Decision Tree

CE417: Introduction to Artificial Intelligence
Sharif University of Technology
Fall 2023

Soleymani

Slides are based on Klein and Abbeel, CS188, UC Berkeley.

Decision Tree

- One of the most intuitive classifiers that is easy to understand and construct
  - However, it also works very (very) well

- Categorical features are preferred. If feature values are continuous, they are discretized first.

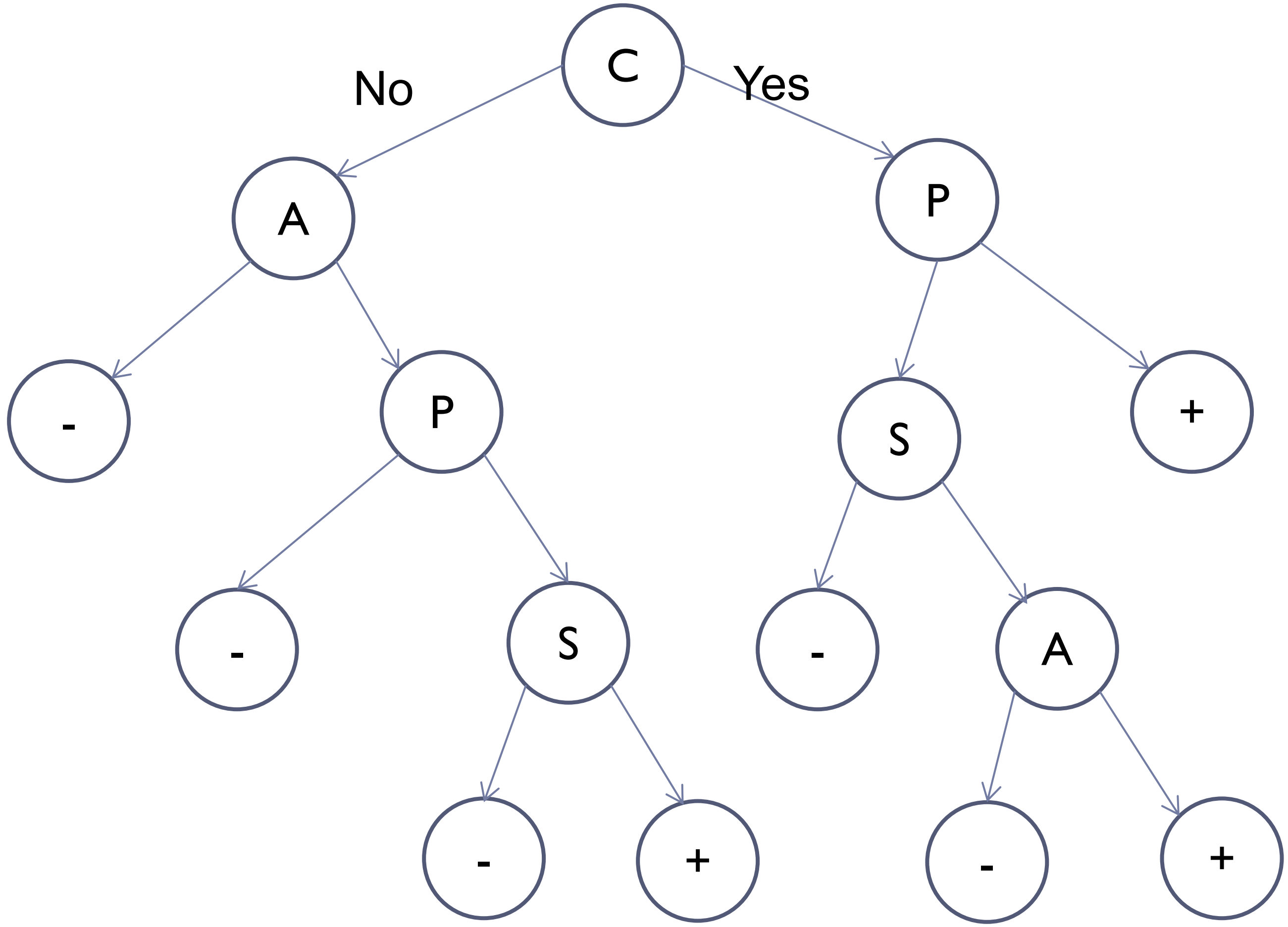
- Application: Database mining

Structure of Decision Tree

- Leaves (terminal nodes) represent target variable
  - Each leaf represents a class label
- Each internal node denotes a test on an attribute
  - Edges to children for each of the possible values of that attribute

Example

- Attributes:
  - A: age>40
  - C: chest pain
  - S: smoking
  - P: physical test

