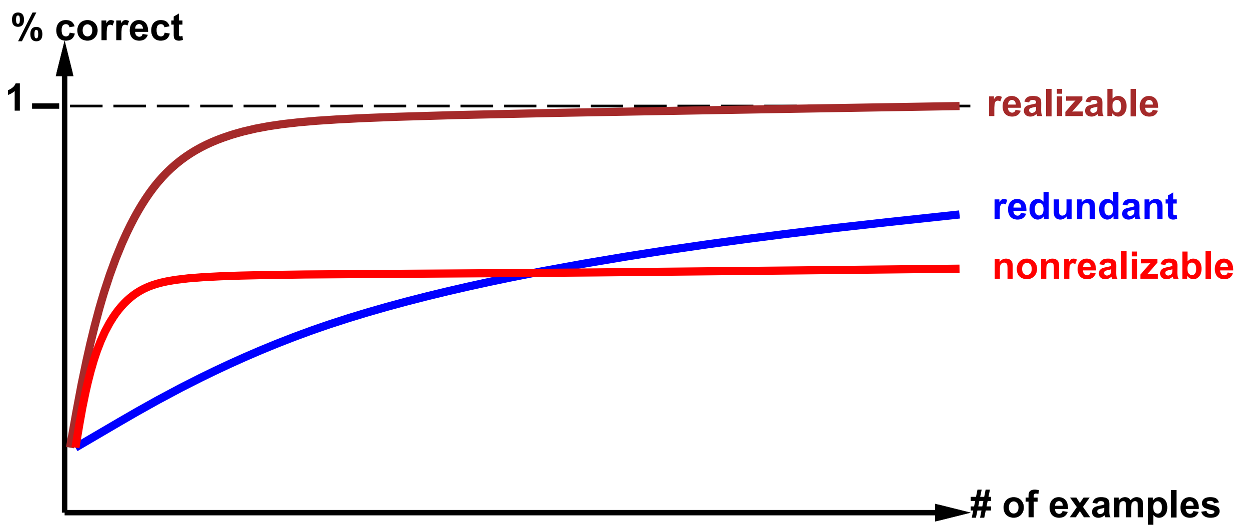
Learning curve = % correct on test set as a function of training set size



Performance measurement contd.

Learning curve depends on...

- **realizable** (can express target function) vs. **non-realizable**
- **non-realizability** can be due to missing attributes or restricted hypothesis class (e.g., thresholded linear function)
- **redundant expressiveness** (e.g., loads of irrelevant attributes)

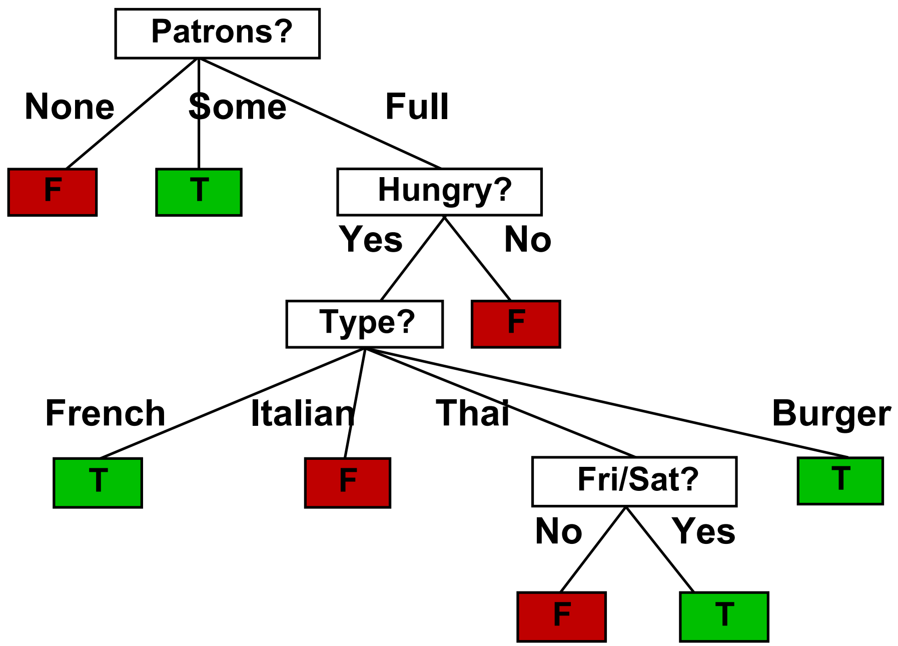


Overfitting in decision trees

Consider adding noisy training examples X_{13} and X_{14}:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	**WillWait**
X_{13}	*F*	*T*	*T*	*T*	*Some*	*$$*	*F*	*T*	*French*	*0–10*	**F**
X_{14}	*F*	*T*	*T*	*T*	*Some*	*$*	*F*	*T*	*Thai*	*0–10*	**F**

Class task:
What is the effect on the tree we learned earlier?



