Artificial Intelligence

Module 4: Learning Approaches (2)
Decision Tree Learning

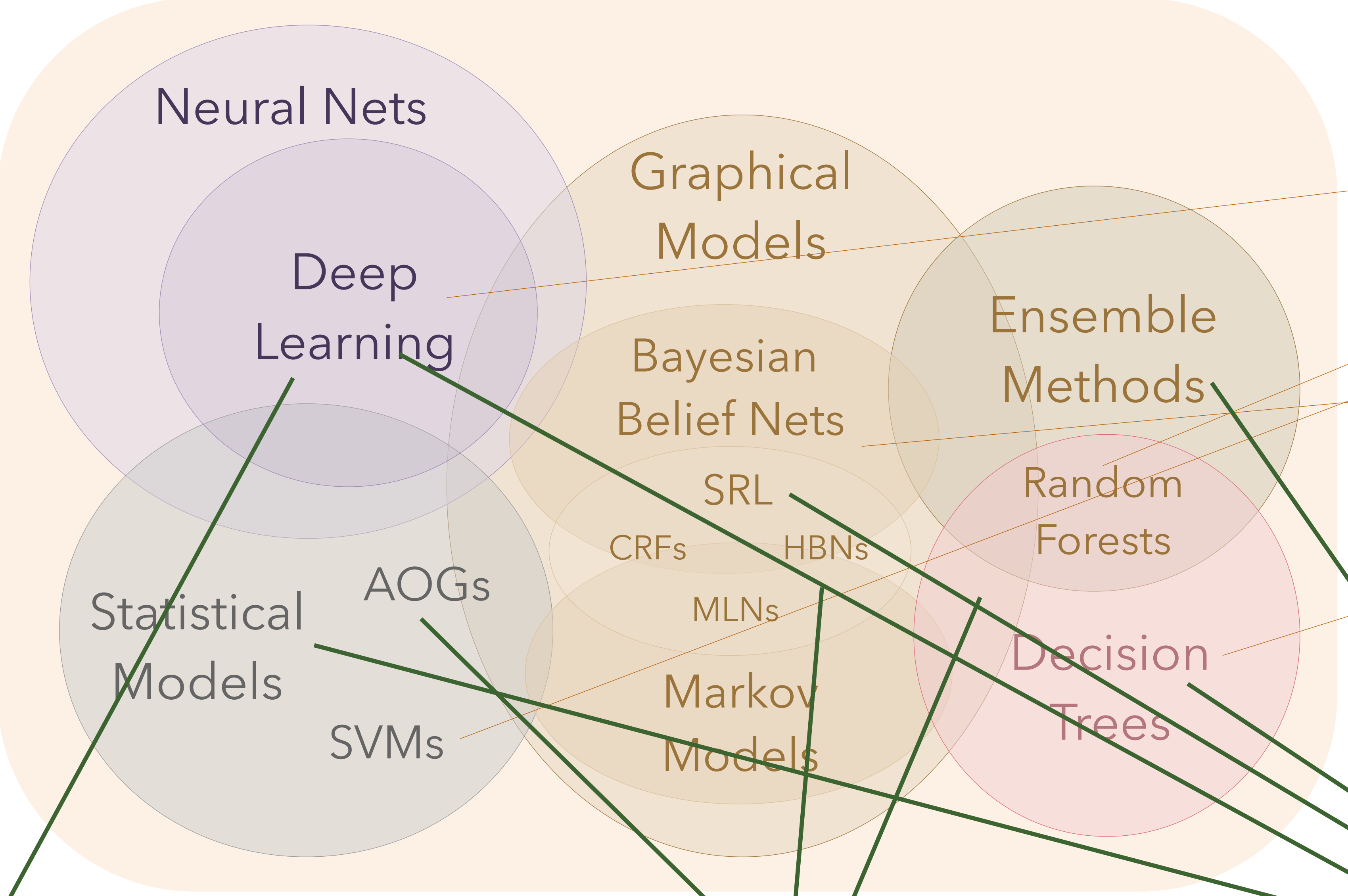
Auckland University of Technology

B.1 Explainable Models

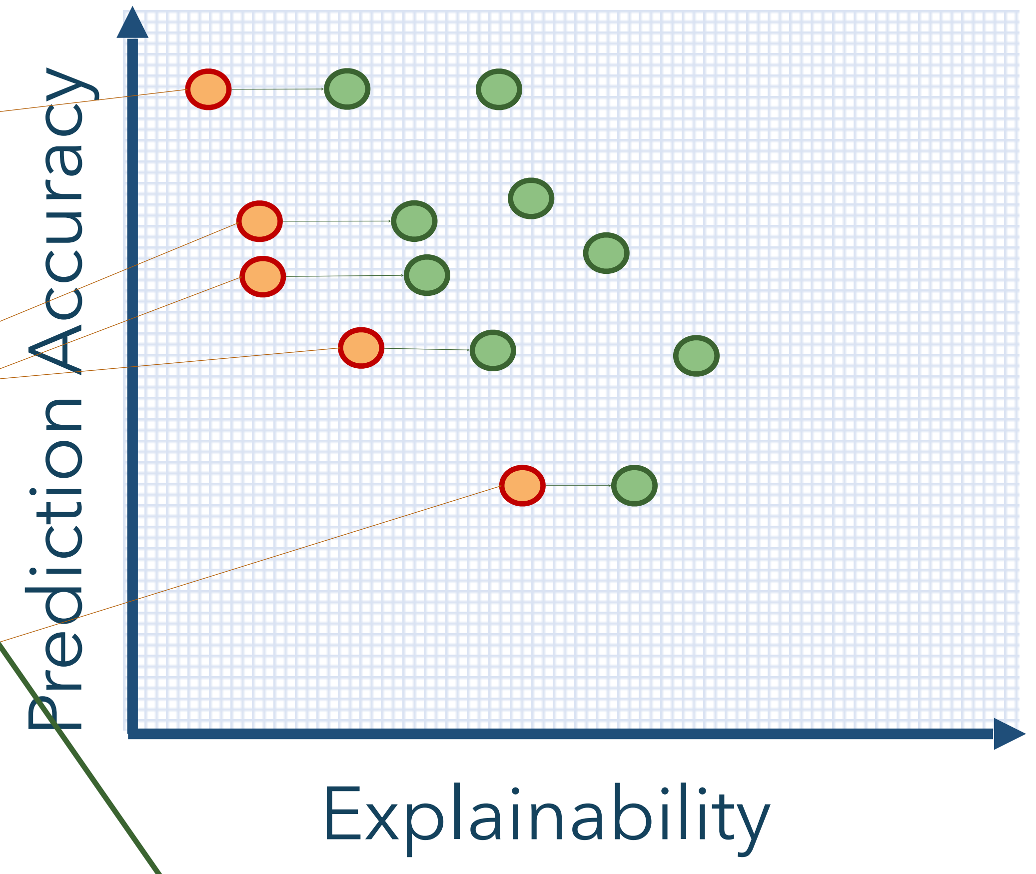
New Approach

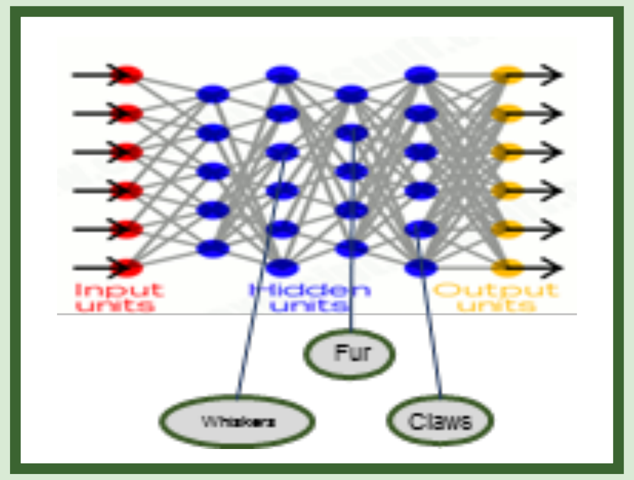
Create a suite of machine learning techniques that produce more explainable models, while maintaining a high level of learning performance

Learning Techniques (today)



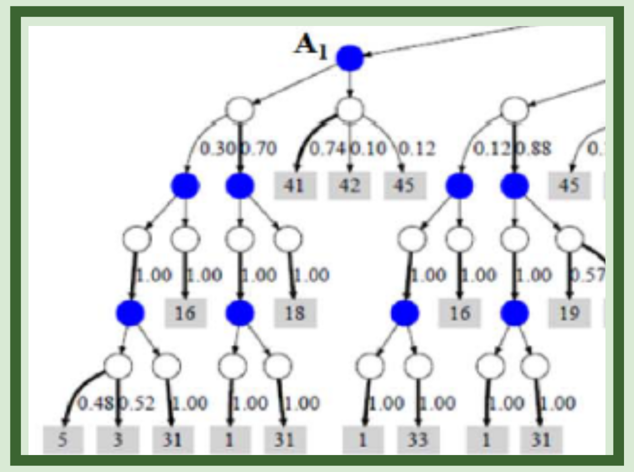
Explainability (notional)





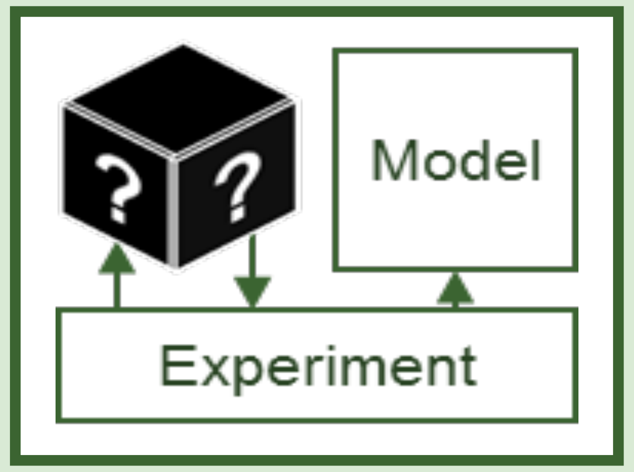
Deep Explanation

Modified deep learning techniques to learn explainable features



Interpretable Models

Techniques to learn more structured, interpretable, causal models



Model Induction

Techniques to infer an explainable model from any model as a black box

https://www.darpa.mil/program/explainable-artificial-intelligence

Learning decision trees

- A **decision tree** represents a function that takes as input *a vector of attribute values* and returns a “decision”—a single output value.
- One of the simplest and successful forms of machine learning
- **Boolean classification**: each example input will be classified as true (a positive example) or false (a negative example).
- A decision tree reaches its decision by performing a sequence of **tests**.

- **Alternate**: whether there is a suitable alternative restaurant nearby.
- **Bar**: whether the restaurant has a comfortable bar area to wait in.
- **Fri/Sat**: true on Fridays and Saturdays.
- **Hungry**: whether we are hungry.
- **Patrons**: how many people are in the restaurant (values are None, Some, and Full).
- **Price**: the restaurant's price range ($, $$, $$$).
- **Raining**: whether it is raining outside.
- **Reservation**: whether we made a reservation.
- **Type**: the kind of restaurant (French, Italian, Thai, or burger).
- **WaitEstimate**: the wait estimated by the host (0–10 minutes, 10–30, 30–60, or >60).

