Lecture 4: Decision Trees

- What is a decision tree?
- Constructing decision trees
- Entropy and information gain
- Issues when using real data

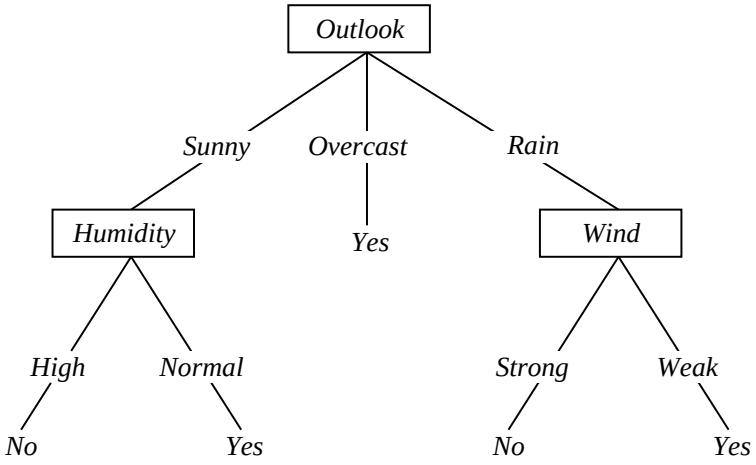
Note: part of this lecture based on notes from Roni Rosenfeld (CMU)

Classification problem example

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	*Sunny*	*Hot*	*High*	*Weak*	*No*
D2	*Sunny*	*Hot*	*High*	*Strong*	*No*
D3	*Overcast*	*Hot*	*High*	*Weak*	*Yes*
D4	*Rain*	*Mild*	*High*	*Weak*	*Yes*
D5	*Rain*	*Cool*	*Normal*	*Weak*	*Yes*
D6	*Rain*	*Cool*	*Normal*	*Strong*	*No*
D7	*Overcast*	*Cool*	*Normal*	*Strong*	*Yes*
D8	*Sunny*	*Mild*	*High*	*Weak*	*No*
D9	*Sunny*	*Cool*	*Normal*	*Weak*	*Yes*
D10	*Rain*	*Mild*	*Normal*	*Weak*	*Yes*
D11	*Sunny*	*Mild*	*Normal*	*Strong*	*Yes*
D12	*Overcast*	*Mild*	*High*	*Strong*	*Yes*
D13	*Overcast*	*Hot*	*Normal*	*Weak*	*Yes*
D14	*Rain*	*Mild*	*High*	*Strong*	*No*

Discover a “rule” for the PlayTennis predicate!

Decision trees



A decision tree consists of:

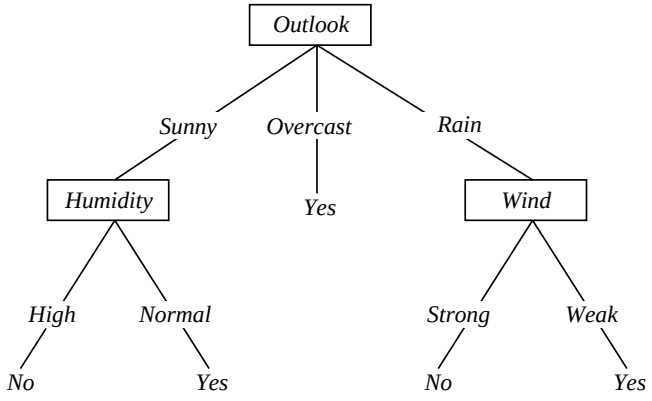
- a set of nodes, where each node tests the value of an attribute and branches on all possible values
- a set of leaves, where each leaf gives a class value

Using decision trees for classification

Suppose we get a new instance:

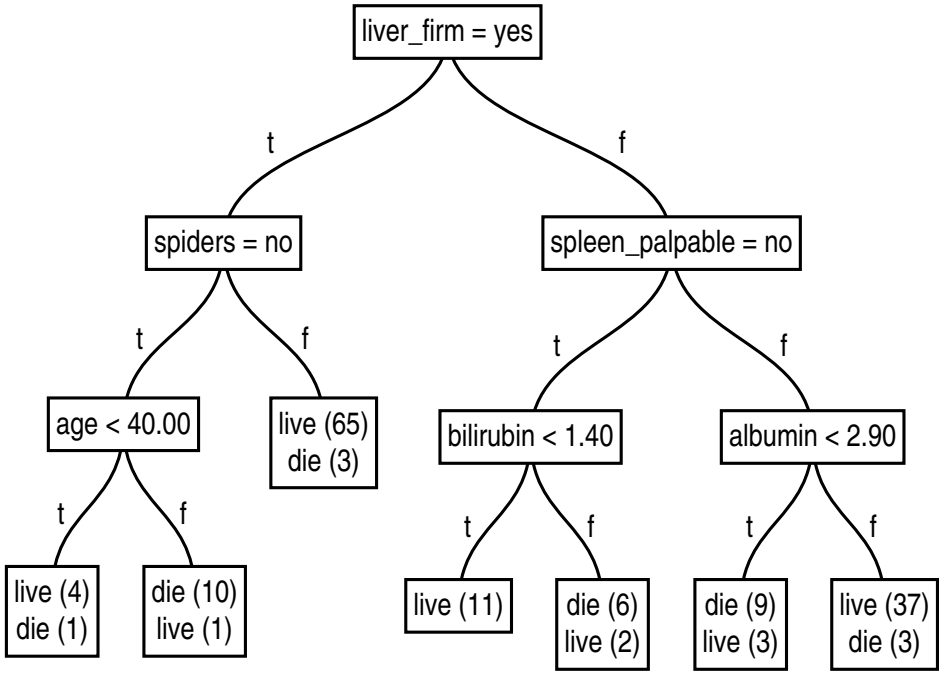
Outlook =Sunny, Temperature=Hot,Humidity=High,Wind=Strong

How do we classify it?



- At every node, test the corresponding attribute
- Send the instance down the appropriate branch of the tree
- If at a leaf, output the corresponding classification

Real example: the “hepatitis” task



Good things about decision trees

- Provide a general representation of classification rules
- Easy to understand!
- Fast learning algorithms (e.g. C4.5, CART)
- Robust to noise (attribute and classification noise, missing values)
- Good accuracy

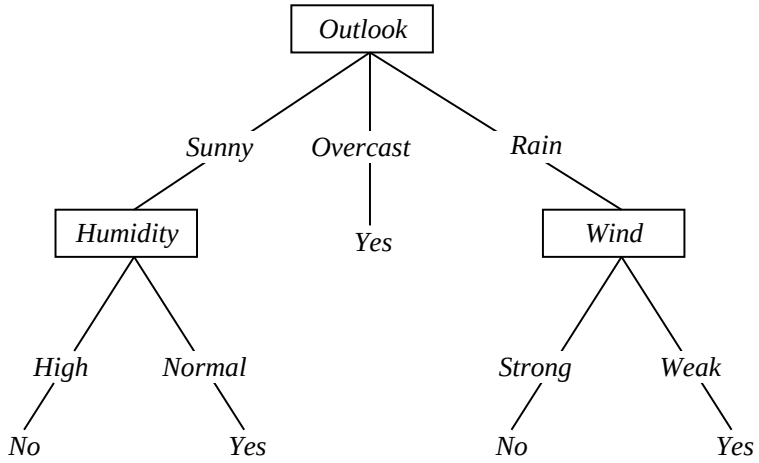
Decision trees are widely used in large, realistic classification problems, e.g.:

- Star classification
- Medical diagnosis
- Industrial applications

Often incorporated in data mining software (e.g. SGI Mineset).

Decision trees as logical representations

Each decision tree has an equivalent representation in propositional logic. For example:



corresponds to:

(Outlook=Sunny $\wedge$ *Humidity=Normal)*

$\vee$ *(Outlook=Overcast)* $\vee$ *(Outlook=Rain* $\wedge$ *Wind=Weak)*

What is easy/hard for decision trees to represent ?

How would we represent:

- $\wedge, \vee$, XOR
- *(A* $\wedge$ *B)* $\vee$ *(C* $\wedge$ *D)*
- *M* of *N*

Natural to represent disjunctions, hard to represent functions like parity, XOR (need exponential-size trees).

