CS 4700:
Foundations of Artificial Intelligence

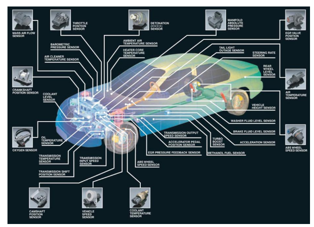
Prof. Bart Selman
selman@cs.cornell.edu

Machine Learning:
Decision Trees
R&N 18.3

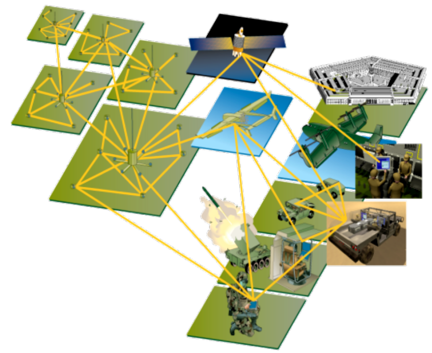
Big Data: Sensors Everywhere

Data collected and stored at
enormous speeds (GB/hour)

Cars
Cellphones
Remote Controls
Traffic lights,
ATM machines
Appliances
Motion sensors
Surveillance cameras
etc etc

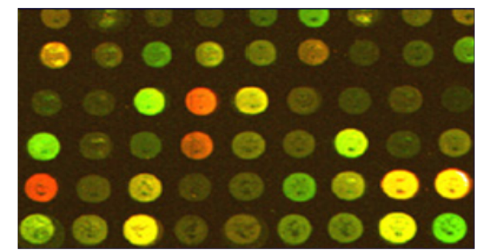


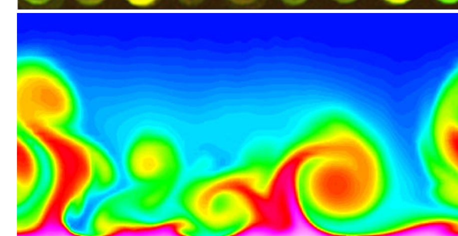
Big Data: Scientific Domains

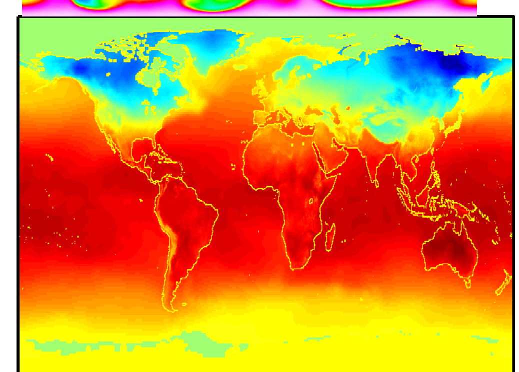


Data collected and stored at enormous speeds (GB/hour)

- remote sensors on a satellite
- telescopes scanning the skies
- microarrays generating gene expression data
- scientific simulations generating terabytes of data







Traditional statistical techniques infeasible to deal with the data TUSNAMI – they don't scale up!!!

→ Machine Learning Techniques

(adapted from Vipin Kumar)

Machine Learning Tasks

Prediction Methods

– Use some variables to predict unknown or future values of other variables.

Description Methods

– Find human-interpretable patterns that describe the data.

Machine Learning Tasks

Supervised learning:

We are given a set of examples with the correct answer -
classification and regression

Unsupervised learning: “just make sense of the data”

Example: Supervised Learning
object recognition
Classification

x

f(x)	giraffe	giraffe	giraffe	llama	llama	llama

Target Function

From: Stuart Russell

Example: Supervised Learning
object recognition
Classification

x






f(x)	giraffe	giraffe	giraffe	llama	llama	llama

Target Function

X=

f(x)=?

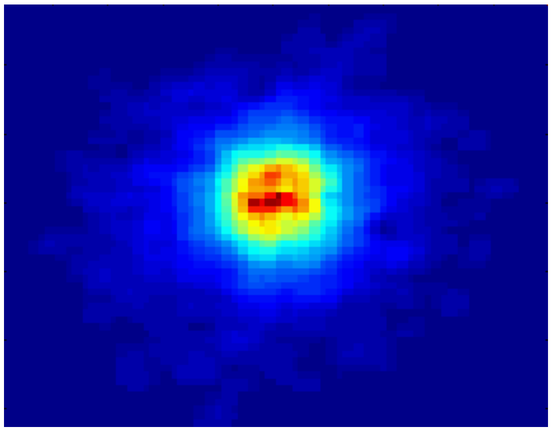
From: Stuart Russell

7

Classifying Galaxies

Courtesy: http://aps.umn.edu

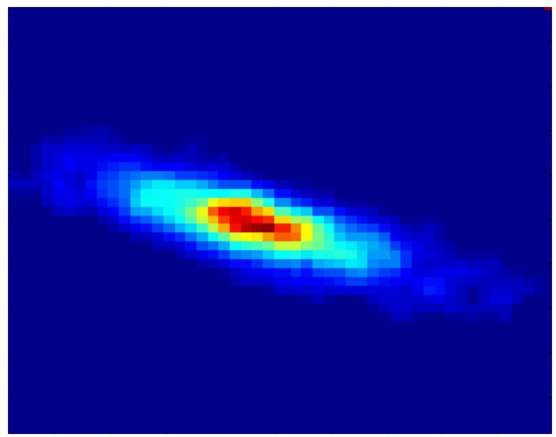
Early



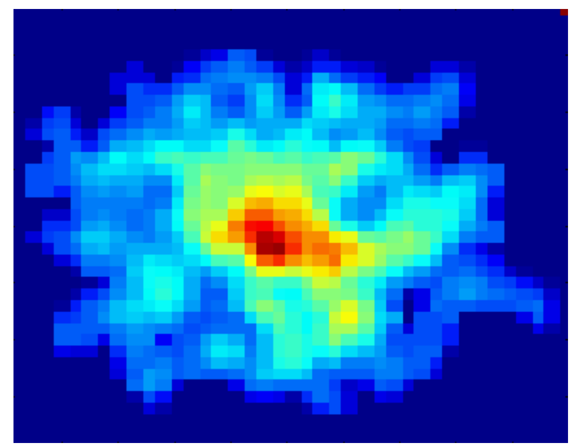
Class:
- Stages of Formation

Attributes:
- Image features,
- Characteristics of light waves received, etc.

Intermediate

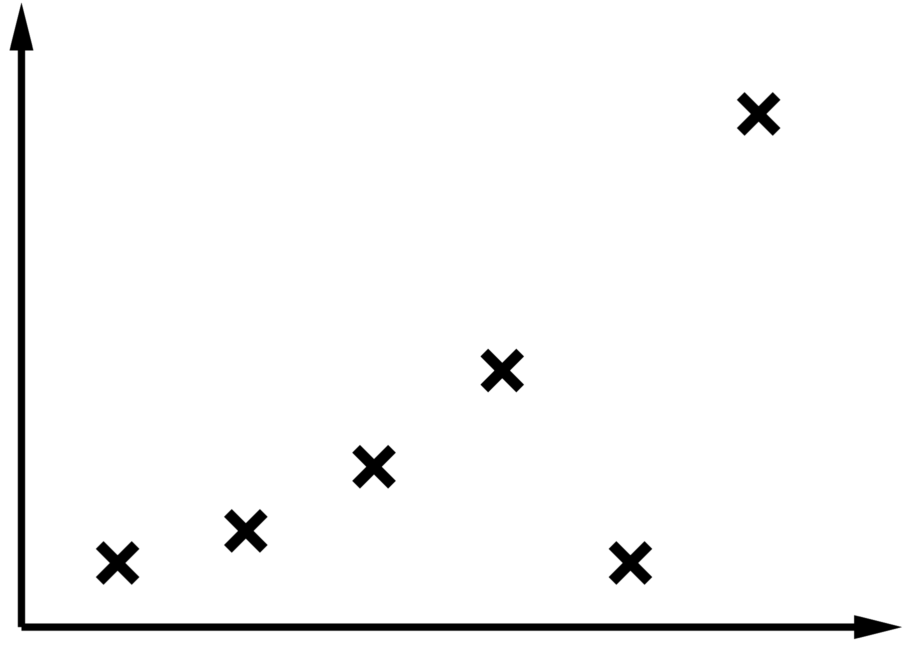


Late

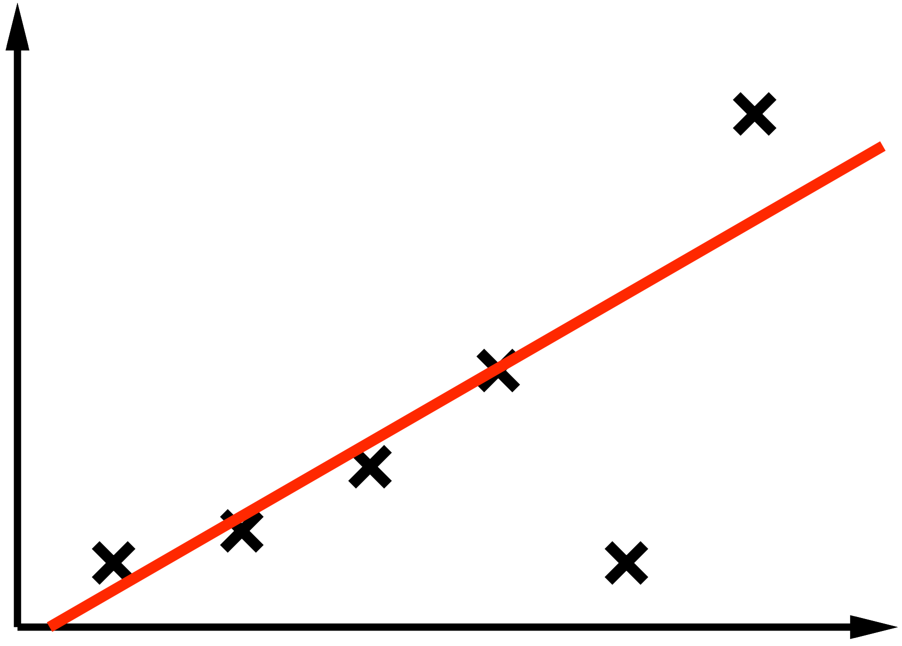


Data Size:
- 72 million stars, 20 million galaxies
- Object Catalog: 9 GB
- Image Database: 150 GB