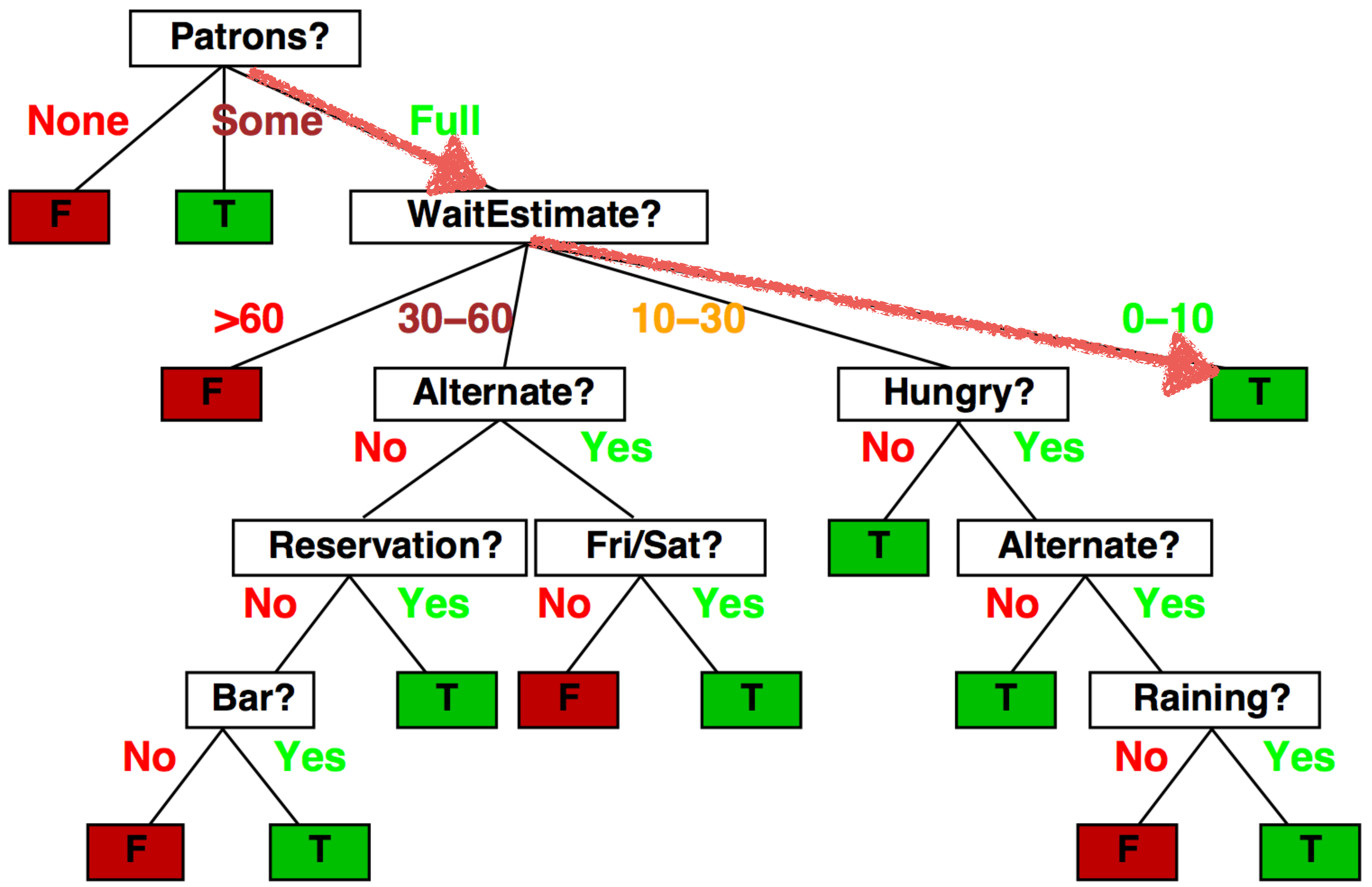
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	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)

Russell & Norvig

Decision tree

Here is the “true” tree for deciding whether to wait:



Russell & Norvig

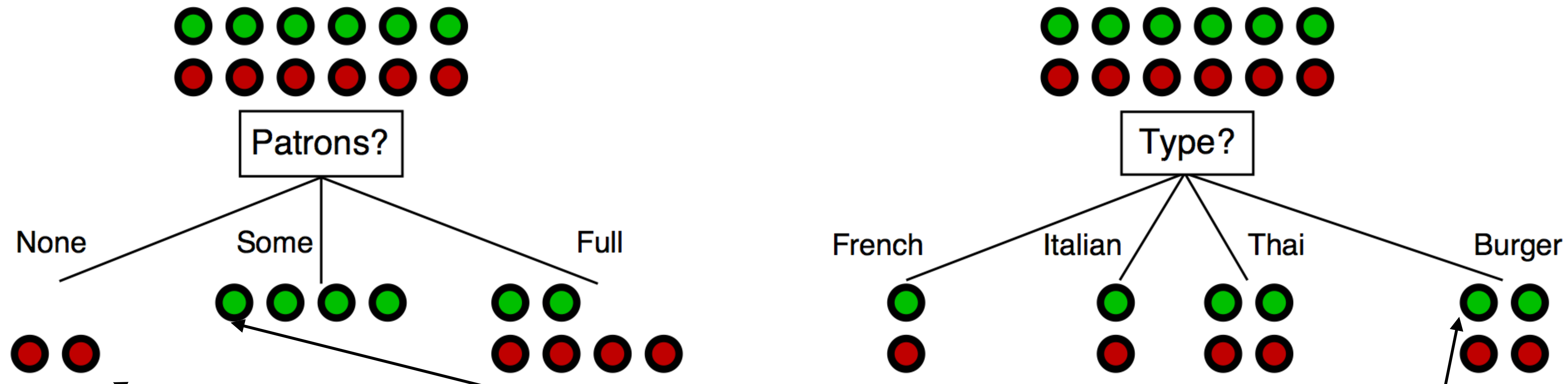
A Boolean decision tree is logically equivalent to the assertion that the **goal attribute** is true if and only if the input attributes satisfy one of the **paths leading to a leaf with value true**.

