CHAPTER 1

PATTERN RECOGNITION: FEATURE SPACE CONSTRUCTION

In this chapter, we proceed with a more detailed discussion on the essence and concepts of pattern recognition. We focus on the initial phase of the overall scheme that is focused on feature formation and analysis as well as feature selection. Let us emphasize that, in general, patterns come in various forms: images, voice recordings, text in some natural language, a sequence of structured information (tuples formed according to some key), and so on. A pattern described through a collection of features can be regarded as a generic chunk of information. Features, generally speaking, are descriptors of the patterns. Naturally, the number of features, their nature, and quality influence the quality of ensuing modeling, especially classification. In this chapter, we look at these issues in detail.

The chapter is structured as follows. First, we formulate a theoretical basis of the problems of pattern recognition. We introduce necessary notation and concepts required in the ensuing discussion across the overall book. We formally define features and pattern recognition process. Next, we present practical approaches to feature extraction applied to visual pattern recognition. In particular, we use symbols of printed music notation as examples of patterns. Then, we discuss some elementary feature transformations. Finally, we present various strategies developed for feature selection.

1.1 CONCEPTS

Formally, a standard pattern recognition problem is a task of splitting a set of objects (patterns)

$$O = \{o_1, o_2, \ldots\}$$

Pattern Recognition: A Quality of Data Perspective, First Edition. Władysław Homenda and Witold Pedrycz.

into subsets composed of objects belonging to the same class

$$O_1, O_2, \ldots, O_C$$

such that

$$O = \bigcup_{l=1}^{C} O_l \quad \text{and} \quad (\forall l, k \in \{1, 2, \ldots, C\}, l \neq k) O_l \cap O_k = \emptyset \tag{1.1}$$

A task that results in the formation of subsets $O_1, O_2, \ldots, O_C$ is defined by a mapping called *a classifier*

$$\Psi: O \to \Theta \tag{1.2}$$

where $\Theta = \{O_1, O_2, \ldots, O_C\}$ is the set of classes under consideration. For the sake of simplicity, we assume that the mapping Ψ takes values from the set of class indices $\Theta = \{1, 2, \ldots, C\}$, that is, class labels, instead of classes themselves.

Pattern recognition is usually performed on some observed set of features that characterize objects, rather than on objects directly. Therefore, we formulate and distinguish a mapping from the space of objects O into the space features X:

$$\varphi: O \to X \tag{1.3}$$

The mapping φ is called a features extractor. Subsequently, we consider a mapping from the space of features into the space of classes

$$\psi: X \to \Theta \tag{1.4}$$

Such a mapping is called a classification algorithm or simply a *classifier*. It is important to notice that the term *classifier* is used in different contexts: classification of objects, classification of features characterizing objects, and, more precisely, classification of vectors of features from features space. The meaning of this term can be inferred from the context. Therefore, in the ensuing considerations we will not distinguish explicitly which meaning is being used. A composition of the previously mentioned two mappings constitutes the classifier: $\Psi = \psi \circ \varphi$. In other words, the mapping $O \xrightarrow{\Psi} \Theta$ can be decomposed into the two mappings realized in series $O \xrightarrow{\varphi} X \xrightarrow{\psi} \Theta$.

In general, a classifier Ψ is not known, that is, we do not know the class that a pattern belongs to. However, in pattern recognition problems, it is assumed that the classifier Ψ is known for some subset of the space of patterns called a *learning set*, namely, labels of patterns are known in supervised learning task. A learning set is a subset of the set of objects, $L \subset O$ for which class labels are known, that is, for any pattern from the learning set $o \in L$, the value $\Psi(o)$ is known. Building a classifier $\Psi: O \to \Theta$ with the use of known classes of objects from a learning set, that is, a known mapping $\Psi: L \to \Theta$ is the ultimate goal of pattern recognition. A premise motivating this action is that we hope that having a *good enough* subset of all objects will let us construct a classifier that will be able to successfully assign correct class labels to all patterns. Summarizing, we explore the pattern recognition problem as a design problem aimed at the development of a classifier regarded as the following mapping:

$$\Psi: O \to \Theta$$

assuming that we are provided with $L \subset O$, a subset of all objects with correctly assigned class labels. Such a classifier is decomposed to a feature extractor

$$\varphi : O \to X$$

and a (features) classifier (or, in other words, a classification algorithm)

$$\psi : X \to \Theta$$

as illustrated in Figure 1.1.

Both the feature extractor and the classification algorithm are built based on a learning set L. The classifier ψ divides features space into so-called decision regions,

$$D_X^{(l)} = \psi^{-1}(l) = \{x \in X : \psi(x) = l\} \quad \text{for every } l \in \Theta \tag{1.5}$$

and then, of course, the features extractor splits the space of objects into classes

$$O_l = \varphi^{-1}\left(D_X^{(l)}\right) = \varphi^{-1}\left(\psi^{-1}(l)\right) \quad \text{for every } l \in \Theta \tag{1.6}$$

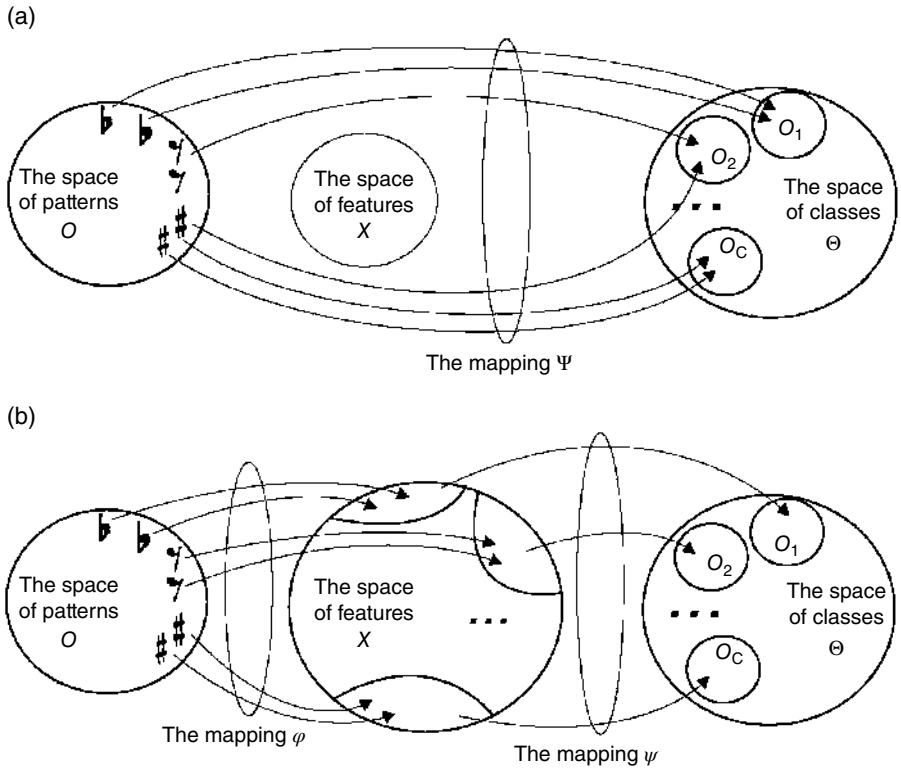

Figure 1.1 Pattern recognition schemes: direct mapping from the space of patterns into the space of classes (a) and composition of mappings from the space of patterns into the space of features and from the space of features into the space of classes (b).

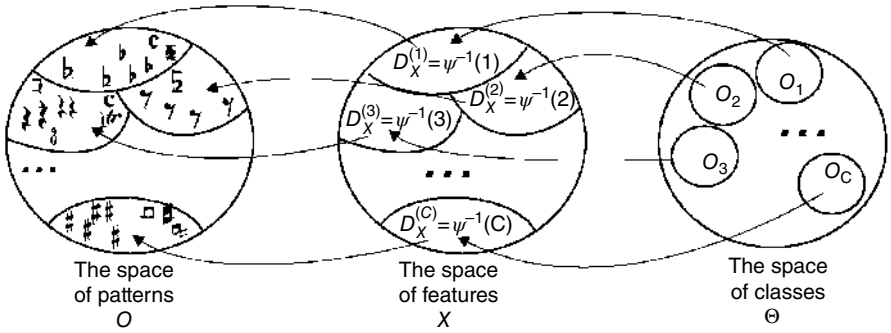

Figure 1.2 A typical pattern recognition scheme.

or equivalently

$$O_l = \Psi^{-1}(l) = (\psi \circ \varphi)^{-1}(l) = \varphi^{-1}\left(\psi^{-1}(l)\right) \quad \text{for every } l \in \Theta \tag{1.7}$$

We assume that the classification algorithm splits the space of feature values, that is, it separates the space X into pairwise disjoint subsets, which cover the entire space X:

$$(\forall l, k \in M, l \neq k) D_X^{(l)} \cap D_X^{(k)} = \emptyset \quad \text{and} \quad \bigcup_{l \in M} D_X^{(l)} = X \tag{1.8}$$

Figure 1.2 illustrates the split of the space of features and the space of objects done by the classifier ψ and the feature extractor φ.

Recognition of objects is usually preceded by an extraction of patterns from a given problem. For instance, dealing with a printed text document or printed sheet music, before proceeding with recognition of symbols, they should be isolated from the environment. In this scenario, a pattern is typically a single symbol (say, a letter or a musical symbol), and patterns are located on a page containing some text message or sheet music with some piece of music. Only after the extraction from the environment are patterns subjected to recognition. If we consider patterns that originate from an image, the task of patterns isolation is usually called segmentation. It is preceded by the stage of preprocessing that facilitates the process of segmentation. In other words, preprocessing aims at introducing various modifications to the source image (e.g., binarization, scaling, etc.) that could help extract patterns of higher quality. For details one may consult in chapter 2 of Krig (2014) where signal preprocessing in pattern recognition is addressed. It is worth noting that not all image acquisition is carried out in a perfect environment, namely, there are a number of possible sources of noise and data of low quality (including imbalanced classes and missing data, among others). There has been a range of studies specifically directed to develop methods for image preprocessing for poor quality signals, for instance, with difficult lighting conditions (Tan and Triggs, 2010) or noise (Haris *et al*., 1998).