Sometimes duplication occurs (same subtree on various paths).

When would one use a decision tree?

- Data is represented as attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data, missing values
- Need to construct a classifier fast
- Need an understandable classifier

Existing applications include:

- Equipment/medical diagnosis
- Learning to fly
- Scene analysis and image segmentation

Standard algorithm developed in the ’80s, now commercially available packages (C4.5). Quite successful in practice

Top-down induction of decision trees

Given a set of labeled training instances:

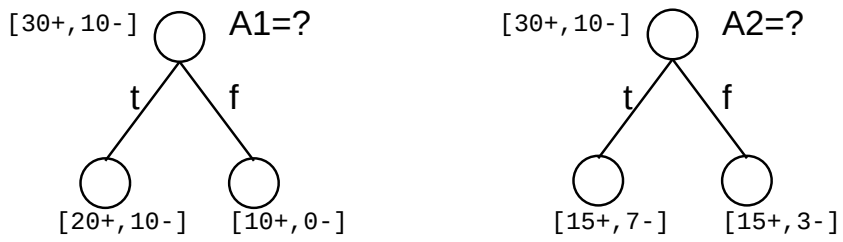
1. If all the training instances have the same class, create a leaf with that class label and exit.
2. Pick the *best* attribute to split the data on
3. Add a node that tests the attribute
4. Split the training set according to the value of the attribute
5. Recurse on each subset of the training data

This is the ID3 algorithm (Quinlan, 1983) and is at the core of C4.5

Which attribute is best?

The attribute should provide **information** about the class label.

Consider we have 29 positive examples, 35 negative ones, and we are considering two attributes, that would give the following splits of instances:



Intuitively, we would like an attribute that *separates* the training instances as well as possible

We need a mathematical measure for the *purity* of a set of instances

Information = Reduction in uncertainty

Imagine:

1. You are about to observe the outcome of a dice roll
2. You are about to observe the outcome of a coin flip

Which one has more uncertainty?

Now suppose:

1. You observe the outcome of the dice roll
2. You observe the outcome of the coin flip

In both cases, now there is no more uncertainty.

Which one provides more information?

Definition of information

Let E be an event that occurs with probability $P(E)$. If we are told that E has occurred with certainty, then we received

$$I(E) = \log_2 \frac{1}{P(E)}$$

bits of **information**.

- You can also think of information as the amount of "surprise" in the outcome (e.g., consider $P(E) = 1$, $P(E) \approx 0$)
- Example: result of a fair coin flip provides $\log_2 2 = 1$ bit of information
- Example: result of a fair dice roll provides $\log_2 6 \approx 2.58$ bits of information.

Information is additive

Suppose you have k independent fair coin tosses. How much information do they give?

$$I(k \text{ fair coin tosses }) = \log_2 \frac{1}{1/2^k} = k \text{ bits}$$

A cute example:

- Consider a random word drawn from a vocabulary of 100,000 words: $I(\text{word}) = \log_2 100,000 \approx 16.61$ bits
- Now consider a 1000 word document drawn from the same source: $I(\text{document}) \approx 16610$ bits
- Now consider a 480×640 gray-scale image with 16 grey levels: $I(\text{picture}) = 307,200 \cdot \log_2 16 = 1,228,800$ bits!

$\Longrightarrow$ A picture is worth (more than) a thousand words!

Entropy

Suppose we have an information source S which emits symbols from an alphabet $\{s_1, \ldots s_k\}$ with probabilities $\{p_1, \ldots p_k\}$. Each emission is independent of the others.

What is the **average amount of information** when observing the output of S?

$$H(S) = \sum_i p_i I(s_i) = \sum_i p_i \log \frac{1}{p_i} = -\sum_i p_i \log p_i$$

Call this **entropy** of S.

Note though that this depends *only on the probability distribution* and not on the actual alphabet (so we can really write $H(P)$).

