Learning and Decision Trees

COMP3411/9814: Artificial Intelligence

Supervised Learning

Given:

- a set of training examples, each with a set of features and target value (or class):

$<x_1, x_2, \ldots, x_n, y>$

- a new example, where only the values for the input features are given

predict the target value of the new example.

Learning decision trees

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

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)

Restaurant Training Data

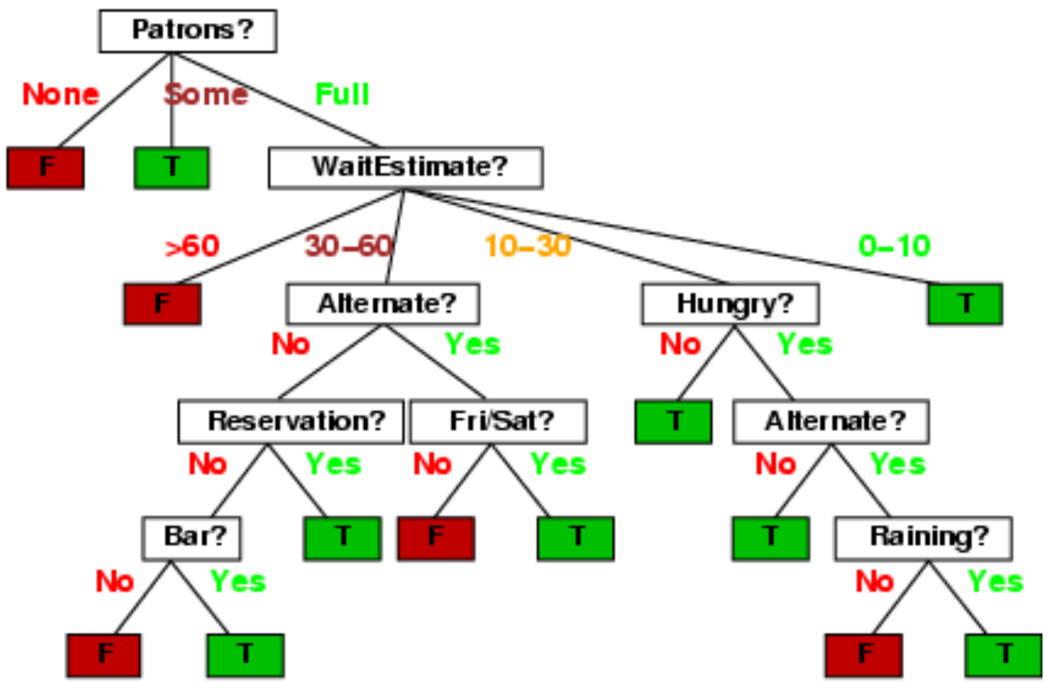
- Examples described by attribute values (Boolean, discrete, continuous)
  - E.g., situations where I will/won't wait for a table:

class value

	Alt	Bar	F/S	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

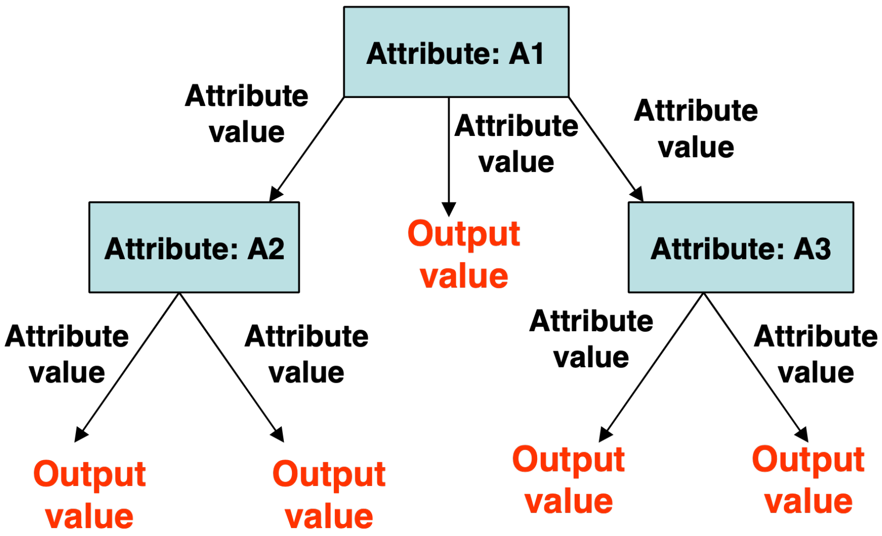
Decision Trees

- One possible representation for hypotheses



Decision Trees

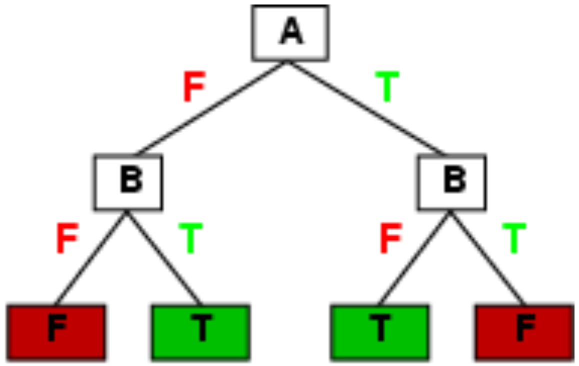
- Output can be multi-valued, not just binary



