node denotes a class prediction. At each node, the algorithm chooses the "best" attribute to partition the data into individual classes.

When decision tree induction is used for attribute subset selection, a tree is constructed from the given data. All attributes that do not appear in the tree are assumed to be irrelevant. The set of attributes appearing in the tree form the reduced subset of attributes. This method of attribute selection is visited again in greater detail in Chapter 5 on concept description.

3.4.3 Data compression

In *data compression*, data encoding or transformations are applied so as to obtain a reduced or "compressed" representation of the original data. If the original data can be *reconstructed* from the compressed data without any loss of information, the data compression technique used is called *lossless*. If, instead, we can reconstruct only an approximation of the original data, then the data compression technique is called *lossy*. There are several well-tuned algorithms for string compression. Although they are typically lossless, they allow only limited manipulation of the data. In this section, we instead focus on two popular and effective methods of lossy data compression: *wavelet transforms*, and *principal components analysis*.

Wavelet transforms

The **discrete wavelet transform (DWT)** is a linear signal processing technique that, when applied to a data vector D, transforms it to a numerically different vector, D', of **wavelet coefficients**. The two vectors are of the same length.

"Hmmm", you wonder. *"How can this technique be useful for data reduction if the wavelet transformed data are of the same length as the original data?"* The usefulness lies in the fact that the wavelet transformed data can be truncated. A compressed approximation of the data can be retained by storing only a small fraction of the strongest of the wavelet coefficients. For example, all wavelet coefficients larger than some user-specified threshold can be retained. The remaining coefficients are set to 0. The resulting data representation is therefore very sparse, so that operations that can take advantage of data sparsity are computationally very fast if performed in wavelet space.

The DWT is closely related to the *discrete Fourier transform (DFT)*, a signal processing technique involving sines and cosines. In general, however, the DWT achieves better lossy compression. That is, if the same number of coefficients are retained for a DWT and a DFT of a given data vector, the DWT version will provide a more accurate approximation of the original data. Unlike DFT, wavelets are quite localized in space, contributing to the conservation of local detail.

There is only one DFT, yet there are several DWTs. The general algorithm for a discrete wavelet transform is as follows.

1. The length, L, of the input data vector must be an integer power of two. This condition can be met by padding the data vector with zeros, as necessary.

2. Each transform involves applying two functions. The first applies some data smoothing, such as a sum or weighted average. The second performs a weighted difference.

3. The two functions are applied to pairs of the input data, resulting in two sets of data of length $L/2$. In general, these respectively represent a smoothed version of the input data, and the high-frequency content of it.

4. The two functions are recursively applied to the sets of data obtained in the previous loop, until the resulting data sets obtained are of desired length.

5. A selection of values from the data sets obtained in the above iterations are designated the wavelet coefficients of the transformed data.

Equivalently, a matrix multiplication can be applied to the input data in order to obtain the wavelet coefficients. For example, given an input vector of length 4 (represented as the column vector $[x_0, x_1, x_2, x_3]$), the 4-point **Haar**

transform of the vector can be obtained by the following matrix multiplication:

$$\begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 & -1/2 \\ 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{3.5}$$

The matrix on the left is *orthonormal*, meaning that the columns are unit vectors (multiplied by a constant) and are mutually orthogonal, so that the matrix inverse is just its transpose. Although we do not have room to discuss it here, this property allows the reconstruction of the data from the smooth and smooth-difference data sets. Other popular wavelet transforms include the Daubechies-4 and the Daubechies-6 transforms.

Wavelet transforms can be applied to multidimensional data, such as a data cube. This is done by first applying the transform to the first dimension, then to the second, and so on. The computational complexity involved is linear with respect to the number of cells in the cube. Wavelet transforms give good results on sparse or skewed data, and data with ordered attributes.

Principal components analysis

Herein, we provide an intuitive introduction to principal components analysis as a method of data compression. A detailed theoretical explanation is beyond the scope of this book.

Suppose that the data to be compressed consists of N tuples or data vectors, from k-dimensions. **Principal components analysis (PCA)** searches for c k-dimensional orthogonal vectors that can best be used to represent the data, where $c << N$. The original data is thus projected onto a much smaller space, resulting in data compression. PCA can be used as a form of dimensionality reduction. However, unlike attribute subset selection, which reduces the attribute set size by retaining a subset of the initial set of attributes, PCA "combines" the essence of attributes by creating an alternative, smaller set of variables. The initial data can then be projected onto this smaller set.

The basic procedure is as follows.

1. The input data are normalized, so that each attribute falls within the same range. This step helps ensure that attributes with large domains will not dominate attributes with smaller domains.
2. PCA computes N orthonormal vectors which provide a basis for the normalized input data. These are unit vectors that each point in a direction perpendicular to the others. These vectors are referred to as the *principal components*. The input data are a linear combination of the principal components.
3. The principal components are sorted in order of decreasing "significance" or strength. The principal components essentially serve as a new set of axes for the data, providing important information about variance. That is, the sorted axes are such that the first axis shows the most variance among the data, the second axis shows the next highest variance, and so on. This information helps identify groups or patterns within the data.
4. Since the components are sorted according to decreasing order of "significance", the size of the data can be reduced by eliminating the weaker components, i.e., those with low variance. Using the strongest principal components, it should be possible to reconstruct a good approximation of the original data.

PCA can be applied to ordered and unordered attributes, and can handle sparse data and skewed data. Multidimensional data of more than two dimensions can be handled by reducing the problem to two dimensions. For example, a 3-D data cube for sales with the dimensions *item_type, branch*, and *year* must first be reduced to a 2-D cube, such as with the dimensions *item_type*, and *branch* $\times$ *year*.

3.4.4 Numerosity reduction

"Can we reduce the data volume by choosing alternative, 'smaller' forms of data representation?" Techniques of *numerosity reduction* can indeed be applied for this purpose. These techniques may be parametric or non-parametric. For *parametric methods*, a model is used to estimate the data, so that typically only the data parameters need be stored, instead of the actual data. (Outliers may also be stored). Log-linear models, which estimate discrete multidimensional probability distributions, are an example. *Non-parametric methods* for storing reduced representations of the data include histograms, clustering, and sampling.