AI2 Module 3
Tutorial 1

Jacques Fleuriot School of Informatics
(Amended by David Talbot January 6, 2005.)

This tutorial will help you develop some intuition about information-based heuristics used in machine learning. The questions are described in terms of encoding information. Their relevance will become apparent when we discuss the decision tree learning algorithm. Many other machine learning algorithms use similar heuristics.

We have a large collection of marbles of which we want to make an inventory. Each marble has a number printed on it and has one of four colours: Red, Green, Blue, Yellow.
More concretely, we have 1024 marbles, and the numbers printed on marbles are $1, \ldots, 1024$. We also know that there are 512 Red marbles, 256 Green marbles, 128 Blue marbles, 128 Yellow marbles.

To make an inventory we can simply write down a table like the one on the right but we quickly realise that there is no need to write the numbers as they are implicit in the order, so instead we write: (Red,Yellow,Red,...)

1	Red
2	Yellow
3	Red
...	...

1. We ask a friend to save our inventory on a computer disk and she requests that we write it down with bits: 0s and 1s. To satisfy that we decide on an encoding:
   (Red=00,Green=01,Blue=10,Yellow=11)
   and rewrite the inventory as: (00,11,00,...)
   we also notice that the comma symbols are not needed since each letter takes exactly 2 bits so we rewrite again as (001100...)

   Question: How many bits are required to encode the inventory (ignoring the space required for the encoding table)?

2. Our friend complains that the inventory takes too much space and suggests that we might improve things by giving a shorter code to Red (that appears many times) through an increase in the length of the code for Yellow (that appears less time). She warns that doing this will this will probably have an effect on the codes used for Blue and Green as well.

   Question: Try to devise such an encoding. How many bits are needed now ? can you give up the separation between symbols as above ?

3. In general, given k kinds of marbles with $n_1, n_2, \ldots, n_k$ of each respectively, one can try to encode the information in an inventory. Let $n = \sum_{i=1}^{k} n_i$ denote the total number of marbles and $p_i = \frac{n_i}{n}$ be the relative frequency. It is known (though not at all obvious) that asymptotically any scheme must use at least an average of

$$I(p_1, \ldots, p_k) = \sum_{i=1}^{k} p_i \log_2 \frac{1}{p_i} = \sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n}{n_i}$$

   bits for a total of $nI(n_1, \ldots, n_k)$. $I()$ is known as the *entropy*. There are known algorithmic ways of computing optimal codes achieving average close to $I()$ but that is beyond the scope for our course.

   Question: Compute $I()$ for the inventory. How close is the improved scheme to it ?

4. Our friend is still not pleased and wants to save more space. She has looked at our inventory and found a regularity that may be used. In particular, the marbles numbered $1, \ldots, 600$ include all Red marbles and 88 of the Yellow marbles. The idea now is that we use two tables so as to encode each part separately. So we have part 1 with: 512 Red and 88 Yellow marbles and part 2 with: 256 Green, 128 Blue, and 40 Yellow marbles.

   Question: Compute the entropy for each part and the total length required by both parts together assuming we have a code that achieves the entropy.

   Question: Try to construct your own codes for each part and compute the total length required. How close does it come to the value using the entropy ?

   Question: is either of the above better than the original scheme ?

Information Content, Entropy and Prediction

This section gives some more examples of entropy and information content.
The *information content* (IC) of an event is the amount of new information communicated when we learn about the event. The information content of the event $X = i$, where X is a random variable and i is the outcome, is defined as

$$IC(X = i) = \log_2 \frac{1}{p(X = i)}.$$

The definition agrees with a number of common-sense ideas regarding 'information'.

1. More surprising events provide more information.

   For example, if X is a random variable representing the current weather in Edinburgh, the information content of the event 'sunny', $p(sunny = 0.001)$, $IC = 11$ bits, is higher than that of the event 'cloudy', $p(cloudy = 0.8)$, $IC = 0.322$ bits.

2. Learning that an event that was bound to happen, did happen, provides no information.

   Such an event has a probility of 1 and since $\log_2 1 = 0$ the IC is 0 bits.

3. Learning the outcome of related random variables reduces the information content.

   For example, the information provided by learning that a randomly selected English character is 'u' is lower when we already know that the previous letter was'q'.

The *entropy* of a random variable, $I(X)$, is an average of the information content over the outcomes of the random variable. If a variable X has N possible outcomes,

$$I(X) = \sum_{i=1}^{N} p(X = i) \log_2 \frac{1}{p(X = i)}.$$

Entropy can be thought of as the average *uncertainty* of the random variable. Prediction is easier when the entropy is lower since we are less uncertain (on average).

For example, the entropy of a coin is maximized when it is fair i.e. when $p(heads = 0.5)$ and $p(tail = 0.5)$, $I(X) = 0.5 \cdot \log_2 \frac{1}{0.5} + 0.5 \cdot \log_2 \frac{1}{0.5} = \log_2 2 = 1$ bit. A fair coin is also most difficult to predict.

When the coin is biased, say, $p(heads = 0.8)$ and $p(tails = 0.2)$, the entropy will be lower i.e. $0.8 \cdot \log_2 \frac{1}{0.8} + 0.2 \cdot \log_2 \frac{1}{0.2} = 0.722$ bits, and we can win money if we predict heads...

Decision trees predict a class variable by asking questions about (hopefully correlated) attributes of an input example. The answers to these questions reduce the entropy (uncertainty) of the class assignment making prediction gradually easier at each node in the tree.