Interpretations of entropy

$$H(P) = \sum_i p_i \log \frac{1}{p_i}$$

- Average amount of information per symbol
- Average amount of surprise when observing the symbol
- Uncertainty the observer has before seeing the symbol
- Average number of bits needed to communicate the symbol

Entropy and coding theory

- Suppose I will get data from a 4-value alphabet $\mathbf{y}_j$ and I want to send it over a channel. I know that the probability of item $\mathbf{y}_j$ is p_j.
- Suppose all values are equally likely. Then I can encode them in two bits each, so on every transmission I need 2 bits
- Suppose now $p_0 = 0.5$, $p_1 = 0.25$, $p_2 = p_3 = 0.125$.
  Then I can encode $\mathbf{y}_0 = 0$, $\mathbf{y}_1 = 10$, $\mathbf{y}_2 = 110$, $\mathbf{y}_3 = 111$.
  What is the expected length of the message that I will have to send over time?

Shannon: there are codes that will communicate the symbols with efficiency arbitrarily close to $H(S)$ bits/symbol. There are no codes that will do it with efficiency less than $H(S)$ bits/symbol.

Properties of entropy

$$H(P) = \sum_{i=1}^{k} p_i \log \frac{1}{p_i}$$

- Non-negative: $H(P) \geq 0$
- $H(P) \leq \log k$ with equality if and only if $p_i = \frac{1}{k}, \forall i$
- The further P is from uniform, the lower the entropy

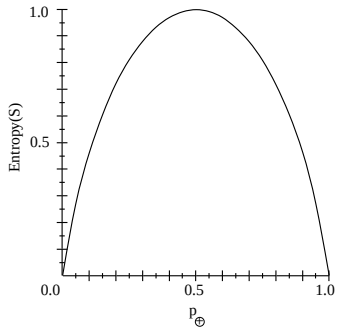
Entropy applied to concept learning

Consider:

- S - a sample of training examples
- $p_\oplus$ is the proportion of positive examples in S
- $p_\ominus$ is the proportion of negative examples in S

Entropy measures the impurity of S:

$$H(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$$



Conditional entropy

Suppose I am trying to predict output Y and I have input X, e.g.:

HasKids	*OwnsDumboVideo*
Yes	*Yes*
Yes	*Yes*
Yes	*Yes*
Yes	*Yes*
No	*No*
No	*No*
Yes	*No*
Yes	*No*

From the table, we can estimate $P(OwnsDumboVideo = Yes)$, $P(OwnsDumboVideo = Yes | HasKids = Yes)$ ...

$H(OwnsDumboVideo) = 1$ (based on the data in the table).

What if we look only at the instances for which $HasKids = No$?

Specific conditional entropy $H(Y|X = v)$ is the entropy of Y among only the instances in which X has value v

Conditional entropy

Conditional entropy, $H(Y|X)$, is the average conditional entropy of Y given specific values for X:

$$H(Y|X) = \sum_v P(X = v)H(Y|X = v)$$

Alternative interpretation: the expected number of bits needed to transmit Y if both the emitter and the receiver know the value of X. In our example:

$$H(O|H) = P(H = Y)H(O|H = Y) + P(H = N)H(O|H = N) = ...$$

Information gain

Suppose I have to transmit Y. How many bits on the average would it save me if both me and the receiver knew X?

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

This is called **information gain**

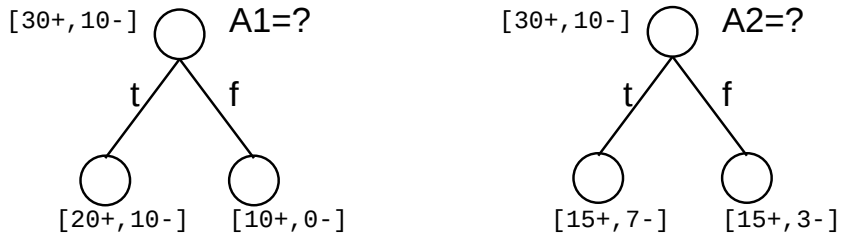
Alternative interpretation: how much reduction in entropy do I get if I know X.

Information gain to determine best attribute

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

where S_v is the subset of instances in which $A = v$.

IG(S,A) = expected reduction in entropy due to sorting on attribute *A*



Check that in this case, A1 wins.

Going back to our example...