Not all pattern recognition tasks have well-defined and clearly delineated preprocessing and symbols extraction (segmentation) stages. Automatic patterns acquisition often produces excessive, undesirable symbols and ordinary garbage. Let us

refer to such patterns as *foreign patterns*, in contrast to *native patterns* of proper, recognized classes (cf. Homenda *et al*., 2014, 2016). In such a case a classification module, which assigns all extracted symbols to designed classes (proper classes of native symbols, labeled and present in the learning set), will produce misclassification for every undesirable symbol and for every garbage symbol. In order to improve the performance of the classification procedure, it is required to construct such classifiers that could assign native symbols to correct class labels and reject undesirable and garbage symbols.

Rejection of symbols can be formally interpreted by considering a new class O_0, to which we classify all undesirable and garbage symbols. In consequence, we can distinguish a classification decision region, which separates foreign symbols from useful ones through the classifier ψ:

$$D_X^0 = \{x \in X : \psi(x) = 0\} \tag{1.9}$$

This new class (decision region) D_X^0 is a distinct subspace of the space X,

$$(\forall l \in C) D_X^{(l)} \cap D_X^{(0)} = \emptyset \quad \text{and} \quad X = D_X^{(0)} \cup \bigcup_{i \in C} D_X^{(l)} \tag{1.10}$$

where, of course, all former classes $D_X^{(l)}, l \in \Theta$ are pairwise disjoint. Rejecting foreign symbols implies a certain problem. Unlike objects of proper classes, foreign symbols are usually not similar to one another and do not create a consistent class. They are not well positioned in the feature space. Moreover, most often they are not available at the stage of classifier construction. Therefore, instead of distinguishing a decision region corresponding to a family of foreign objects, it is reasonable to separate areas outside of decision regions of native objects (cf. Homenda *et al*., 2014). Of course, in such a case, we assume that decision regions of native symbols cover only their own areas and do not exhaust the whole feature space X. An area outside of decision regions of native objects can be formally defined in the following form:

$$D_X^0 = X - \bigcup_{i \in C} D_X^{(i)} \tag{1.11}$$

This is illustrated in Figure 1.3.

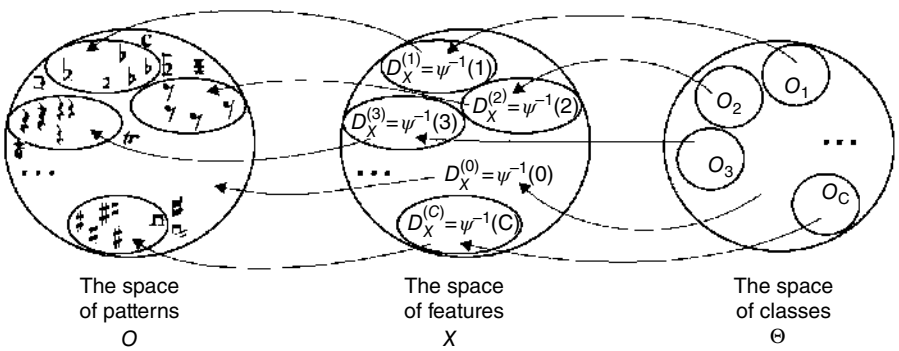

Figure 1.3 Pattern recognition with rejection.

1.2 FROM PATTERNS TO FEATURES

From now on we will use the term *pattern* for objects being recognized as well as for features describing and representing these objects. The exact meaning of this term can be inferred from the context of its usage and, if necessary, it could be explicitly indicated.

Let us distinguish two kinds of features characterizing patterns:

- Numerical features—features whose values form a set of real numbers
- Categorical features—features that are valued in a finite set of values

Values of categorical features can be of any type, for instance, names over some alphabet. Since sets of values of categorical features are finite, they can be enumerated, and values of categorical features can be cast on their indices. Therefore, for the sake of simplicity, categorical features are considered to be numerical ones.

The space of features X is the Cartesian product of individual features X_1, X_2, …, X_M, that is, $X = X_1 \times X_2 \times \cdots \times X_M$. Therefore, mappings φ and ψ operate on vectors $(x_1, x_2, \ldots, x_M)^T$. Such vectors are values of the mapping φ and arguments of the mapping ψ. An i-th element of the vector is denoted x_i, $i = 1, 2, \ldots, M$, and it describes the value of the i-th feature value. For the sake of simplicity, a vector of values of features $\mathbf{x} = (x_1, x_2, x_3, \ldots, x_M)$ will be simply called a vector of features or a feature vector.

Now, let us focus on patterns represented as monochrome images, say black and white images, which are in fact rectangular tables with elements called black/white pixels. In this book, we concentrate the discussion on scanned handwritten digits, handwritten letters, and symbols of printed music notation, and we rarely switch to other types of patterns. This choice is motivated by the fact that, in the context of the methods studied here, such patterns have superior illustrative properties. However, it should be clear that the studied methods are of a general nature and could be easily applied to other types of patterns.

As mentioned before, pattern recognition rarely applies to patterns directly, that is, to patterns present in their original form. In almost each case of pattern recognition, features describing patterns are processed. This observation motivates us to discuss in subsequent sections selected features of monochrome images. These features are especially relevant if we consider processing of printed or handwritten letters, digits, symbols of printed music notation, symbols of geodetic maps, and so on. It is worth stressing that different features can be acquired and applied in order to process other kinds of patterns, for example, those present in speech recognition, signal processing, and others.

In our experiments with recognition of handwritten digits, letters, and symbols of printed music notation, we used the following groups of features: numerical, vectorial, vectorial transformed to vectorial, and vectorial transformed to numerical.

Let us consider a treble clef, an example of a pattern taken from a dataset of music notation symbols. A treble clef is depicted in Figure 1.4. It is a monochrome (black and white) image. We will be referring to such a pattern in terms of a raster scan: a rectangular collection of pixels that we locate inside a *bounding box* of width

Figure 1.4 A treble clef, a symbol belonging to a data set of printed music notation, taken as an example of a pattern. The pattern is surrounded by a bounding box of width $W = 22$ and height $H = 60$ pixels. The bounding box is not part of the pattern; it has been added only for illustrative purposes.

W and height H (cf. Figure 1.4). In other words, a bounding box is the smallest rectangle part of an image enclosing a pattern. In Figure 1.4, the bounding box is identified as a frame used in order to clearly identify the smallest rectangle of pixels enclosing a pattern.

Specifically, a raster scan pattern is represented as a mapping:

$$I: \langle 1, H \rangle \times \langle 1, W \rangle \rightarrow \{0, 1\} \quad I(i,j) = \begin{cases} 1 & \text{for black pixel} \\ 0 & \text{for white pixel} \end{cases} \tag{1.12}$$

1.2.1 Vectorial Features

Only a very limited number of numerical features, effectively employed in pattern recognition problems, can be derived directly from patterns. These features are discussed later in this chapter. However, many numerical features are derived indirectly from vectorial ones.

Vectorial features are usually created based on a bounding box of a given pattern (cf. Figures 1.5 and 1.6). Now, let us discuss the most prominent examples of vectorial features of monochrome images: projections, margins, and transitions.

1. Horizontal and vertical projections:
   - Horizontal projection is a vector of numbers of black pixels in rows.
   - Vertical projection is a vector of numbers of black pixels in columns.

   Therefore, horizontal projection is a vector (of numbers) of length equal to the height of the bounding box (H), while vertical projection is a vector (of numbers) of the length equal to the width of the bounding box (W):

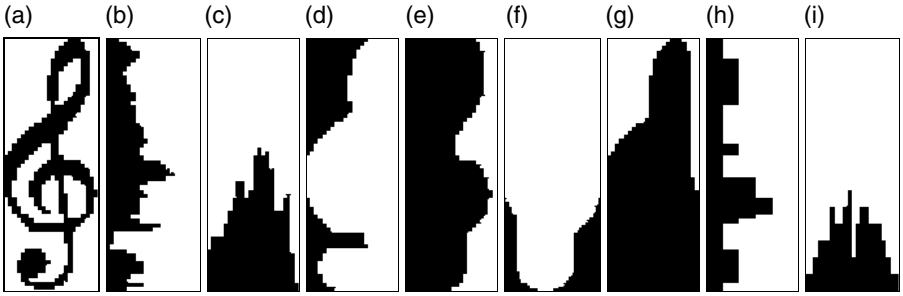

Figure 1.5 Vectorial features: (a) original pattern, (b) horizontal projection, (c) vertical projection, (d) left margin, (e) right margin, (f) bottom margin, (g) top margin, (h) horizontal transition, and (i) vertical transition. Please note that the transition values are very small, so in order to enhance visibility, we multiplied them by 4.