Expressiveness

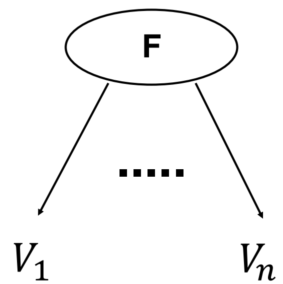
- Decision trees can express any function of the input attributes.
  - E.g., for Boolean functions, truth table row → path to leaf:

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



- There is a consistent decision tree for any training set with one path to leaf for each example (unless *f* nondeterministic in *x*) but it probably won't generalise to new examples
- Prefer to find more compact decision trees

ID3 (Quinlan)



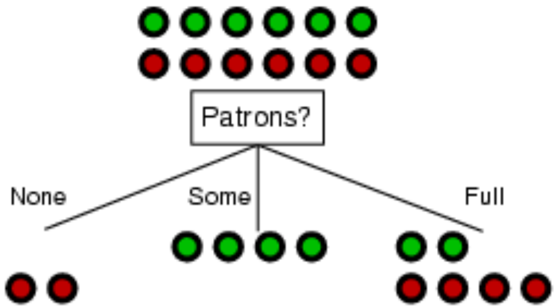
- The algorithm operates over a set of training instances, *E*.
- If all instances in *E* are in class *C*, create a node *C* and stop, otherwise select a feature *F* and create a decision node.
- Partition the training instances in *E* into subsets according to the values, V_i of *F*.
- Apply the algorithm recursively to each of the subsets E_i

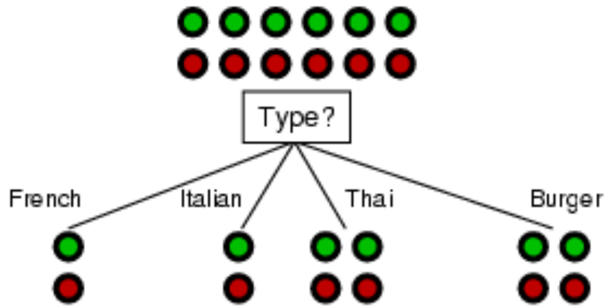
Generalisation

- Training data must not be inconsistent
    - see later how to handle inconsistent data
- Can split attributes in any order and still produce a tree that correctly classifies all examples in training set
- However, want a tree that is likely to generalise to correctly classify (unseen) examples in test set.
- Prefer simpler hypothesis, i.e. a smaller tree.
- How can we choose attributes to produce a small tree?