Overfitting

Consider error of hypothesis h over

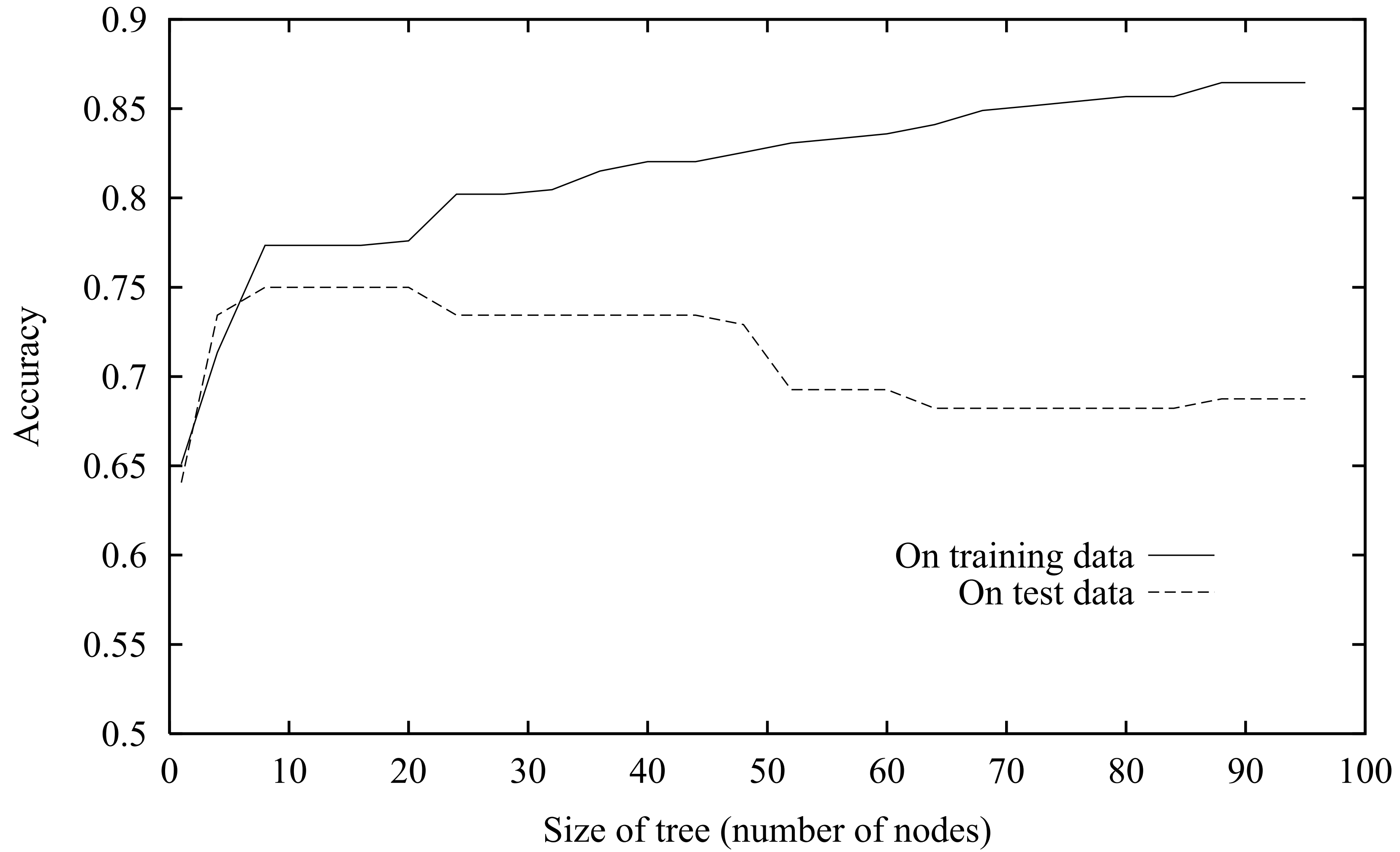
- Training data: $\text{error}_t(h)$
- Entire distribution $\mathcal{D}$ of data (often approximated by measurement on test-set): $\text{error}_{\mathcal{D}}(h)$

Overfitting

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$\text{error}_t(h) < \text{error}_t(h') \text{ and } \text{error}_{\mathcal{D}}(h) > \text{error}_{\mathcal{D}}(h')$$