$$goal \Leftrightarrow (path_1 \vee path_2 \vee \cdots)$$

e.g. $path_{rightmost} = (Patrons = Full \wedge WaitEstimate = \text{0-10})$

Decision tree learning

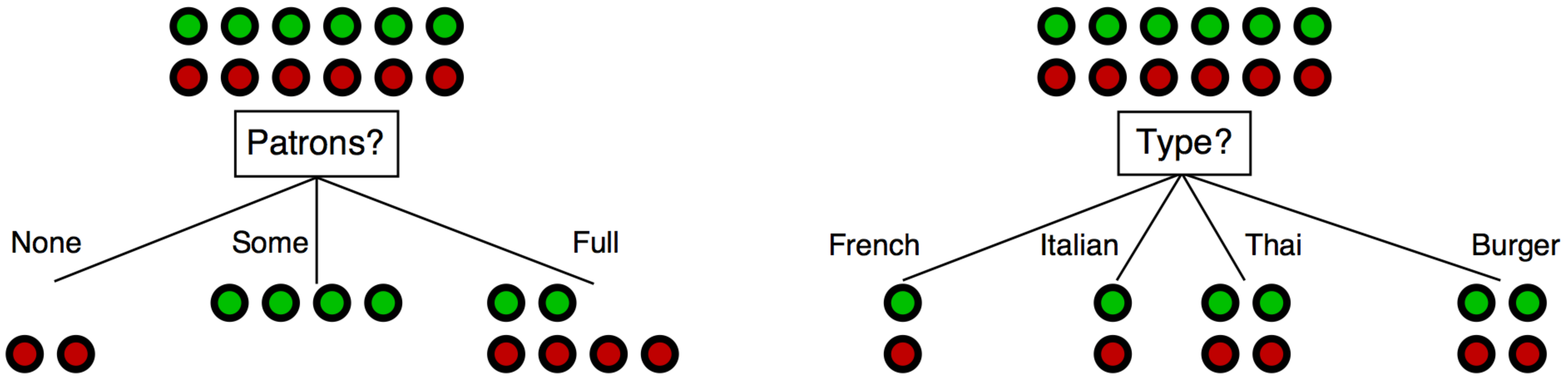
- Aim: find a small tree **consistent** with the training examples



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*

Decision tree learning

- Aim: find a small tree **consistent** with the training examples



Which is a better choice for classification?

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

Patrons? is better - gives more information about the classification

Decision tree learning

Entropy H(S) is a measure of the amount of uncertainty in the set S.

$$H(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

- S – The current (data) set for which entropy is being calculated (changes every iteration of the algorithm)
- X – Set of classes in S

$H(S) = 0$ if perfectly classified

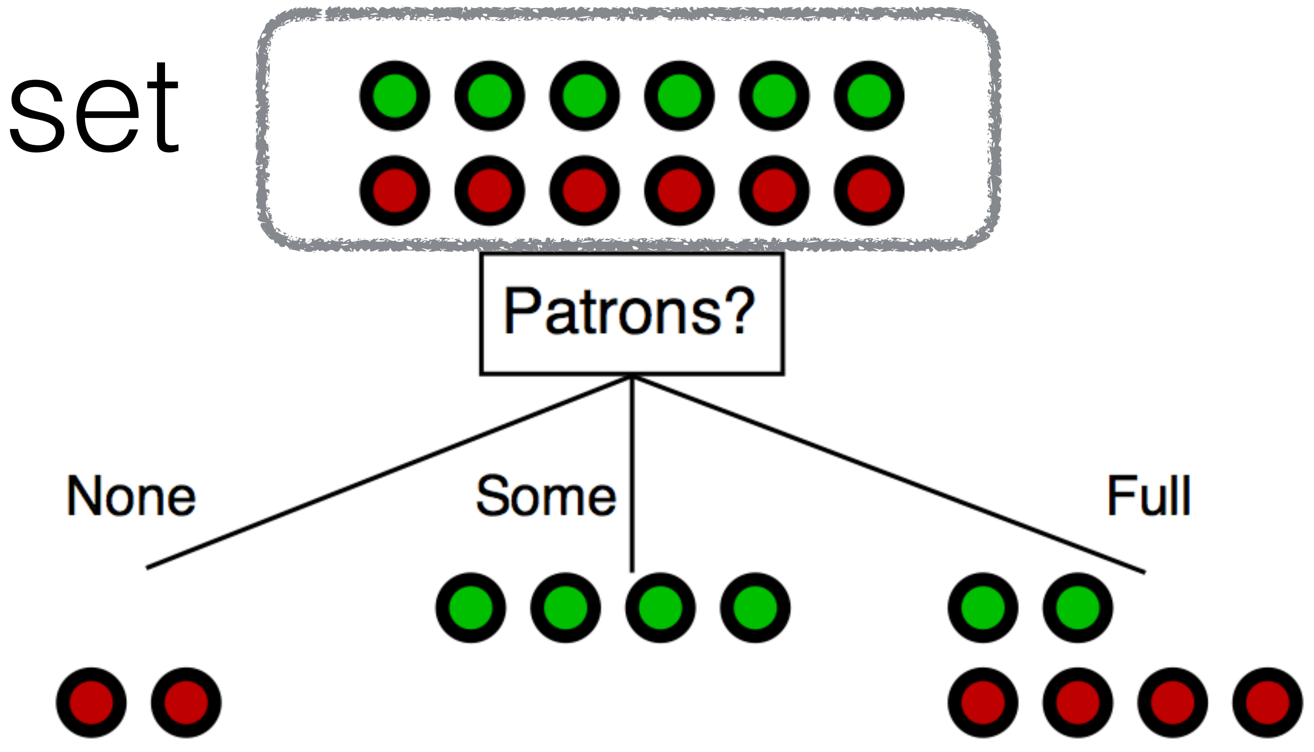
- p(x) – The proportion of the number of elements in class x to the number of elements in set S