Choosing an attribute

	Alt	Bar	F/S	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

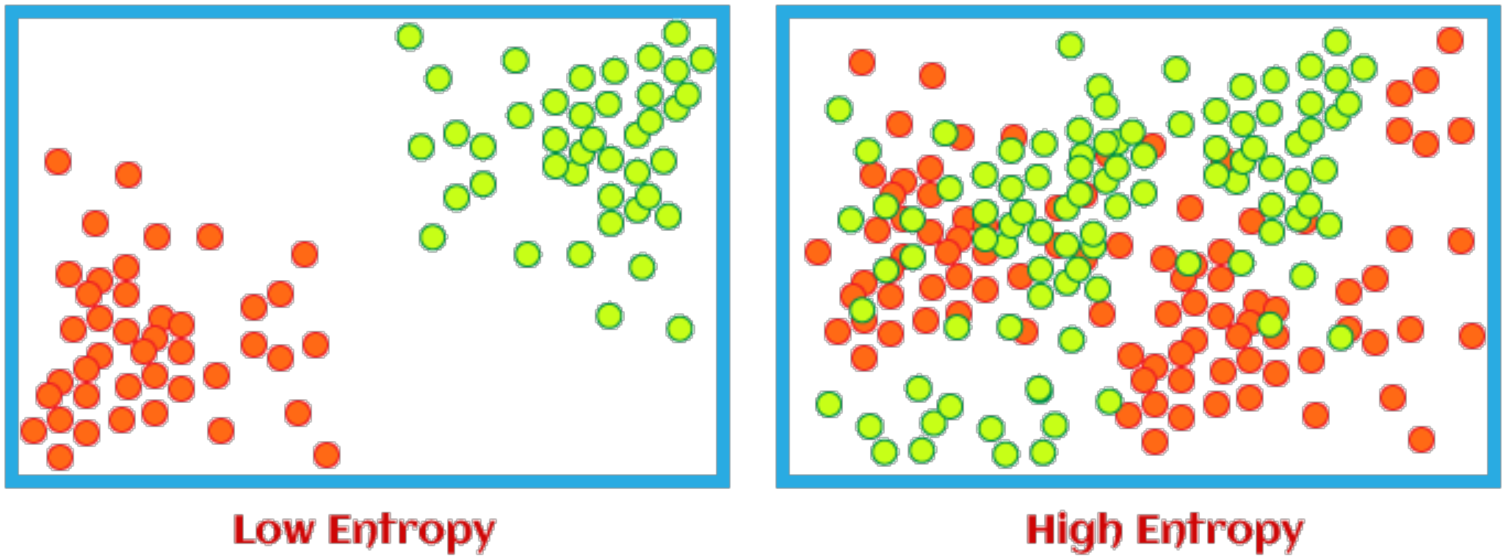




- Patrons is a more “informative” attribute than Type, because it splits the examples more nearly into sets that are “all positive” or “all negative”.
- “Informativeness” can be quantified using the mathematical concept of “entropy”.
- Tree can be built by minimising entropy at each step

Entropy

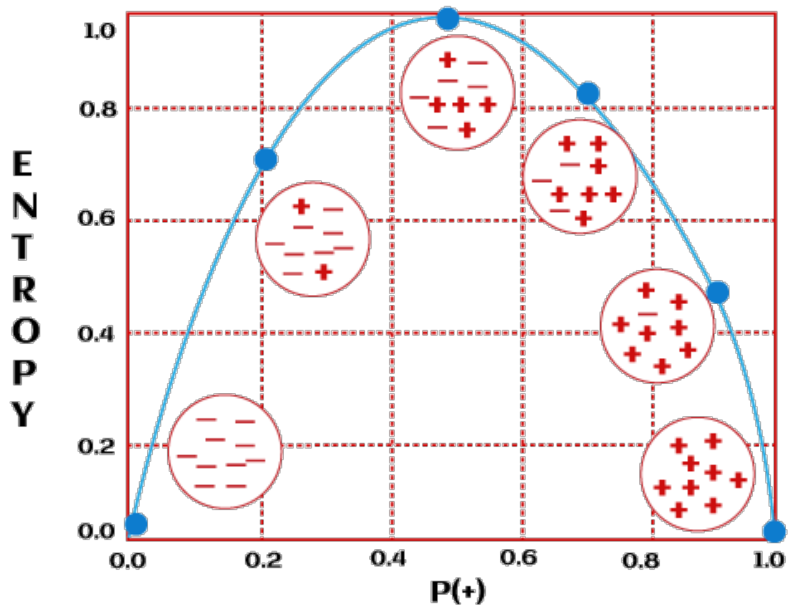
- Entropy is a measure of the randomness or uniformity.



Entropy

- If the prior probabilities of the n class values are $p_1, \ldots, p_n$ then the entropy is

$$H(\langle p_{1,} \ldots, p_{n,} \rangle) = \sum_{n=1} -p_i \log_2 p_i$$



- This is the minimum number of bits needed to encode an example.

- Frequently occurring classes require fewer bits.

Entropy

- Suppose we have p positive and n negative examples in a node.
- $H\left(\left\langle\frac{p}{p+n}, \frac{n}{p+n}\right\rangle\right)$ bits needed to encode 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

  $H\left(\left\langle\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right\rangle\right)$ bits needed to encode an example

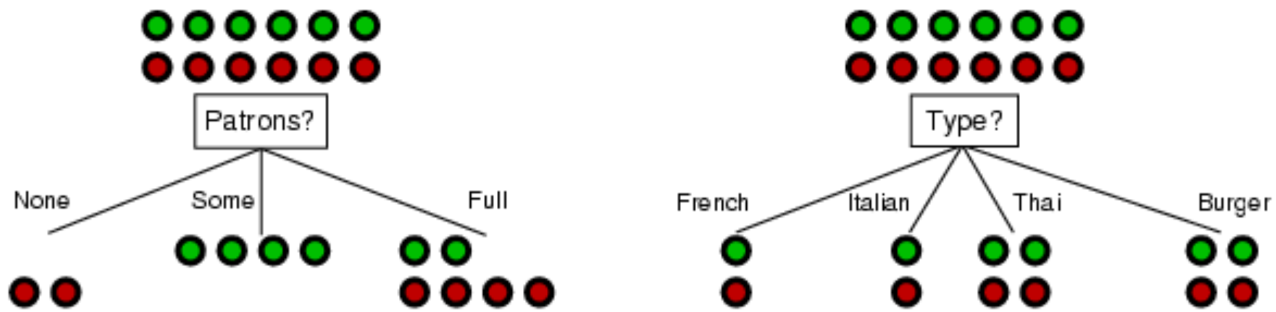
  expected number of bits per example over all branches is

  $$\sum_i \boxed{\frac{p_i + n_i}{p + n}} H\left(\left\langle\frac{p_i}{p_i + n}, \frac{n_i}{p_i + n_i}\right\rangle\right)$$