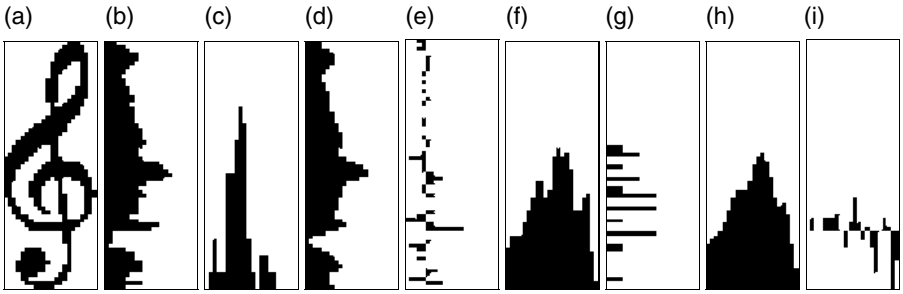

Figure 1.6 Vectorial to vectorial transformations: (a) original pattern, (b) horizontal projection, (c) its histogram, (d) its smoothing, (e) its differentiation, (f) vertical projection, (g) its histogram, (h) its smoothing, and (i) its differentiation. Note: The values of the vertical histogram are multiplied by 4.

$$ProjH(i) = \sum_{j=1}^{W} I(i,j) \quad i = 1,2,\ldots,H$$
$$ProjV(j) = \sum_{i=1}^{H} I(i,j) \quad j = 1,2,\ldots,W \tag{1.13}$$

2. Left, right, bottom, and top margins:

- The left margin are indices of the last white pixels preceding the first black ones, pixel indexing starts from 1 from the left side of each row. It is zero if a row begins with a black pixel; it is W (the width of the bounding box) if no black pixel is in the row.
- The right margin are indices of the last black pixels, pixel indexing starts from 1 from the left side in each row; it is 0 if no black pixel is in the row.
- The bottom and top margins are defined like left and right margins, indexing starts from 1 and goes from bottom to top in each column.

Hence, left and right margins are vectors (of numbers) of lengths equal to the height of the bounding box, while bottom and top margins are vectors (of numbers) of length equal to its width; the detailed formulas for margins are as follows:

$$
\begin{aligned}
MargL(i) &= \begin{cases} W & \text{if } \sum_{j=1}^{W} I(i,j) = 0 \\ \arg\min_{1 \le j \le W} \{I(i,j) = 1\} - 1 & \text{otherwise} \end{cases} \quad i = 1,2,\ldots,H \\
MargR(i) &= \begin{cases} 0 & \text{if } \sum_{j=1}^{W} I(i,j) = 0 \\ \arg\max_{1 \le j \le W} \{I(i,j) = 1\} & \text{otherwise} \end{cases} \quad i = 1,2,\ldots,H \\
MargB(j) &= \begin{cases} H & \text{if } \sum_{i=1}^{H} I(i,j) = 0 \\ \arg\min_{1 \le i \le H} \{I(i,j) = 1\} - 1 & \text{otherwise} \end{cases} \quad j = 1,2,\ldots,W \\
MargT(j) &= \begin{cases} 0 & \text{if } \sum_{i=1}^{H} I(i,j) = 0 \\ \arg\max_{1 \le i \le H} \{I(i,j) = 1\} & \text{otherwise} \end{cases} \quad j = 1,2,\ldots,W
\end{aligned}
\tag{1.14}
$$

3. Horizontal and vertical transitions:

- The horizontal transition is the number of pairs of two consecutive white and black pixels in given rows.
- The vertical transition is the number of such pairs in columns.

Like in the case of projections, transitions are vectors of numbers of respective length.

$$
\begin{aligned}
&TranH: \langle 1, H \rangle \to \langle 1, W \rangle^{H} \quad TranH(j) = \sum_{j=2}^{W} \max\{0, I(i,j) - I(i,j-1)\} \\
&TranV: \langle 1, W \rangle \to \langle 1, H \rangle^{W} \quad TranV(i) = \sum_{i=2}^{H} \max\{0, I(i,j) - I(i-1,j)\}
\end{aligned}
\tag{1.15}
$$

1.2.2 Transformations of Features: From Vectorial to Vectorial

An interesting collection of numerical features can be derived from vectorial features transformed to other vectorial features. Let us present several important vectorial to

vectorial mappings: histograms, smoothings, and differentiations. Illustrative examples of such transformations are presented in Figure 1.6:

1. A histogram and a cumulative histogram are defined on a vector V of length L. Let us assume that elements of vector V are integers located in an interval $\langle 1, L_h \rangle$, that is, $V(i) \in \langle 1, L_h \rangle$. Let us also consider a vector V_h of length L_h. The histogram is a mapping from the vector V to the vector V_h that assigns the number of elements that have value i in the vector V to the i-th element of V_h, that is, assigns this number to the $V_h(i)$, $i = 1, 2, \ldots, L_h$. Given these assumptions we define histogram *Hist* and cumulative histogram *HistC* as follows:

$$V_h(j) = \sum_{i=1}^{L} \begin{cases} 1 & V(i) = j \\ 0 & V(i) \neq j \end{cases} \quad \text{for} \quad j = 1, 2, \ldots, L_h$$
$$V_h(j) = \sum_{i=1}^{L} \begin{cases} 1 & V(i) \leq j \\ 0 & V(i) > j \end{cases} \quad \text{for} \quad j = 1, 2, \ldots, L_h \tag{1.16}$$

   For instance, a histogram of a vertical projection is defined for each integer number i between 0 and the number of rows (H). It counts the number of columns, in which the number of black pixels is equal to i.

2. Smoothing is a mapping defined on a vector that replaces a given value by the mean of this value and its p left and p right neighbors. Both original and result vectors have the same length L. For instance, for $p = 1$ the value is replaced by the mean of this value and its left and right neighbors in the vector. Note that, for $p = 1$, the first and the last elements of the vectors do not have their left and right neighbors, respectively. By analogy, for values of p greater than one, some neighbors of p left and p right elements are missing for leftmost and rightmost elements. The following formulas define smoothing mapping Smth_p for any $p < L/2$:

$$V_{\text{Smth}}(i) = \frac{1}{r - l + 1} \sum_{j=i-l}^{i+r} V(j) \quad i = 1, 2, \ldots, L$$
$$l = \max\{1, i - p\}, \quad r = \min\{L, i + p\} \tag{1.17}$$

3. Differentiation assigns a difference between current and previous elements of vector V to the second and next elements of the result vector V_d:

$$\text{Diff}: V \rightarrow V_d \quad V_d(i) = V(i) - V(i-1) \quad \text{for} \quad i = 2, 3, \ldots, L \quad \text{and} \quad V_d(1) = 0 \tag{1.18}$$

   Notice that differential values may be negative, positive, or equal to 0. The first element of the result differential vector is arbitrarily set to 0.

1.2.3 Transformations of Features: Vectorial to Numerical

As we have already mentioned, a pattern recognition task usually employs numerical features. We have also shown that quite a few interesting characteristics describing

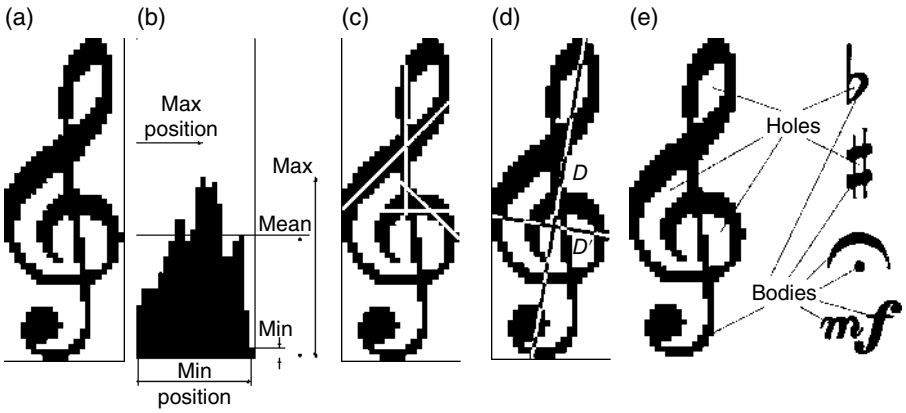

Figure 1.7 Vectorial to numerical transformations: (a) original pattern, (b) numerical features of vertical projection (min = 2, mean = 23, max = 34, min position = 22, max position = 13), (c) directions—white lines on the black pattern (W–E = 13, N–S = 28, NE–SW = 20, NW–SE = 11), (d) eccentricity, and (e) Euler numbers (treble clef: −2, flat: 0, sharp: 0, fermata: 2, mezzo forte: 2).

images could be gathered in the corresponding vectors. Therefore, it becomes imperative to derive numerical characteristics from vectorial features. In this section we discuss principal numerical characteristics of vectorial features. These characteristics can be applied to vectors discussed in the previous sections: projections, margins, and transitions and then histograms, smoothings, and differentiations of projections, margins, and transitions. Transformations from vectorial to numerical features are outlined in the following text and illustrated in Figure 1.7.

1. Minimum, mean, and maximum values of a vector. These transformations can be applied to projections, margins, and transitions. Let V be a vector of length L. Then the following obvious formulas define these concepts:

$$\text{Min value} = \min_{1 \le i \le L} \{V(i)\} \quad \text{Mean value} = \frac{1}{L}\sum_{i=1}^{L} V(i) \quad \text{Max value} = \max_{1 \le i \le L} \{V(i)\} \tag{1.19}$$

2. Positions of minimum and maximum values are just indices of vector's elements with the minimal and maximal values, respectively. If the minimal or maximal value appears more than once in a vector, then the position can be chosen arbitrarily. In the following formulas, the first occurrence is taken as the position. Let V be a vector of length L, and then the following formulas define these features:

$$\begin{aligned} \text{Position of min value} &= \arg \min_{1 \le i \le L} \{V(i) = \text{min value}\} \\ \text{Position of max value} &= \arg \min_{1 \le i \le L} \{V(i) = \text{max value}\} \end{aligned} \tag{1.20}$$

where *min value* and *max value* are defined in (1.19).