Supervised learning: curve fitting
Regression

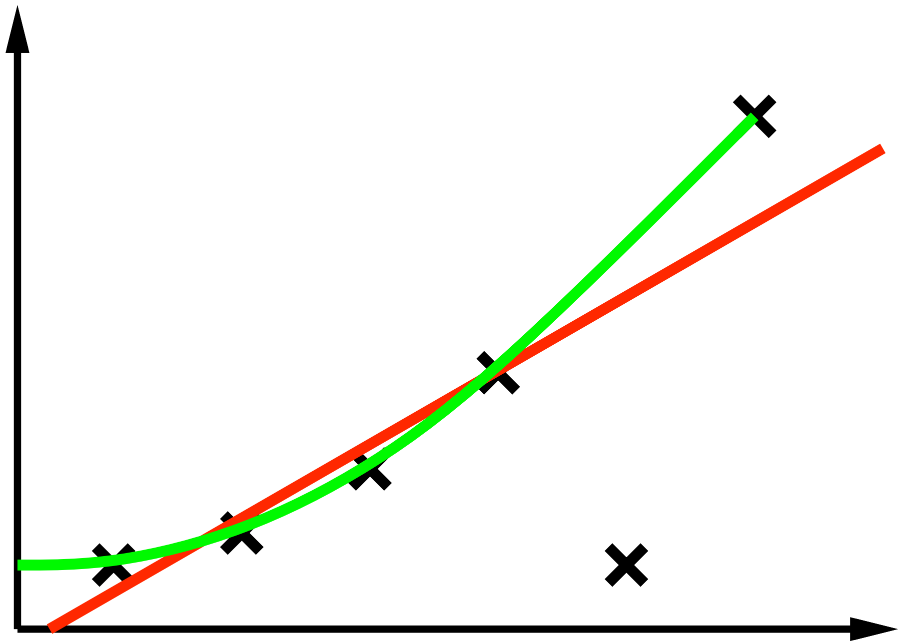


Supervised learning: curve fitting
Regression

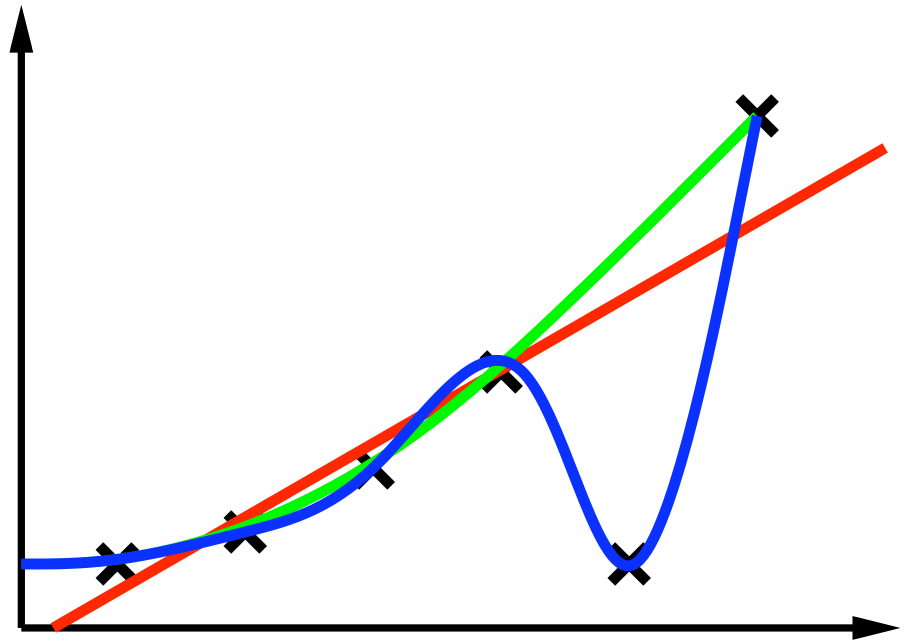


10

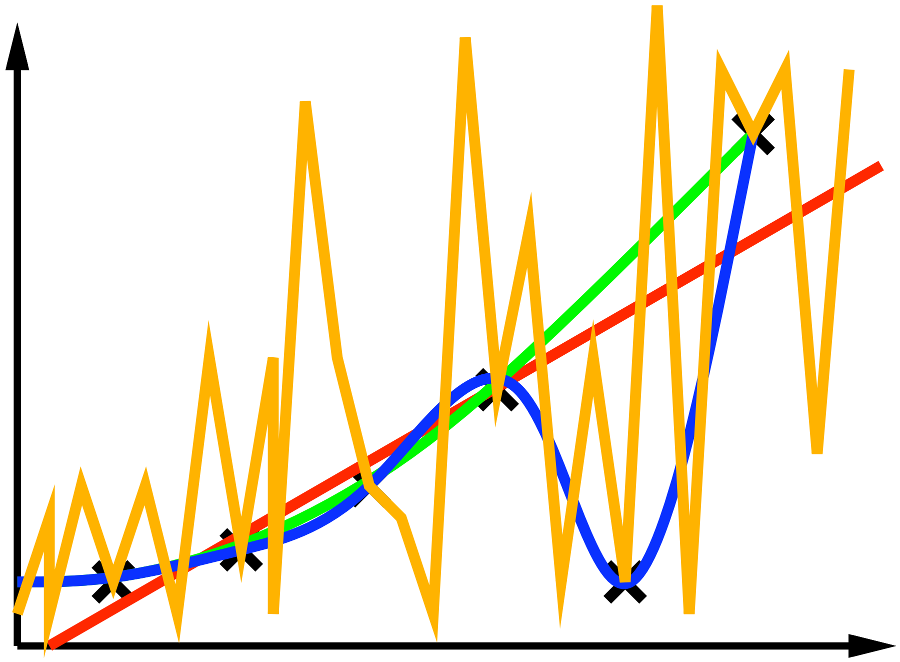
Supervised learning: curve fitting
Regression



Supervised learning: curve fitting
Regression

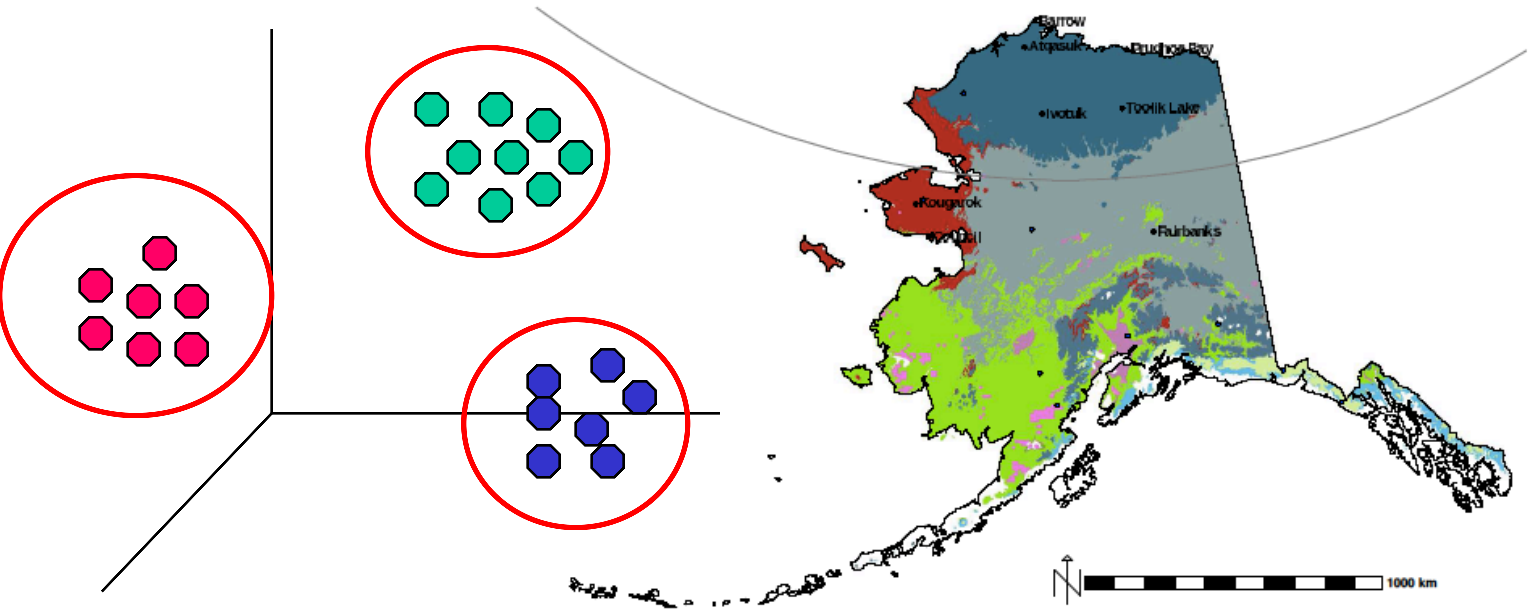


Supervised learning: curve fitting
Regression



Unsupervised Learning: Clustering

10 Alaska Ecoregions (2000–2009)



Each ecoregion is a different random color. Blue filled circles mark locations most representative of mean conditions of each region.

<u>Ecoregion Analysis of Alaska using clustering</u>

"Representativeness-based Sampling Network Design for the State of Alaska." Hoffman, Forrest M., Jitendra Kumar, Richard T. Mills, and William W. Hargrove. 2013. Landscape Ecology

14

Machine Learning

In classification – inputs belong two or more classes.

Goal: the learner must produce a model that assigns unseen inputs to one (or multi-label classification) or more of these classes. Typically supervised learning.

- Example –
- Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".

In regression, also typically supervised, the outputs are continuous rather than discrete.

In clustering, a set of inputs is to be divided into groups. Typically done in an unsupervised way (i.e., no labels, the groups are not known beforehand).

Supervised learning: Big Picture

Goal: To learn an unknown *target function* f

Input: a *training set* of *labeled examples* (x_j, y_j) where $y_j = f(x_j)$

- E.g., x_j is an image, $f(x_j)$ is the label "giraffe"
- E.g., x_j is a seismic signal, $f(x_j)$ is the label "explosion"

Output: *hypothesis* h that is "close" to f, i.e., predicts well on unseen examples ("*test set*")

Many possible hypothesis families for h

- Linear models, logistic regression, neural networks, support vector machines, decision trees, examples (nearest-neighbor), grammars, kernelized separators, etc etc

Today: Decision Trees!

Big Picture of Supervised Learning

Learning can be seen as fitting a function to the data. We can consider different target functions and therefore different hypothesis spaces.