  Probability of going down branch *i*

- For Patrons, this is 0.459 bits, for Type this is (still) 1 bit ➔ split on Patrons

Choosing and Attribute



$$\text{For Patrons, Entropy} = \frac{1}{6}(0) + \frac{1}{3}(0) + \frac{1}{2}\left[-\frac{1}{3}\log(\frac{1}{3}) - \frac{2}{3}\log(\frac{2}{3})\right]$$

$$= 0 + 0 + \frac{1}{2}\left[\frac{1}{3}(1.585) + \frac{2}{3}(0.585)\right] = 0.459$$

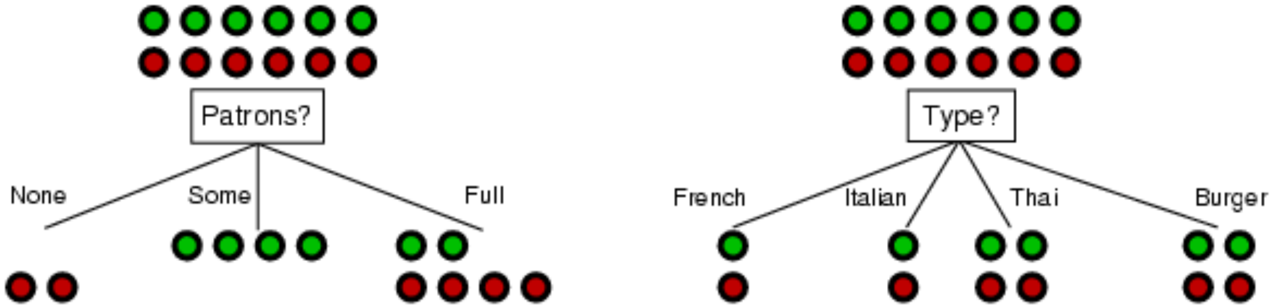
$$\text{For Type, Entropy} = \frac{1}{6}(1) + \frac{1}{6}(1) + \frac{1}{3}(1) + \frac{1}{3}(1) = 1$$

$$Prior(none) = \frac{2}{12} = \frac{1}{6}$$

$$P_{wait}(false) = 1$$

$$-P_{wait}(false) log_2 P_{wait}(false) = 1 \times 0 = 0$$

Choosing and Attribute



$$\text{For Patrons, Entropy} = \frac{1}{6}(0) + \frac{1}{3}(0) + \frac{1}{2}\left[-\frac{1}{3}\log(\frac{1}{3}) - \frac{2}{3}\log(\frac{2}{3})\right]$$

$$= 0 + 0 + \frac{1}{2}\left[\frac{1}{3}(1.585) + \frac{2}{3}(0.585)\right] = 0.459$$

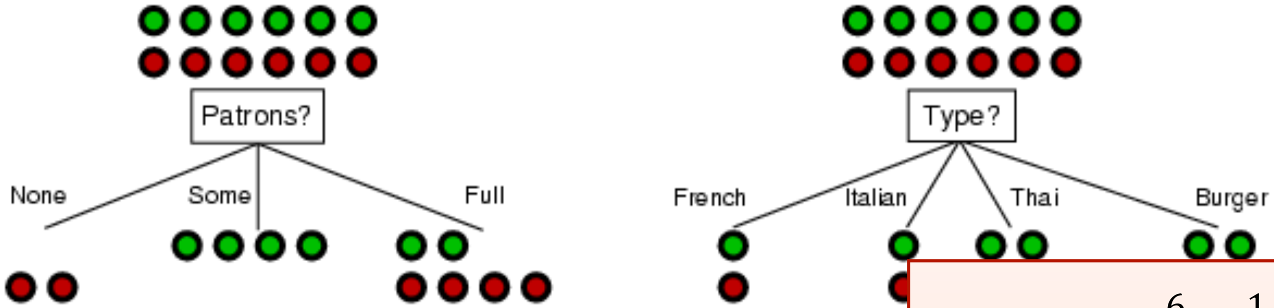
$$\text{For Type, Entropy} = \frac{1}{6}(1) + \frac{1}{6}(1) + \frac{1}{3}(1) + \frac{1}{3}(1) = 1$$