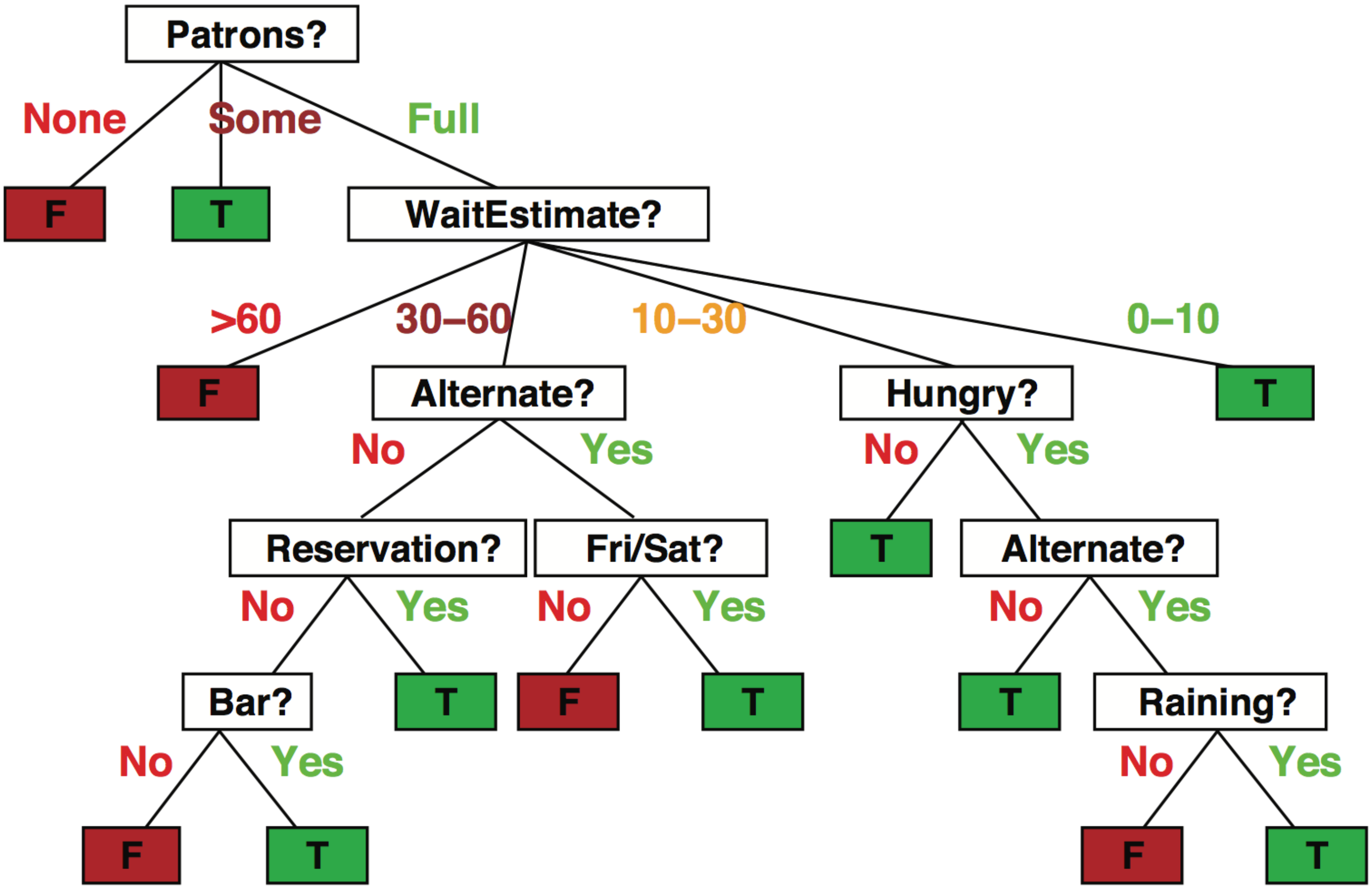


- Label:
  - Heart disease (+), No heart disease (-)

Example : Restaurant

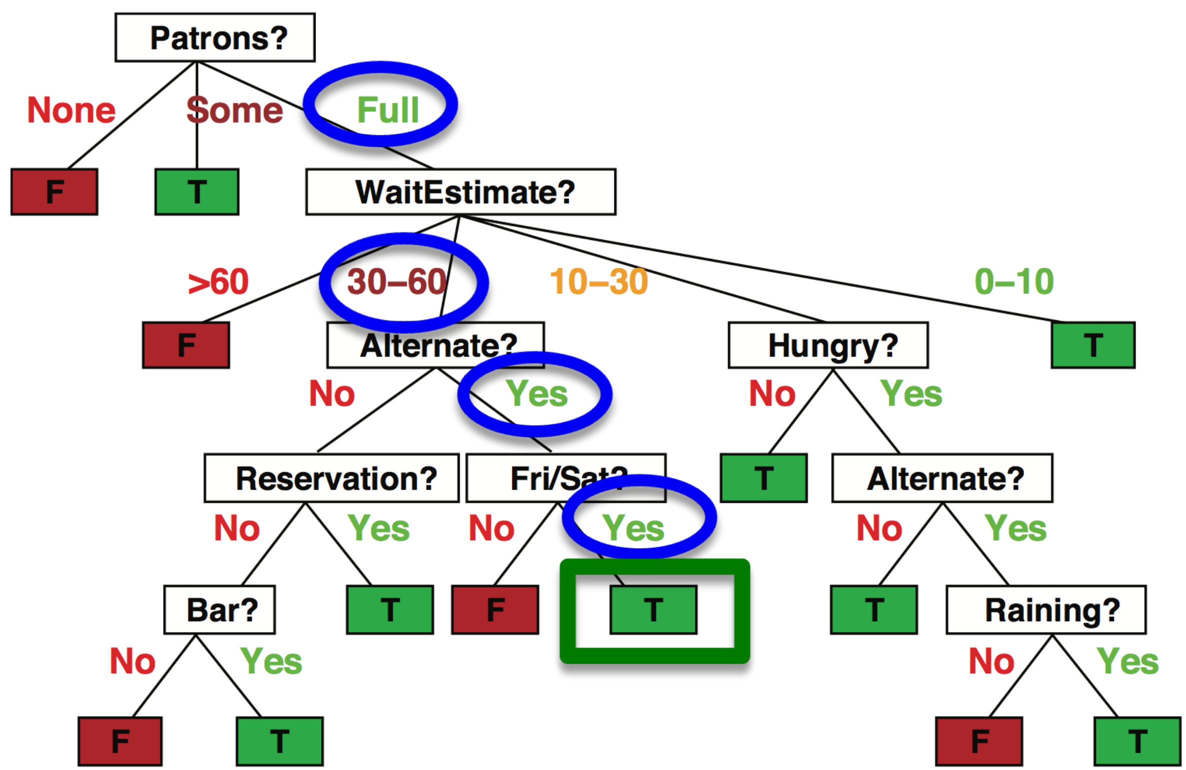
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3 =$ Yes
$\mathbf{x}_4$	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10–30	$y_4 =$ Yes
$\mathbf{x}_5$	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	$y_5 =$ No
$\mathbf{x}_6$	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0–10	$y_6 =$ Yes
$\mathbf{x}_7$	No	Yes	No	No	None	$	Yes	No	Burger	0–10	$y_7 =$ No
$\mathbf{x}_8$	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0–10	$y_8 =$ Yes
$\mathbf{x}_9$	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	$y_9 =$ No
$\mathbf{x}_{10}$	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10–30	$y_{10} =$ No
$\mathbf{x}_{11}$	No	No	No	No	None	$	No	No	Thai	0–10	$y_{11} =$ No
$\mathbf{x}_{12}$	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30–60	$y_{12} =$ Yes