Information gain $$IG(A, S) = H(S) - \sum_{t \in T} p(t) H(t)$$

- T – The subsets created from splitting set S by attribute A such that $S = \bigcup_{t \in T} t$
- p(t) – The proportion of the number of elements in t to the number of elements in set S

Decision tree learning

- S = The current (data) set
- X = {Green, Red}



$$H(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

Entropy H(S)=H(6,6) = -6/12 $\log_2$(6/12)-6/12 $\log_2$(6/12) = 1

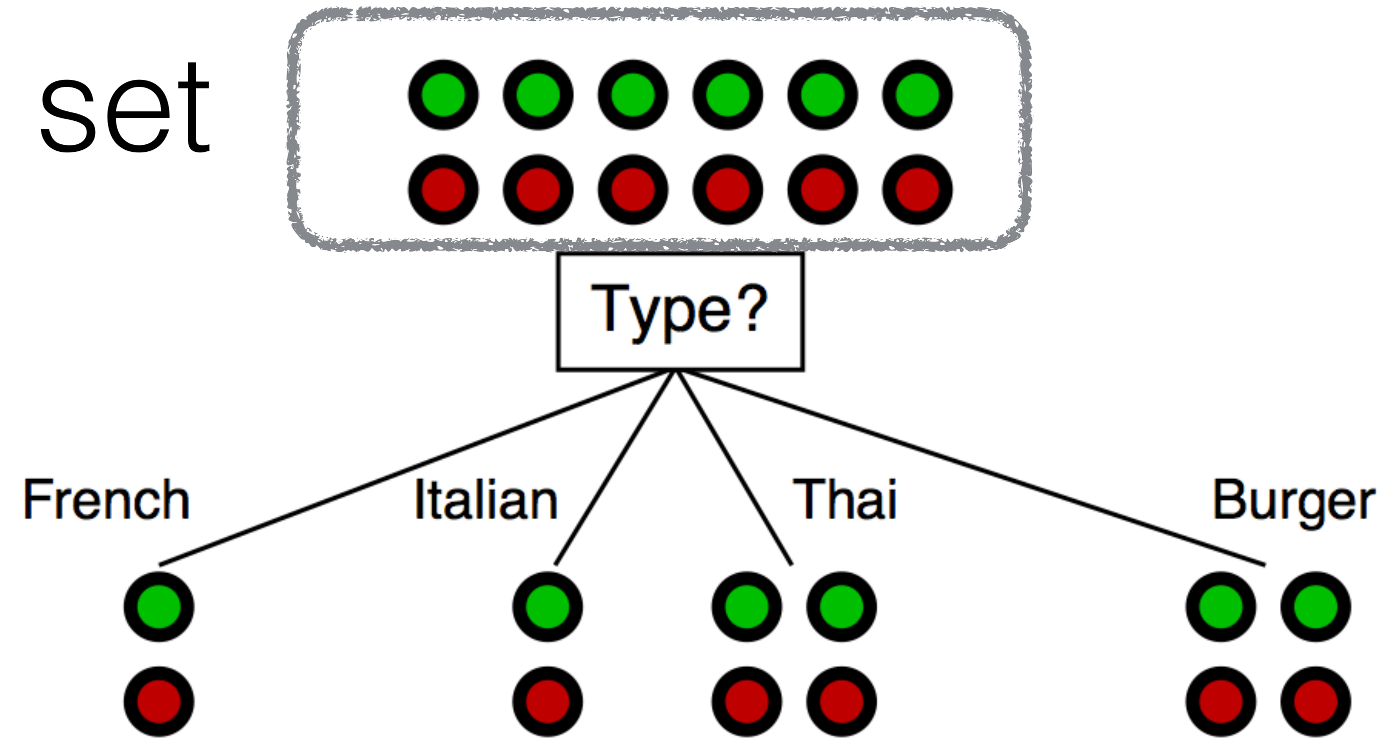
T=
Case t= None: H(0,2) =0
Case t= Some: H(4,0) =0
Case t= Full: H(2,4)= -2/6 $\log_2$(2/6)-4/6 $\log_2$(4/6) = 0.92

IG(Patrons?, S) = 1 - 2/12*0 - 4/12*0 - 6/12*0.92 = 0.54

$$IG(A,S) = H(S) - \sum_{t \in T} p(t) H(t)$$

Decision tree learning

- S = The current (data) set
- X = {Green, Red}



$$H(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

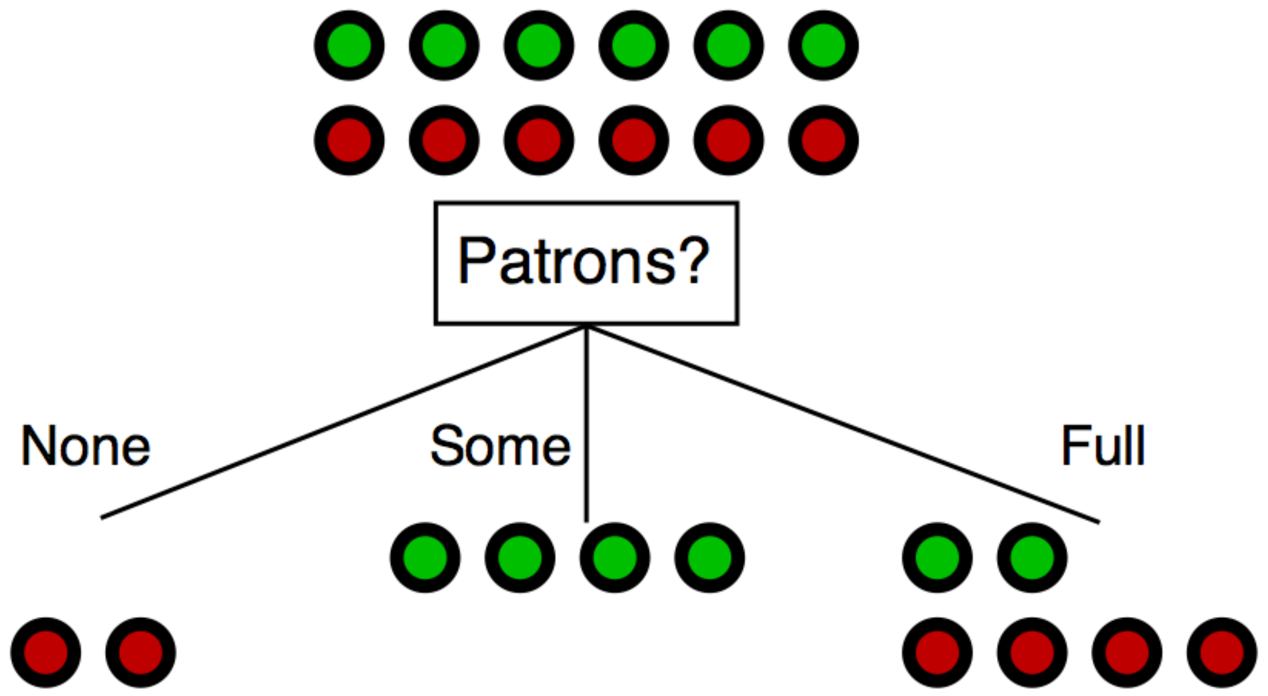
Entropy $H(S)=H(6,6) = -6/12 \log_2(6/12)-6/12 \log_2(6/12) = 1$