$$Prior(some) = \frac{4}{12} = \frac{1}{3}$$

$$P_{wait}(true) = 1$$

$$-P_{wait}(true)log_2P_{wait}(true) = 1\times 0 = 0$$

Choosing and Attribute



$$Prior(full) = \frac{6}{12} = \frac{1}{2}$$

$$P_{wait}(false) = \frac{2}{6} = \frac{1}{3}$$

$$P_{wait}(true) = \frac{4}{6} = \frac{2}{3}$$

$$-P_{wait}(false)log_2 P_{wait}(false) = \frac{1}{3}log(\frac{1}{3})$$

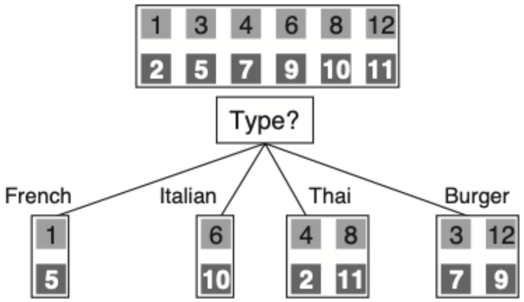
$$-P_{wait}(true)log_2 P_{wait}(true) = \frac{2}{3}log(\frac{2}{3})$$

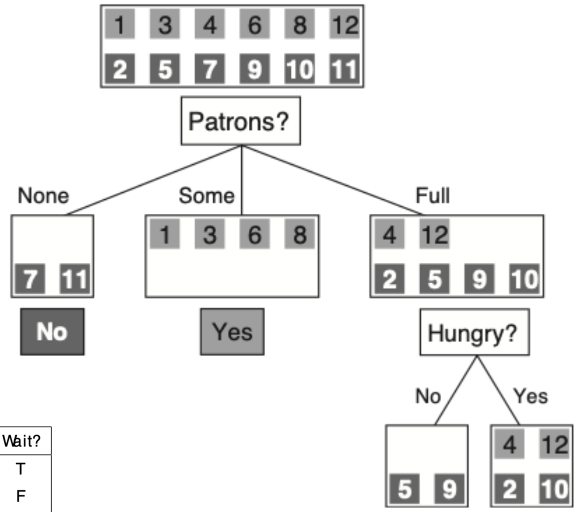
For Patrons, Entropy $= \frac{1}{6}(0) + \frac{1}{3}(0) + \frac{1}{2}\left[-\frac{1}{3}\log(\frac{1}{3}) - \frac{2}{3}\log(\frac{2}{3})\right]$

$= 0 + 0 + \frac{1}{2}\left[\frac{1}{3}(1.585) + \frac{2}{3}(0.585)\right] = 0.459$

For Type, Entropy $= \frac{1}{6}(1) + \frac{1}{6}(1) + \frac{1}{3}(1) + \frac{1}{3}(1) = 1$

Choosing Next Attribute



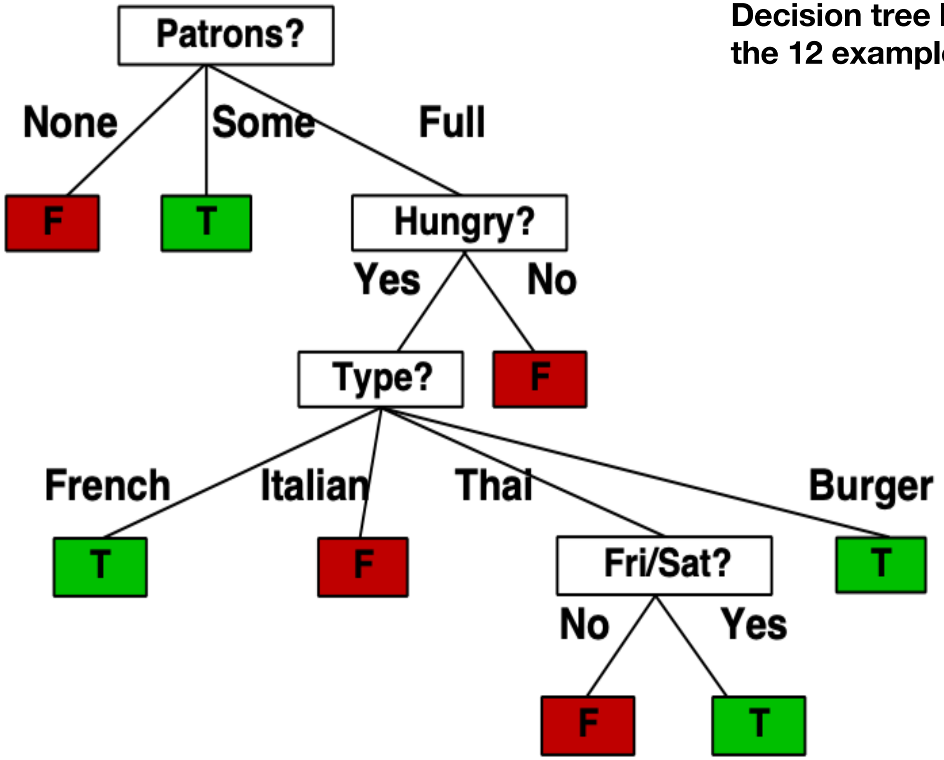


$$\sum_i \frac{p_i + n_i}{p + n} H\left(\left\langle \frac{p_i}{p_i + n}, \frac{n_i}{p_i + n_i} \right\rangle\right)$$

