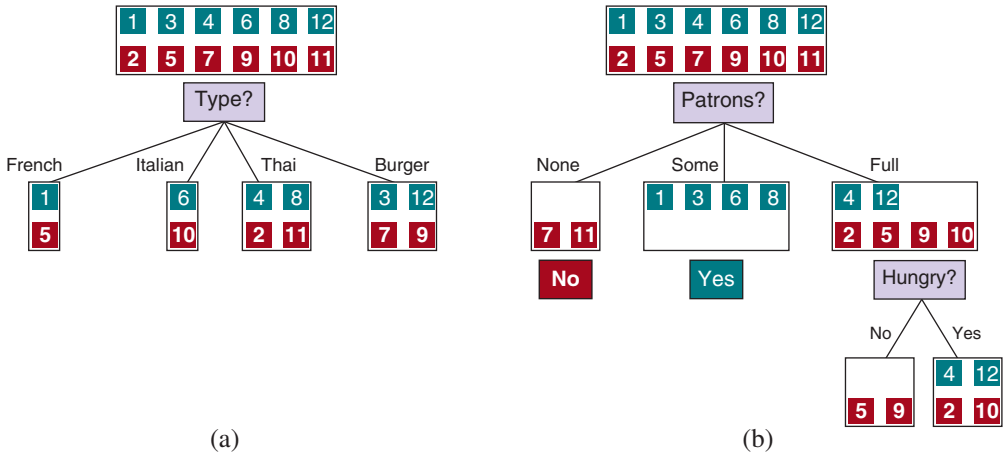

Figure 19.4 Splitting the examples by testing on attributes. At each node we show the positive (green boxes) and negative (red boxes) examples remaining. (a) Splitting on *Type* brings us no nearer to distinguishing between positive and negative examples. (b) Splitting on *Patrons* does a good job of separating positive and negative examples. After splitting on *Patrons*, *Hungry* is a fairly good second test.

outcomes, each of which has the same number of positive as negative examples. On the other hand, in (b) we see that *Patrons* is a fairly important attribute, because if the value is *None* or *Some*, then we are left with example sets for which we can answer definitively (*No* and *Yes*, respectively). If the value is *Full*, we are left with a mixed set of examples. There are four cases to consider for these recursive subproblems:

1. If the remaining examples are all positive (or all negative), then we are done: we can answer *Yes* or *No*. Figure 19.4(b) shows examples of this happening in the *None* and *Some* branches.
2. If there are some positive and some negative examples, then choose the best attribute to split them. Figure 19.4(b) shows *Hungry* being used to split the remaining examples.
3. If there are no examples left, it means that no example has been observed for this combination of attribute values, and we return the most common output value from the set of examples that were used in constructing the node's parent.
4. If there are no attributes left, but both positive and negative examples, it means that these examples have exactly the same description, but different classifications. This can happen because there is an error or **noise** in the data; because the domain is nondeterministic; or because we can't observe an attribute that would distinguish the examples. The best we can do is return the most common output value of the remaining examples.

Noise

The LEARN-DECISION-TREE algorithm is shown in Figure 19.5. Note that the set of examples is an input to the algorithm, but nowhere do the examples appear in the tree returned by the algorithm. A tree consists of tests on attributes in the interior nodes, values of attributes on the branches, and output values on the leaf nodes. The details of the IMPORTANCE function are given in Section 19.3.3. The output of the learning algorithm on our sample training set is shown in Figure 19.6. The tree is clearly different from the original tree shown in Fig-