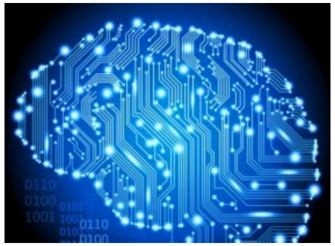
Lecture 12: Learning 1

http://cs.nju.edu.cn/yuy/course_ai17.ashx



Previously...

Search

- Path-based search
- Iterative improvement search

Knowledge

- Propositional Logic
- First Order Logic (FOL)

Uncertainty

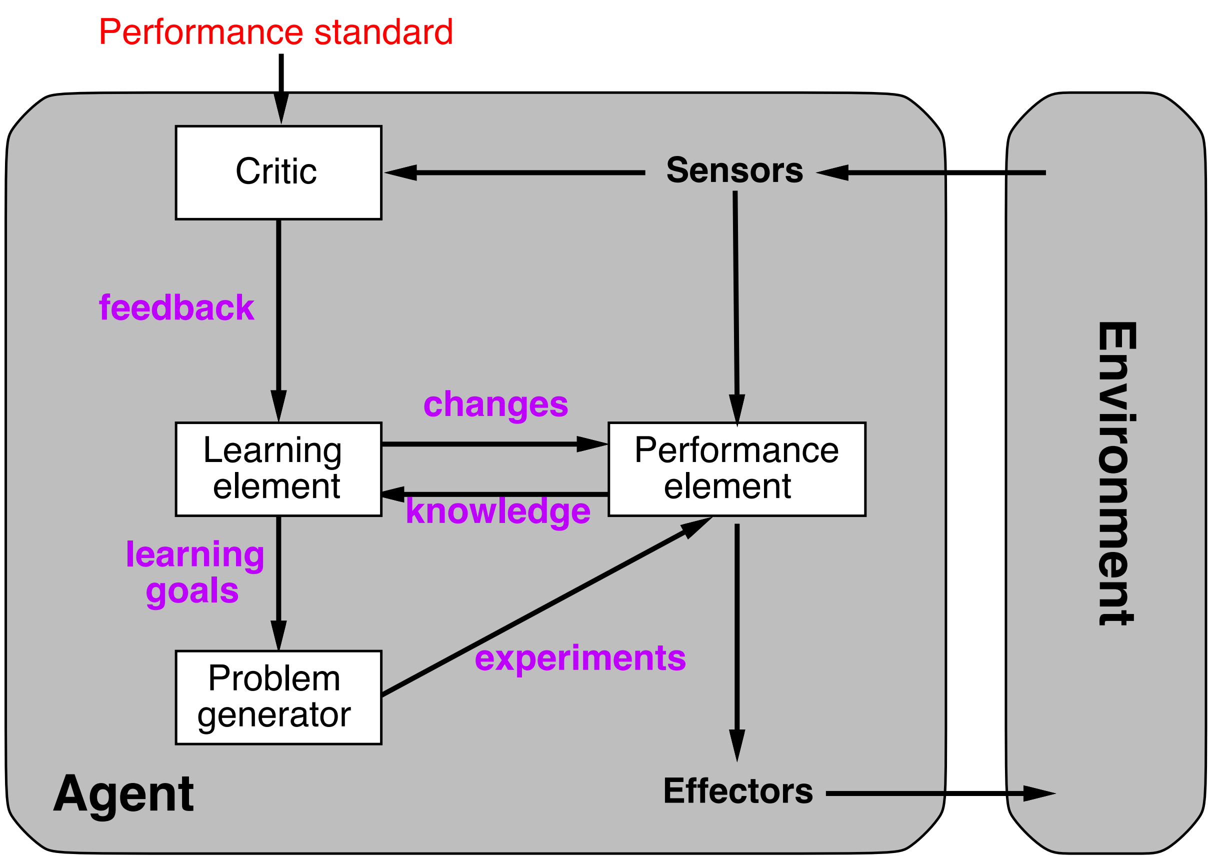
- Bayesian network

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



Inductive Learning

Simplest form: learn a function from examples (**tabula rasa**)