After splitting on Patrons, split on Hungry

	Alt	Bar	F/S	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

Induced Tree



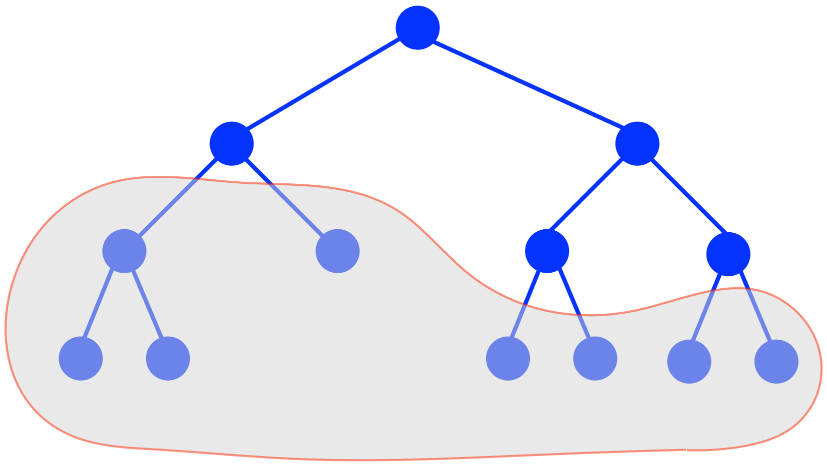
Decision tree learned from the 12 examples.

Decision Trees

- Decision tree learning is a method for approximating discrete value target functions, in which the learned function is represented by a decision tree.
- Decision trees can also be represented by if-then-else rule
- Decision tree learning is one of the most widely used approach for inductive inference

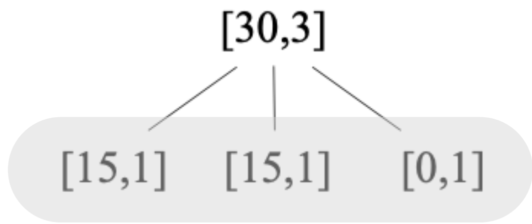
Overfitting

- What if training data are noisy?
    - Misclassified examples
    - Incorrect measurements
- Making decision tree classify every example (including noise) could make it less accurate
    - Tries to classify bad examples
    - ➢ Misclassifies correct examples in test set
- This is called *overfitting*
- Can improve accuracy by pruning branches created by try to classify noise.



Tree Pruning

- The top node of this tree has
  - 30 +ve examples
  - 3 –ve examples



- It is split 3 ways.

- If we decide to prune it, the top node becomes a leaf node and the decision value is the majority class, i.e. +ve.

- Only prune if the expected error is less than the expected error with the children

Laplace Error and Pruning

When a node becomes a leaf, all items will be assigned to the majority class at that node. We can estimate the error rate on the (unseen) test items using the Laplace error:

$$E = 1 - \frac{n+1}{N+k}$$

N = total number of training examples

n = number of training examples in the majority class

k = number of classes

If the average Laplace error of the children exceeds that of the parent node, we prune off the children.

How do we get the Laplace Error?

- Suppose a node as 99 +ve and 1 –ve: $\frac{n}{N} = \frac{99}{100}$ is the probability of the majority class.
- If we decide to prune, the expected error will be $1 - \frac{99}{100} = 0.01$
- Good estimate if N is large, but if N is small, better to rely on prior probability of class, i.e. $\frac{1}{k} -$
- So the Laplace error adds a bias for small N. When N is large, correction is irrelevant

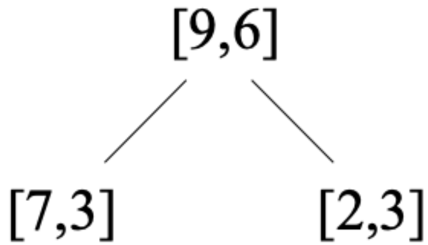
$$E = 1 - \frac{n}{N} + \frac{1}{k}$$

N = total number of training examples

n = number of training examples in the majority class

k = number of classes

Minimal Error Pruning



Should the children of this node be pruned or not?

Left child has class frequencies [7, 3]

$$E = 1 - \frac{n+1}{N+k} = 1 - \frac{7+1}{10+2} = 0.333$$

Right child has $E = 0.429$

Parent node has $E = 0.412$

Average for left and right child is:

$$E = \frac{10}{15} \times 0.333 + \frac{5}{15} \times 0.429 = 0.365$$

Since $0.365 < 0.412$, children should NOT be pruned

Minimal Error Pruning

Should the children of this node be pruned or not?