3. The zero-order moment ρ_0, the first-order raw moment ρ_1, and the mean value μ_1,

$$\rho_0 = \sum_{i=1}^{L} V(i) \quad \rho_1 = \sum_{i=1}^{L} i \cdot V(i) \quad \mu_1 = \frac{\sum_{i=1}^{L} i \cdot V(i)}{\sum_{i=1}^{L} V(i)} = \frac{\rho_1}{\rho_0}$$

and the second-order raw ρ_2 and central μ_2 moments of a vector V of length L,

$$\rho_2 = \sum_{i=1}^{L} i^2 \cdot V(i) \quad \eta_2 = \sum_{i=1}^{L} (i - \mu_1)^2 \cdot V(i) \tag{1.21}$$

1.2.4 Numerical Features

Several important features can be extracted directly from an image. We discuss here the following features: shape of the bounding box (height to width proportion), blackness, raw and central moments, eccentricity, and Euler numbers. In the following text we present descriptions of the listed features and illustrate the discussion with Figures 1.4 and 1.7.

1. Proportion of the bounding box is just the proportion of its height H to width W:

$$\frac{H}{W} \tag{1.22}$$

2. Blackness is the proportion of the number of black pixels to all pixels in the bounding box:

$$\frac{\sum_{i=1}^{H} \sum_{j=1}^{W} I(i,j)}{H \cdot W} \tag{1.23}$$

3. Raw and central moments. Raw moments of an image are defined as follows:

$$\rho_{kl} = \sum_{i=1}^{H} \sum_{j=1}^{W} i^k j^l \cdot I(i,j) \tag{1.24}$$

where $k + l$ is an order of a moment. Please notice that the moment of order zero is equal to the area of the image (it is the number of black pixels) and the first-order moments ρ_{10} and ρ_{01} define the center of the image (which can be interpreted as the mean value or the center of gravity).

Central moments are defined by the formula

$$\mu_{kl} = \sum_{i=1}^{W} \sum_{j=1}^{H} (i - \rho_{10})^k (j - \rho_{01})^l \cdot I(i,j) \tag{1.25}$$

Notice that $\mu_{00} = \rho_{00}$ and $\mu_{10} = \mu_{01} = 0$. If we compare moments of an image with moments of horizontal and vertical projections of this image, we come to a conclusion that they are identical, that is, the first-order moment ρ_{10} of an image and the first-order moment of its horizontal projection ρ_1 are equal:

$$\rho_{10} = \sum_{i=1}^{H}\sum_{j=1}^{W} i^1 j^0 \cdot I(i,j) = \sum_{i=1}^{H}\left(i \cdot \sum_{j=1}^{W} I(i,j)\right) = \sum_{i=1}^{H}(i \cdot ProjH(i)) = \rho_1 \tag{1.26}$$

Alike, the first-order moment ρ_{01} is equal to the first-order moment ρ_1 of its vertical projection. Analogously, the second-order raw moments ρ_{20} and ρ_{02} and the second-order moments of its vertical and horizontal projections are equal. The same correspondence concerns central moments μ_{20} and μ_{02} and the respective moments of vertical and horizontal projection μ_2.

4. Eccentricity E is defined as the proportion of the length of diameter D to the length of diameter D' perpendicular to D. Diameter D of a pattern is an interval of the maximal length connecting two black pixels of the pattern

$$\text{Length}(D) = \max_{\substack{1 \le i,k \le H \\ 1 \le j,l \le W}} \{d(I(i,j), I(k,l)) : I(i,j) = 1 = I(k,l)\} \tag{1.27}$$

The following formula allows a simple computation of this feature:

$$E = \frac{(\mu_{20} - \mu_{02}) + 4\,\mu_{11}^2}{\mu_{00}} \tag{1.28}$$

where $\mu_{20}, \mu_{02}, \mu_{11}$ are central moments of the second order and μ_{00} is the area of the pattern (equal to the number of black pixels) (cf. Hu, 1962; Sonka *et al*., 1998).

5. Euler numbers 4, 6, and 9. Euler numbers of a pattern represented as a monochrome image describe topological properties of the pattern regardless of its geometrical shape. The Euler number of a binary image is the difference between the number of connected components (*NCC*) and the number of holes (*NH*) (Sossa-Azuela *et al*., 2013):

$$EN = NCC - NH \tag{1.29}$$

A connected component is a region of black (foreground) pixels, which are connected. A hole is a region of connected white (background) pixels enclosed by black pixels. For instance:

- The treble clef has one connected component and three holes, $EN = 1 - 3 = -2$.
- The mezzo forte pattern has two connected components and no holes, $EN = 2 - 0 = 2$.
- The sharp has one connected component and one hole, $EN = 1 - 1 = 0$.

Connectivity depends on the definition of connected components. We consider three types of connectivity:

- 4-Connectivity is computed in a 4-pixel neighborhood, that is, to a given pixel $I(i,j)$, connected are its horizontal $I(i \pm 1, j)$ and vertical $I(i, j \pm 1)$ neighbors.

- 8-Connectivity is calculated in an 8-pixel neighborhood, that is, to a given pixel $I(i,j)$, connected are its horizontal, vertical, and diagonal $I(i\pm 1, j\pm 1)$, $I(i\pm 1, j\mp 1)$ neighbors.
- 6-Connectivity is based in a 6-pixel neighborhood, that is, to a given pixel $I(i,j)$, connected are its horizontal and vertical neighbors and two neighbors on an arbitrarily chosen diagonal, and in this study we consider right-to-left diagonal $I(i\pm 1, j\pm 1)$ neighbors.

6. Directions: vertical (N–S, short for North–South), horizontal (W–E, short for West–East), and diagonal (NW–SE and NE–SW). In brief, direction is the length of the longest run of black pixels of a line in a given direction. For instance, the formulas for vertical and NW–SE diagonal directions read as follows:

$$\begin{aligned} &\max_{1\le i\le H,\,1\le j\le W}\ \max_{l\ge 0,\,r\ge 0}\left\{ l+r+1=\sum_{k=-l}^{r} I(i+k,j)\right\} \\ &\max_{1\le i\le H,\,1\le j\le W}\ \max_{l\ge 0,\,r\ge 0}\left\{ l+r+1=\sum_{i=-l}^{r} I(i+k,j-k)\right\} \end{aligned} \tag{1.30}$$

with an assumption that for a given i and j, values l and r are such that the following obvious inequalities hold: $1\le i-l\le i+r\le H$ and $1\le j-l\le j+r\le W$.

7. A peak in a vector (e.g., in a horizontal/vertical projection, margin, etc.) is an element of this vector that is not smaller than its left and right neighbors, and at the same time it is either not smaller than the 3/4 of the maximum in this vector, or it is not smaller than half of the maximum in this vector, and it exceeds its left and right neighbors by a quarter of the maximum. The following formula defines the number of peaks in a vector V of length L assuming that $\mathrm{MAX}=\max_{1\le i\le L}\{V(i)\}$ is the maximal element in the vector V:

$$\sum_{i=2}^{L-1}\begin{cases} 1 & V(i)>3/4\cdot\mathrm{MAX}\wedge V(i)-\max(V(i-1),V(i+1))\ge 0 \\ 1 & 3/4\cdot\mathrm{MAX}\ge V(i)>1/2\cdot\mathrm{MAX}\wedge V(i)-\max(V(i-1),V(i+1))\ge 1/4\cdot\mathrm{MAX} \\ 0 & \text{otherwise}\end{cases} \tag{1.31}$$

To sum up, a primary step of pattern recognition described in this section aims at the generation of numerical features from images. The proposed mechanisms generate, in total, 171 features (viz., the feature vector contains 171 elements). It turned out that for the two datasets that we have tackled in the detailed experiments (viz., handwritten digits and printed music notation), some of the computed features were constant, so we removed them. As a result, we obtained a list of 159 features; they are listed in Appendix 1.A.

There are numerous papers on feature extraction in pattern recognition. Many of them discuss very specific cases of image analysis, for instance, involving face recognition (Turk and Pentland, 1991), high-speed methods for feature extraction (Bay *et al*., 2008), texture analysis (Manjunath and Ma, 1996), and real-world applications in which we aim at matching between different views of an object or a scene (Lowe, 2004).

1.3 FEATURES SCALING

The values assumed by different features may vary, for instance, comparison of salaries and age. Features in their *natural* form are called *raw features* or we may just say *raw values*. Some raw features may *weigh* much more than the other features. As a consequence, when processing raw features with certain algorithms, we risk that *heavy* features will overshadow the *light* ones. Therefore, there is a need to *scale* raw features, and this process aims at unifying ranges of their values. We consider two types of unification: normalization to the unit interval and standardization based on unification of mean and standard deviation. In both types of scaling, a linear transformation is applied, so characteristics of patterns are preserved.

1.3.1 Features Normalization

A typical normalization linearly transforms raw values of features to the unipolar unit interval [0, 1] or to the bipolar unit interval $[-1, 1]$. In order to give details of such transformation, let us assume that patterns are described by features $X_1, X_2, \ldots, X_M$ and that $x_{i,\min}$ and $x_{i,\max}$ are the minimal and the maximal values of a feature X_i for all patterns located in a learning set. Hence, for a value $x_{i,j}$ of this feature for a given pattern o_j represented as a vector of features $\mathbf{x}_j = \left(x_{1,j}, x_{2,j}, \ldots, x_{M,j}\right)^T$, the corresponding unipolar value is computed as follows:

$$a_{i,j} = \frac{x_{i,j} - x_{i,\min}}{x_{i,\max} - x_{i,\min}} \tag{1.32}$$

The corresponding bipolar value is given by

$$b_{i,j} = 2 \cdot \frac{x_{i,j} - x_{i,\min}}{x_{i,\max} - x_{i,\min}} - 1 \tag{1.33}$$

In Table 1.1, we present example values of features. Numerical features (minimal, maximal, and mean values, positions of minimal and maximal values) are derived from two vectorial features: vertical projection and differential of vertical projections. We outline features of printed music notation symbols. In this example, we consider only eight classes and take a single pattern (symbol) coming from each class. In the consecutive segments of this table, we present raw values, parameters of raw values of the entire learning set (minimal, maximal, mean values, and standard deviation), values normalized to the unipolar and bipolar unit intervals, and standardized values. It is worth noting that normalization and standardization may give undefined values such as mean of differential of vertical projection. This feature is constant; therefore, the denominators in formulas (1.32), (1.33), and (1.34) are 0.

