

data cleaning and was addressed in Section 3.2.2. Section 3.2.3 on the data cleaning process also discussed ETL tools, where users specify transformations to correct data inconsistencies. Attribute construction and aggregation were discussed in Section 3.4 on data reduction. In this section, we therefore concentrate on the latter three strategies.

Discretization techniques can be categorized based on how the discretization is performed, such as whether it uses class information or which direction it proceeds (i.e., top-down vs. bottom-up). If the discretization process uses class information, then we say it is *supervised discretization*. Otherwise, it is *unsupervised*. If the process starts by first finding one or a few points (called *split points* or *cut points*) to split the entire attribute range, and then repeats this recursively on the resulting intervals, it is called *top-down discretization* or *splitting*. This contrasts with *bottom-up discretization* or *merging*, which starts by considering all of the continuous values as potential split-points, removes some by merging neighborhood values to form intervals, and then recursively applies this process to the resulting intervals.

Data discretization and concept hierarchy generation are also forms of data reduction. The raw data are replaced by a smaller number of interval or concept labels. This simplifies the original data and makes the mining more efficient. The resulting patterns mined are typically easier to understand. Concept hierarchies are also useful for mining at multiple abstraction levels.

The rest of this section is organized as follows. First, normalization techniques are presented in Section 3.5.2. We then describe several techniques for data discretization, each of which can be used to generate concept hierarchies for numeric attributes. The techniques include *binning* (Section 3.5.3) and *histogram analysis* (Section 3.5.4), as well as *cluster analysis*, *decision tree analysis*, and *correlation analysis* (Section 3.5.5). Finally, Section 3.5.6 describes the automatic generation of concept hierarchies for nominal data.

### 3.5.2 Data Transformation by Normalization

The measurement unit used can affect the data analysis. For example, changing measurement units from meters to inches for *height*, or from kilograms to pounds for *weight*, may lead to very different results. In general, expressing an attribute in smaller units will lead to a larger range for that attribute, and thus tend to give such an attribute greater effect or “weight.” To help avoid dependence on the choice of measurement units, the data should be *normalized* or *standardized*. This involves transforming the data to fall within a smaller or common range such as  $[-1, 1]$  or  $[0.0, 1.0]$ . (The terms *standardize* and *normalize* are used interchangeably in data preprocessing, although in statistics, the latter term also has other connotations.)

Normalizing the data attempts to give all attributes an equal weight. Normalization is particularly useful for classification algorithms involving neural networks or distance measurements such as nearest-neighbor classification and clustering. If using the neural network backpropagation algorithm for classification mining (Chapter 9), normalizing the input values for each attribute measured in the training tuples will help speed up the learning phase. For distance-based methods, normalization helps prevent

attributes with initially large ranges (e.g., *income*) from outweighing attributes with initially smaller ranges (e.g., binary attributes). It is also useful when given no prior knowledge of the data.

There are many methods for data normalization. We study *min-max normalization*, *z-score normalization*, and *normalization by decimal scaling*. For our discussion, let  $A$  be a numeric attribute with  $n$  observed values,  $v_1, v_2, \dots, v_n$ .

**Min-max normalization** performs a linear transformation on the original data. Suppose that  $\min_A$  and  $\max_A$  are the minimum and maximum values of an attribute,  $A$ . Min-max normalization maps a value,  $v_i$ , of  $A$  to  $v'_i$  in the range  $[\text{new\_min}_A, \text{new\_max}_A]$  by computing

$$v'_i = \frac{v_i - \min_A}{\max_A - \min_A}(\text{new\_max}_A - \text{new\_min}_A) + \text{new\_min}_A. \quad (3.8)$$

Min-max normalization preserves the relationships among the original data values. It will encounter an “out-of-bounds” error if a future input case for normalization falls outside of the original data range for  $A$ .