Left child has class frequencies [3, 2]

$$E = 1 - \frac{n+1}{N+k} = 1 - \frac{3+1}{5+2} = 0.429$$

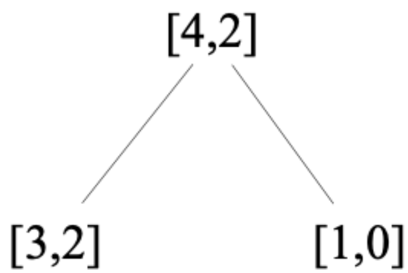
Right child has E = 0.333

Parent node has E = 0.375

Average for left and right child is:

$$E = \frac{5}{6} \times 0.429 + \frac{1}{6} \times 0.333 = 0.413$$

Since 0.413 > 0.375, children should be pruned

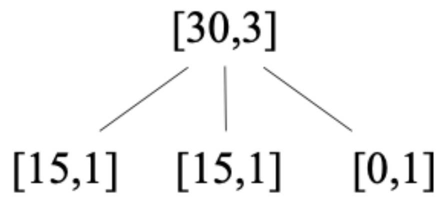


Minimal Error Pruning

Should the children of this node be pruned or not?

Left child has class frequencies [15, 1]

$$E = 1 - \frac{n+1}{N+k} = 1 - \frac{15+1}{16+2} = 0.111$$



Right child has $E = 0.333$

Parent node has $E = \frac{4}{35} = 0.114$

Average for left and right child is:

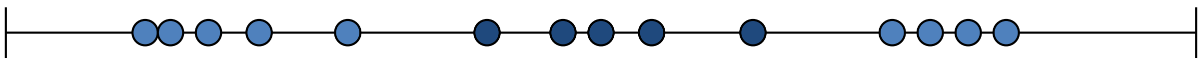
$$E = \frac{16}{33} \times 0.111 + \frac{16}{33} \times 0.111 + \frac{1}{33} \times 0.333 = 0.118$$

Since 0.118 > 0.114, children should be pruned

Numerical Attributes

- ID3 algorithm is designed for attributes that have discrete values.
- How can we handle attributes with continuous numerical values?

➢ Must discretise values.



Problems Suitable for Decision Trees

- Instances are represented by attribute-value pairs
- Instances are described by a fixed set of attributes (e.g., Temperature) and their values (e.g., Hot).
  - Easiest domains for decision tree learning is when each attribute takes on a small number of disjoint possible values (e.g., Hot, Mild, Cold).
  - Extensions allow handling real-valued attributes as well (e.g., representing temperature numerically).
- The target function has discrete output values
- The training data
  - The training data may contain errors
  - The training data may contain missing attribute values

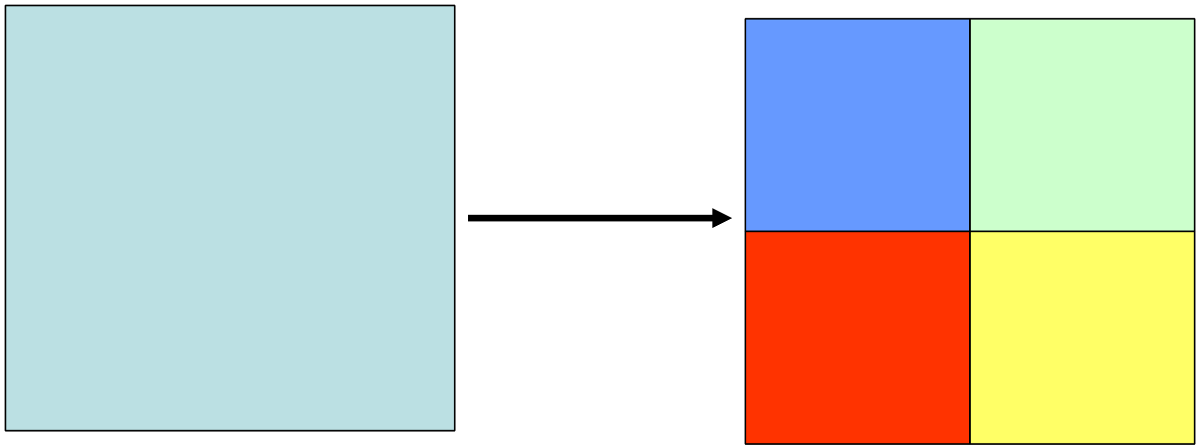
Training and Testing

- For classification problems, a classifier's performance is measured in terms of the *error rate.*
- The classifier predicts the class of each instance: if it is correct, that is counted as a *success;* if not, it is an *error.*
- The error rate is just the proportion of errors made over a whole set of instances, and it measures the overall performance of the classifier.