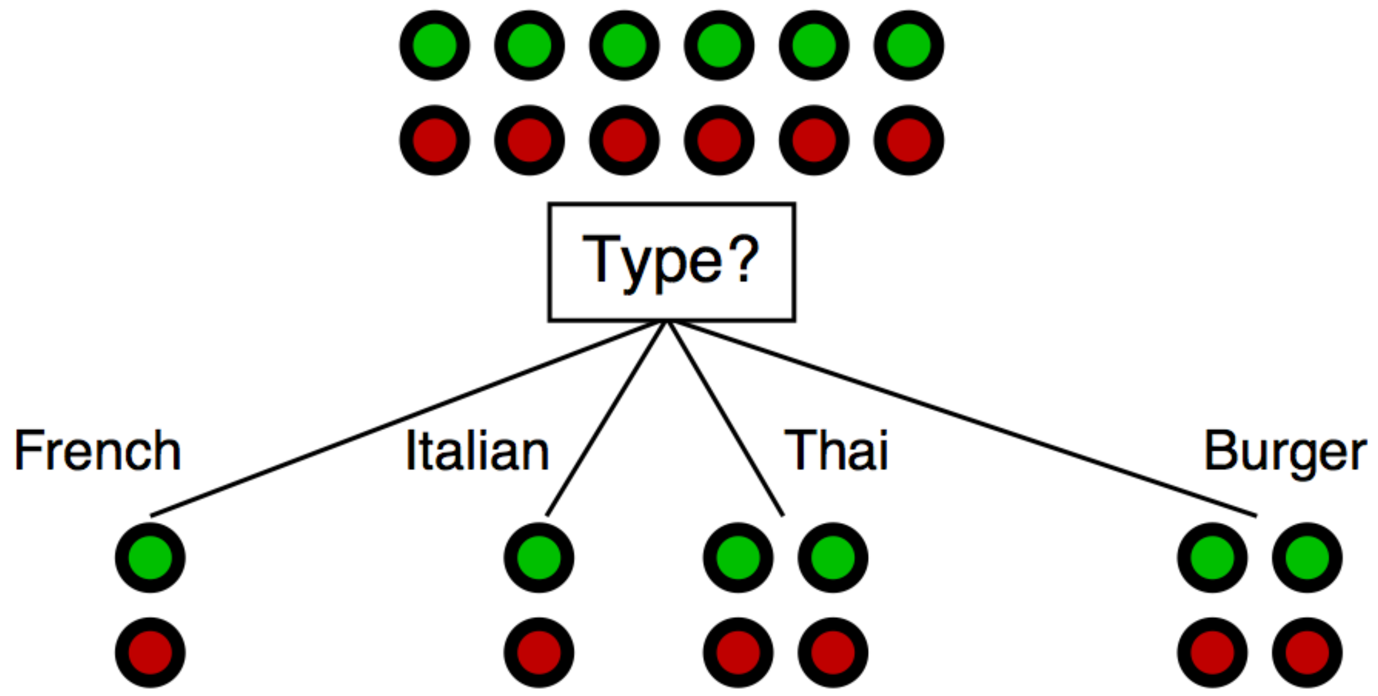
T= Case t=French, Italian: H(1,1) =1
Case t=Thai, Burger: H(2,2) =1

IG(Type?, S)=1-2/12*1-2/12*1-4/12*1-4/12*1=0

$$IG(A,S) = H(S) - \sum_{t \in T} p(t) H(t)$$

Decision tree learning





Better IG(Patrons?, S) = 1 - 2/12*0 - 4/12*0 - 6/12*0.92 = 0.54

IG(Type?, S)=1-2/12*1-2/12*1-4/12*1-4/12*1=0

Decision tree learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose “most significant” attribute as root of (sub)tree
- ID3 (Iterative Dichotomiser 3) Algorithm:
    1. Calculate the entropy of every attribute using the data set S
    2. Split the set S into subsets using the attribute for which the resulting entropy (after splitting) is minimum (or, equivalently, information gain is maximum)
    3. Make a decision tree node containing that attribute
    4. Recurse on subsets using remaining attributes

