

Artificial Intelligence

V. Machine Learning

1. Decision Tree Learning

F. C. Langbein

School of Computer Science
and Informatics
Cardiff University



1.4

Overview

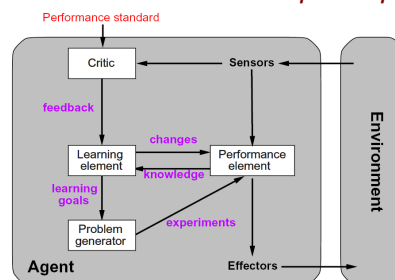
- Machine Learning
 - Supervised, unsupervised and reinforcement learning
 - Inductive learning
 - Overfitting
- Decision Tree Learning
 - Entropy
- Learning Curves
 - Realisability
 - Redundant expressiveness

F. C. Langbein, Artificial Intelligence – V. Machine Learning; 1. Decision Tree Learning

1

Machine Learning

- Essential for *unknown environments*
 - when designer lacks omniscience
- Useful as a system *construction method*
 - expose agent to reality instead of trying to write it down
- *Modifies* decision mechanism to *improve performance*



F. C. Langbein, Artificial Intelligence – V. Machine Learning; 1. Decision Tree Learning

2

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

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

- Classify input based on feedback

Types of Learning

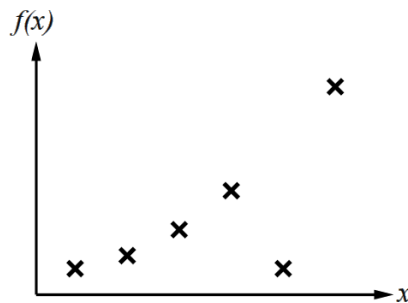
- **Supervised learning**
 - Learn input/output patterns from given, correct output for some inputs
 - Split data into **distinct** *test* and *training* set
 - Train element on training data
 - Evaluate performance on test data
- **Unsupervised learning**
 - Learn input/output patterns without known output
- **Reinforcement learning**
 - Learn from occasional rewards
 - Sub-problem: learn how the environment works

Inductive Learning (a.k.a. Science)

- Simplest form: learn a function *from examples*
 - f is the target function
 - An example is a pair $(x, f(x))$, e.g. $\left(\begin{array}{c|c|c} O & O & X \\ \hline & X & \\ \hline X & & \end{array} \right), +1$
 - Problem: find a **hypothesis** h such that $h \approx f$, given training set of examples
- Highly simplified model of real learning:
 - Ignores prior knowledge
 - Assumes a deterministic, observable environment
 - Assumes examples are given
 - Assumes that the agent wants to learn f — why?

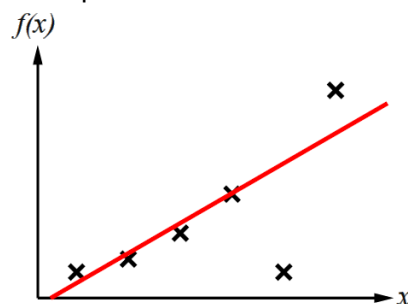
Inductive Learning Method

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



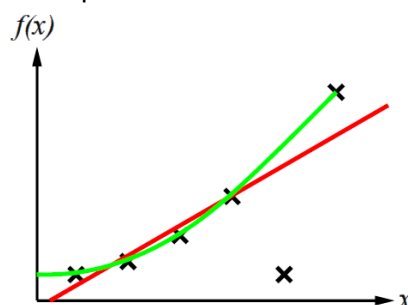
Inductive Learning Method

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



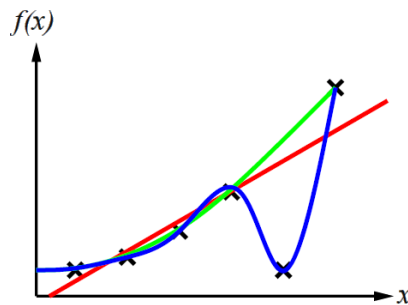
Inductive Learning Method

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



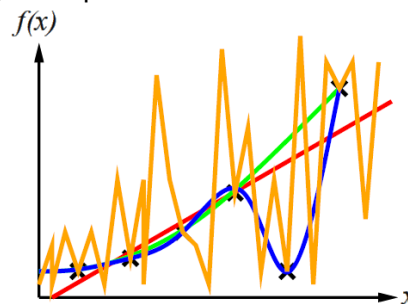
Inductive Learning Method

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



Inductive Learning Method

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



- Ockham's razor: *prefer the simplest hypothesis consistent with the data*

Attribute-based Representation

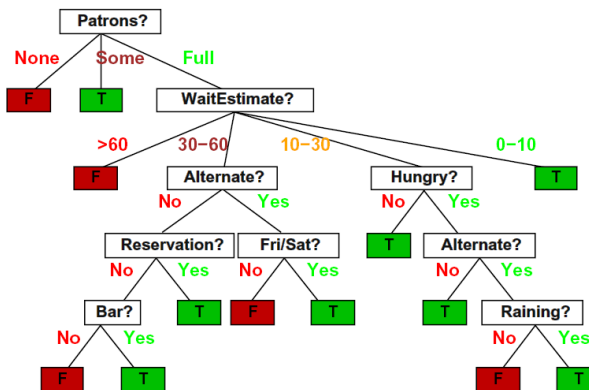
- Examples described by **attribute values**
(Boolean, discrete, continuous **feature vector** of input/world)
- E.g., situations where I will/won't wait for a table

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	\$	F	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)

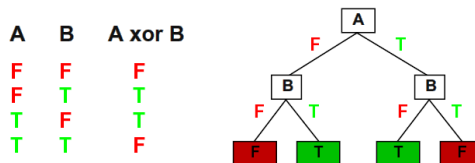
Decision Trees

- ▶ One possible representation for hypothesis
 - E.g. the “true” tree for deciding whether to wait