Examples:

Propositional if-then rules

Decision Trees

First-order if-then rules

First-order logic theory

Linear functions

Polynomials of degree at most k

Neural networks

Java programs

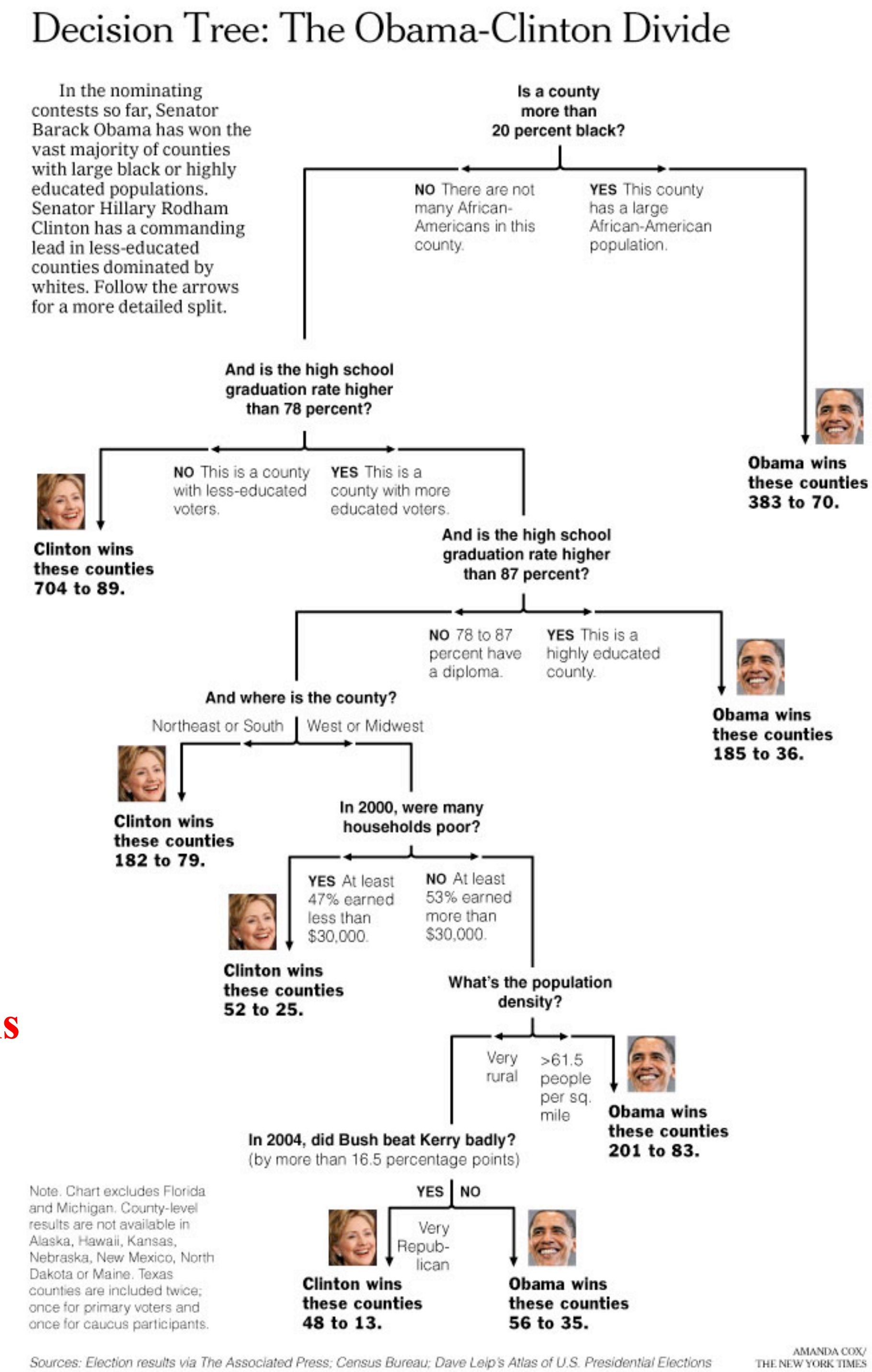
Turing machine

Etc

A learning problem is realizable if its hypothesis space contains the true function.

Tradeoff between expressiveness of a hypothesis space and the complexity of finding simple, consistent hypotheses within the space.

Can we learn how counties vote?



New York Times
April 16, 2008

Decision Trees:
a sequence of tests.
Representation very natural for humans.
Style of many "How to" manuals and trouble-shooting procedures.

Decision Tree: The Obama-Clinton Divide

In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.

Is a county more than 20 percent black?

NO There are not many African-Americans in this county.

YES This county has a large African-American population.

Note: order of tests matters (in general)!

Obama wins these counties 383 to 70.

And is the high school graduation rate higher than 78 percent?

NO This is a county with less-educated voters.

YES This is a county with more educated voters.

Clinton wins these counties 704 to 89.

And is the high school graduation rate higher than 87 percent?

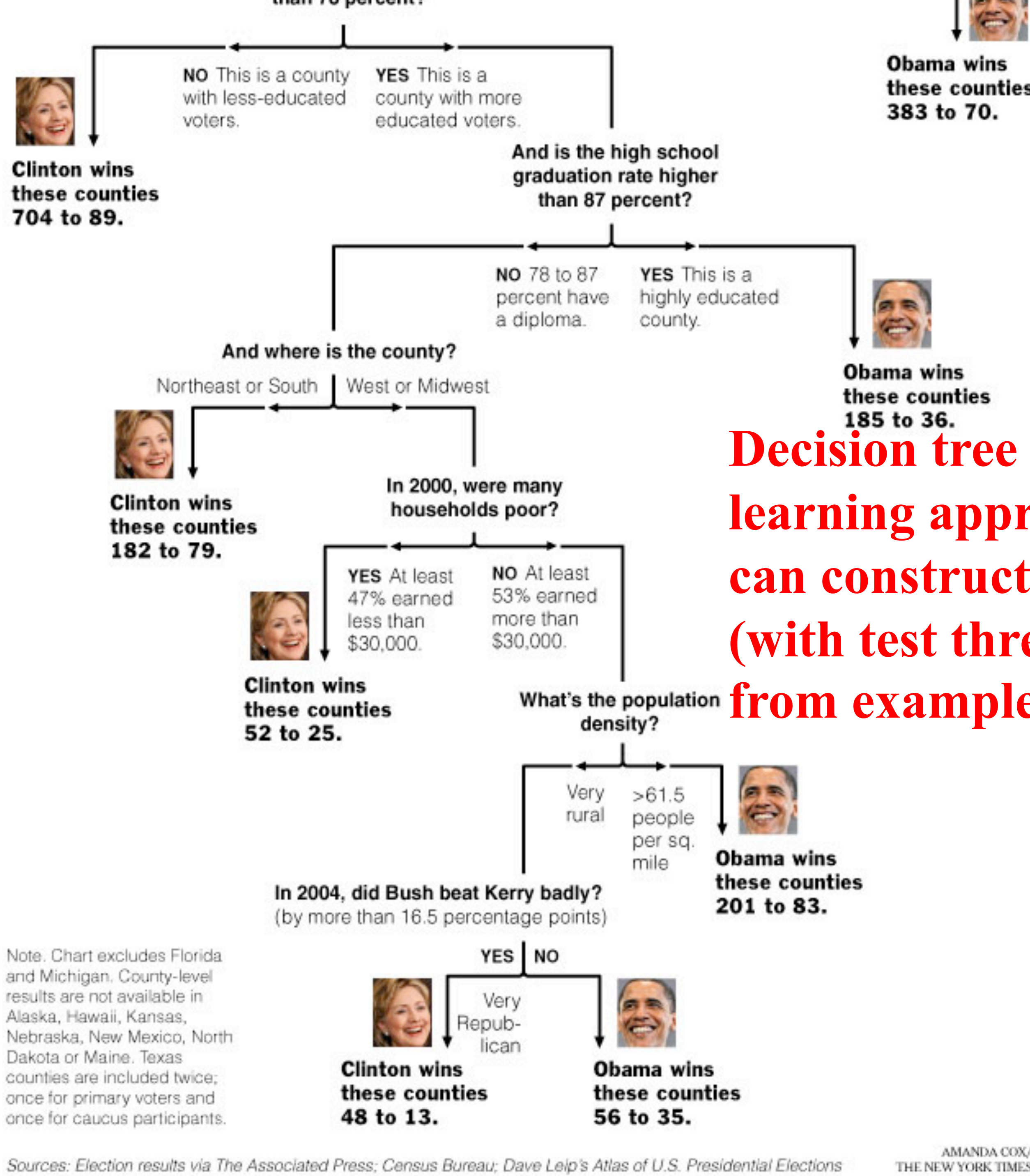
NO 78 to 87 percent have a diploma.

YES This is a highly educated county.

Obama wins these counties 185 to 36.

And where is the county?

Northeast or South | West or Midwest



Decision tree learning approach can construct tree (with test thresholds) from example counties.

Decision Tree Learning

Decision Tree Learning

Task:

– Given: collection of examples (x, f(x))

– Return: a function *h* (*hypothesis*) that approximates *f*

– *h is a decision tree*

Input: an object or situation described by a set of attributes (or features)

Output: a "decision" – the predicts output value for the input.

The input attributes and the outputs can be discrete or continuous.

We will focus on decision trees for Boolean classification:

each example is classified as positive or negative.

Decision Tree

What is a decision tree?

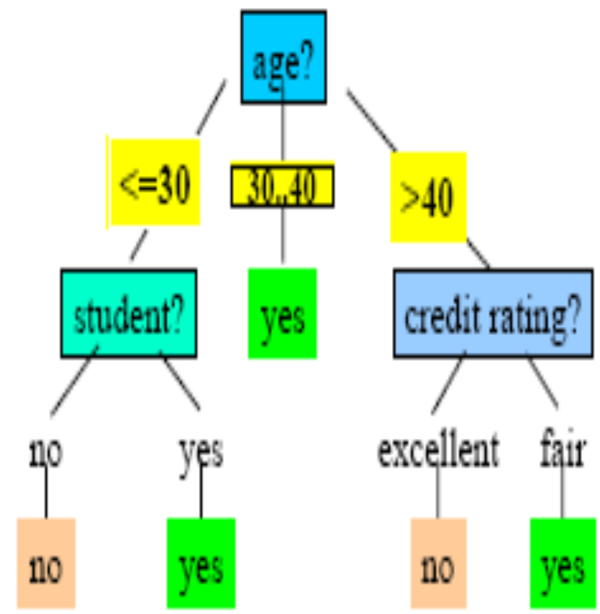
A tree with two types of nodes:

Decision nodes
Leaf nodes

Decision node: Specifies a choice or test of some attribute with 2 or more alternatives;
→ every decision node is part of a path to a leaf node

Leaf node: Indicates classification of an example

- Decision Tree example (is a customer going to buy a computer or not):



Big Tip Example

Food (3)	Chat (2)	Fast (2)	Price (3)	Bar (2)	BigTip
great	yes	yes	normal	no	yes
great	no	yes	normal	no	yes
mediocre	yes	no	high	no	no
great	yes	yes	normal	yes	yes

Etc.

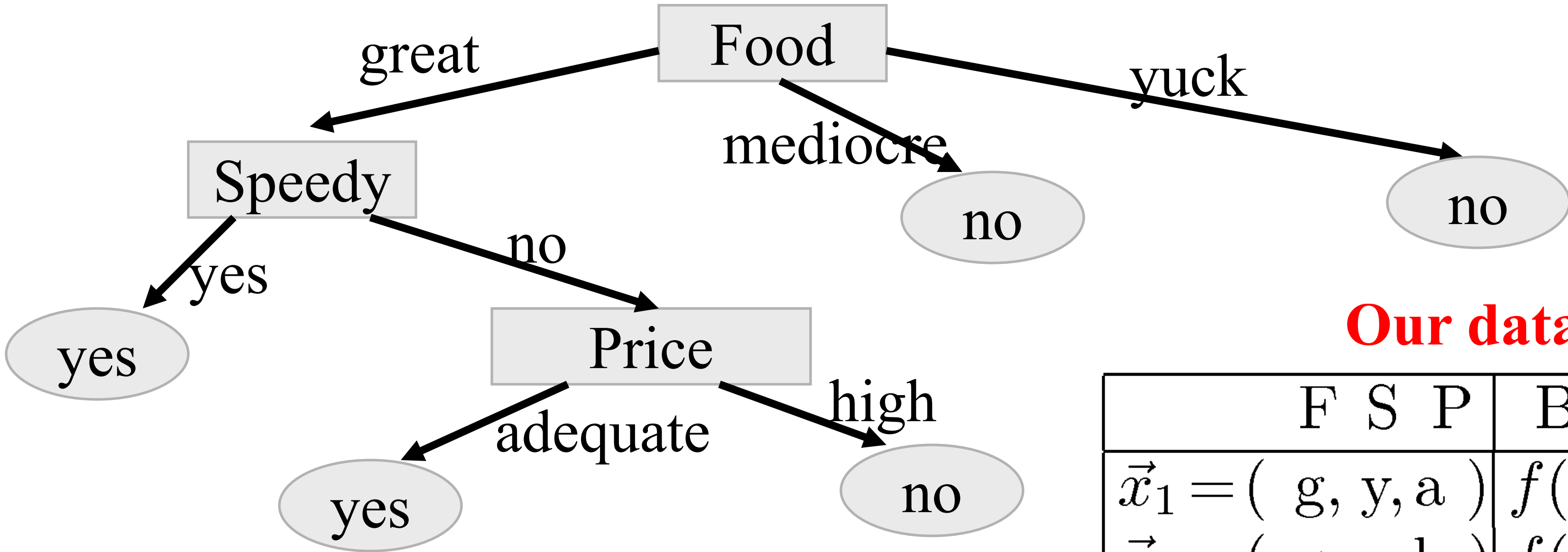
Instance Space X: Set of all possible objects described by attributes (often called features).