In the normalized learning set of patterns, values fall into the unit interval. However, when we consider patterns not belonging to the learning set, for example, patterns subjected to recognition after the original normalization has been performed, their feature values may be less than the minimum or greater than the maximum of a given feature. We can easily normalize new, incoming patterns, but it may happen that the result values will fall out of the desired unit interval. Hence, such irregularity

TABLE 1.1 Numerical features derived from two vectorial features: vertical projection and differential of vertical projections

	Vertical Projection						Differential of Vertical Projection					
Class Name	Min	Min Position	Max	Max Position	Mean	ρ_1	Min	Min Position	Max	Max Position	Mean	ρ_1
					Raw values							
Mezzo forte	3	0	22	13	10	15.07	−9	15	11	11	0	11.81
Sharp	12	26	30	7	13	13.93	−7	9	10	6	0	12.40
Flat	9	31	32	1	17	12.24	−8	6	6	0	0	10.80
Clef G	3	0	17	11	11	15.69	−6	24	6	7	0	15.07
Clef C	9	0	32	4	23	14.88	−14	10	18	1	0	14.79
Quarter rest	4	0	25	16	12	14.68	−4	20	8	3	0	14.02
Eighth rest	3	3	14	15	9	15.88	−5	17	3	3	0	15.26
1/16 rest	2	30	22	15	11	14.04	−3	16	3	1	0	14.64
…							…					
Mean value	4.73	15.47	23.27	11.63	12.53	14.67	−8.67	15.50	8.10	7.27	0.00	14.17
Standard deviation	2.83	13.68	6.76	7.55	4.26	1.23	6.20	8.68	5.62	7.12	0.00	2.62
Min	1.00	0.00	11.00	0.00	7.00	11.23	−25.00	1.00	2.00	0.00	0.00	8.38
Max	12.00	31.00	32.00	23.00	23.00	16.63	6.20	30.00	21.00	27.00	0.00	19.54
					Values normalized to the unit interval							
Mezzo forte	0.18	0.00	0.52	0.57	0.19	0.71	0.70	0.48	0.47	0.41	NA	0.31
Sharp	1.00	0.84	0.90	0.30	0.38	0.50	0.58	0.28	0.42	0.22	NA	0.36
Flat	0.73	1.00	1.00	0.04	0.63	0.19	0.54	0.17	0.21	0.00	NA	0.22
Clef G	0.18	0.00	0.29	0.48	0.25	0.83	0.83	0.79	0.21	0.26	NA	0.60
Clef C	0.73	0.00	1.00	0.17	1.00	0.68	0.48	0.31	0.84	0.04	NA	0.57
Quarter rest	0.27	0.00	0.67	0.70	0.31	0.64	0.91	0.66	0.32	0.11	NA	0.51
Eighth rest	0.18	0.10	0.14	0.65	0.13	0.86	0.87	0.55	0.05	0.11	NA	0.62
1/16 rest	0.09	0.97	0.52	0.65	0.25	0.52	0.96	0.52	0.05	0.04	NA	0.56

Values normalized to the bipolar unit interval												
Mezzo forte	−0.64	−1.00	0.05	0.13	−0.63	0.42	0.39	−0.03	−0.05	−0.19	NA	−0.38
Sharp	1.00	0.68	0.81	−0.39	−0.25	0.00	0.15	−0.45	−0.16	−0.56	NA	−0.28
Flat	0.45	1.00	1.00	−0.91	0.25	−0.63	0.09	−0.66	−0.58	−1.00	NA	−0.57
Clef G	−0.64	−1.00	−0.43	−0.04	−0.50	0.65	0.65	0.59	−0.58	−0.48	NA	0.20
Clef C	0.45	−1.00	1.00	−0.65	1.00	0.35	−0.04	−0.38	0.68	−0.93	NA	0.15
Quarter rest	−0.45	−1.00	0.33	0.39	−0.38	0.28	0.83	0.31	−0.37	−0.78	NA	0.01
Eighth rest	−0.64	−0.81	−0.71	0.30	−0.75	0.72	0.74	0.10	−0.89	−0.78	NA	0.23
1/16 rest	−0.82	0.94	0.05	0.30	−0.50	0.04	0.91	0.03	−0.89	−0.93	NA	0.12
Standardized values												
Mezzo forte	−0.61	−1.13	−0.19	0.18	−0.60	0.33	−0.05	−0.06	0.52	0.52	NA	−0.90
Sharp	2.57	0.77	1.00	−0.61	0.11	−0.60	0.27	−0.75	0.34	−0.18	NA	−0.68
Flat	1.51	1.14	1.29	−1.41	1.05	−1.97	0.11	−1.09	−0.37	−1.02	NA	−1.29
Clef G	−0.61	−1.13	−0.93	−0.08	−0.36	0.83	0.43	0.98	−0.37	−0.04	NA	0.34
Clef C	1.51	−1.13	1.29	−1.01	2.46	0.17	−0.86	−0.63	1.76	−0.88	NA	0.24
Quarter rest	−0.26	−1.13	0.26	0.58	−0.13	0.01	0.75	0.52	−0.02	−0.60	NA	−0.06
Eighth rest	−0.61	−0.91	−1.37	0.45	−0.83	0.99	0.59	0.17	−0.91	−0.60	NA	0.41
1/16 rest	−0.97	1.06	−0.19	0.45	−0.36	−0.51	0.91	0.06	−0.91	−0.88	NA	0.18

Outlined are features of single patterns (symbols) of a printed music notation.

must be considered in some way. In order to deal with such case, we recommend selecting processing algorithms prone to slight variations in feature values, which with the current advancements in the machine learning area is not a problem.

Alternatively, if there is no other option, one may truncate values located outside the unit interval.

1.3.2 Standardization

Standardization is another method of features unification. Standardization considers not only raw values themselves but also dispersion of values, that is, we employ the mean value and standard deviation of a given feature. Let the following vector $\mathbf{x}_j = \left(x_{1,j}, x_{2,j}, \ldots, x_{M,j}\right)^T$ represent a j-th pattern. The following formula realizes a standardization procedure,

$$u_{i,j} = \frac{x_{i,j} - \bar{x}_i}{\sigma_i} \tag{1.34}$$

where $\bar{x}_i$ is the mean of the feature X_i and σ_i is the standard deviation of this feature:

$$\bar{x}_i = \frac{1}{N}\sum_{j=1}^{N} x_{i,j}, \quad \sigma_i = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(x_{i,j} - \bar{x}_i\right)^2} \tag{1.35}$$

N is the number of patterns in the learning set.

Standardized values of selected features are displayed in the bottom segment of Table 1.1. As mentioned earlier, we outline features of one pattern (symbol) from each of 8 classes of printed music notation. Unlike in the case of normalization, there is no fixed interval that includes the values of a feature, so in order to process a standardized dataset, one cannot choose a classifier that requires all features to fall into some arbitrarily predefined interval.

1.3.3 Empirical Evaluation of Features Scaling

In this section, we discuss the influence of feature values scaling on classification quality. We tested two datasets: musical symbols and handwritten digits. Let us recall that handwritten digits are examples of a well-balanced dataset. In contrast, patterns in the dataset with musical symbols are imbalanced with regard to size, shape, and cardinality.

There are some measures that can be taken to mitigate problems of class imbalance. Even though pattern shape is a property that cannot be modified, the other two aspects—samples cardinality and size—can be adjusted to *balance* an imbalanced dataset. Therefore, we *balanced* values of features, that is, standardized and normalized values. Also, we *balanced* cardinalities of classes.

The dataset of musical symbols consists of 20 classes; see Section 3.4 for details. The majority of classes contain from 200 to 3000 samples. One class is extremely small; it includes only 26 patterns. Based on the original dataset, we built two balanced datasets in which all classes contain exactly 500 patterns, that is, we

obtained two balanced datasets with 10,000 patterns in each of them. In order to construct these datasets, we had to oversample less frequent classes and undersample the frequent ones. For those classes in which there were more than 500 samples, we randomly selected 500 patterns. With regard to rare classes, new patterns were generated in order to reach the total of 500 patterns in a given class. We applied two distinct methods for generation of new patterns, and hence we obtained two balanced datasets.

The essence of the first method for samples generation is defined as follows:

Algorithm 1.1

On intervals oversampling rare class

Data: class O of patterns of cardinality N
Set of Features
N_B the assumed cardinality of the balanced class

Algorithm: initialize the balanced class $O_B = O$
repeat
pick up randomly two native patterns from O:
$\mathbf{X} = (x_1, x_2, \ldots, x_M)$ and $\mathbf{Y} = (y_1, y_2, \ldots, y_M)$
create new pattern $\mathbf{Z} = (z_1, z_2, \ldots, z_M)$ such that for $l = 1, 2, \ldots, M$
z_l is a random value from the interval $[\min(x_l, y_l), \max(x_l, y_l)]$
add Z to O_B
until number of patterns in O_B is not less than N_B

Results: the balanced class O_B of patterns

In the procedure shown earlier, we generate a number of patterns so that the total number of patterns (original patterns plus the ones that are generated) is equal to 500. On the input to this procedure, we pass original samples from a given rare class. Technically, we operate on a data frame of M features describing some patterns. We call this method *on intervals* as new patterns are generated with the use of random intervals formed between feature values of existing patterns.