Example: Restaurant (Wait or Go?)



Example: Restaurant

- It's Friday night and you're hungry
- You arrive at your favorite cheap but really cool happening burger place
- It's full up and you have no reservation but there is a bar
- The host estimates a 45 minute wait
- There are alternatives nearby but it's raining outside

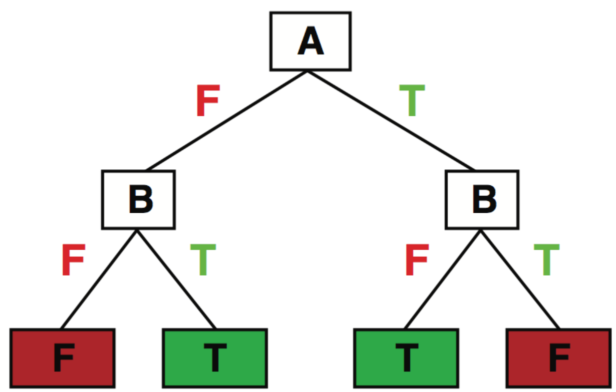


Decision Tree: Expressiveness

- Discrete decision trees can express ***any function*** of the input in propositional logic

- E.g., for Boolean functions, build a path from root to leaf for each ro

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



Decision Tree: Learning

- Learning an optimal decision tree is NP-Complete
  - Instead, we use a greedy search based on a heuristic
  - We cannot guarantee to return the globally-optimal decision tree.
- The most common strategy for DT learning is a greedy top-down approach
  - chooses a variable at each step that best splits the set of items.
- Tree is constructed by splitting samples into subsets based on an attribute value test in a recursive manner

Decision Tree: Learning

- Decision tree learning: construction of a decision tree from training samples.
  - Decision trees used in data mining are usually classification trees
- Many specific decision-tree learning algorithms, such as:
  - ID3
  - C4.5
- Approximates functions of usually discrete domain
  - The learned function is represented by a decision tree

Decision Tree: Construct

- We prefer decisions leading to a simple, compact tree with few nodes

- Which attribute at the root?
  - Measure: how well the attributes split the set into homogeneous subsets (having same value of target)
    - Homogeneity of the target variable within the subsets.

- How to form descendant?
  - Descendant is created for each possible value of A
    - Training examples are sorted to descendant nodes

Decision Tree: Construct

```
Function FindTree(S,A)
  If empty(A) or all labels of the samples in S are the same
     status = leaf
     class = most common class in the labels of S
  else
     status = internal
     a ←bestAttribute(S,A)
     LeftNode = FindTree(S(a=1),A \ {a})
     RightNode = FindTree(S(a=0),A \ {a})
  end
end
```

S: samples, A: attributes

Recursive calls to create left and right subtrees
S(a=1) is the set of samples in S for which a=1

Top down, Greedy, No backtrack

ID3

ID3 (Examples, Target_Attribute, Attributes)

Create a root node for the tree

If all examples are positive, return the single-node tree Root, with label = +

If all examples are negative, return the single-node tree Root, with label = -

If number of predicting attributes is empty then

  return Root, with label = most common value of the target attribute in the examples

else

  A = The Attribute that best classifies examples.

  Testing attribute for Root = A.

  for each possible value, v_i, of A

    Add a new tree branch below Root, corresponding to the test A = v_i .