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	*Sunny*	*Hot*	*High*	*Weak*	*No*
D2	*Sunny*	*Hot*	*High*	*Strong*	*No*
D3	*Overcast*	*Hot*	*High*	*Weak*	*Yes*
D4	*Rain*	*Mild*	*High*	*Weak*	*Yes*
D5	*Rain*	*Cool*	*Normal*	*Weak*	*Yes*
D6	*Rain*	*Cool*	*Normal*	*Strong*	*No*
D7	*Overcast*	*Cool*	*Normal*	*Strong*	*Yes*
D8	*Sunny*	*Mild*	*High*	*Weak*	*No*
D9	*Sunny*	*Cool*	*Normal*	*Weak*	*Yes*
D10	*Rain*	*Mild*	*Normal*	*Weak*	*Yes*
D11	*Sunny*	*Mild*	*Normal*	*Strong*	*Yes*
D12	*Overcast*	*Mild*	*High*	*Strong*	*Yes*
D13	*Overcast*	*Hot*	*Normal*	*Weak*	*Yes*
D14	*Rain*	*Mild*	*High*	*Strong*	*No*

Which attribute will have the highest information gain?

Problem: Attributes with multiple values

- If an attribute splits the data perfectly, it will always be preferred by information gain.

  E.g. a unique ID for each data point!
- But that has very poor generalization performance!
- Possible solutions:
  - Using better criteria (based on information)
  - Ensuring that all attributes have the same number of values

A better criterion: Gain ratio

For a set of instances S and an attribute A with v possible values

$$GainRatio(S, A) = \frac{IG(S, A)}{H(A)}$$

where

$$H(A) = -\sum_{i=1}^{v} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$

and S_v is the subset of S for which $A = v$.

So for an attribute that splits the data into many partitions mostly uniformly, $H(A)$ will be high

Problem: It can actually become too high!

Solution: First use information gain, then use gain ratio only for attributes with information gain above average

Other such metrics are also used.

Ensuring the same number of values

If an attribute A has $v > 2$ possible values, $Val_1..Val_v$, replace it by v Boolean attributes, $A_k, k = 1..v$, where:

$$A_k = \begin{cases} 1 & \text{if } A = Val_k \\ 0 & \text{otherwise} \end{cases}, \forall k = 1..v$$

This is called *1-of-v encoding*

Used more generally to encode learning data (e.g. in neural networks)

Decision tree construction as search

- State space: all possible trees
- Actions: which attribute to test
- Goal: tree consistent with the training data
- Depth-first search, no backtracking
- Heuristic: information gain (or gain ratio)
- Can get stuck in a local minimum, but is fairly robust (because of the heuristic)

Inductive bias of decision tree construction

- The hypothesis space is complete! We can represent any Boolean function of the attributes
- So there is *no representational bias*
- Outputs a single hypothesis: the “shortest” tree, as anticipated by the information gain
- Because there is no backtracking, it is subject to local minima
- But because the search choices are statistically based, it is robust to noise in the data
- *Algorithmic bias: prefer shorter (smaller) trees; prefer trees that place attributes with high information gain close to the root*

Using decision trees for real data

Lots of issues to deal with!

- How to test real-valued attributes
- How we estimate classifier error
- How to deal with noise in the data
- How to deal with missing attributes
- How to incorporate attribute costs

Example: CRX data, UCI Repository

```
|  This file concerns credit card applications.  All attribute names
|  and values have been changed to meaningless symbols to protect
|  confidentiality of the data.
6
+, -.   | classes

A1:     b,a.
A2:     continuous.
A3:     continuous.
A4:     u, y, l, t.
A5:     g, p, gg.
A6:     c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.
A7:     v, h, bb, j, n, z, dd, ff, o.
A8:     continuous.
A9:     t,f.
A10:    t,f.
A11:    continuous.
A12:    t,f.
A13:    g, p, s.
A14:    continuous.
A15:    continuous.
```


Attributes with continuous values

Example:

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

A decision tree needs to perform tests on these attributes as well

What kind of test do we want?

Value of the attribute less than a cut point!

What cut points should we consider?

We need to consider only cut points where the class label changes!