In general, if the \( n \) possible answers \( v_i \) have probabilities \( P(v_i) \), then the entropy \( H \) of the actual answer is
\[
H(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} - P(v_i) \log_2 P(v_i)
\]

For example, \( H(\frac{1}{2}, \frac{1}{2}) = 2(-\frac{1}{2} \log_2(\frac{1}{2})) = 1 \) bit

In choosing attribute tests, we want to calculate the change of the value distribution \( P(C) \) of the class attribute \( C \), if the training set \( S \) is divided into subsets according to the value of attribute \( a \)
\[
\text{Gain}_S(P(C) \mid a) = \sum_{S_i} (|S_i|/|S|) \cdot H_{S_i}(P(C))
\]
where
\[
\text{Gain}_S(P(C) \mid a) = \sum_{S_i} (|S_i|/|S|) \cdot H_{S_i}(P(C))
\]
when \( a \) divides \( S \) in subsets \( S_i \).

Let the training set \( S \) contain 14 apples and 6 oranges

Hence,
\[
H_S(P(C)) = H(0.7, 0.3) \approx 0.7 \times 0.515 + 0.3 \times 1.737 \approx 0.881
\]

Suppose that attribute \( a \) divides the data s.t.
\[
S_1 = \{7 \text{ apples}, 3 \text{ oranges}\}, \quad S_2 = \{7 \text{ apples}\}, \quad S_3 = \{3 \text{ oranges}\}
\]
then
\[
\text{Gain}_S(P(C) \mid a) = \sum_{S_i} (|S_i|/|S|) \cdot H_{S_i}(P(C))
\]
\[
\approx (10/20) \times H(0.7, 0.3) + 0 + 0
\]
\[
\approx \frac{1}{2} \times 0.881 \approx 0.441
\]
Assessing performance of learning algorithms

- Divide the set of examples into disjoint training set and test set
- Apply the training algorithm to the training set, generating a hypothesis $h$
- Measure the percentage of examples in the test set that are correctly classified by $h$: $h(x) = y$ for an $(x,y)$ example
- Repeat the above-mentioned steps for different sizes of training sets and different randomly selected training sets of each size
- The result of this procedure is a set of data that can be processed to give the average prediction quality as a function of the size of the training set
- Plotting this function on a graph gives the learning curve

An alternative (better) approach to testing is cross-validation

- The idea in k-fold cross-validation is that each example serves double duty as training data and test data
- First we split the data into $k$ equal subsets
- We then perform $k$ rounds of learning; on each round $1/k$ of the data is held out as a test set and the remaining examples are used as training data
- The average test set score of the $k$ rounds should then be a better estimate than a single score
- Popular values for $k$ are 5 and 10 – enough to give an estimate that is statistically likely to be accurate, at the cost of 5 to 10 times longer computation time
- The extreme is $k = n$, also known as leave-one-out cross-validation (LOO[CV], or jackknife)
Generalization and overfitting

- If there are two or more examples with the same description (in terms of attributes) but different classifications $\Rightarrow$ no consistent decision tree exists
- The solution is to have each leaf node report either
  - The majority classification for its set of examples, if a deterministic hypothesis is required, or
  - the estimated probabilities of each classification using the relative frequencies
- It is quite possible, and in fact likely, that even when vital information is missing, the learning algorithm will find a consistent decision tree
- This is because the algorithm can use irrelevant attributes, if any, to make spurious distinctions among the examples

Consider trying to predict the roll of a die on the basis of
- The day and
- The month in which the die was rolled, and
- Which is color of the die,
then as long as no two examples have identical descriptions, the learning algorithm will find an exact hypothesis
- Such a hypothesis will be totally spurious
- The more attributes there are, the more likely it is that an exact hypothesis will be found
- The correct tree to return would be a single leaf node with probabilities close to 1/6 for each roll
- This problem is an example of overfitting, a very general phenomenon afflicting every kind of learning algorithm and target function, not only random concepts
Decision tree pruning

- A simple approach to deal with overfitting is to prune the decision tree.
- Pruning works by preventing recursive splitting on attributes that are not clearly relevant.
- Suppose we split a set of examples using an irrelevant attribute.
- Generally, we would expect the resulting subsets to have roughly the same proportions of each class as the original set.
- In this case, the information gain will be close to zero.
- How large a gain should we require in order to split on a particular attribute?

A statistical significance test begins by assuming that there is no underlying pattern (the so-called null hypothesis) and then analyzes the actual data to calculate the extent to which they deviate from a perfect absence of pattern.
- If the degree of deviation is statistically unlikely (usually taken to mean a 5% probability or less), then that is considered to be good evidence for the presence of a significant pattern in the data.
- The probabilities are calculated from standard distributions of the amount of deviation one would expect to see in random sampling.
- Null hypothesis: the attribute at hand is irrelevant and, hence, its information gain for an infinitely large sample is zero.
- We need to calculate the probability that, under the null hypothesis, a sample of size $v = n + p$ would exhibit the observed deviation from the expected distribution of examples.
Let the numbers positive and negative examples in each subset be \( p_i \) and \( n_i \), respectively.

Their expected values, assuming true irrelevance, are

\[
\begin{align*}
    p'_i &= p \cdot (p_i + n_i) / (p + n) \\
    n'_i &= n \cdot (p_i + n_i) / (p + n)
\end{align*}
\]

where \( p \) and \( n \) are the total numbers of positive and negative examples in the training set.

A convenient measure for the total deviation is given by

\[
D = \sum_{i=1}^{v} \left( (p_i - p'_i)^2 / p'_i + (n_i - n'_i)^2 / n'_i \right)
\]

Under the null hypothesis, the value of \( D \) is distributed according to the \( \chi^2 \) (chi-squared) distribution with \((v-1)\) degrees of freedom.

The probability that the attribute is really irrelevant can be calculated with the help of standard \( \chi^2 \) tables.

The above method is known as \( \chi^2 \) (pre-)pruning.

Pruning allows the training examples to contain noise and it also reduces the size of the decision trees and makes them more comprehensible.

More common than pre-pruning are post-pruning methods in which

- One first constructs a decision tree that is as consistent as possible with the training data and
- Then removes those subtrees that have likely been added due to the noise.

In cross-validation the known data is divided in \( k \) parts, each of which is used as a test set in its turn for a decision tree that has been grown on the other \( k-1 \) subsets.

Thus one can approximate how well each hypothesis will predict unseen data.
Broadening the applicability of decision trees

- In practice decision tree learning has to answer also the following questions
  - Missing attribute values: while learning and in classifying instances
  - Multivalued discrete attributes: value subsetting or penalizing against too many values
  - Numerical attributes: split point selection for interval division
  - Continuous-valued output attributes
- Decision trees are used widely and many good implementations are available (for free)
- Decision trees fulfill understandability, contrary to neural networks, which is a legal requirement for financial decisions