    Let Examples(v_i) be the subset of examples that have the value for A

    if Examples(v_i) is empty then

      below this new branch add a leaf node with label = most common target value in the examples

    else below this new branch add subtree **ID3 (Examples(v_i), Target_Attribute, Attributes – {A})**

return Root

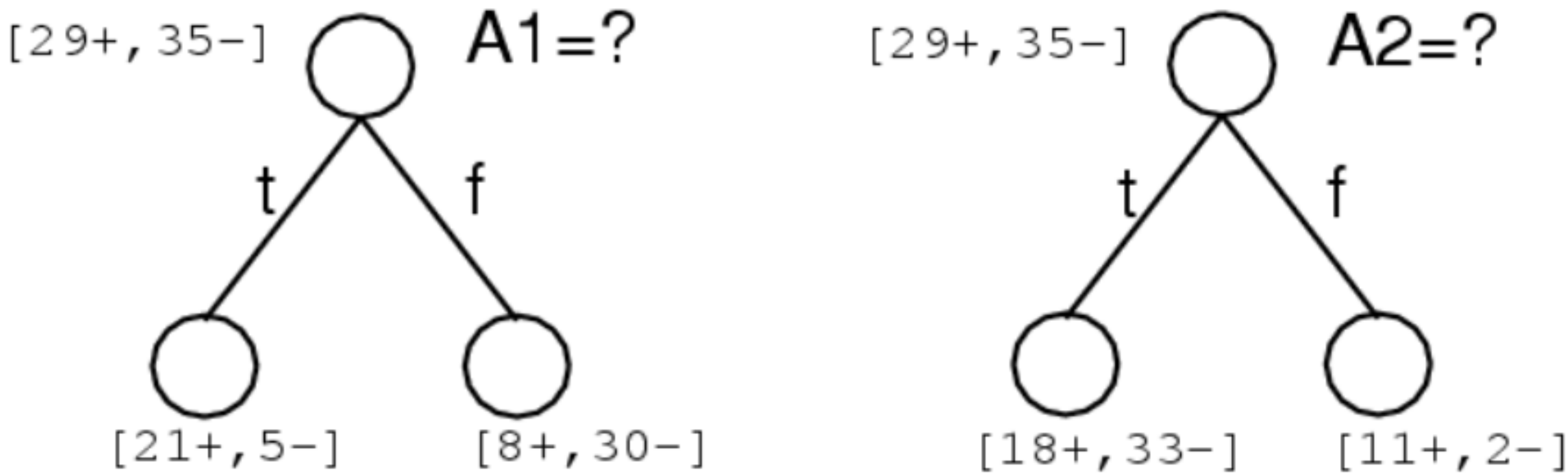
ID3: Properties

- The algorithm
  - either reaches homogenous nodes
  - or runs out of attributes

- Guaranteed to find a tree consistent with any conflict-free training set
  - ID3 hypothesis space of all DTs contains all discrete-valued functions
  - Conflict free training set: identical feature vectors always assigned the same class

- But not necessarily find the simplest tree (containing minimum number of nodes).
  - a greedy algorithm with locally-optimal decisions at each node (no backtrack).

Which Attribute is the best?



Entropy

- Entropy *H(X)* of a random variable *X*:

$$H(X) = -\sum_{x_i \in X} P(x_i) \log_2 P(x_i)$$

- More uncertainty, more entropy!

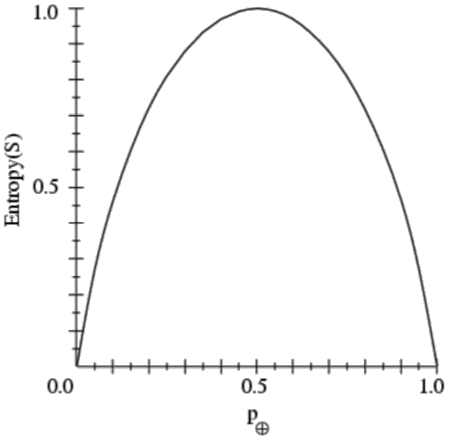
- Information Theory interpretation:
  - H(X) is the expected number of bits needed to encode a randomly drawn value of X (under most efficient code)

Entropy for a Boolean Variable

$$P(Y = T) = \frac{5}{6}, P(Y = F) = \frac{1}{6}$$

$$H(Y) = -\sum_{y_i \in Y} P(y_i) \log P(y_i)$$

$$H(Y) = -\frac{5}{6} \log_2 \frac{5}{6} - \frac{1}{6} \log_2 \frac{1}{6} \approx 0.65$$



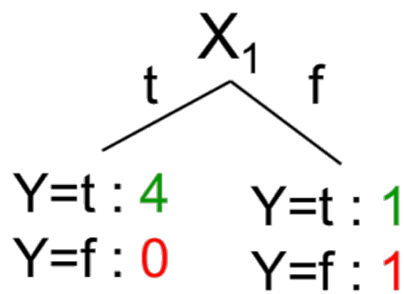
X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

Conditional Entropy

- Conditional Entropy *H(Y|X)* of a random variable *Y* conditioned on a random variable *X*

$$H(Y|X) = -\sum_{i=1}^{v} P(X = x_j) \sum_{j=1}^{k} P(Y = y_i | X = x_j) \log P(Y = y_i | X = x_j)$$

Example:



X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

$P(X_1=t) = 4/6$

$P(X_1=f) = 2/6$

$$H(Y|X_1) = -\ 4/6\ (1 \log_2 1 + 0 \log_2 0)$$
$$-\ 2/6\ (1/2 \log_2 1/2 + 1/2 \log_2 1/2)$$
$$= 2/6$$
$$= 0.33$$