Target Function f: Mapping from Attributes to Target Feature (often called label) (f is unknown)

Hypothesis Space H: Set of all classification rules h_i we allow.

Training Data D: Set of instances labeled with Target Feature

Decision Tree Example: "BigTip"



Our data

	F	S	P		BigTip
$\vec{x}_1 = ($	g,	y,	a	$)$	$f(\vec{x}_1) = 1$
$\vec{x}_2 = ($	g,	n,	h	$)$	$f(\vec{x}_2) = 0$
$\vec{x}_3 = ($	g,	y,	h	$)$	$f(\vec{x}_3) = 1$
$\vec{x}_4 = ($	g,	n,	a	$)$	$f(\vec{x}_4) = 1$
$\vec{x}_5 = ($	m,	y,	a	$)$	$f(\vec{x}_5) = 0$
$\vec{x}_6 = ($	y,	y,	a	$)$	$f(\vec{x}_6) = 0$
$\vec{x}_7 = ($	g,	y,	a	$)$	$f(\vec{x}_7) = 1$
$\vec{x}_8 = ($	g,	y,	h	$)$	$f(\vec{x}_8) = 1$
$\vec{x}_9 = ($	m,	y,	a	$)$	$f(\vec{x}_9) = 0$
$\vec{x}_{10} = ($	g,	y,	a	$)$	$f(\vec{x}_{10}) = 1$

Is the decision tree we learned consistent?

Yes, it agrees with all the examples!

Data: Not all 2x2x3 = 12 tuples
Also, some repeats! These are literally "observations."

Learning decision trees: Another example (waiting at a restaurant)

Problem: decide whether to wait for a table at a restaurant. What attributes would you use?

Attributes used by R&N

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Goal predicate: WillWait?

Aside:
What about using restaurant name?
It could be great for generating a small tree …
But it doesn't generalize!

Attribute-based representations

Examples described by attribute values (Boolean, discrete, continuous)

E.g.

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
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

12 examples

6 +

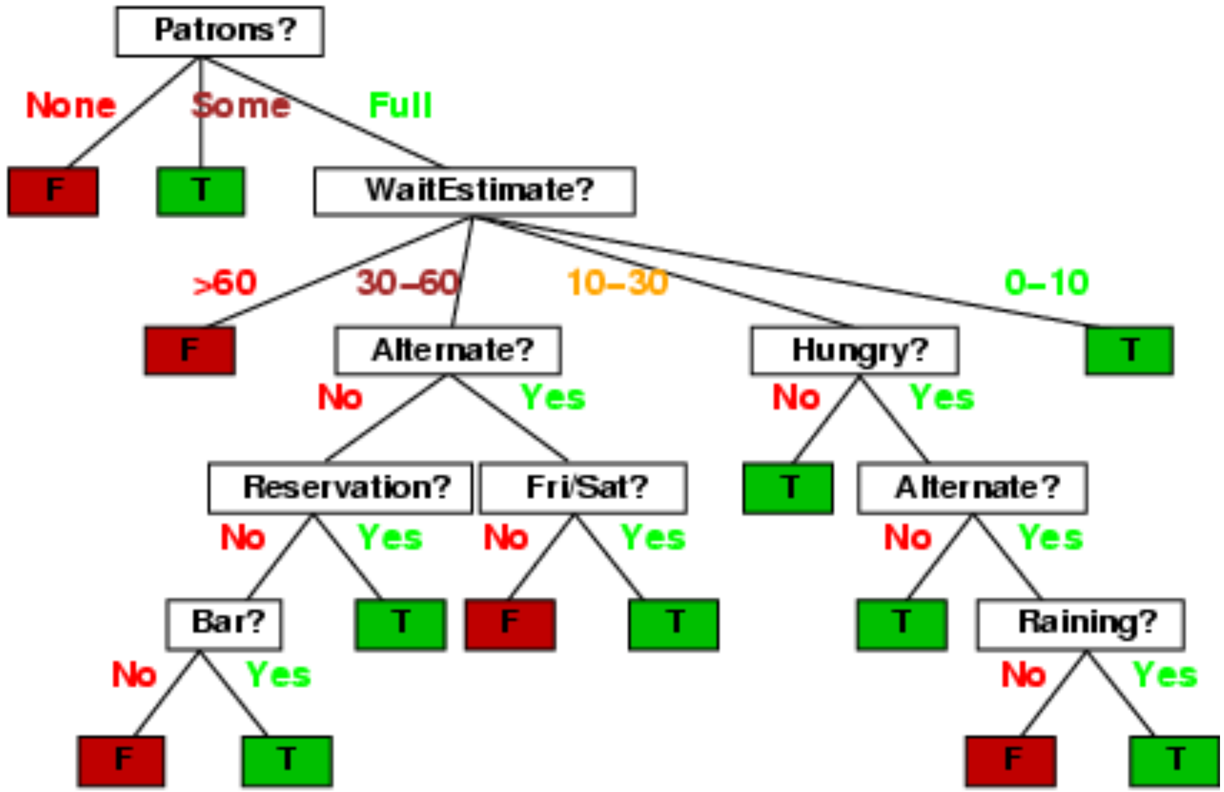
6 -

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

Decision trees

One possible representation for hypotheses

E.g., here is a tree for deciding whether to wait:



Decision tree learning Algorithm

Decision trees can express any Boolean function.

Goal: Finding a decision tree that agrees with training set.

We could construct a decision tree that has one path to a leaf for each example, where the path tests sets each attribute value to the value of the example.

What is the problem with this from a learning point of view?

Problem: This approach would just memorize example.
How to deal with new examples? It doesn't generalize!

(But sometimes hard to avoid --- e.g. parity function, 1, if an even number of inputs, or majority function, 1, if more than half of the inputs are 1).

Overall Goal: get a good classification with a small number of tests.

We want a compact/smallest tree.
But finding the smallest tree consistent with the examples is NP-hard!

“most significant”
In what sense?

Basic DT Learning Algorithm

Goal: find a *small* tree consistent with the training examples

Idea: (recursively) choose "most significant" attribute as root of (sub)tree;

Use a top-down greedy search through the space of possible decision trees.

Greedy because there is no backtracking. It picks highest values first.

Variations of known algorithms ID3, C4.5 (Quinlan -86, -93)

Top-down greedy construction

(ID3 Iterative Dichotomiser 3)

- Which attribute should be tested?
  - Heuristics and Statistical testing with current data
- Repeat for descendants

Big Tip Example

10 examples:

6+ 1 3 4 7 8 10

4- 2 5 6 9

Attributes:

•Food with values g,m,y

•Speedy? with values y,n

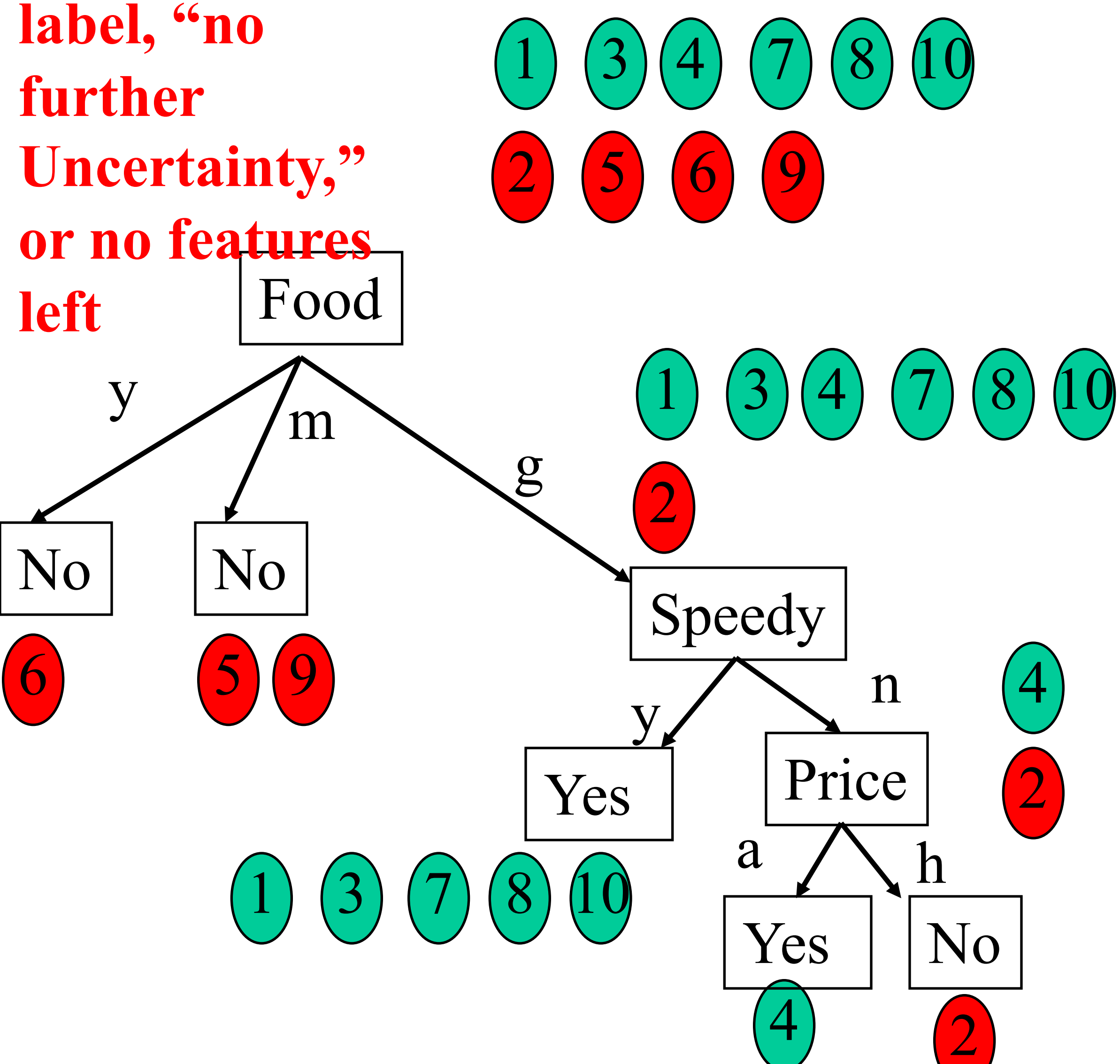
•Price, with values a, h

F S P	BigTip
$\vec{x}_1 = (\ g, y, a\ )$	$f(\vec{x}_1) = 1$
$\vec{x}_2 = (\ g, n, h\ )$	$f(\vec{x}_2) = 0$
$\vec{x}_3 = (\ g, y, h\ )$	$f(\vec{x}_3) = 1$
$\vec{x}_4 = (\ g, n, a\ )$	$f(\vec{x}_4) = 1$
$\vec{x}_5 = (\ m, y, a\ )$	$f(\vec{x}_5) = 0$
$\vec{x}_6 = (\ y, y, a\ )$	$f(\vec{x}_6) = 0$
$\vec{x}_7 = (\ g, y, a\ )$	$f(\vec{x}_7) = 1$
$\vec{x}_8 = (\ g, y, h\ )$	$f(\vec{x}_8) = 1$
$\vec{x}_9 = (\ m, y, a\ )$	$f(\vec{x}_9) = 0$
$\vec{x}_{10} = (\ g, y, a\ )$	$f(\vec{x}_{10}) = 1$

Let's build our decision tree starting with the attribute Food, (3 possible values: g, m, y).