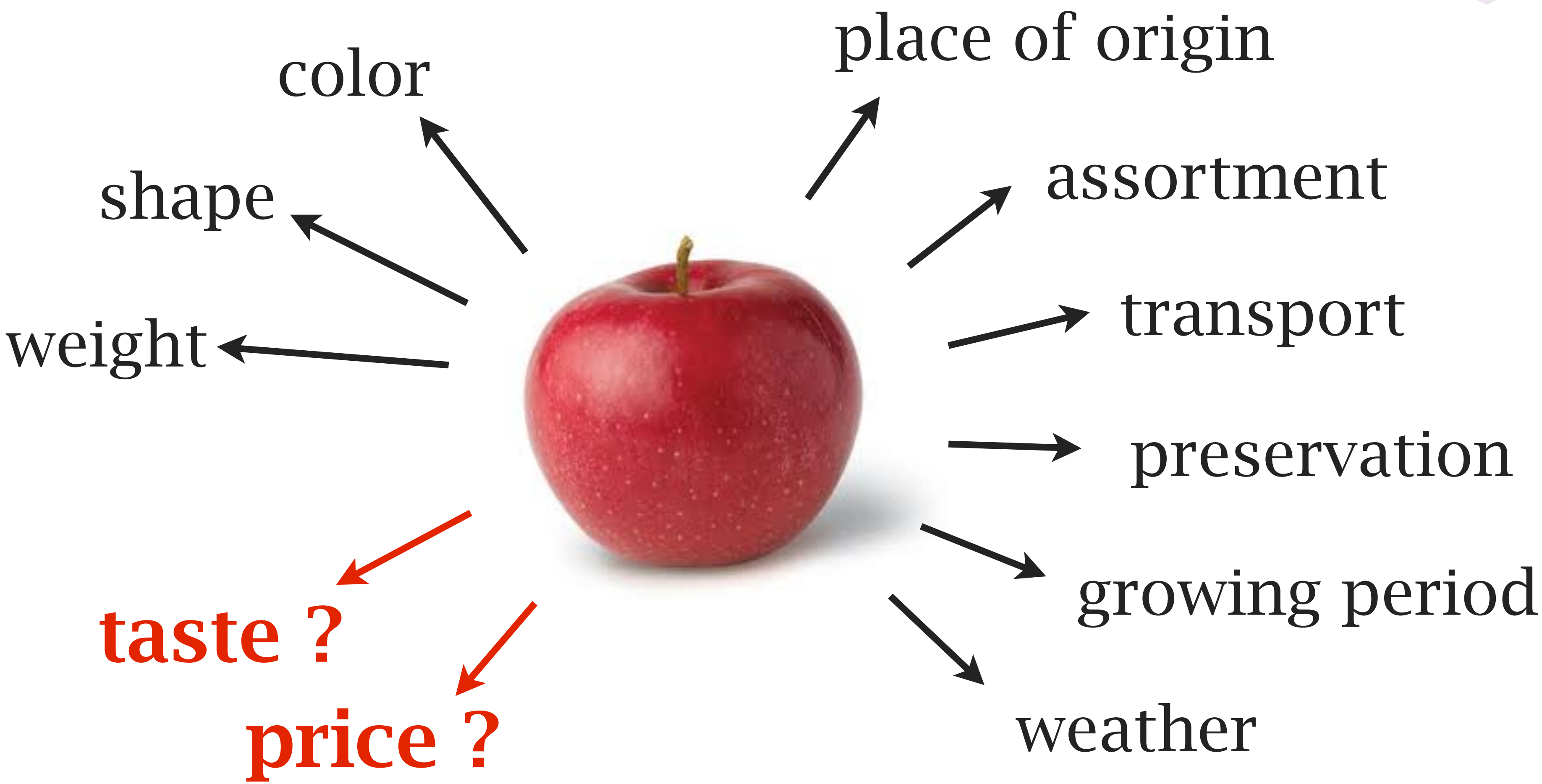
f is the target function

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

Problem: find a(n) hypothesis h
such that $h \approx f$
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**
- **Assumes that the agent wants to learn f—why?**)

Attribute-based representations



Attribute-based representations

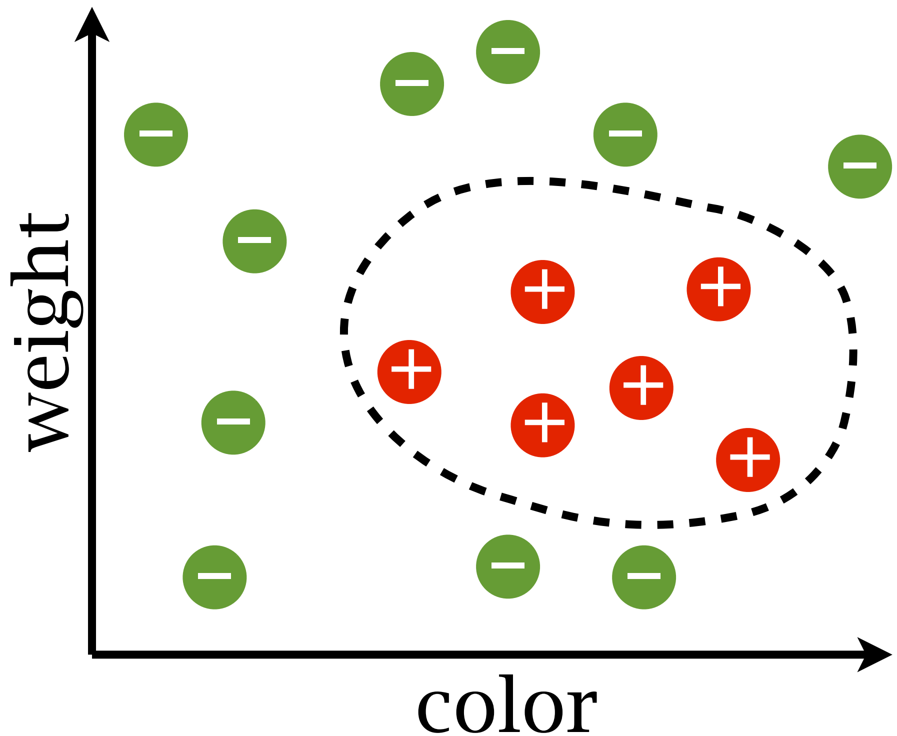
Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

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)

Learning task: Classification

Features: color, weight
Label: taste is sweet (positive/+) or not (negative/-)



(color, weight) $\rightarrow$ sweet ?