Information Gain

- For each split, compare entropy before and after
  - Difference is the information gain

$$IG(X) = H(Y) - H(Y|X)$$

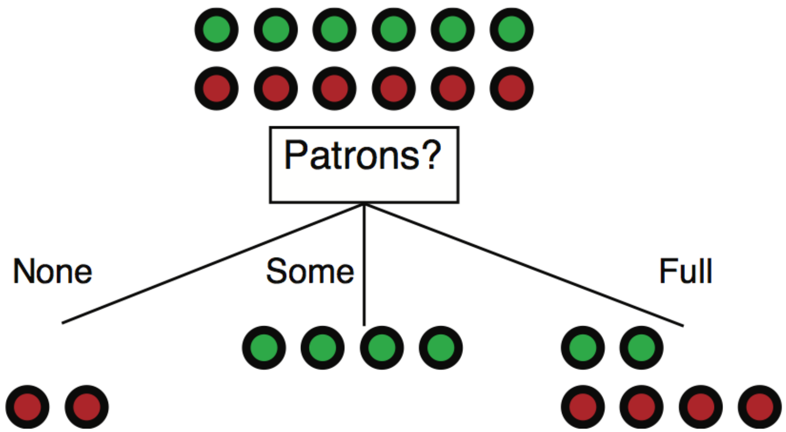
In our running example:

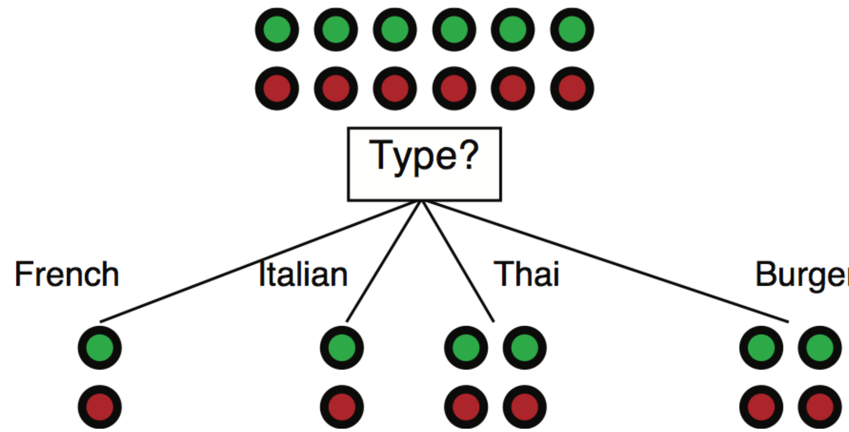
$IG(X_1) = H(Y) - H(Y|X_1)$
$= 0.65 - 0.33$

$IG(X_1) > 0$ → we prefer the split!

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

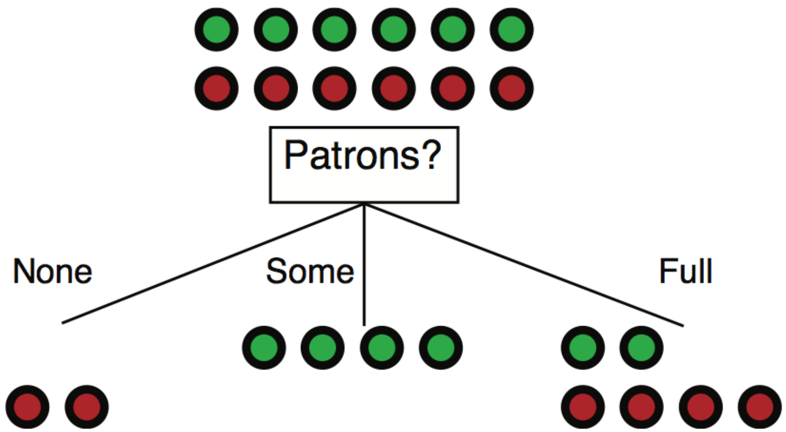
Example: Restaurant

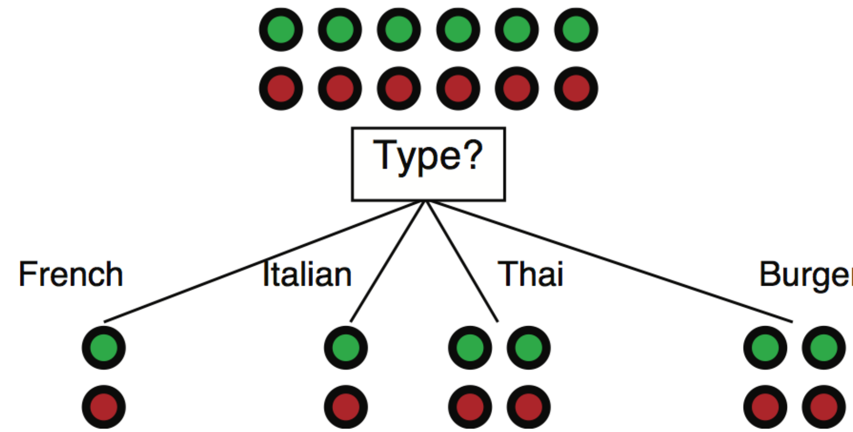




Information gain?