Overfitting (cont'd)



Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data (statistical tests needed)
- Measure performance over separate validation data set

Reduced-Error Pruning

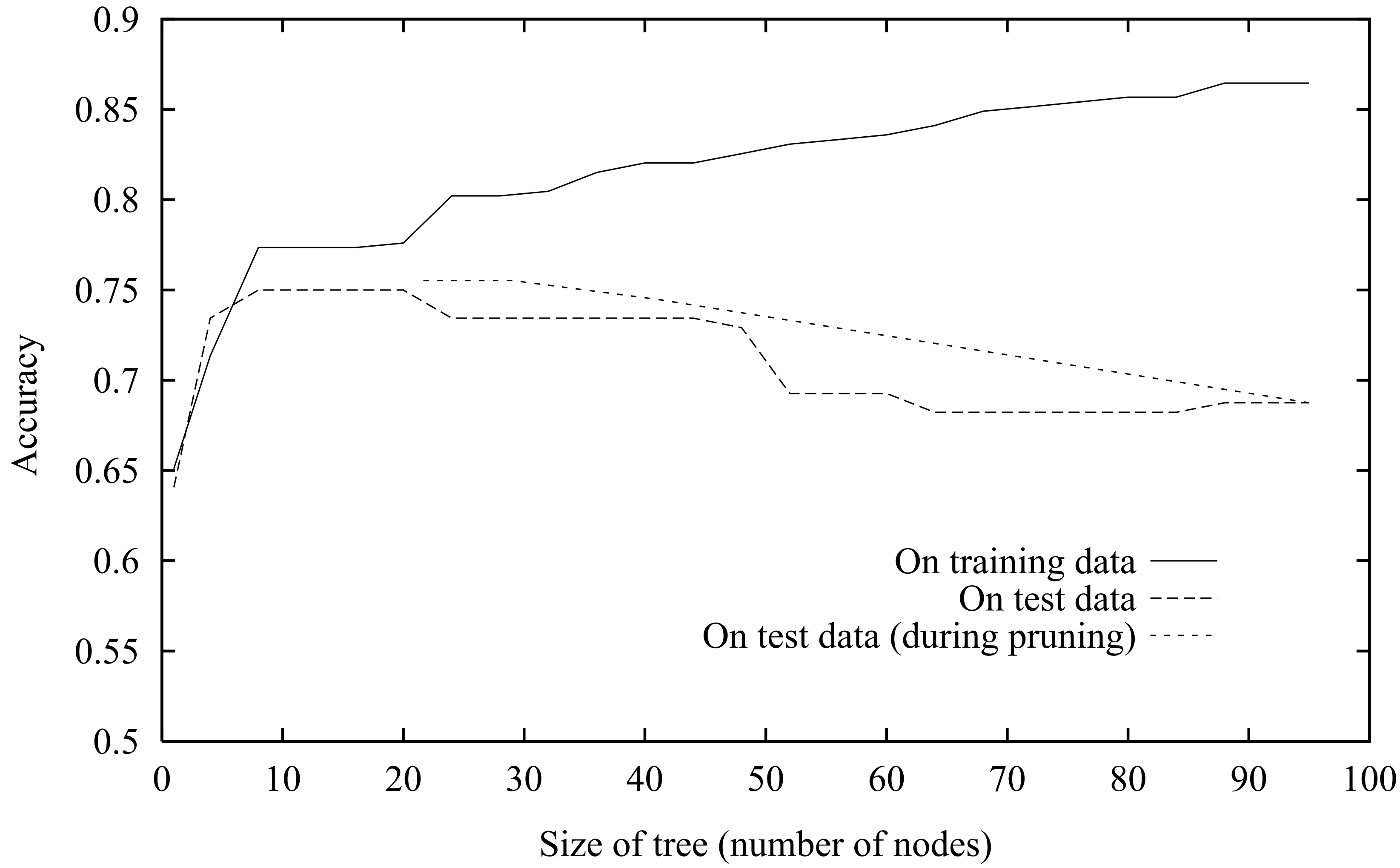
Split data into ***training*** and ***validation*** set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

$\Rightarrow$ Produces smallest version of most accurate subtree

Effect of Reduced-Error Pruning



Summary

- Learning needed for **unknown environments**, **lazy designers**
- **Learning agent = performance element + learning element**
- Learning method depends on **type of performance element**, available **feedback**, type of **component to be improved**, and its **representation**
- For **supervised learning**, the aim is to find a simple hypothesis that is approximately consistent with training examples
- Decision tree learning using **information gain**
- **Learning performance = prediction accuracy measured on test set**