$\mathcal{X} \qquad \rightarrow \{-1, +1\}$

ground-truth function f

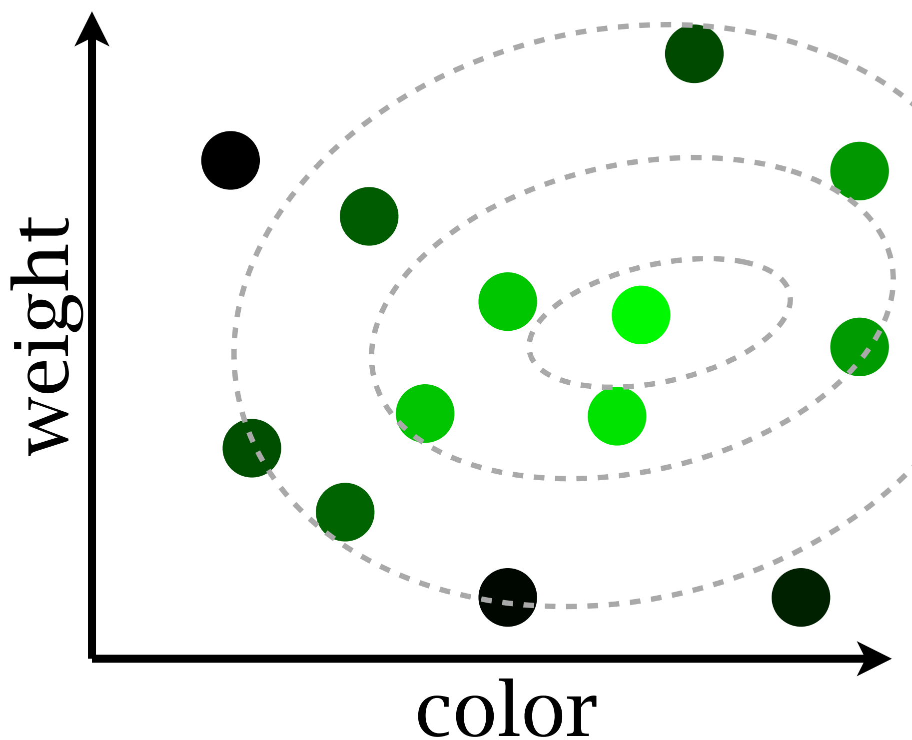
examples/training data:
$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$

$y_i = f(\boldsymbol{x}_i)$

learning: <u>find</u> an f' that is <u>close</u> to f

Learning task: Regression

Features: color, weight
Label: price [0,1]



(color, weight) → price

$\mathcal{X} \rightarrow [0, +1]$

ground-truth function f

examples/training data:
$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$

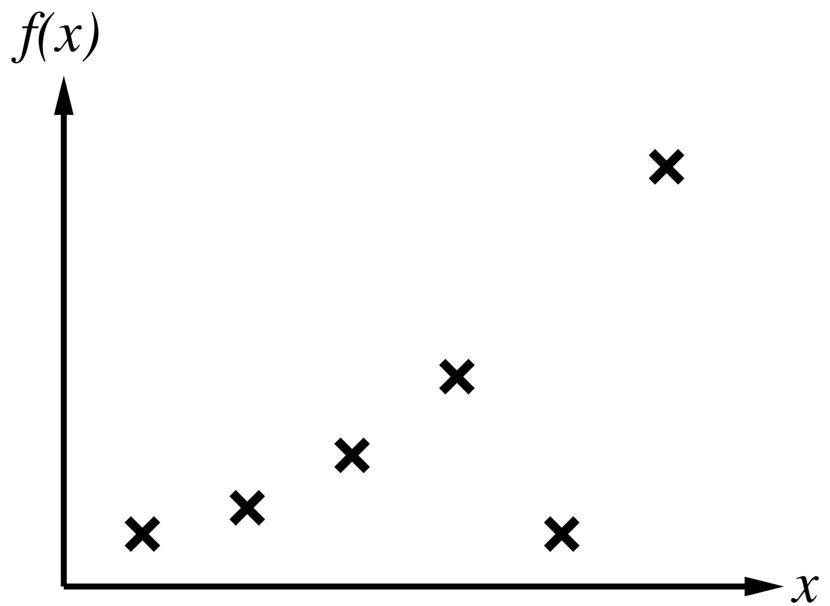
$y_i = f(\boldsymbol{x}_i)$

learning: find an f' that is close to f

Learning task: Regression

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

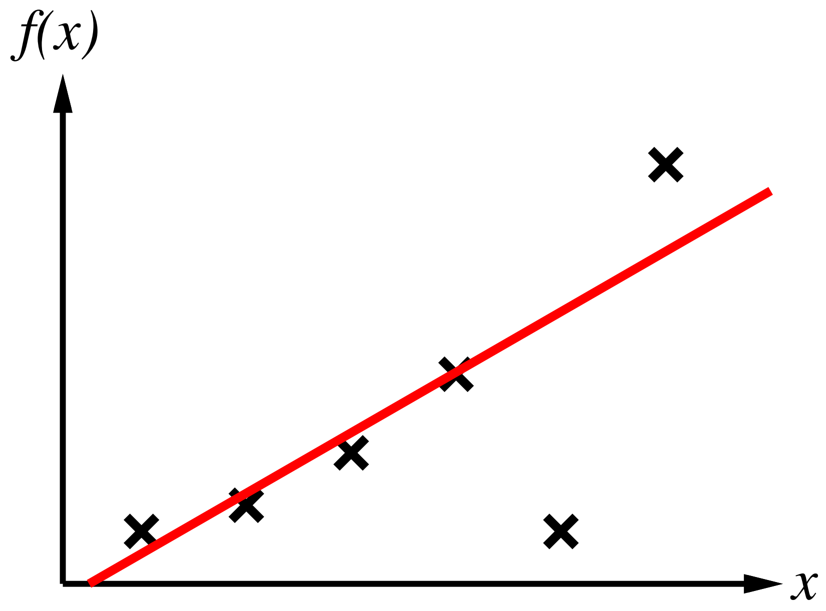
E.g., curve fitting:



Learning task: Regression

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

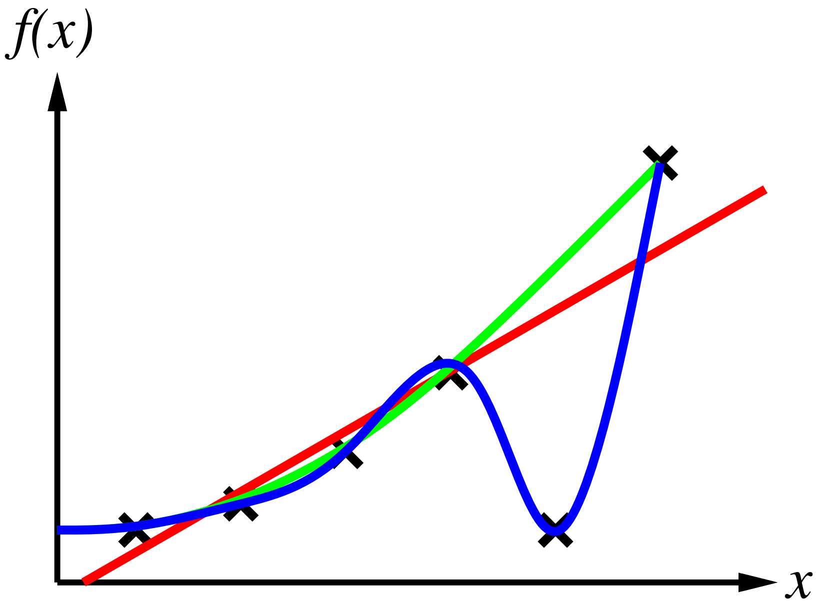
E.g., curve fitting:



Learning task: Regression

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

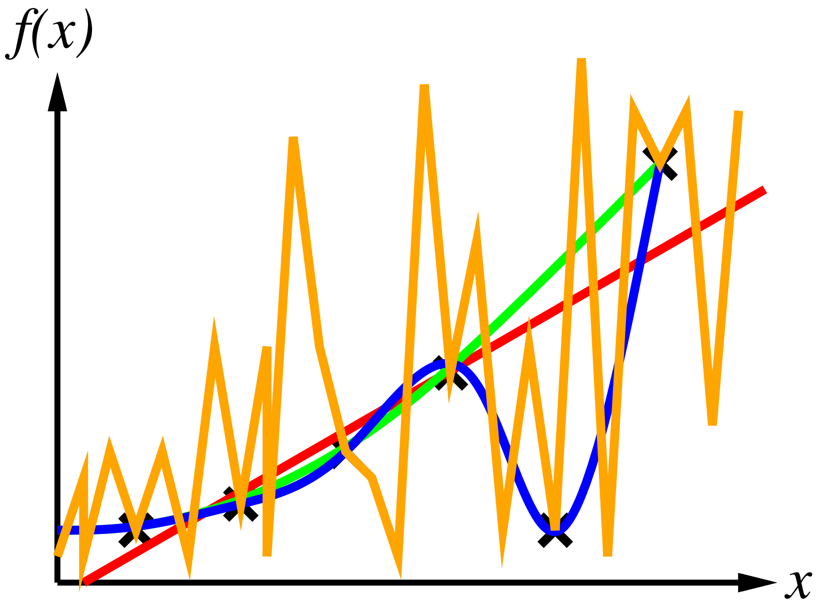
E.g., curve fitting:



Learning task: Regression

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

E.g., curve fitting:



how to learn? why it can learn?

Learning algorithms

Decision tree

Neural networks

Linear classifiers

Bayesian classifiers

Lazy classifiers

...