Top-Down Induction of Decision Tree: Big Tip Example

Node "done" when uniform label, "no further Uncertainty," or no features left



10 examples:

6+

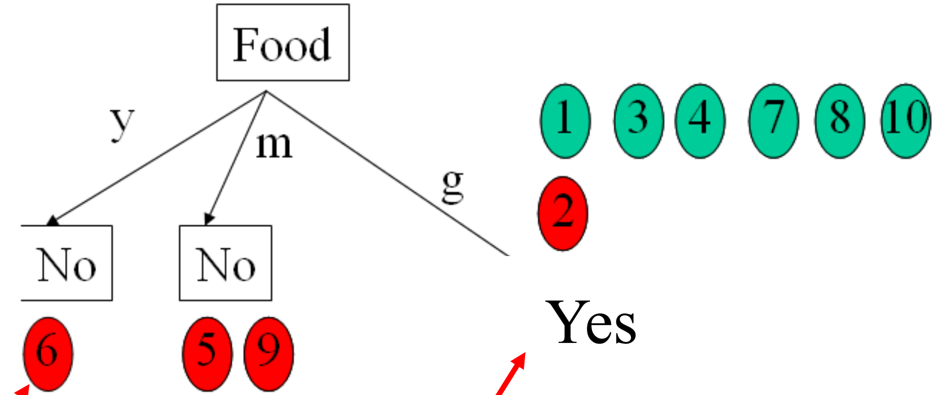
4-

F S P	BigTip
$\vec{x}_1 = (\text{ g, y, a })$	$f(\vec{x}_1) = 1$
$\vec{x}_2 = (\text{ g, n, h })$	$f(\vec{x}_2) = 0$
$\vec{x}_3 = (\text{ g, y, h })$	$f(\vec{x}_3) = 1$
$\vec{x}_4 = (\text{ g, n, a })$	$f(\vec{x}_4) = 1$
$\vec{x}_5 = (\text{ m, y, a })$	$f(\vec{x}_5) = 0$
$\vec{x}_6 = (\text{ y, y, a })$	$f(\vec{x}_6) = 0$
$\vec{x}_7 = (\text{ g, y, a })$	$f(\vec{x}_7) = 1$
$\vec{x}_8 = (\text{ g, y, h })$	$f(\vec{x}_8) = 1$
$\vec{x}_9 = (\text{ m, y, a })$	$f(\vec{x}_9) = 0$
$\vec{x}_{10} = (\text{ g, y, a })$	$f(\vec{x}_{10}) = 1$

How many + and - examples per subclass, starting with y?

Let's consider next the attribute Speedy

Top-Down Induction of DT (simplified)



TDIDF(D,c_{def})

IF(all examples in D have same class c)

– Return leaf with class c (or class c_{def}, if D is empty)

ELSE IF(no attributes left to test)

– Return leaf with class c of majority in D

ELSE

– Pick A as the "best" decision attribute for next node

– FOR each value v_i of A create a new descendent of node

- $D_i = \{(\vec{x}, y) \in D : \text{attribute A of } \vec{x} \text{ has value } v_i\}$
- Subtree t_i for v_i is TDIDT(D_i,c_{def})

– RETURN tree with A as root and t_i as subtrees

Training Data: $D = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\}$

Picking the Best Attribute to Split

Ockham's Razor:

- **All other things being equal, choose the simplest explanation**

Decision Tree Induction:

- **Find the smallest tree that classifies the training data correctly**

Problem

- **Finding the smallest tree is computationally hard ☹!**

Approach

- **Use heuristic search (greedy search)**

Key Heuristics:

- **Pick attribute that *maximizes information (Information Gain)***
  i.e. "most informative"
- **Other statistical tests**

Attribute-based representations

Examples described by attribute values (Boolean, discrete, continuous)

E.g.

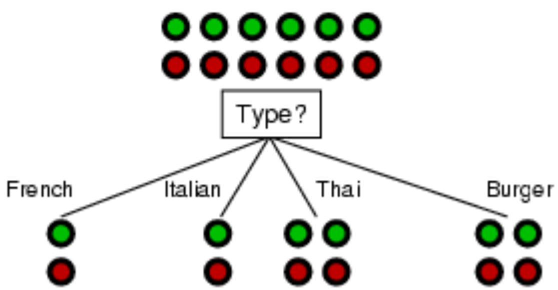
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F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

12 examples
6 +
6 -

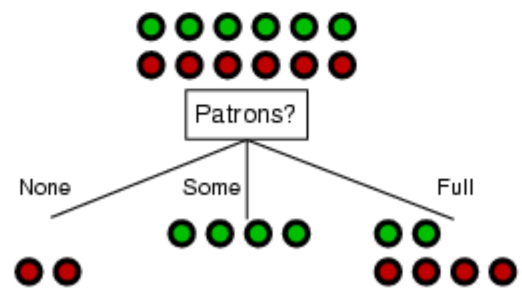
Classification of examples is positive (T) or negative (F)

Choosing an attribute: Information Gain

Goal: trees with short paths to leaf nodes



Is this a good attribute to split on?



Which one should we pick?

A perfect attribute would ideally divide the examples into sub-sets that are all positive or all negative… i.e. **maximum information gain**.

Information Gain

Most useful in classification

- how to measure the 'worth' of an attribute ***information gain***
- how well attribute separates examples according to their classification

Next

- precise definition for gain

→ measure from Information Theory

Shannon and Weaver 49

One of the most successful and impactful mathematical theories known.

Information

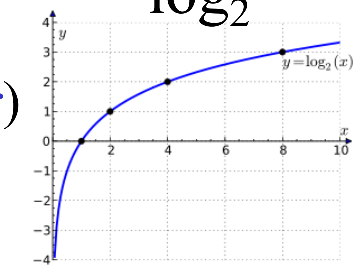
"Information" answers questions. Entropy is a measure of *unpredictability* of *information content*.

The more clueless I am about a question, the more information the **answer** to the question contains.

Example – fair coin → prior <0.5,0.5>

$E[-\log_2(P(x))]$

$\log_2$



By definition Information of the prior (or entropy of the prior)

$I(P1,P2) = - P1 \log_2(P1) - P2 \log_2(P2) =$

$I(0.5,0.5) = -0.5 \log_2(0.5) - 0.5 \log_2(0.5) = 1$

We need 1 bit to convey the outcome of the flip of a fair coin.

Scale: 1 bit = answer to Boolean question with prior <0.5, 0.5>

Does a biased coin have more or less information? Why?

Information (or Entropy)

Information in an answer given possible answers $v_1, v_2, \ldots v_n$:

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

— $v_1, \ldots, v_n$ possible answers

— $P(v_i)$ probabibilty of answer v_i

(Also called entropy of the prior.)

Example – biased coin → prior <1/100,99/100>

$I(1/100,99/100) = -1/100 \log_2(1/100) - 99/100 \log_2(99/100)$
= 0.08 bits (so not much information gained from "answer.")

Example – fully biased coin → prior <1,0>

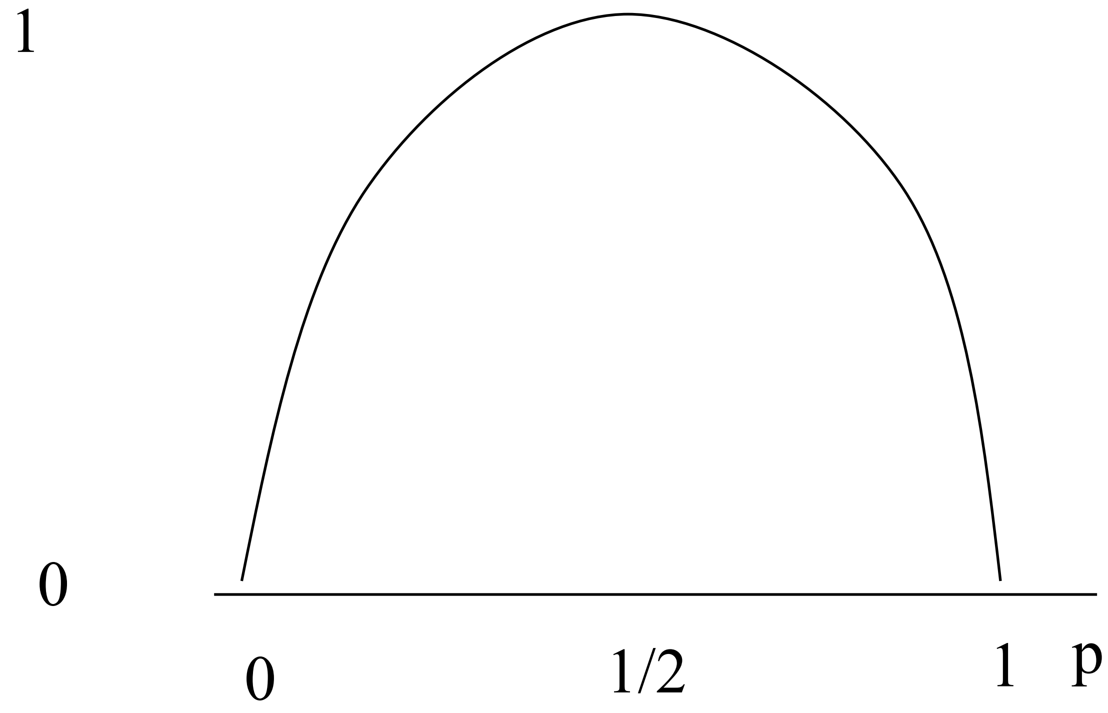
$I(1,0) = -1 \log_2(1) - 0 \log_2(0) = 0$ bits

$0 \log_2(0) = 0$

i.e., no uncertainty left in source!

Shape of Entropy Function

Roll of an unbiased die



The more uniform the probability distribution, the greater is its entropy.

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

Information or Entropy

Information or Entropy measures the "randomness" of an arbitrary collection of examples.