Example: Restaurant



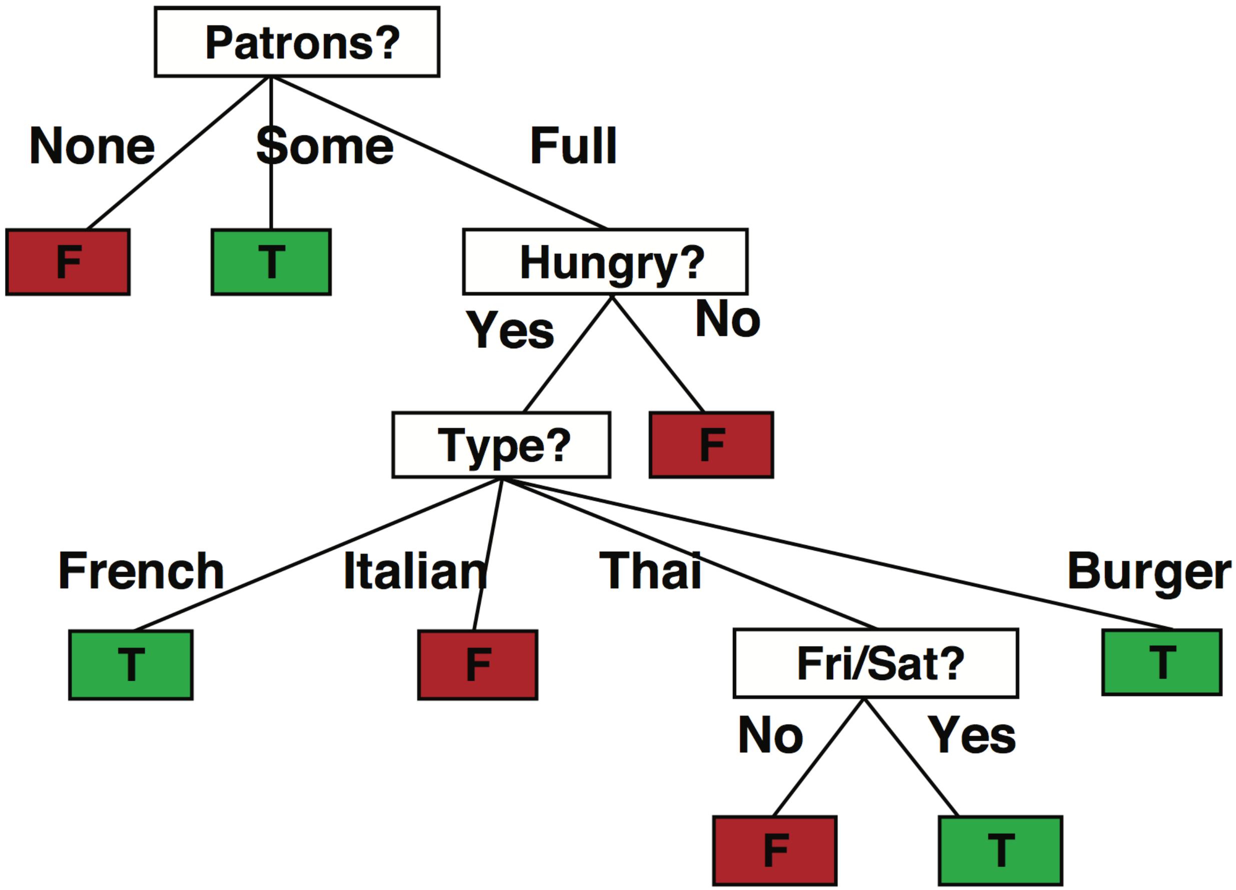


$$1 - \frac{6}{12}(-\frac{2}{6}\log_2\frac{2}{6} - \frac{4}{6}\log_2\frac{4}{6}) \approx 0.541$$

$$1 - (\frac{2}{12} + \frac{2}{12} + \frac{4}{12} + \frac{4}{12})(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}) = 0$$

Results for Restaurant Data

- Decision tree learned from the 12 examples:



Decision Tree: Learning

function Decision-Tree-Learning(*examples,attributes,parent_examples*)
returns a tree

if *examples* is empty **then return** Plurality-Value(*parent_examples*)

else if all *examples* have the same classification then return the classification

else if *attributes* is empty then return Plurality-Value(*examples*)

else $A \leftarrow \text{argmax}_{a \in attributes}$ **Importance**(*a*, *examples*)

    tree ← a new decision tree with root test *A*

    for each value *v* of *A* **do**

        exs ← the subset of *examples* with value *v* for attribute *A*

        subtree ← Decision-Tree-Learning(exs, *attributes* $- A$, *examples*)

