Figure 6.1 Example of subtree raising, where node C is "raised" to subsume node B.

It is no use taking the training set error as the error estimate: that would not lead to any pruning because the tree has been constructed expressly for that particular training set. One way of coming up with an error estimate is the standard verification technique: hold back some of the data originally given and use it as an independent test set to estimate the error at each node. This is called reduced-error pruning. It suffers from the disadvantage that the actual tree is based on less data.

The alternative is to try to make some estimate of error based on the training data itself. That is what C4.5 does, and we will describe its method here. It is a heuristic based on some statistical reasoning, but the statistical underpinning is rather weak and ad hoc. However, it seems to work well in practice. The idea is to consider the set of instances that reach each node and imagine that the majority class is chosen to represent that node. That gives a certain number of "errors," $E$, out of the total number of instances, $N$. Now imagine that the true probability of error at the node is $q$, and that the $N$ instances are generated by a Bernoulli process with parameter $q$, of which $E$ turn out to be errors.

This is almost the same situation as we considered when looking at the holdout method in Section 5.2, where we calculated confidence intervals on the true success probability $p$ given a certain observed success rate. There are two differences. One is trivial: here we are looking at the error rate $q$ rather than the success rate $p$; these are simply related by $p + q = 1$. The second is more serious: here the figures $E$ and $N$ are measured from the training data, whereas in Section 5.2 we were considering independent test data instead. Because of this difference, we make a pessimistic estimate of the error rate by using the upper confidence limit rather than by stating the estimate as a confidence range.
The mathematics involved is just the same as before. Given a particular confidence $c$ (the default figure used by C4.5 is $c = 25\%$), we find confidence limits $z$ such that

$$
\Pr\left[ \frac{f - q}{\sqrt{q(1-q)/N}} > z \right] = c,
$$

where $N$ is the number of samples, $f = E/N$ is the observed error rate, and $q$ is the true error rate. As before, this leads to an upper confidence limit for $q$. Now we use that upper confidence limit as a (pessimistic) estimate for the error rate $e$ at the node:

$$
e = \frac{f + z^2}{2N} + z\sqrt{\frac{f - f^2}{N} + \frac{z^2}{4N^2}},
$$

$$
1 + \frac{z^2}{N}.
$$

Note the use of the $+$ sign before the square root in the numerator to obtain the upper confidence limit. Here, $z$ is the number of standard deviations corresponding to the confidence $c$, which for $c = 25\%$ is $z = 0.69$.

To see how all this works in practice, let’s look again at the labor negotiations decision tree of Figure 1.3, salient parts of which are reproduced in Figure 6.2 with the number of training examples that reach the leaves added. We use the preceding formula with a 25% confidence figure, that is, with $z = 0.69$. Consider the lower left leaf, for which $E = 2$, $N = 6$, and so $f = 0.33$. Plugging these figures into the formula, the upper confidence limit is calculated as $e = 0.47$. That means that instead of using the training set error rate for this leaf, which is 33%, we will use the pessimistic estimate of 47%. This is pessimistic indeed, considering that it would be a bad mistake to let the error rate exceed 50% for a two-class problem. But things are worse for the neighboring leaf, where $E = 1$ and $N = 2$, because the upper confidence becomes $e = 0.72$. The third leaf has the same value of $e$ as the first. The next step is to combine the error estimates for these three leaves in the ratio of the number of examples they cover, 6 : 2 : 6, which leads to a combined error estimate of 0.51. Now we consider the error estimate for the parent node, health plan contribution. This covers nine bad examples and five good ones, so the training set error rate is $f = 5/14$. For these values, the preceding formula yields a pessimistic error estimate of $e = 0.46$. Because this is less than the combined error estimate of the three children, they are pruned away.

The next step is to consider the working hours per week node, which now has two children that are both leaves. The error estimate for the first, with $E = 1$ and $N = 2$, is $e = 0.72$, and for the second it is $e = 0.46$ as we have just seen. Combining these in the appropriate ratio of 2 : 14 leads to a value that is higher than
the error estimate for the working hours node, so the subtree is pruned away and replaced by a leaf node.

The estimated error figures obtained in these examples should be taken with a grain of salt because the estimate is only a heuristic one and is based on a number of shaky assumptions: the use of the upper confidence limit; the assumption of a normal distribution; and the fact that statistics from the training set are used. However, the qualitative behavior of the error formula is correct and the method seems to work reasonably well in practice. If necessary, the underlying confidence level, which we have taken to be 25%, can be tweaked to produce more satisfactory results.

**Complexity of decision tree induction**

Now that we have learned how to accomplish the pruning operations, we have finally covered all the central aspects of decision tree induction. Let's take stock and consider the computational complexity of inducing decision trees. We will use the standard order notation: $O(n)$ stands for a quantity that grows at most linearly with $n$, $O(n^2)$ grows at most quadratically with $n$, and so on.

Suppose that the training data contains $n$ instances and $m$ attributes. We need to make some assumption about the size of the tree, and we will assume that its depth is on the order of $\log n$, that is, $O(\log n)$. This is the standard rate of growth of a tree with $n$ leaves, provided that it remains "bushy" and doesn't degenerate into a few very long, stringy branches. Note that we are tacitly assum-