**Example 3.4 Min-max normalization.** Suppose that the minimum and maximum values for the attribute *income* are \$12,000 and \$98,000, respectively. We would like to map *income* to the range  $[0.0, 1.0]$ . By min-max normalization, a value of \$73,600 for *income* is transformed to  $\frac{73,600 - 12,000}{98,000 - 12,000}(1.0 - 0) + 0 = 0.716$ . ■

In **z-score normalization** (or *zero-mean normalization*), the values for an attribute,  $A$ , are normalized based on the mean (i.e., average) and standard deviation of  $A$ . A value,  $v_i$ , of  $A$  is normalized to  $v'_i$  by computing

$$v'_i = \frac{v_i - \bar{A}}{\sigma_A}, \quad (3.9)$$

where  $\bar{A}$  and  $\sigma_A$  are the mean and standard deviation, respectively, of attribute  $A$ . The mean and standard deviation were discussed in Section 2.2, where  $\bar{A} = \frac{1}{n}(v_1 + v_2 + \dots + v_n)$  and  $\sigma_A$  is computed as the square root of the variance of  $A$  (see Eq. (2.6)). This method of normalization is useful when the actual minimum and maximum of attribute  $A$  are unknown, or when there are outliers that dominate the min-max normalization.

**Example 3.5 z-score normalization.** Suppose that the mean and standard deviation of the values for the attribute *income* are \$54,000 and \$16,000, respectively. With z-score normalization, a value of \$73,600 for *income* is transformed to  $\frac{73,600 - 54,000}{16,000} = 1.225$ . ■

A variation of this z-score normalization replaces the standard deviation of Eq. (3.9) by the *mean absolute deviation* of  $A$ . The *mean absolute deviation* of  $A$ , denoted  $s_A$ , is

$$s_A = \frac{1}{n}(|v_1 - \bar{A}| + |v_2 - \bar{A}| + \dots + |v_n - \bar{A}|). \quad (3.10)$$

Thus, z-score normalization using the mean absolute deviation is

$$v'_i = \frac{v_i - \bar{A}}{s_A}. \quad (3.11)$$

The mean absolute deviation,  $s_A$ , is more robust to outliers than the standard deviation,  $\sigma_A$ . When computing the mean absolute deviation, the deviations from the mean (i.e.,  $|x_i - \bar{x}|$ ) are not squared; hence, the effect of outliers is somewhat reduced.

**Normalization by decimal scaling** normalizes by moving the decimal point of values of attribute  $A$ . The number of decimal points moved depends on the maximum absolute value of  $A$ . A value,  $v_i$ , of  $A$  is normalized to  $v'_i$  by computing

$$v'_i = \frac{v_i}{10^j}, \quad (3.12)$$

where  $j$  is the smallest integer such that  $\max(|v'_i|) < 1$ .

**Example 3.6 Decimal scaling.** Suppose that the recorded values of  $A$  range from  $-986$  to  $917$ . The maximum absolute value of  $A$  is  $986$ . To normalize by decimal scaling, we therefore divide each value by  $1000$  (i.e.,  $j = 3$ ) so that  $-986$  normalizes to  $-0.986$  and  $917$  normalizes to  $0.917$ . ■

Note that normalization can change the original data quite a bit, especially when using z-score normalization or decimal scaling. It is also necessary to save the normalization parameters (e.g., the mean and standard deviation if using z-score normalization) so that future data can be normalized in a uniform manner.

### 3.5.3 Discretization by Binning

Binning is a top-down splitting technique based on a specified number of bins. Section 3.2.2 discussed binning methods for data smoothing. These methods are also used as discretization methods for data reduction and concept hierarchy generation. For example, attribute values can be discretized by applying equal-width or equal-frequency binning, and then replacing each bin value by the bin mean or median, as in *smoothing by bin means* or *smoothing by bin medians*, respectively. These techniques can be applied recursively to the resulting partitions to generate concept hierarchies.

Binning does not use class information and is therefore an unsupervised discretization technique. It is sensitive to the user-specified number of bins, as well as the presence of outliers.

### 3.5.4 Discretization by Histogram Analysis

Like binning, histogram analysis is an unsupervised discretization technique because it does not use class information. Histograms were introduced in Section 2.2.3. A histogram partitions the values of an attribute,  $A$ , into disjoint ranges called *buckets* or *bins*.