        add a branch to *tree* with label ($A = v_k$) and subtree *subtree*

return *tree*

Decision Tree Learning: Function Approximation Problem

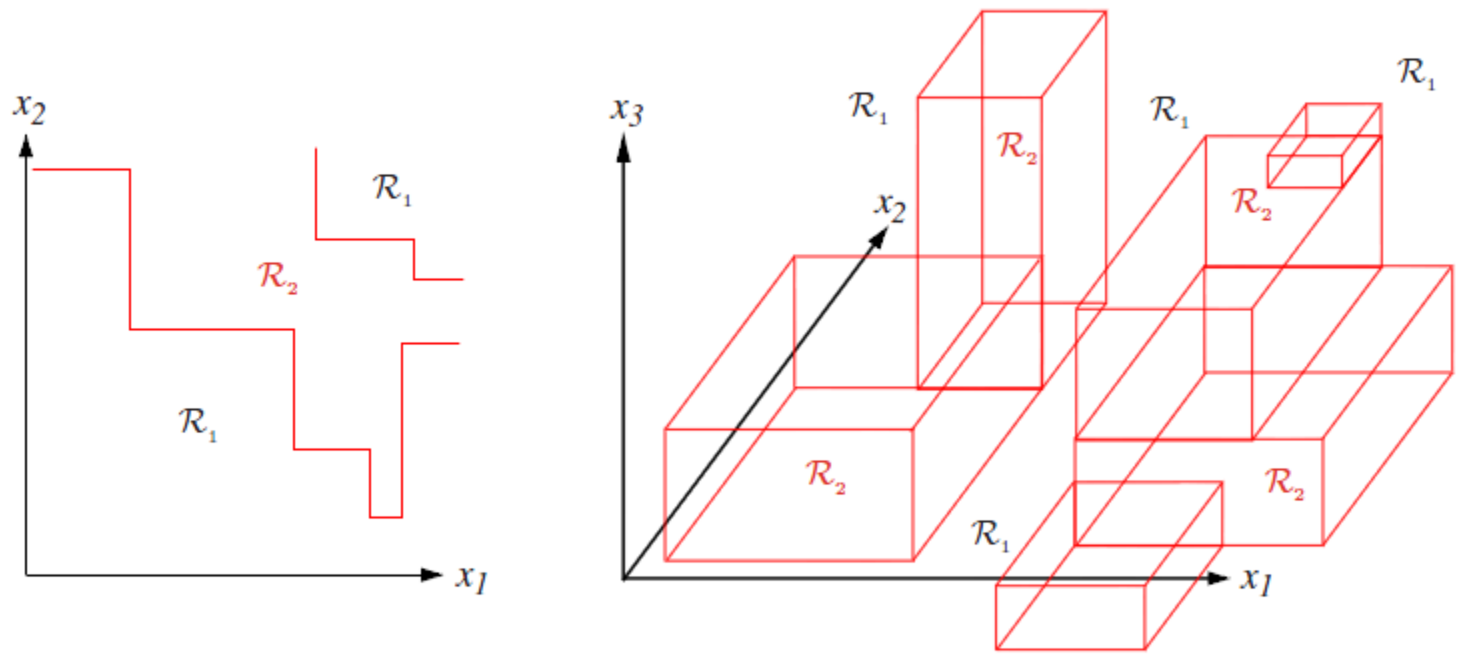
- Problem Setting:
  - Set of possible instances X
  - Unknown target function $f: X \rightarrow Y$ (Y is discrete valued)
  - Set of function hypotheses $H = \{ h \mid h : X \rightarrow Y \}$
    - h is a DT where tree sorts each $\boldsymbol{x}$ to a leaf which assigns a label y

- Input:
  - Training examples $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$ of unknown target function f

- Output:
  - Hypothesis $h \in H$ that best approximates target function f

How Partition Instance Space?

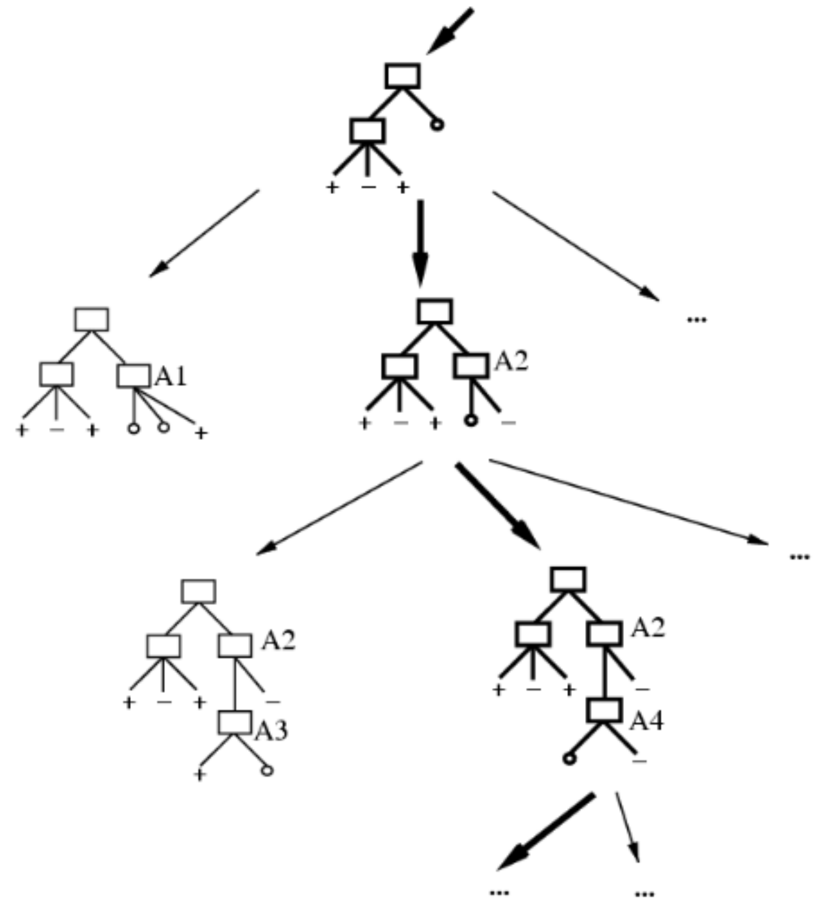
- Decision tree
  - Partition the instance space into axis-parallel regions, labeled with class value



[Duda & Hurt 's Book]

ID3 as a Search in the Space of Trees

- ID3: heuristic search through space of DTs
  - Performs a simple to complex hill-climbing search (begins with empty tree)
  - prefers simpler hypotheses due to using IG as a measure of selecting attribute test
- IG gives a bias for trees with minimal size.
  - ID3 implements a search (preference) bias instead of a restriction bias.