The alternative method for samples generation is realized as follows:

Algorithm 1.2

Gaussian oversampling rare class

Data: class O of patterns of cardinality N
Set of Features
N_B—assumed cardinality of the balanced class

Algorithm: initialize the balanced class $O_B = O$
compute center $\bar{\mathbf{X}} = (x_1, x_2, \ldots, x_M)$ of the class O
repeat
create new pattern $\mathbf{Z} = (z_1, z_2, \ldots, z_M)$ such that for $l = 1, 2, \ldots, M$
use Gaussian probability distribution $N(\mu, \sigma)$ to sample z_l,
where μ and σ are computed according to (1.35)
add Z to O_B
until number of patterns in O_B is not less than N_B

Results: the balanced class O_B of patterns

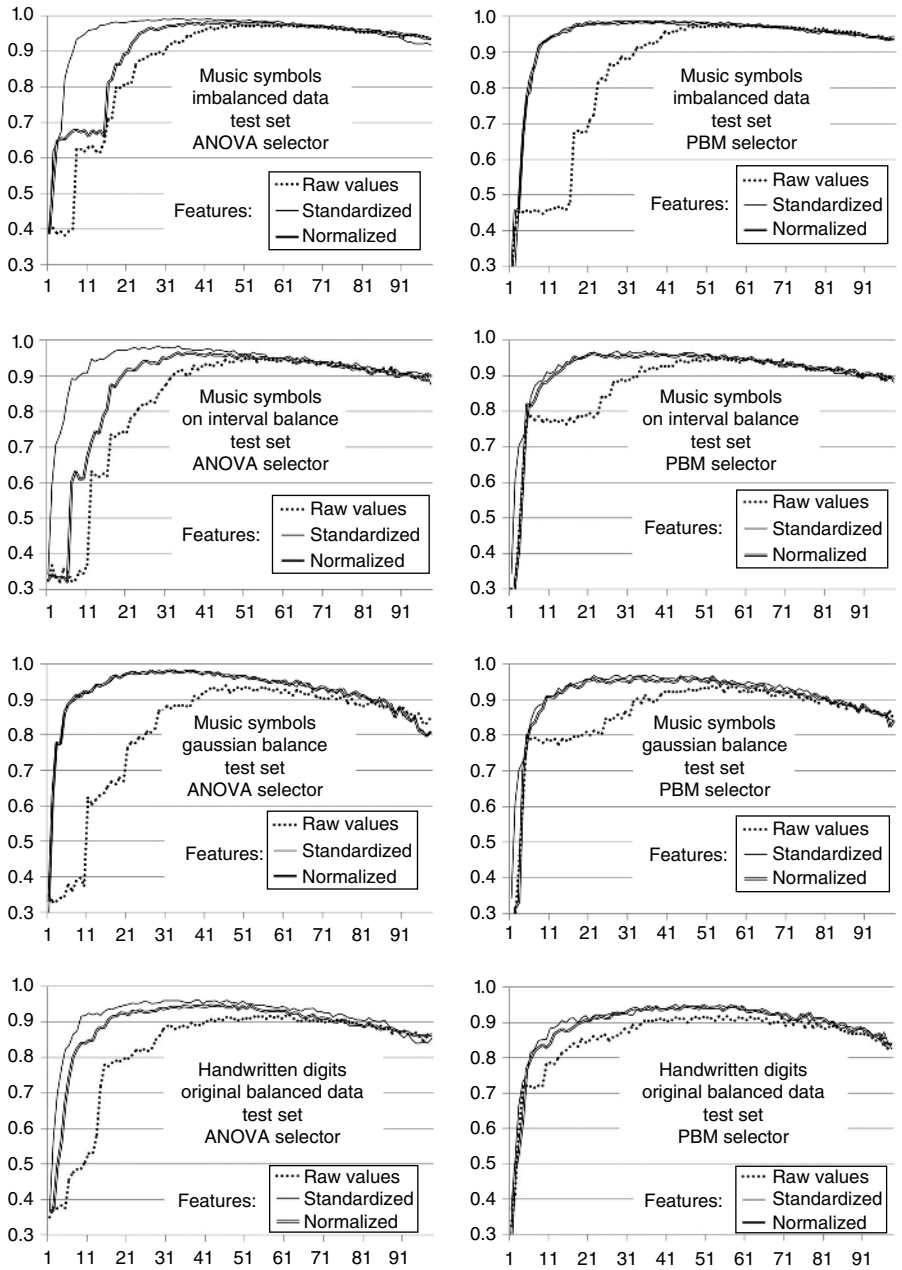

Figure 1.8 Quality of different sets of features selected with the greedy search method. The procedure was adding features one by one: in each iteration one best feature was added. Feature evaluation was performed using the ANOVA F-test and PBM index. We display accuracy (vertical axis) versus feature sets cardinality (horizontal axis). Results concern sets of features ranging from 1 to 100. Plots present accuracy measured on test sets. Plots concern different sets: original, normalized, standardized digits, and musical symbols for the dataset (Homenda *et al*., 2017). Information about the kind of data is presented in each individual plot.

In this procedure, we use normal (Gaussian) probability distribution to approximate distribution of patterns in a given class. We call this data generation procedure *Gaussian* generation.

The dataset of handwritten digits consisted of 10 classes with around 1000 patterns in each class. The total number of patterns is exactly 10,000. This dataset is balanced, and there is no need to undersample the patterns coming from the dominant class or oversample the patterns that form a minority class.

In Figure 1.8, we present results of empirical tests done for the four datasets described earlier: original dataset of handwritten digits, original (unmodified) dataset of music notation, music notation dataset balanced with the *on intervals* method, and music notation dataset balanced with the *Gaussian* method. In addition, in each individual plot in Figure 1.8, we present (for comparative purposes) results achieved for raw data versus normalized data versus standardized data. We display the accuracy of SVM classifiers, which were built using sets of features of various sizes. We checked sets of features of all cardinalities between 1 and 100. In order to select a particular set of features, we ran a greedy search algorithm employing the ANOVA F-test and PBM index for evaluation of features quality (cf. Section 1.4 for the details of the features selection process).

In the experiment with the results plotted in Figure 1.8, the classes forming learning sets were split randomly into the training and test sets in proportion 70–30%. Training sets were used to construct classifiers, while the accuracy was evaluated on the test sets.

The results displayed in Figure 1.8 indicate that the standardization of a dataset helps achieve better numerical accuracy. The results for raw data are worse than for the standardized one. The results for normalized data are worse or in a few cases similar to the results achieved with standardized data. The differences are clearly visible for feature sets of size ranging from 10 to 50. For very large feature sets, the classification results tend to be very similar, no matter which dataset was used. Naturally, such large feature sets are not recommended as we see a clear effect of overfitting that makes test accuracy drop after achieving some peak. Peaks in the accuracy test occur for feature sets consisting of more than 20 but less than 30 features.

It is important to notice that balancing raw (not normalized and not standardized) musical symbols dataset improves accuracy a lot, especially for the PBM index. On the other hand, accuracy obtained on original (imbalanced) musical symbols dataset for standardized features (both ANOVA F-measure and PBM index) and for normalized features (PBM index) is slightly better than for balanced datasets.

1.4 EVALUATION AND SELECTION OF FEATURES

In the previous sections of this chapter, we have presented the general idea of how to represent patterns using features. The discussion was focused on extracting various features from monochrome, segmented images of patterns. Up to this point we have

avoided a very crucial topic: quality of features. In this section, we take a closer look at this issue.

Taking up a pattern recognition task, we have to be aware of the fact that extracted features may be of poor quality. This problem manifests itself in a few aspects. First, we may be in possession of too many features and our processing algorithm may not be able to handle them efficiently. The second apparent flaw is that features may be redundant (duplicated), constant, or may carry utterly useless information (such as the National Insurance Number that is unique for each person and does not carry any predictive strength). Last, we may have heavily correlated features or features of general poor quality (e.g., with missing values). In any case, before actual pattern recognition it is obligatory to get to know the data we process. When it comes to features, we need to evaluate them. The objective of this task is to select such a subset that would be valuable for classification. In the following section, we discuss problems of features evaluation and selection of an optimal feature subset.

1.4.1 Correlation

It may happen that some features are very similar or even identical in the sense that, for instance, their values are linearly dependent for the considered learning set of patterns. A few linearly dependent features do not carry more information than only one of them. If we have a pair of correlated features, only one of them is needed in the process of classifier construction. Redundant features should be dropped because they do not carry new information. Moreover, they complicate the construction of the model, and in this way we risk constructing a poor model.

To illustrate dependence between features, let us consider mean values of vertical and horizontal projections and blackness. These features are defined by formulas (1.13) and (1.23). It is obvious that they are proportional: their values are equal to the number of black pixels in a bounding box of a pattern divided by width, height, and the product of width and height of the box, respectively:

$$\text{Mean}_{\text{vert,proj}} = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} I(i,j)}{W} \quad \text{Mean}_{\text{hor, proj}} = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} I(i,j)}{H} \quad \text{Blackness} = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} I(i,j)}{W \cdot H} \tag{1.36}$$

Hence, these three features are strictly correlated. It is sufficient to multiply values of one feature by a constant in order to obtain the values of some other features:

$$\text{Mean}_{\text{hor, proj}} = \frac{W}{H} \cdot \text{Mean}_{\text{vert,proj}} \quad \text{Blackness} = \frac{1}{H} \cdot \text{Mean}_{\text{vert,proj}} \tag{1.37}$$

and, of course, two out of these three features should be eliminated.