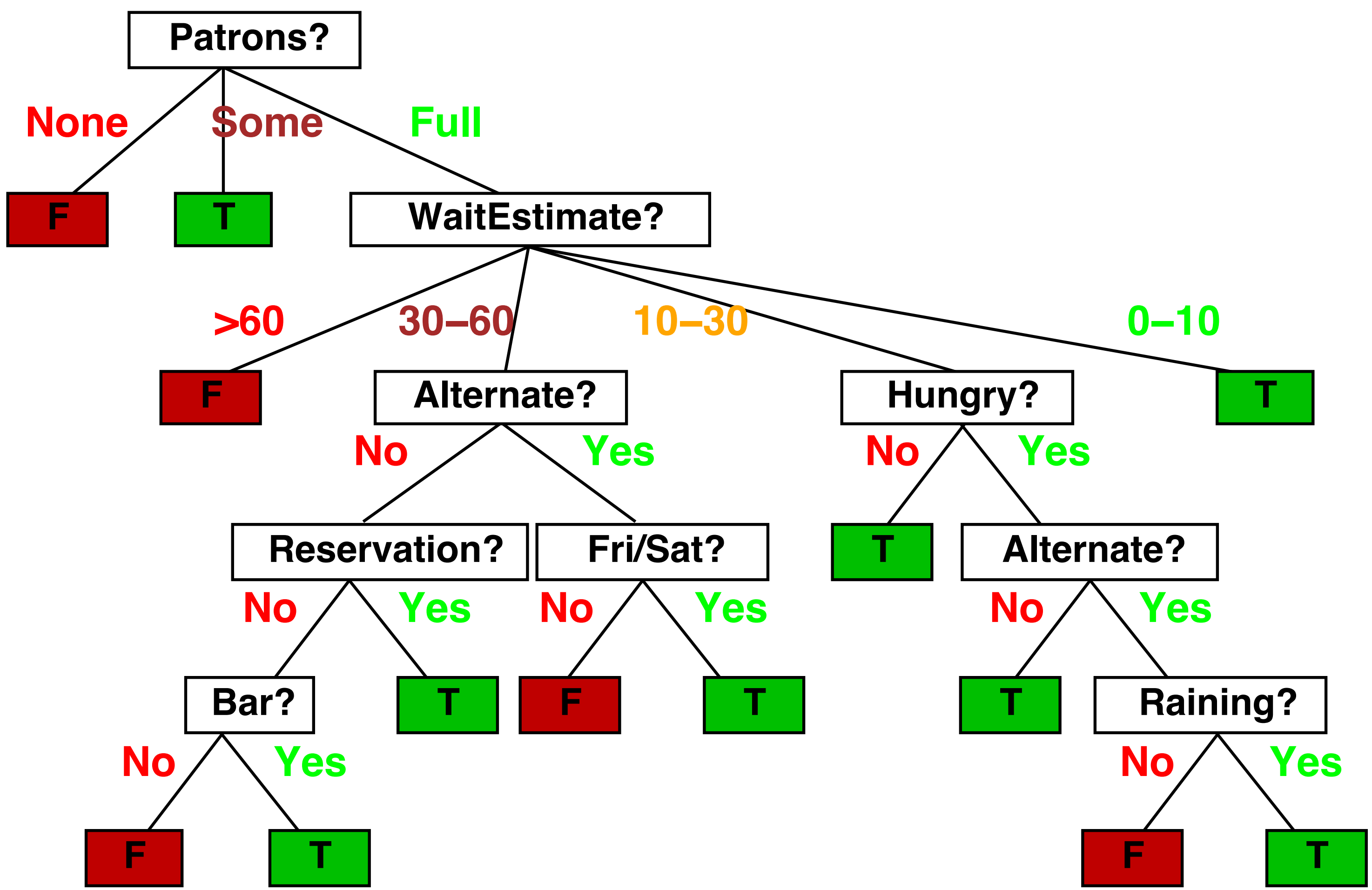
Why different classifiers?

- heuristics
- viewpoint
- performance

Decision tree learning

what is a decision tree

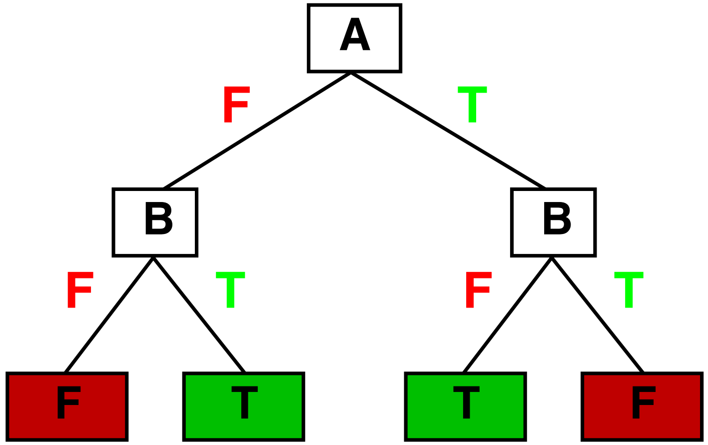
One possible representation for hypotheses
E.g., here is the "true" tree for deciding whether to wait:



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



Trivially, 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 (all possible trees)