Decision Tree Expressiveness

- ▶ Decision trees can express any function of the input attributes
 - E.g. for Boolean features, truth table row is path to leaf



- ▶ There is a consistent decision tree for any training set
 - with one path to leaf for each example (unless $f(x)$ non-deterministic in x)
 - but it will probably not *generalise* to new examples
- ▶ Try to find more *compact* decision trees

Hypothesis Spaces

- ▶ Number of *distinct decision trees* with n Boolean attributes
 - = number of Boolean functions
 - = number of distinct truth tables with 2^n rows
 - = 2^{2^n}
- ▶ Number of purely *conjunctive hypotheses* (E.g., $\text{Hungry} \wedge \neg \text{Rain}$)
 - Each attribute can be in (positive), in (negative), or out
 - ▶ 3^n distinct conjunctive hypothesis
- ▶ E.g. for 6 Boolean attributes, there are 729 hypotheses and $18,446,744,073,709,551,616 \approx 1.8 * 10^{19}$ trees

Decision Tree Learning

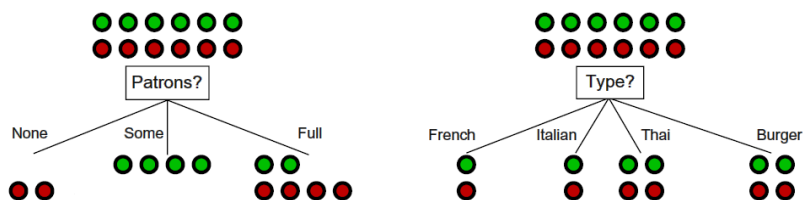
- More expressive hypothesis space than hypotheses
 - Increases chance that target function can be expressed
 - But also increases number of hypotheses consistent with training set
 - ➔ May get worse predictions
- Aim:
 - Find a small tree consistent with the training examples
- Idea:
 - Recursively choose “most significant” attribute as root of sub-tree

Decision Tree Learning Algorithm

```
function DTL(examples, attributes, default)
if empty(examples)
return default
else if HAVE_SAME_CLASSIFICATION(examples)
return CLASSIFICATION(examples)
best ← CHOOSE_ATTRIBUTE(attributes, examples)
tree ← DECISION_TREE_WITH_ROOT(best)
for  $v_1 \in \text{VALUES}(best)$ 
examples1 ← {examples with best =  $v_1$ }
subtree ← DTL(examples1, attributes - best,
MODE(examples))
tree ← ADD_SUBTREE(tree,  $v_1$ , subtree)
return tree
```

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 **more information** about the classification

Entropy

- Information answers questions
 - The more clueless I am about the answer initially, the more information is contained in the answer
- Scale: 1 bit = answer to question with prior $\langle 0.5, 0.5 \rangle$
- Information in an answer when prior is $\langle P_1, \dots, P_n \rangle$ is defined as

$$H(\langle P_1, \dots, P_n \rangle) = \sum_{l=1}^n -P_l \log_2 P_l$$
 - Also called **entropy** of the prior
- For p positive and n negative examples we need $H(\langle \frac{p}{p+n}, \frac{n}{p+n} \rangle)$ bits to classify a new example
 - E.g. for 12 restaurant examples with $p = n = 6$: we need 1 bit

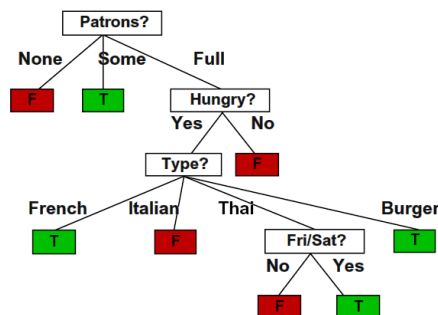
Minimise Entropy

- An attribute splits the examples E into subsets E_1, \dots, E_n
 - Each attribute needs less information to complete the classification (we hope)
- Let E_l have p_l positive and n_l negative examples
 - $H(\langle \frac{p_l}{p_l+n_l}, \frac{n_l}{p_l+n_l} \rangle)$ bits needed to classify a new example
 - *Expected* number of bits per example over all branches:

$$\sum_l \frac{p_l + n_l}{p + n} H(\langle \frac{p_l}{p_l + n_l}, \frac{n_l}{p_l + n_l} \rangle)$$
- For Patrons, this is 0.459 bits, for Type this is (still) 1 bit
 - ➔ Choose the attribute that minimises the remaining information needed

Example

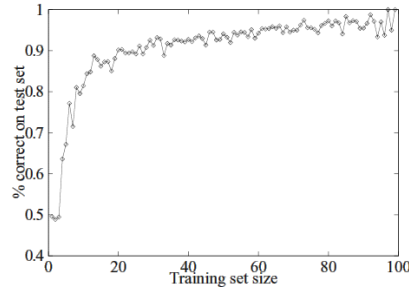
- Decision tree learned from the 12 examples



- Substantially simpler than “true” tree
 - A more complex hypothesis is not justified by small amount of data

Performance Measurement

- ▶ Hume's *Problem of Induction*: How do we know that $h \approx f$?
 - (1) Use theorems of statistical learning theory
 - (2) Try h on a new test set of examples: test data (*same distribution over sample space* as training data)
- ▶ *Learning curve*: % correct on test data vs. training data size



Learning Curves

- ▶ Learning curve depends on
 - *Realisable* vs. *non-realisable*
 - ➔ can target function be expressed?
 - non-realisability can be due to missing attributes or restricted hypothesis class (e.g. linear function)
 - *Redundant expressiveness* (e.g. loads of irrelevant attr.)