The strength of correlation between two features is expressed by the Pearson correlation coefficient. To compute this coefficient, let us consider two numerical features X_k and X_l and their values for all N patterns from the learning set: $x_{1,k}, x_{2,k}, \ldots, x_{N,k}$

and $x_{1,l}, x_{2,l}, \ldots, x_{N,l}$, respectively. The Pearson correlation coefficient is defined as follows:

$$r_{k,l} = \frac{\sum_{i=1}^{N} (x_{i,k} - \bar{x}_k)(x_{i,l} - \bar{x}_l)}{\sqrt{\sum_{i=1}^{N} (x_{i,k} - \bar{x}_k)^2} \sqrt{\sum_{i=1}^{N} (x_{i,l} - \bar{x}_l)^2}} \tag{1.38}$$

where the mean values of these features are computed according to the formulas as

$$\bar{x}_k = \frac{1}{N} \sum_{i=1}^{N} x_{i,k}, \quad \bar{x}_l = \frac{1}{N} \sum_{i=1}^{N} x_{i,l} \tag{1.39}$$

The Pearson correlation coefficient is a number from the interval $[-1, 1]$. It assumes the value equal to 1 for a pair of strictly correlated features, that is, a pair of features that are in a linear relationship such that when one feature increases, the second one increases as well. Such behavior is exhibited by any two features outlined in (1.36). This conclusion is trivially obtained by replacing feature values in (1.38) with the product of the values of another feature and respective coefficient. On the other hand, two linearly dependent features with a negative linear coefficient produce the correlation value equal to -1. As an example of such two features, we may give blackness and whiteness. Another example of a correlation with a negative coefficient is the mean value of horizontal projection and the mean value of a cumulative histogram of vertical projection. This pair of features has a correlation coefficient very close to -1.

For a given set of features, we compute a correlation matrix, identify groups of correlated features, and then eliminate all but one feature from every such group. It should be mentioned that, typically, we do not see strictly correlated features, that is, pairs with the absolute value of coefficients equal to 1, as in the cases outlined earlier. Usually, we have strongly correlated features, that is, pairs of features such that their absolute correlation values are high. We note that it is up to the model designer to determine a cut threshold that allows making a decision on which pair of features is correlated. Assuming that we have selected such threshold, say, the absolute value of 0.6, we shall investigate pairs of features correlated to a degree exceeding 0.6, and we should drop redundant features.

Table 1.2 presents correlation coefficients between features listed before in Table 1.1. Of course, the matrix is symmetrical and has 1s at the main diagonal. In this matrix each pair of features with the absolute value of the correlation coefficient greater than 0.6 is highlighted in boldface. For example, the following pairs of features are correlated: the raw moment of the differential of vertical projection is correlated with the position of the maximal value of vertical projection (0.64) and the first raw moment of vertical projection with the values of vertical projection (0.65) and with the position of the minimum value of differential of this projection (0.64). If we want to eliminate these correlations, we should either remove the raw moment of the differential of vertical projection or two other correlated features. There is no general rule on which one of them should be removed. If we have, for instance, evaluated the quality

TABLE 1.2 Matrix of the Pearson correlation coefficients for features outlined in Table 1.1

		Vertical Projection						Differential of Vertical Projection					
		Min	Min Position	Max	Max Position	Mean	ρ_1	Min	Min Position	Max	Max Position	Mean	ρ_1
Vertical Projection	Min	**1**	0.17	0.32	−0.04	**0.72**	−0.09	0.01	0.04	−0.11	0.10	0.02	0.03
	Min Position	0.17	**1**	0.07	−0.07	0.19	−0.38	0.05	0.02	−0.13	0.04	−0.02	0.06
	Max	0.32	0.07	**1**	−0.39	**0.60**	−0.39	**−0.63**	−0.24	0.54	−0.04	−0.02	−0.41
	Max Position	−0.04	−0.07	−0.39	**1**	−0.03	0.57	0.45	0.53	−0.38	0.20	0.03	**0.64**
	Mean	**0.72**	0.19	**0.60**	−0.03	**1**	−0.06	−0.09	0.16	0.02	0.02	0.02	0.10
	ρ_1	−0.09	−0.38	−0.39	0.57	−0.06	**1**	0.35	0.41	−0.22	0.24	0.07	**0.65**
Differential Vertical Projection	Min	0.01	0.05	**−0.63**	0.45	−0.09	0.35	**1**	0.29	**−0.71**	0.02	0.04	0.43
	Min Position	0.04	0.02	−0.24	0.53	0.16	0.41	0.29	**1**	−0.29	0.11	0.03	**0.64**
	Max	−0.11	−0.13	0.54	−0.38	0.02	−0.22	**−0.71**	−0.29	**1**	−0.15	0.01	−0.48
	Max Position	0.10	0.04	−0.04	0.20	0.02	0.24	0.02	0.11	−0.15	**1**	−0.04	0.39
	Mean	0.02	−0.02	−0.02	0.03	0.02	0.07	0.04	0.03	0.01	−0.04	**1**	0.01
	ρ_1	0.03	0.06	−0.41	**0.64**	0.10	**0.65**	0.43	**0.64**	−0.48	0.39	0.01	**1**

The bold font is used to highlight coefficients of the absolute value greater than 0.6. Coefficients at the main diagonal are obviously equal to 1.

of single features (cf. Section 1.4.2 for such evaluations), then the weaker and not the stronger one(s) may be removed.

1.4.2 Evaluation of Features: Two Approaches

Features correlation allows to identify similar features and then eliminate dependent ones. Equally important is features evaluation with regard to their usefulness in classification. The challenge is to find a possibly small set of features, which would guarantee construction of a high quality classifier. Unfortunately, there is no single suitable method of low computational cost to select the best subset of features for all data. In practice, we distinguish two kinds of approaches for feature selection:

- Index-based methods
- Wrapper-based methods

The first family of methods relies on relatively simple indices that evaluate features. They rely on dependencies such as relationships between the feature and dependent variables and relationships between features themselves. The upside of applying index-based approaches is that they require moderate computational effort.

The so-called wrapper-based methods rely on constructing multiple models based on various features subsets. After forming collection of classification models, we compare their efficiency and select the best one. In order to be sure that for a given classification method, we selected the best subset of features, and we need to build classifiers for all possible subsets of features. In practice, however, this is not a feasible option, especially when the full feature set is large. To limit computational overhead induced by a brute force feature search, we can apply several greedy algorithms that limit the number of checked feature subsets in some way. Still, this method is computationally costly and time consuming. In addition, if we switch to another classification algorithm, then in order to obtain the same high quality model, it will be desirable to repeat the whole procedure. Even though the mentioned negative aspects are hard to overlook, it shall be mentioned that wrapper-based methods provide models of superior numerical quality. For this reason in further parts of this chapter, we take a closer look at wrapper-based feature search methods.

The literature of the topic offers a wide range of papers in which we find quite elaborate examples of how appropriate feature selection ensures proper processing abilities. One may read on unsupervised similarity-based feature selection in Mitra *et al*. (2002), mutual information-based feature selection in Peng *et al*. (2005), feature selection for bioinformatics data in Saeys *et al*. (2007) and Guyon *et al*. (2002), rough set-based feature selection in Swiniarski and Skowron (2003), and more. There are also papers in which we find elaborations on feature selection techniques directed to be used with some particular classifier, for instance, SVM in Huang and Wang (2006). It is also worth to consult a general-purpose surveys on feature selection, in, for instance, Trier *et al*. (1996) and Kudo and Sklansky (2000).

1.4.3 Index-Based Feature Evaluation: Single Feature Versus Feature Set Evaluation

Let us now address an important distinction arising in a scenario when we apply index-based methods for feature selection. Let us stress that index-based feature evaluation could be applied in two variants:

- To evaluate the quality of a single feature
- To evaluate the quality of a subset of k features

The first strategy, outlined in Algorithm 1.3, simply requires evaluating features one by one.

Algorithm 1.3
Index-based evaluation of features: one-by-one scheme
Data: Set of Features for Evaluation
Learning Set of Patterns
Index for Feature Evaluation
Algorithm: **for each** feature in the Set of Features
evaluate feature **using** Index for Feature Evaluation
Results: vector with quality score for each feature in the Set of Features

If we sort the output of Algorithm 1.3, we construct feature ranking. Subsequently, the model designer will be able to select a subset of features individually evaluated as the best, in the hope that such features together will provide us with a high quality model. Selection of k (the number of features) could be performed with a plot of index values. Looking for a knee-point present in the plot is a standard way when making an empirically guided selection of parameters.

The second strategy evaluates features as a group. This, however, entails a certain problem as to how to select a features subset for evaluation. This issue will be tackled later in this chapter. At this point let us focus on a description of indices for feature evaluation in the context of single feature evaluation. Later on, this discussion will be extended into the second scenario in which we evaluate subsets of features.

1.4.4 Indices for Feature Evaluation

We turn attention to two kinds of methods of low computational complexity: statistical indices for feature evaluation and indices used in clustering quality verification.

Even though clustering is an assignment with a motivation different than pattern recognition, we see a lot of similarities between those two. Intuitively, we see an analogy between clusters and classes: we assume that classes, like clusters, gather similar objects or, in other words, they reside in some designed subspace of the feature space. Therefore, we perceive cluster validity indices as feasible candidates for feature evaluation indices.

In the following sections we describe four indices: ANOVA F-test (statistical index) and three clustering indices.