We don't have exact probabilities but our training **data provides an estimate of the probabilities of positive vs. negative examples given a set of values for the attributes.**

For a collection S, entropy is given as:

$$I(\frac{p}{p+n}, \frac{n}{p+n}) = -\frac{p}{p+n}\ \ log_2(\frac{p}{p+n}) - \frac{n}{p+n}\ \ log_2(\frac{n}{p+n})$$

For a collection S having positive and negative examples

p - # positive examples;

n - # negative examples

Attribute-based representations

Examples described by attribute values (Boolean, discrete, continuous)
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*	*Wait*
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

12 examples
6 +
6 -

What' s the entropy of this collection of examples?

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

p = n = 6; I(0.5,0.5) = -0.5 log2(0.5) –0.5 log2(0.5) = 1

So, we need 1 bit of info to classify a randomly picked example,
assuming no other information is given about the example. (Makes sense!)

Choosing an attribute: Information Gain

Intuition: Pick the attribute that reduces the entropy (the uncertainty) the most.

So we measure the information gain after testing a given attribute A:

$$Gain(A) = I(\frac{p}{p+n}, \frac{n}{p+n}) - Remainder(A)$$

Remainder(A) → gives us the remaining uncertainty after getting info on attribute A.

Choosing an attribute: Information Gain

Remainder(A)

→ **gives us the amount information we still need after testing on A.**

Assume A divides the training set E into E_1, E_2, … E_v, corresponding to the different v distinct values of A.

Each subset E_i has p_i positive examples and n_i negative examples.

So for total information content, we need to weigh the contributions of the different subclasses induced by A

Weight (relative size) of each subclass

$$Remainder(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} \, I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

Choosing an attribute: Information Gain

Measures the expected reduction in entropy. The higher the Information Gain (IG), or just Gain, with respect to an attribute A , the more is the expected reduction in entropy.

Weight of each subclass

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where Values(A) is the set of all possible values for attribute A,

S_v is the subset of S for which attribute A has value v.

Interpretations of gain

Gain(S,A)

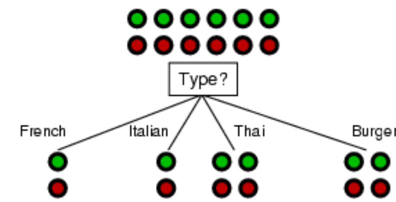
- expected reduction in entropy caused by knowing A
- information provided about the target function value given the value of A
- number of bits saved in the coding a member of S knowing the value of A

Used in ID3 (Iterative Dichotomiser 3) Ross Quinlan

Information gain

For the training set, $p = n = 6$, $I(6/12, 6/12) = 1$ bit

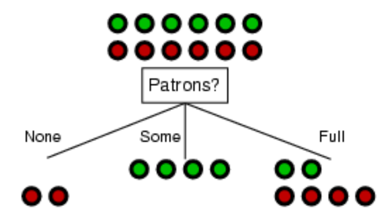
Consider the attributes *Type* and *Patrons*:



Info gain?

$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = 0.541 \text{ bits}$$

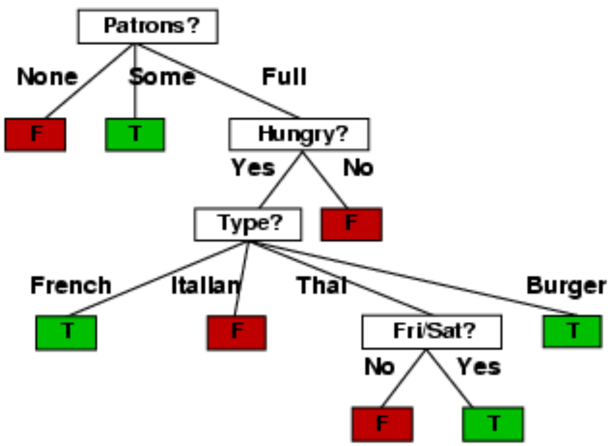


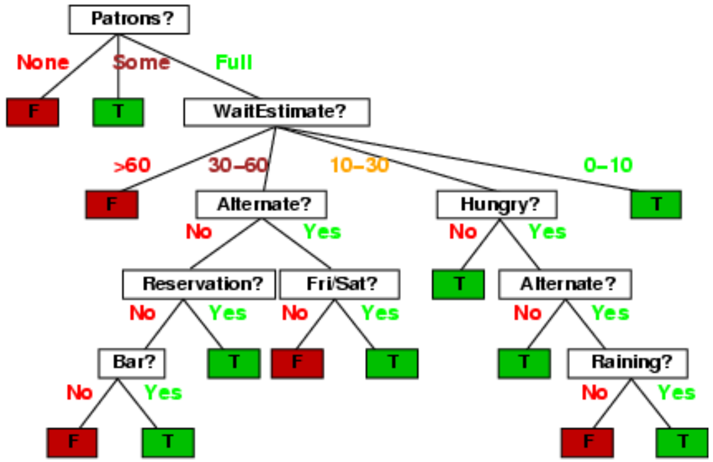
Patrons **has the highest IG of all attributes and so is chosen by the DTL algorithm as the root.**

Example contd.

Decision tree learned from the 12 examples:

“personal R&N Tree”





Substantially simpler than “true” tree ---
but a more complex hypothesis isn’t justified
from just the data.

Expressiveness of Decision Trees

Any particular decision tree hypothesis for WillWait goal predicate can be seen as a disjunction of a conjunction of tests, i.e., an assertion of the form:

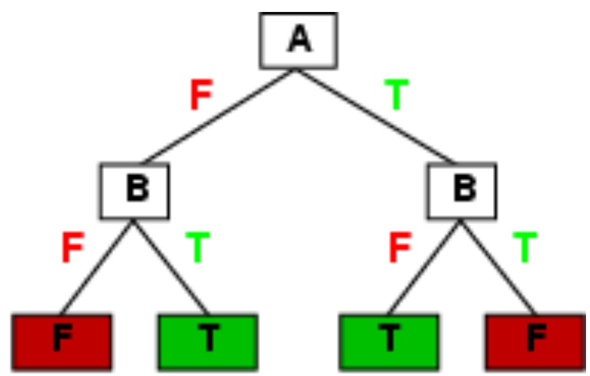
$$\forall s \;\; WillWait(s) \leftrightarrow (P1(s) \lor P2(s) \lor \ldots \lor Pn(s))$$

Where each condition Pi(s) is a conjunction of tests corresponding to the path from the root of the tree to a leaf with a positive outcome.

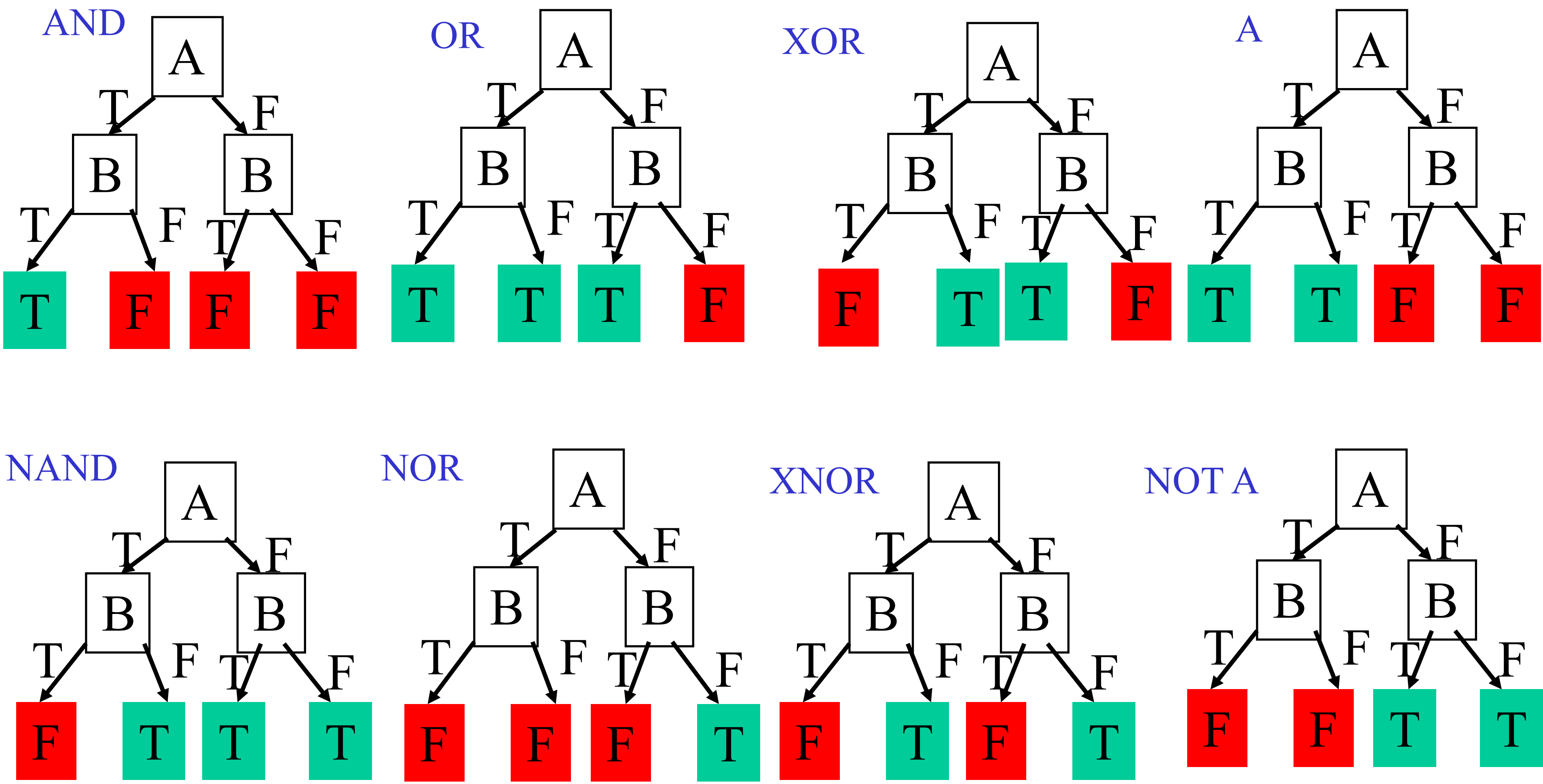
Expressiveness

Decision trees can express any Boolean function of the input attributes.
E.g., for Boolean functions, truth table row → path to leaf:

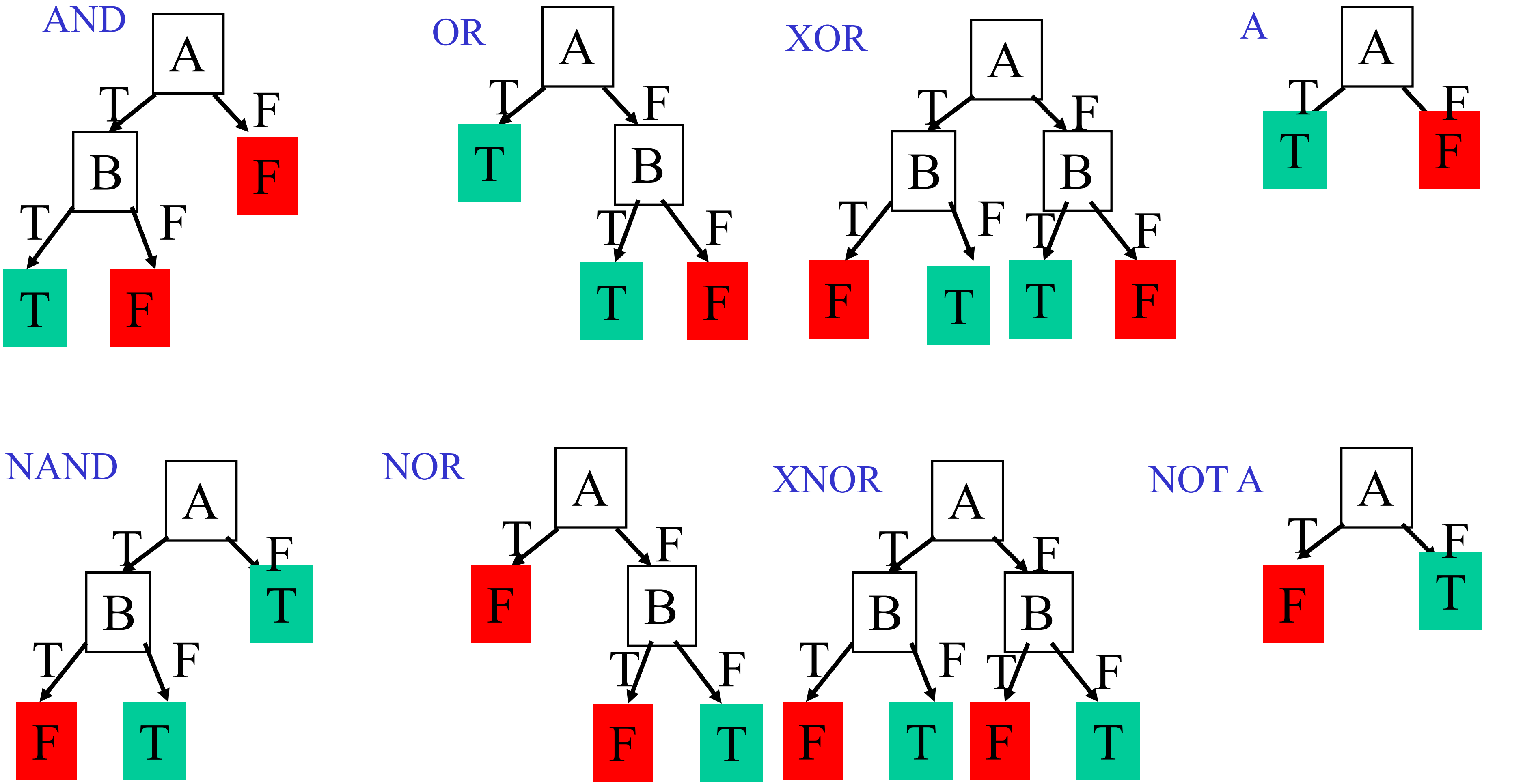
A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Expressiveness:
Boolean Function with 2 attributes → 2^{2^2} DTs

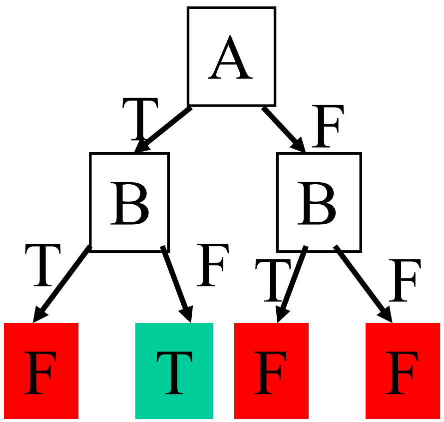


Expressiveness: 2 attribute → 2^{2^2} DTs

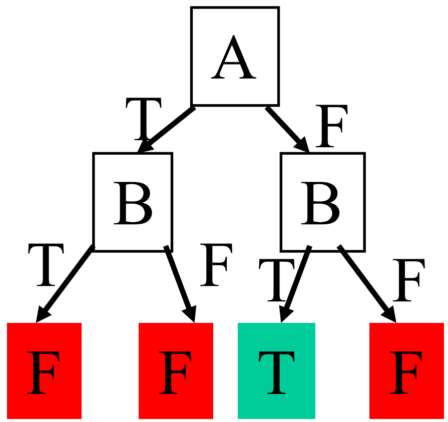


Expressiveness: 2 attribute → 2^{2^2} DTs

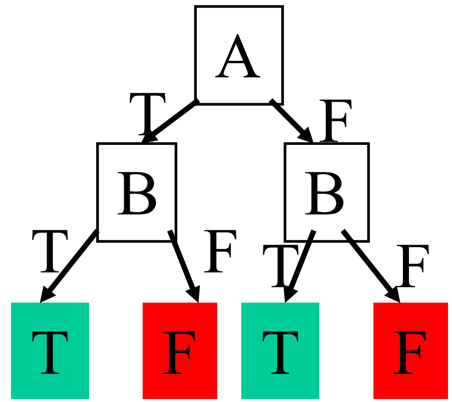
A AND-NOT B



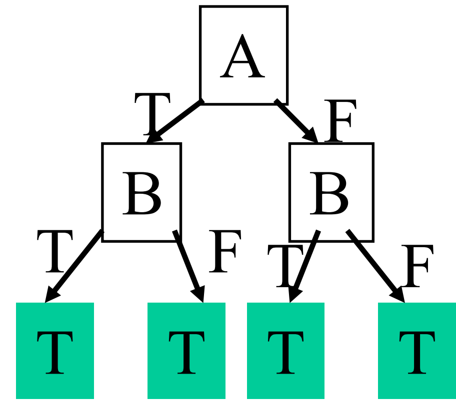
NOT A AND B



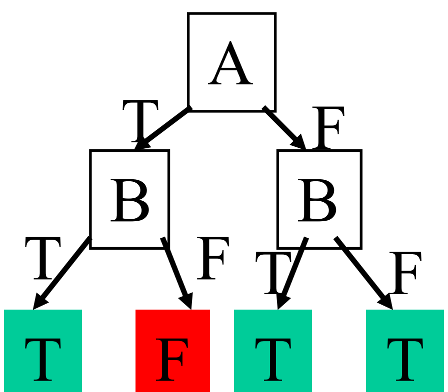
B



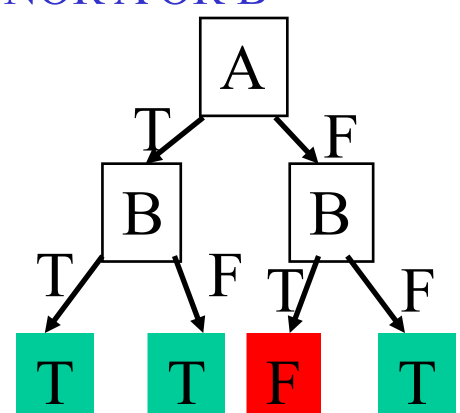
TRUE



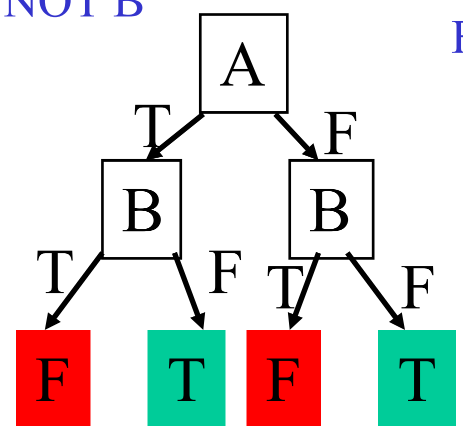
A OR NOT B



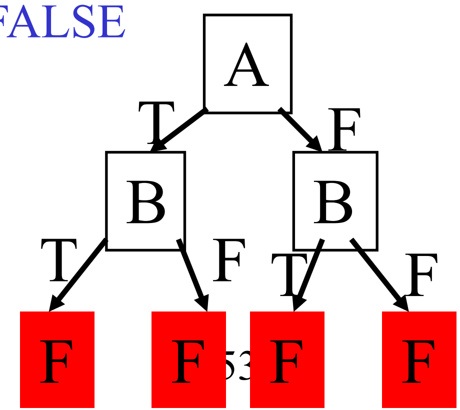
NOR A OR B



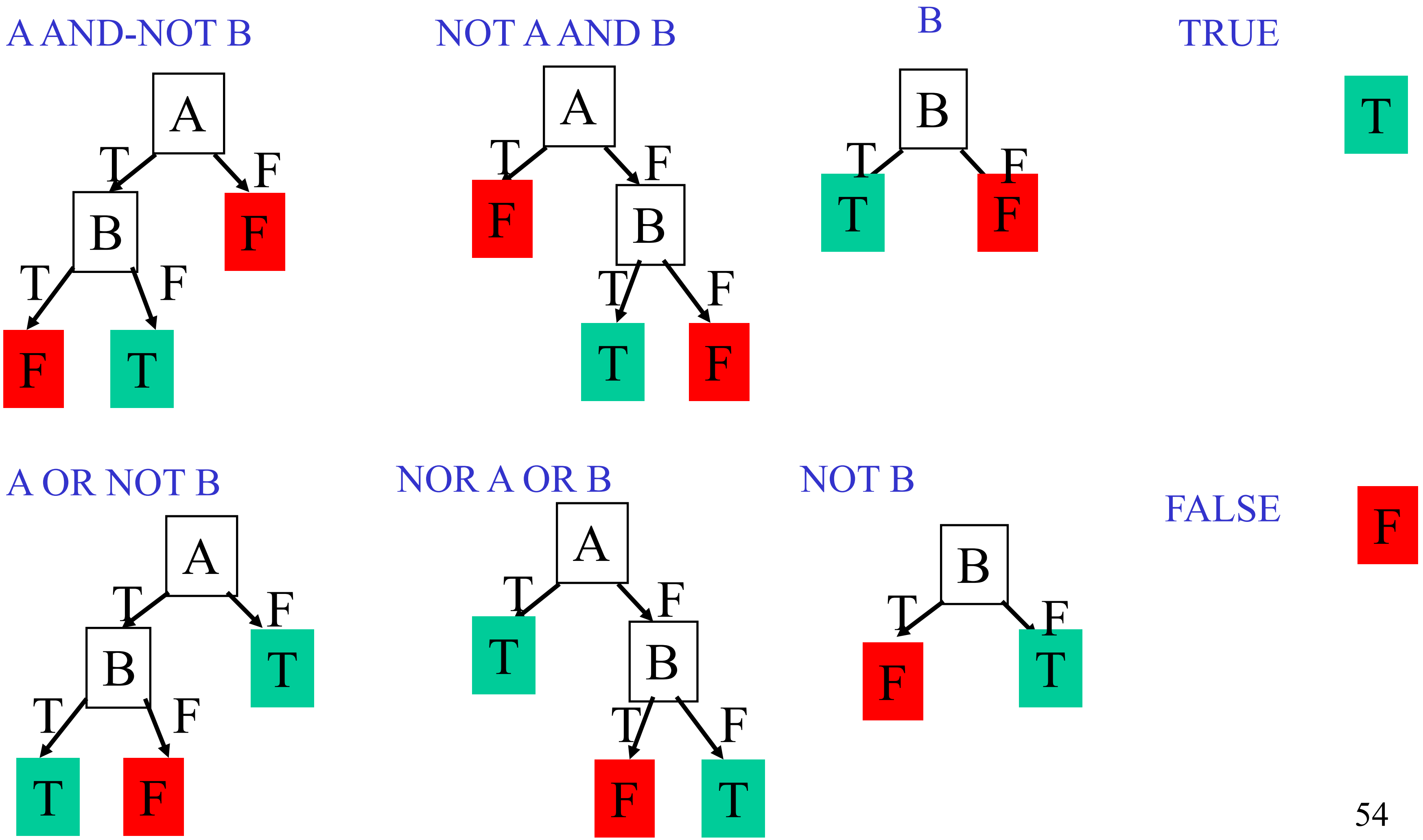
NOT B



FALSE



Expressiveness:
2 attribute → 2^{2^2} DTs



Number of Distinct Decision Trees

How many distinct decision trees with *10* Boolean attributes?