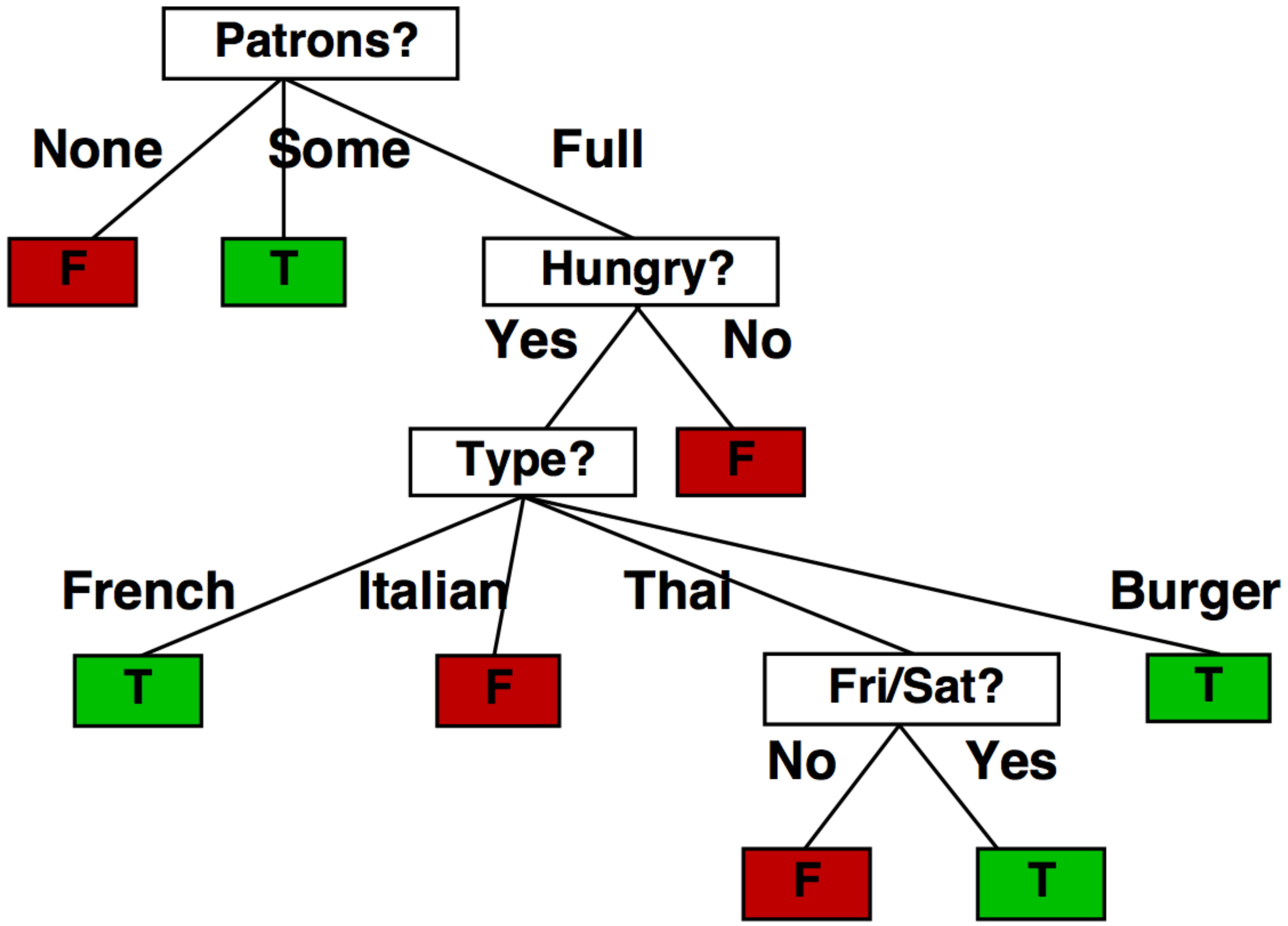
Decision tree learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose “most significant” attribute as root of (sub)tree