Training and Testing

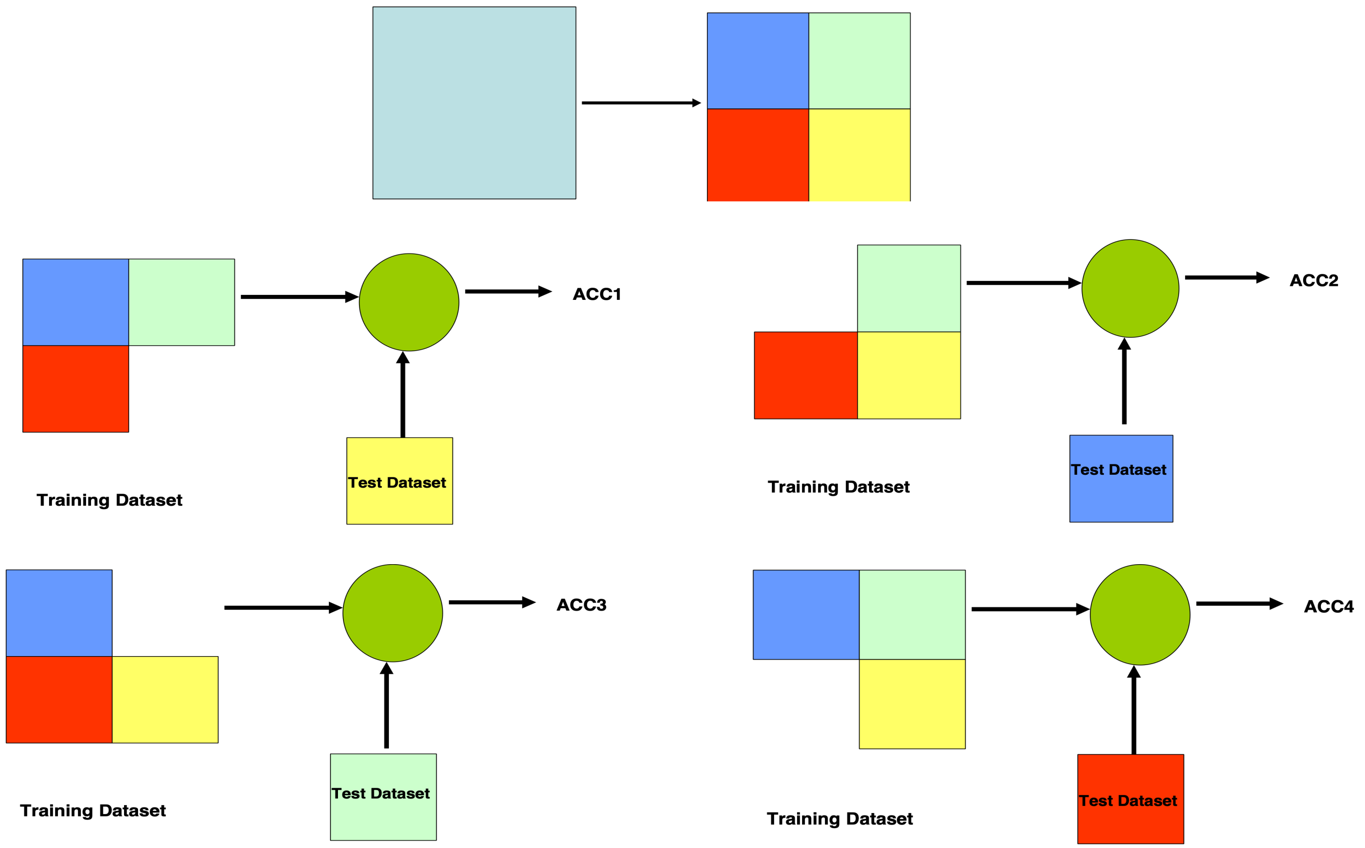
- Self-consistency test: when training and test dataset are the same
- Hold out strategy: reserve some examples for testing and use the rest for training (set part of that aside for validation, if required)
- K-fold Cross validation:
  - If don’t have many examples
  - Partition dataset randomly into k equal-sized sets.
  - Train and test classifier k times using one set for testing and other $k - 1$ sets for training

4-Fold Cross-validation



ACC = (ACC1 + ACC2 + ACC3 + ACC4) / 4

4-Fold Cross-validation



K-Fold Cross Validation

- Train multiple times, leaving out a disjoint subset of data each time for test.
- Average the test set accuracies.

Partition data into K disjoint subsets
for $k \in 1..K$:
$\quad testData \leftarrow k_{th} subset$
$\quad h \leftarrow$ classifier trained on all data except for $testData$
$\quad accuracy(k) =$ accuracy of h on $testData$
end
$FinalAccuracy =$ mean of the K recorded test set accuracies

Summary

- Supervised Learning
  - Training set and test set
  - Try to predict target value based on input attributes
- Ockham's Razor
  - Tradeoff between simplicity and accuracy
- Decision Trees
  - Improve generalisation by building a smaller tree (using entropy)
  - Prune nodes based on Laplace error
  - Other ways to prune Decision Trees

References

- Poole & Mackworth, Artificial Intelligence: Foundations of Computational Agents, Chapter 7.
- Russell & Norvig, *Artificial Intelligence: a Modern Approach*, Chapter 18.1, 18.2, 18.3