= number of Boolean functions with 10 propositional symbols

Input features	Output
0 0 0 0 0 0 0 0 0 0	0/1
0 0 0 0 0 0 0 0 0 1	0/1
0 0 0 0 0 0 0 0 1 0	0/1
0 0 0 0 0 0 0 1 0 0	0/1
…	
1 1 1 1 1 1 1 1 1 1	0/1

How many entries does this table have?

2^{10}

So how many Boolean functions with 10 Boolean attributes are there, given that each entry can be 0/1?

$$= 2^{2^{10}}$$

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. how many Boolean functions on 6 attributes? A lot…

With 6 Boolean attributes, there are 18,446,744,073,709,551,616 possible trees!

Googles calculator could not handle 10 attributes ☺!

Evaluation Methodology
General for Machine Learning

Evaluation Methodology

How to evaluate the quality of a learning algorithm, i.e.,:

How good are the hypotheses produce by the learning algorithm?
How good are they at classifying unseen examples?

Standard methodology **("Holdout Cross-Validation")**:

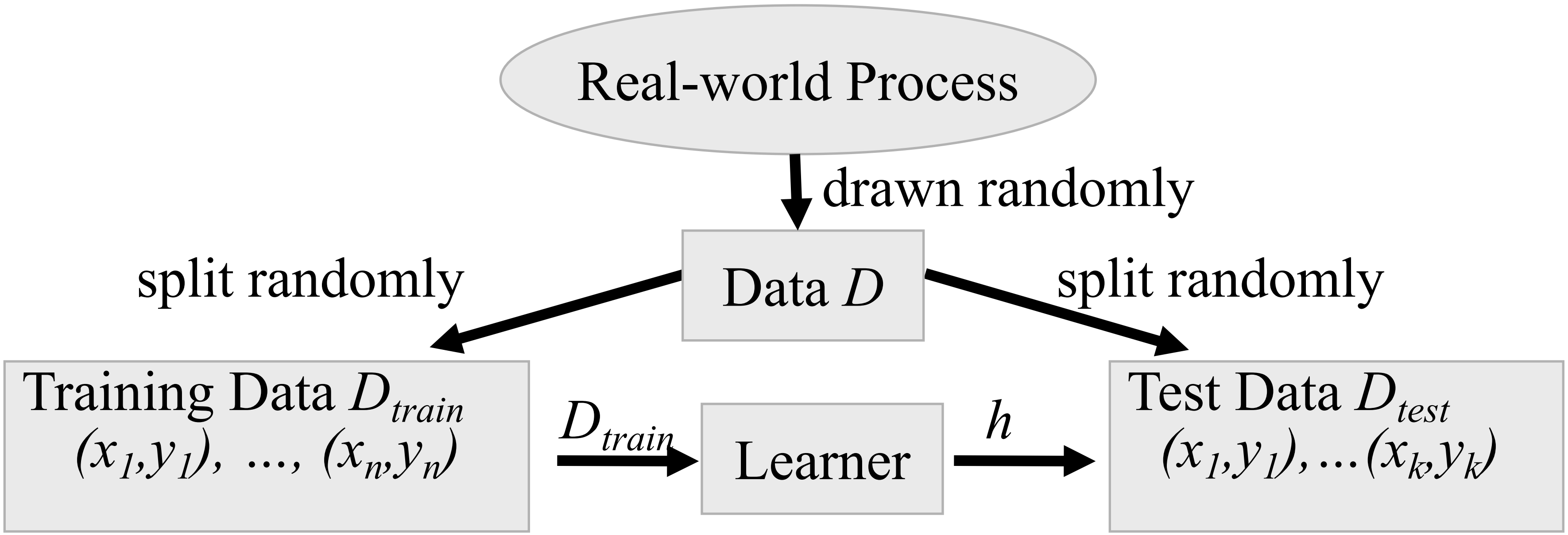
1. Collect a large set of examples.
2. Randomly divide collection into two disjoint sets: **training set** and **test set**.
3. Apply learning algorithm to training set generating hypothesis *h*
4. Measure performance of *h* w.r.t. test set (a form of cross-validation)

→ measures generalization to unseen data

Important: keep the training and test sets disjoint! "No peeking"!

Note: The first two questions about any learning result: Can you describe your training and your test set? What's your error on the test set?

Test/Training Split



Also validation set for meta-parametres.

Measuring Prediction Performance

Definition: *The* **training error** $Err_{D_{train}}(h)$ *on training data* $D_{train} = ((\vec{x}_1, y_1), ..., (\vec{x}_n, y_n))$ *of a hypothesis* h *is* $Err_{D_{train}}(h) = \frac{1}{n}\sum_{i=1}^{n} \Delta(h(\vec{x}_i), y_i)$.

Definition: *The* **test error** $Err_{D_{test}}(h)$ *on test data* $D_{test} = ((\vec{x}_1, y_1), ..., (\vec{x}_k, y_k))$ *of a hypothesis* h *is* $Err_{D_{test}}(h) = \frac{1}{k}\sum_{i=1}^{k} \Delta(h(\vec{x}_i), y_i)$.

Definition: *The* **prediction/generalization/true error** $Err_P(h)$ *of a hypothesis* h *for a learning task* $P(X, Y)$ *is*

$$Err_P(h) = \sum_{\vec{x} \in X, y \in Y} \Delta(h(\vec{x}), y) P(X = \vec{x}, Y = y).$$

Definition: *The* zero/one-loss function $\Delta(a, b)$ *returns* 1 *if* $a \neq b$ *and* 0 *otherwise.*

Performance Measures

Error Rate

– Fraction (or percentage) of false predictions

Accuracy

– Fraction (or percentage) of correct predictions

Precision/Recall

Example: binary classification problems (classes pos/neg)

– Precision: Fraction (or percentage) of correct predictions among all examples predicted to be positive

– Recall: Fraction (or percentage) of correct predictions among all real positive examples

(Can be generalized to multi-class case.)

Extensions of the Decision Tree Learning Algorithm

Noisy data

Overfitting and Model Selection

Cross Validation

Missing Data (R&N, Section 18.3.6)

Using gain ratios (R&N, Section 18.3.6)

Real-valued data (R&N, Section 18.3.6)

Generation of rules and pruning

DT Ensembles

Regression DT

How well does it work?

Many case studies have shown that decision trees are at least as accurate as human experts.

- A study for diagnosing breast cancer had humans correctly classifying the examples 65% of the time, and the decision tree classified 72% correct.
- British Petroleum designed a decision tree for gas-oil separation for offshore oil platforms that replaced an earlier rule-based expert system.
- Cessna designed an airplane flight controller using 90,000 examples and 20 attributes per example.

Bird Distributions
Machine Learning and Citizen Science

eBird

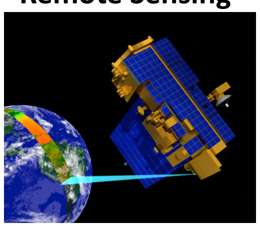
Bird Observations

300K+ volunteer birders	300M+ bird observations	22M+ hours of field work (2500+years)

Land Cover

Weather

Remote Sensing



Environmental Data

TeraGrid

80,000+
CPU Hours
(~ 10 Years!!!)

Adaptive Spatio-Temporal Machine Learning Models and Algorithms (STEM Models)

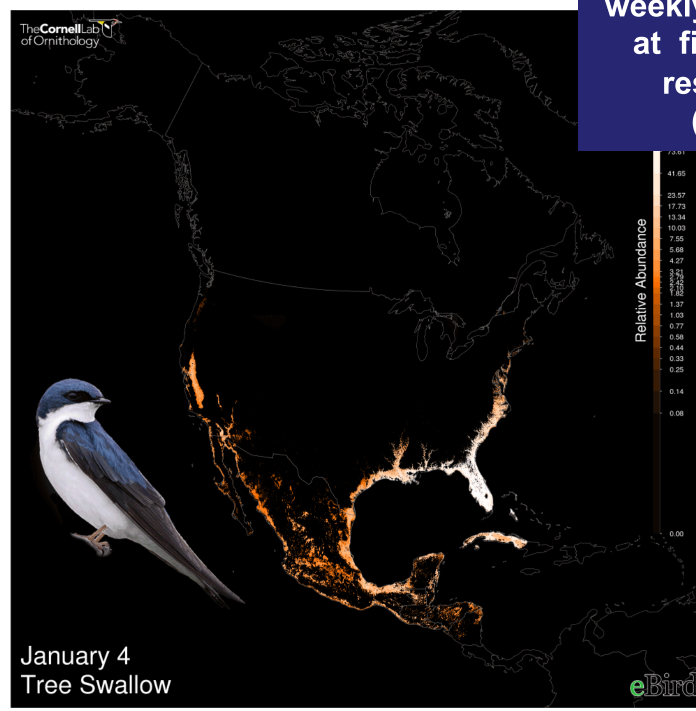
Boosted Regression DT Ensemble

Relate environmental predictors to observed patterns of occurrences and absences

State of the Birds Report
(officially released by Secretary of Interior)

Novel Approaches
To Conservation
Based on eBird Models

Distribution Models for 400+ species with weekly estimates at fine spatial resolution ($3km^2$)



Patterns of occurrence of the Tree Swallow for different months of the year **Source : Daniel Fink**

Bird Distribution Models, Revealing, at a fine resolution, Species' Habitat Preferences

Summary:
When to use Decision Trees

Instances presented as attribute-value pairs

Method of approximating discrete-valued functions

Target function has discrete values: classification problems

Robust to noisy data:

Training data may contain

- errors
- missing attribute values

Typical bias: prefer smaller trees (Ockham's razor)

Widely used, practical and easy to interpret results

Inducing decision trees is one of the most widely used learning methods in practice

Can outperform human experts in many problems

Strengths include

- Fast
- simple to implement
- human readable
- can convert result to a set of easily interpretable rules
- empirically valid in many commercial products
- handles noisy data

Can be a legal requirement! Why?

Weaknesses include:

- "Univariate" splits/partitioning using only one attribute at a time so limits types of possible trees
- large decision trees may be hard to understand
- requires fixed-length feature vectors
- non-incremental (i.e., batch method)