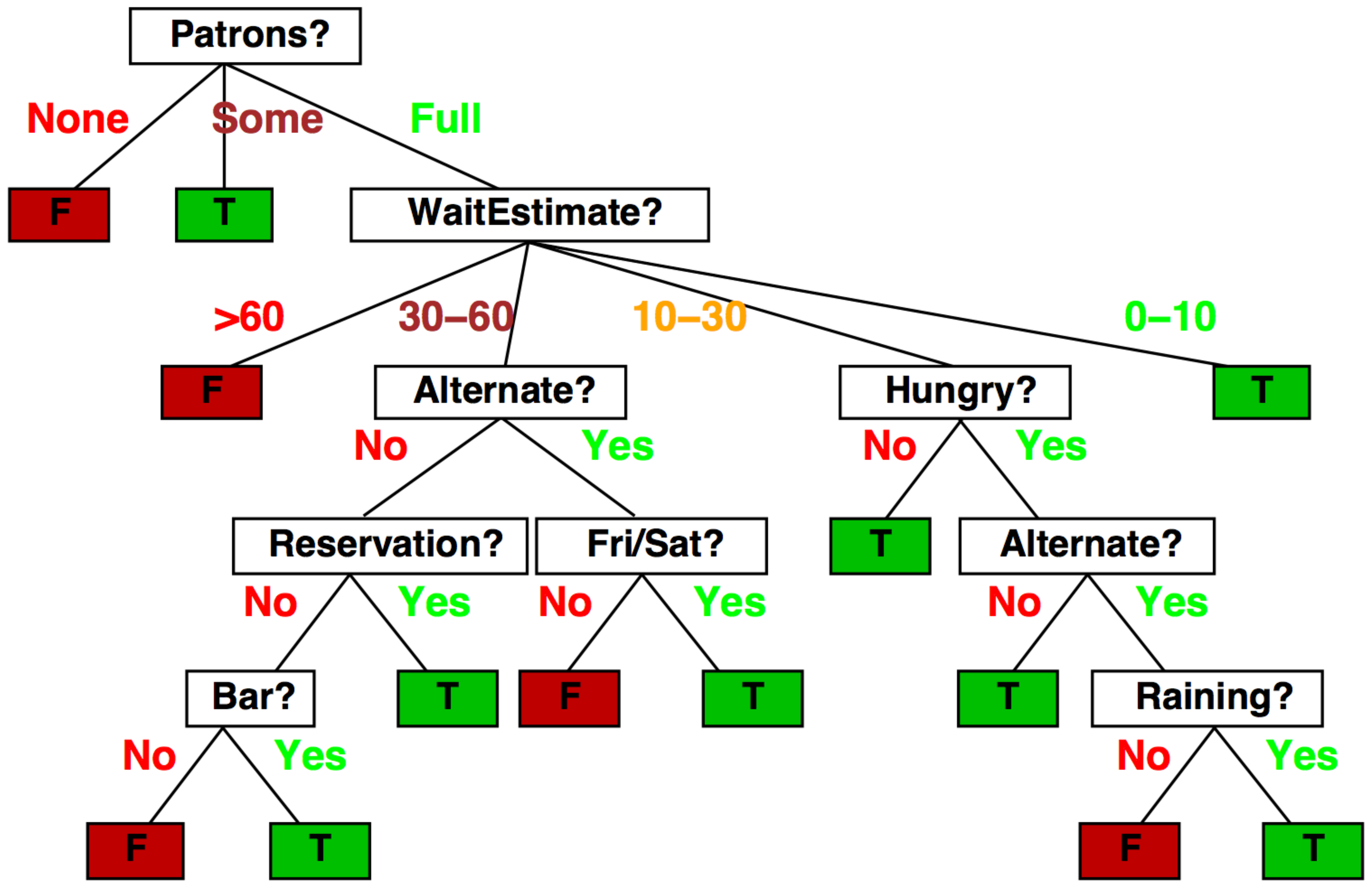
```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Decision tree learning

Decision tree learned from the 12 examples



True decision tree

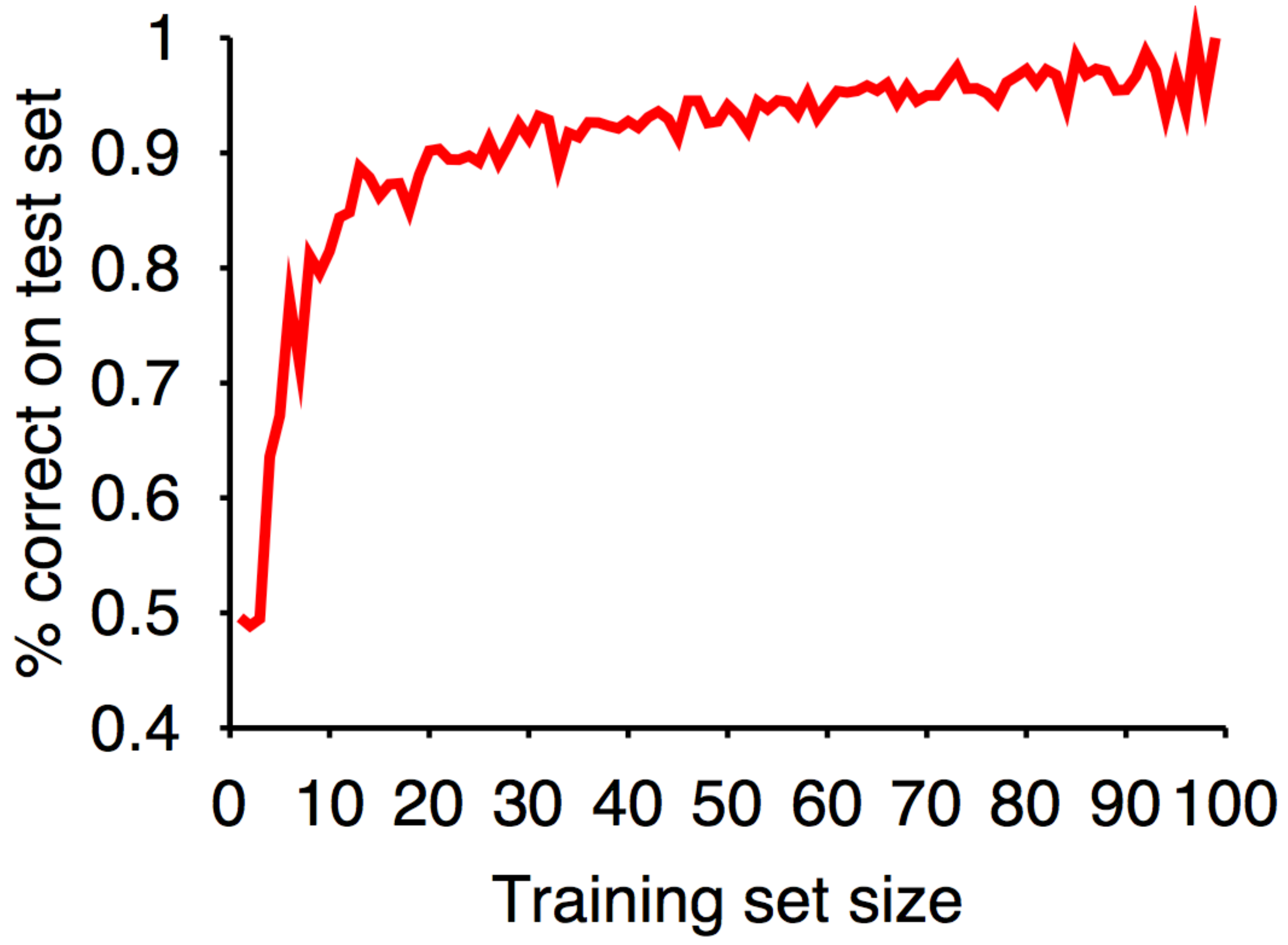


Substantially simpler than "true" tree — a more complex hypothesis isn't justified by small amount of data

Performance measurement

How do we know that hypothesis **h ≈ f**?

- Use theorems of computational/statistical learning theory
- 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

Reference

- AIMA book: chapter 19