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

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

Decision tree learning algorithm

Aim: find a small tree consistent with the training examples

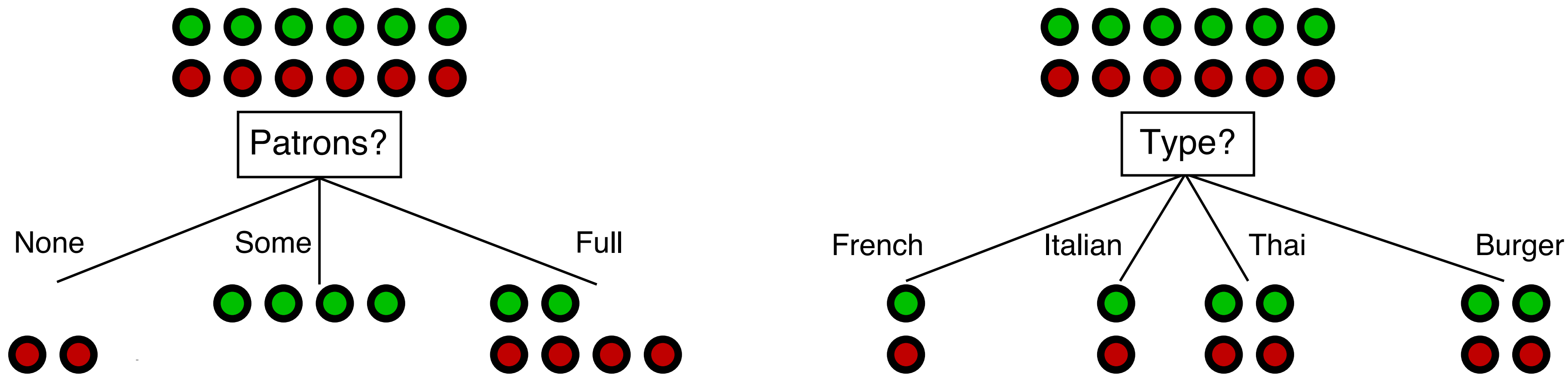
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

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*

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 I am 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

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

(also called entropy of the prior)

Information

Suppose we have p positive and n negative examples at the root
$\Rightarrow$ $H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a 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, each of which (we hope) needs less information to complete the classification

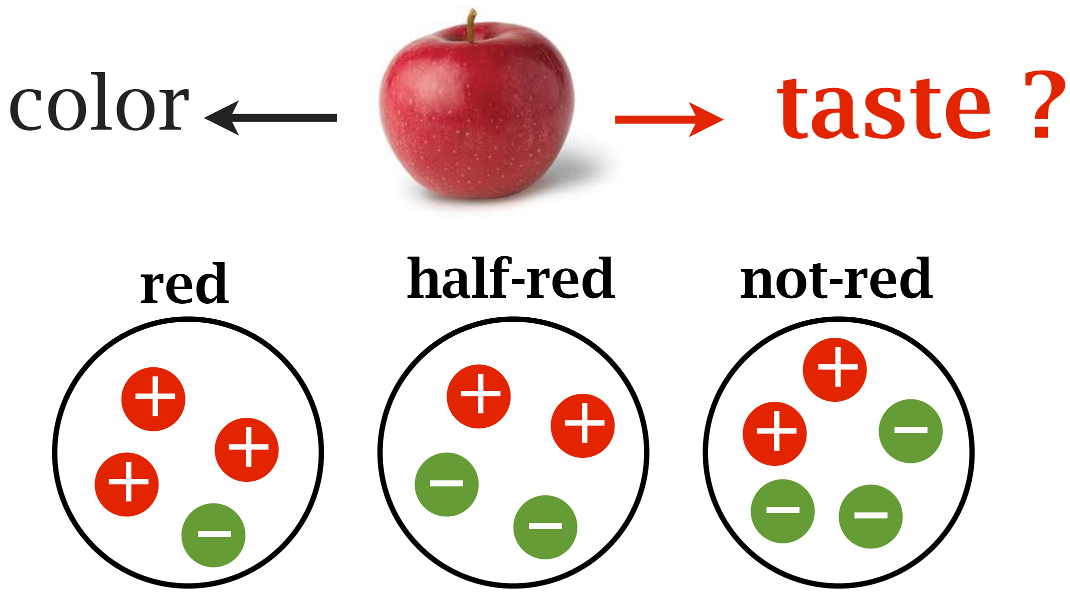
Let E_i have p_i positive and n_i negative examples
$\Rightarrow$ $H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify a new example
$\Rightarrow$ **expected** number of bits per example over all branches is

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

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

$\Rightarrow$ choose the attribute that minimizes the remaining information needed

Example



id	color	taste
1	red	sweet
2	red	sweet
3	half-red	sweet
4	not-red	sweet
5	not-red	not-sweet
6	half-red	sweet
7	red	not-sweet
8	not-red	not-sweet
9	not-red	sweet
10	half-red	not-sweet
11	red	sweet
12	half-red	not-sweet
13	not-red	not-sweet

information gain:

entropy before split: $H(X) = -\sum_i ratio(class_i) \ln ratio(class_i) = 0.6902$

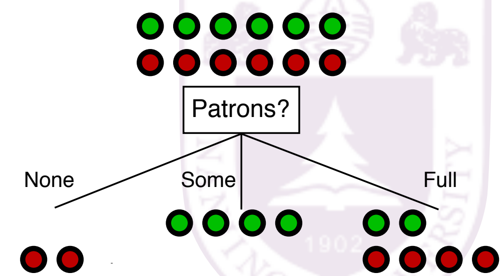
entropy after split: $I(X;\text{split}) = \sum_i ratio(split_i) H(split_i)$

$$= \frac{4}{13}0.5623 + \frac{4}{13}0.6931 + \frac{5}{13}0.6730 = 0.6452$$

information gain:

$$Gain(X; split) = H(X) - I(X;\text{split}) = 0.045$$

Decision tree learning algorithm



Aim: find a small tree consistent with the training examples

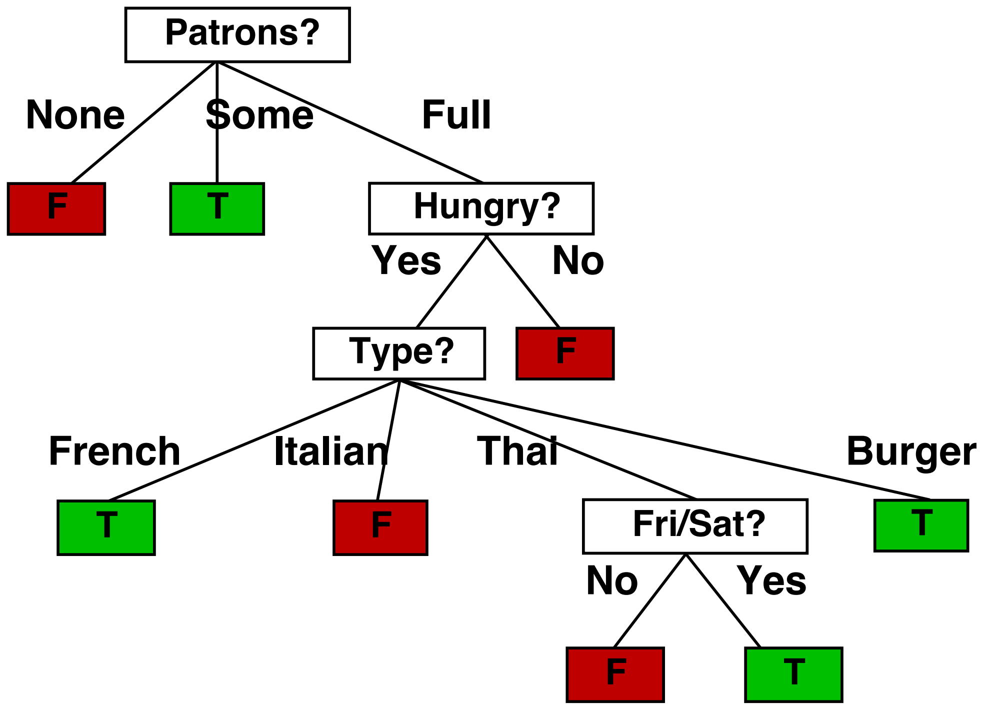
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

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**
            $examples_i \leftarrow$ {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*

Example of learned tree

Decision tree learned from the 12 examples:

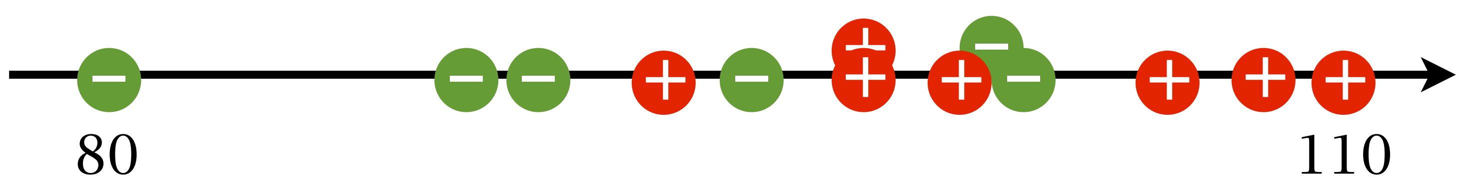


Substantially simpler than “true” tree—a more complex hypothesis isn’t justified by small amount of data

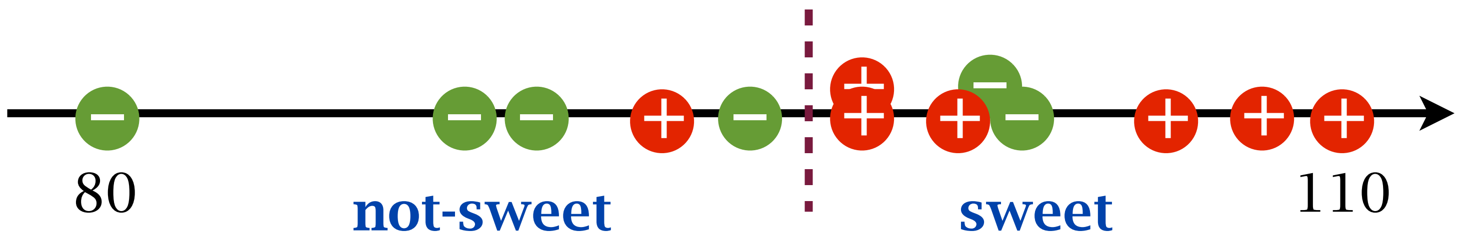
Continuous attribute

weight  taste ?

id	weight	taste
1	110	sweet
2	105	sweet
3	100	sweet
4	93	sweet
5	80	not-sweet
6	98	sweet
7	95	not-sweet
8	102	not-sweet
9	98	sweet
10	90	not-sweet
11	108	sweet
12	101	not-sweet
13	89	not-sweet