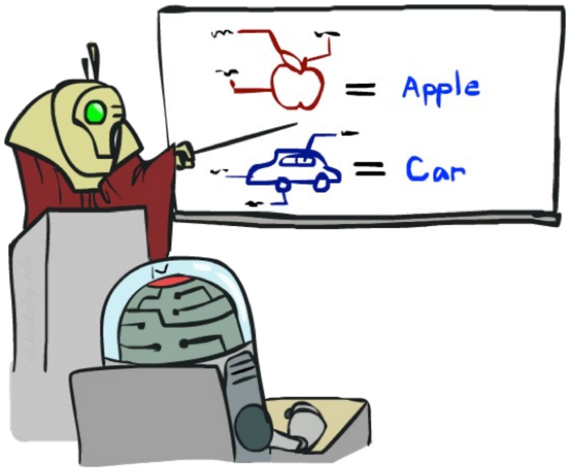
[Mitchell's Book]

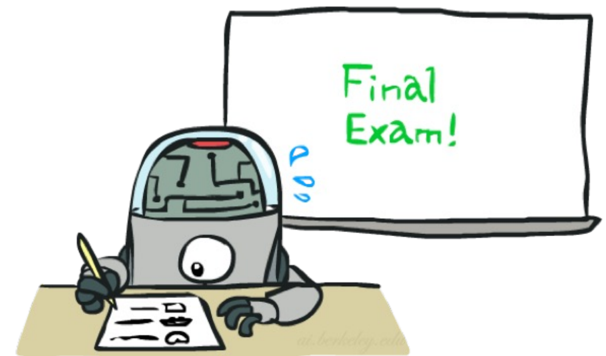
Why Prefer Short Hypotheses?

- Why is the optimal solution the smallest tree?

- Fewer short hypotheses than long ones
  - a short hypothesis that fits the data is less likely to be a statistical coincidence
    - Lower variance of the smaller trees

Training and Testing

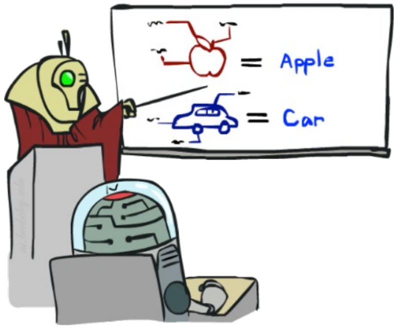




Recap: Important Points About Learning

- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set (Validation set)
  - Test set
- Features: attribute-value pairs which characterize each x
- Evaluation
  - Accuracy: fraction of instances predicted correctly
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
    (Tune hyperparameters on held-out set)
  - Compute accuracy of test set
  - Very important: never "peek" at the test set!
- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well
  - Underfitting: fits the training set poorly

Training Data



Held-Out Data (Validation set)

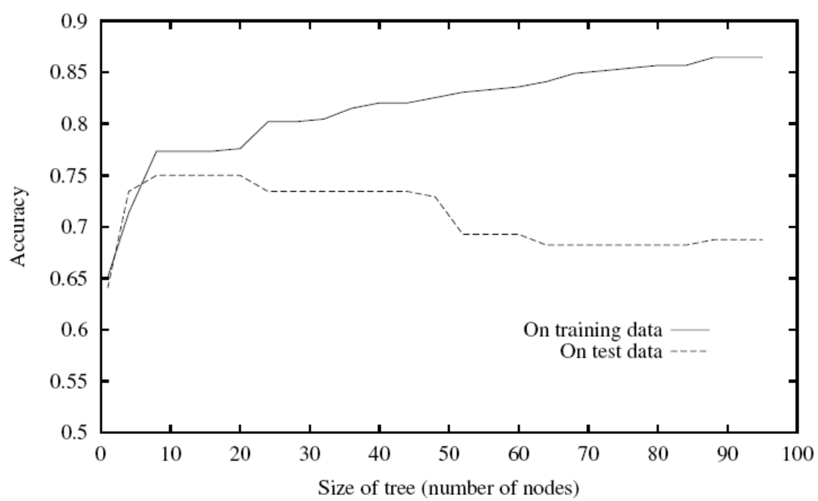
Test Data

Overfitting

- ID3 perfectly classifies training data (for consistent data)
  - It tries to memorize every training data
  - Poor decisions when very little data (it may not reflect reliable trends)
    - Noise in the training data: the tree is erroneously fitting.
    - A node that "should" be pure but had a single (or few) exception(s)?
- For many (non relevant) attributes, the algorithm will continue to split nodes
  - Leads to overfitting!
- Must introduce some bias towards *simpler* trees

Overfitting in Decision Tree Learning

- Hypothesis space H: decision trees
- Training (emprical) error of $h \in H$: $error_{train}(h)$
- Expected error of $h \in H$: $error_{true}(h)$
- h overfits training data if there is a $h' \in H$ such that
  - $error_{train}(h) < error_{train}(h')$
  - $error_{true}(h) > error_{true}(h')$



[Mitchell's Book]

Avoiding Overfitting

- **Stop growing** when the data split is not statistically significant.
  - Bound depth or # leaves
  - Doesn't work well in practice
- Grow full tree and then **prune** it.
  - Optimize on a held-out set
  - More successful than stop growing in practice.

Reduced-Error Pruning

- Split data into train and validation set
- Build tree using training set
- Do until further pruning is harmful:
  - Evaluate impact on validation set when pruning sub-tree rooted at each node
    - Temporarily remove sub-tree rooted at node
    - Replace it with a leaf labeled with the current majority class at that node
    - Measure and record error on validation set
  - Greedily remove the one that most improves validation set accuracy (if any).

Produces smallest version of the most accurate sub-tree.

Decision Tree: Advantages

- Simple to understand and interpret
- Requires little data preparation and also can handle both numerical and categorical data
- Time efficiency of learning decision tree classifier
  - Cab be used on large datasets
- Robust: Performs well even if its assumptions are somewhat violated