Continuous attribute



for every split point

information gain:

entropy before split: $H(X) = -\sum_i ratio(class_i) \ln ratio(class_i) = 0.6902$

entropy after split: $I(X; \text{split}) = \sum_i ratio(split_i) H(split_i)$

$$= \frac{5}{13} 0.5004 + \frac{8}{13} 0.5623 = 0.5385$$

information gain:

$$Gain(X; split) = H(X) - I(X; \text{split}) = 0.1517$$

Non-generalizable feature

id	color	weight	taste
1	red	110	sweet
2	red	105	sweet
3	half-red	100	sweet
4	not-red	93	sweet
5	not-red	80	not-sweet
6	half-red	98	sweet
7	red	95	not-sweet
8	not-red	102	not-sweet
9	not-red	98	sweet
10	half-red	90	not-sweet
11	red	108	sweet
12	half-red	101	not-sweet
13	not-red	89	not-sweet

the system may not know non-generalizable features

$$\mathrm{IG} = H(X) - 0$$

Gain ratio as a correction:

$$\text{Gain ratio}(X) = \frac{H(X) - I(X;\text{split})}{IV(\text{split})}$$

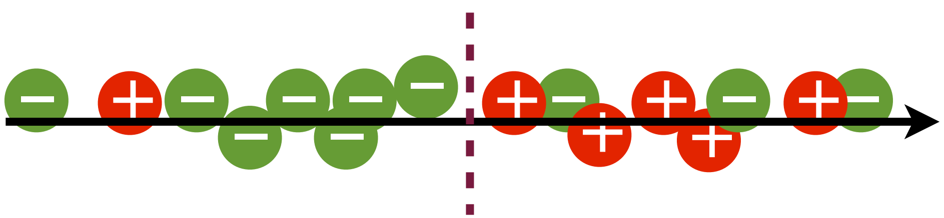
$$IV(\text{split}) = H(\text{split})$$

Alternative to information: Gini index

Gini index (CART):

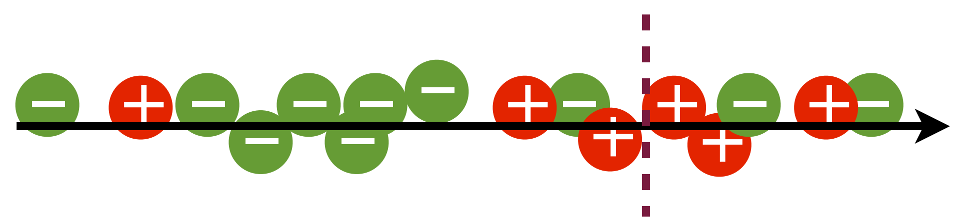
Gini: $Gini(X) = 1 - \sum_i p_i^2$

Gini after split: $\frac{\#\text{left}}{\#\text{all}} Gini(\text{left}) + \frac{\#\text{right}}{\#\text{all}} Gini(\text{right})$



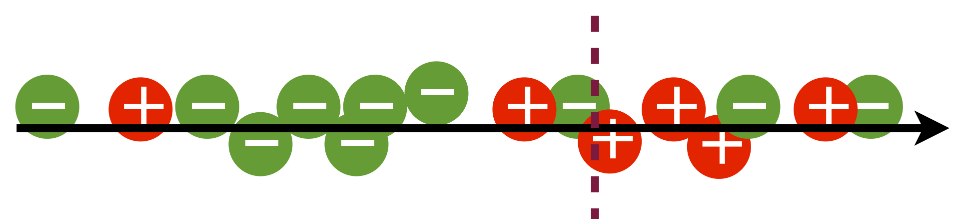
$\text{IG} = H(X) - 0.5192$

$Gini = 0.3438$



$\text{IG} = H(X) - 0.6132$

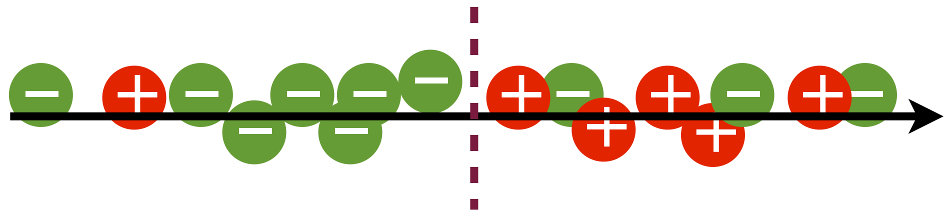
$Gini = 0.4427$



$\text{IG} = H(X) - 0.5514$

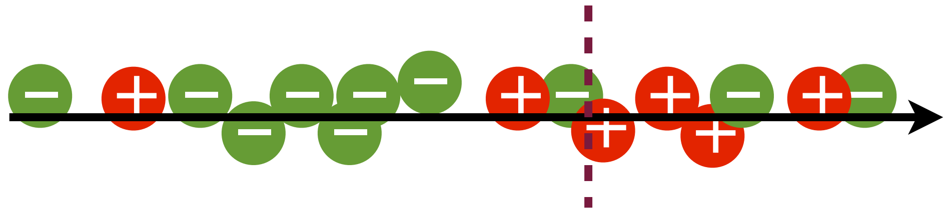
$Gini = 0.3667$

Training error v.s. Information gain



training error: 4

information gain: $\mathrm{IG} = H(X) - 0.5192$



training error: 4

information gain: $\mathrm{IG} = H(X) - 0.5514$

training error is less smooth

Decision tree learning algorithms

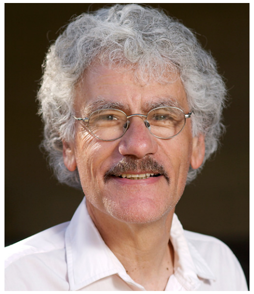
ID3: information gain

C4.5: gain ratio, handling missing values

Ross Quinlan

CART: gini index

Leo Breiman 1928-2005



Jerome H. Friedman