ANOVA F-Test

The ANOVA F-test (*an*alysis *o*f *va*riance) is a statistical test used to assess whether the expected values of a quantitative variable within several predefined groups differ from each other. This test is used to evaluate the ability of a feature to differentiate classes of native patterns. Roughly speaking, the evaluation is described by the following proportion:

$$F = \frac{\text{between}-\text{class variability}}{\text{within}-\text{class variability}} \tag{1.40}$$

Let N_i, $i = 1, 2, \ldots, C$ be the cardinalities of classes of native patterns, where of course $N_1 + N_1 + \cdots + N_C = N$ and $x_{i,j}$ stands for the value of the (considered) feature of the j-th pattern from the i-th class, $j = 1, 2, \ldots, N_i$, $i = 1, 2, \ldots, C$. Let $\bar{x}_i$ $i = 1, 2, \ldots, C$ and $\bar{x}$ be the mean values of the feature in respective classes and in the whole learning set:

$$\bar{x}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} x_{i,j}, \quad i = 1, 2, \ldots C, \quad \bar{x} = \frac{1}{N}\sum_{i=1}^{C}\sum_{j=1}^{N_i} x_{i,j} \tag{1.41}$$

Then the ANOVA F-test for a given feature is defined by the following formula:

$$F = \frac{\dfrac{1}{C-1}\displaystyle\sum_{i=1}^{C} N_i(\bar{x}_i - \bar{x})^2}{\dfrac{1}{N-C}\displaystyle\sum_{i=1}^{C}\sum_{j=1}^{N_i}\left(x_{i,j} - \bar{x}_i\right)^2} \tag{1.42}$$

It is clear that the more dispersed centers of classes and the more compact classes inside are, the greater the value of the ANOVA F-test becomes. This observation implies that the greater the ANOVA F-test is, the easier it is to separate classes from each other. Finally, the quality of features is consistent with the values of the ANOVA F-test: the greater the value of this test is, the better class separation the feature provides. Interestingly, ANOVA F-test turned out to be a great match for feature selection in natural language processing (Elssied *et al.*, 2014).

Clustering Indices

Among a multitude of cluster validity indices, let us select a few that turned out, after a series of empirical experiments, to be well suited to express the quality of features. These are the McClain–Rao (MCR) index, the generalized Dunn index (GDI), and the PBM index.

The McClain–Rao Index (MCR)

The first index considered here was proposed in McClain and Rao (1975) and Charrad (2014). This index expresses the ratio of two terms that is the average distance between pairs of points of the same cluster and pairs of points of different clusters. The following formula defines this index:

$$\text{MCR} = \frac{P_2(N) - \Sigma P_2(N_i)}{\Sigma P_2(N_i)} \cdot \frac{\displaystyle\sum_{i=1}^{C}\sum_{1 \le k < l \le N_i} |x_{i,k} - x_{i,l}|}{\displaystyle\sum_{1 \le i < j \le C}\sum_{1 \le k \le N_i}\sum_{1 \le l \le N_j} |x_{i,k} - x_{i,l}|}$$

where

$$P_2(N) = N(N-1)/2 \quad \text{and} \quad \Sigma P_2(N_i) = \sum_{i=1}^{C} N_i(N_i-1)/2 \tag{1.43}$$

and $P_2(N)$ is the number of all distances computed between points of all clusters (more precisely, the number of all sets consisting of two points from all clusters) and $\Sigma P_2(N_i)$ is the number of all distances between points in the same cluster (the number of all sets consisting of two points from the same cluster). The meaning of other symbols is analogous to the notation being used in the section devoted to the ANOVA F-test.

Considering (1.42) to be the product of two terms, we may interpret the second term as the sum of within-cluster distances divided by the sum of between-cluster distances. The first term counts the number of the between-cluster distances divided by the number of the within-cluster distances. The product of both terms is just a proportion of the average within-cluster distance and the average between-cluster distance.

The minimum value of the index is used to indicate optimal clustering. Therefore, the smaller the value of this index is, the better the feature quality is. Finally, for the sake of consistency with the other indices, we reverse the rank of features formed with the MCR.

The Generalized Dunn Index (GDI)

The Dunn index (Dunn, 1973) defines the ratio between the minimal between-cluster distance to the maximal within-cluster distance. This index was generalized to the GDI by using different definitions of the minimal between-cluster distance and the maximal within-cluster distance (cf. Desgraupes, 2013). We use the version GDI_{41} given by the following formula (cf. Desgraupes, 2016). Of course, when applying clustering indices for feature set evaluation in the context of classification, we replace cluster belongingness with class memberships:

$$\mathrm{GDI}_{41} = \frac{\min\limits_{1 \le k < l \le C} |\bar{x}_k - \bar{x}_l|}{\max\limits_{1 \le k \le C} \max\limits_{1 \le i < j \le N_k} |x_{k,i} - x_{k,j}|} \tag{1.44}$$

If the dataset contains compact and well-separated clusters, the diameters of clusters are expected to be small, and the distances between clusters are expected to be large. Thus, the Dunn index should be maximized. Therefore, the higher the value of this index is, the better the quality of the feature is.

The PBM Clustering Index

The third cluster validity index considered here was proposed in Bandyopadhyay *et al*. (2004), and it is called PBM (after the names of the authors, Pakhira, Bandyopadhyay, and Maulik). Adapting the notation used in the ANOVA F-measure, the following the PBM index is proposed:

$$\mathrm{PBM} = \left(\frac{D_B}{C} \cdot \frac{\sum\limits_{i=1}^{C} \sum\limits_{j=1}^{N_i} |x_{i,j} - \bar{x}|}{\sum\limits_{i=1}^{C} \sum\limits_{j=1}^{N_i} |x_{i,j} - \bar{x}_i|} \right)^2 \tag{1.45}$$

where $D_B = \max_{1 \le i < j \le C} \left| \bar{x}_i - \bar{x}_j \right|$ is the largest distance between mean values in the set of all clusters (centers of clusters). As in the case of the ANOVA F-test, the greater the dispersion between clusters and the more compact each cluster is, the greater the PBM index is. In conclusion, the greater the PBM index is, the better class separation the feature provides and the better the feature quality is.

1.4.5 Selection between Index-Based Methods and Wrapper-Based Methods

Let us recall that classifiers can serve as a feature evaluation method—following the so-called wrapper-based approach to feature selection. Applying classifiers to feature evaluation is in primal conflict with index-based feature search as it drastically increases computational complexity. Wrappers evaluate the *final product* (classification accuracy), while index-based methods are more sublime: they evaluate *constituents*, features making the final model.

At the same time, we have to face a realistic expectation that a computationally inexpensive index-based method would be only valuable if its results do not fall far behind a superior model constructed at a higher computational cost.

In the next section, we aim at a fair comparison of index-based methods for single feature evaluation with classifier-based single feature evaluation. By analogy, in later parts of this chapter, we aim at comparing index-based methods for multiple features selection with classifier-based multiple features selection. In experimental tests we analyze the consistency of a given index-based evaluation method with the accuracy of a trained model.

In this study we use k-NN and SVM classifiers. An extended description of these classifiers is provided in Chapter 2.

1.4.6 Single Feature Evaluation Scheme Using Indices and Classifiers

There are many indices used for evaluating clustering quality. We investigated more than 20 indices in order to choose a few, which would evaluate features consistently with classifiers; that is, we look for indices that rank features in a way similar to classifiers. Moreover, we discuss selected features in order to illustrate some properties such as the correlation of features or the scoring results. Outlined in Table 1.3 are results produced by classifiers SVM and k-NN ($k = 1$) and the ANOVA F-test and several clustering indices: the Calinski–Harabasz index, the Baker–Hubert gamma index, the G+ index, the GDI with parameters δ_4 and Δ_1 (cf. Dunn, 1973; Desgraupes, 2013), the MCR index, the PBM index, and the point-biserial index (cf. Desgraupes, 2013, 2016). For each index listed in the first column of Table 1.3, given are 15 features (their numbers) with the best rank, in descending order of this rank. The full names of these features and assigned numbers are given in Appendix 1.A. In Table 1.3, we would like to draw attention to rows corresponding to the ANOVA F-test and the Calinski–Harabasz, which are identical, which means that these two indices are fully

TABLE 1.3 Features ranking with two classifiers and different indices applied: classifiers SVM and *k*-NN ($k = 1$), ANOVA index, and several clustering indices

SVM	154	120	152	89	116	3	134	67	91	106	130	148	61	150	20
k-NN ($k = 1$)	154	134	120	152	116	81	106	130	95	156	70	89	97	92	148
ANOVA	154	3	145	61	8	67	80	64	10	143	17	32	7	22	69
Calinski–Harabasz	154	3	145	61	8	67	80	64	10	143	17	32	7	22	69
Gamma	12	40	126	34	140	26	87	101	52	98	112	46	59	41	115
G+	12	40	126	34	140	26	87	101	52	98	112	46	59	41	115
GDI-41	6	149	95	144	57	96	146	154	61	72	25	33	45	65	116
McClain–Rao	154	32	80	3	145	44	76	17	143	67	61	69	116	8	78
PBM	154	3	61	80	67	76	145	69	156	64	17	8	32	135	152
Point-biserial	40	12	34	126	26	140	87	101	59	41	98	52	115	112	129

This rank was performed for 159 features out of 171 ones in total (prior to the evaluation we removed 12 features that were constant), cf. Appendix 1.A.

correlated (perfectly consistent) and one should be dropped. Further properties of indices are outlined later on in this chapter.

Feature ranks are investigated to evaluate the usefulness of indices. We applied two schemes to compute the score for pairs of indices/classifiers, both based on computing a *distance* between positions of features. Then we look for indices consistent with SVM and *k*-NN classifiers.

The Distance by Rank (*DR*)

The first criterion used for measuring similarity is the *distance by rank* (*DR*) *score*. This score computes the sum of differences between positions of features in ranks. Let us assume that R_1 and R_2 are ranks based on two indices and that $R_1(X_i)$ and $R_2(X_i)$ are indices of the feature X_i in these ranks. This score is defined as follows:

$$DR(R_1, R_2) = \sum_{i=1}^{M} |R_1(X_i) - R_2(X_i)| \tag{1.46}$$

The results of the DR are shown in Table 1.4. In this table, we display the score for these indices that were used in Table 1.3. The smaller the value of this score is, the more similar indices are. It is equal to 0 for the two perfectly consistent indices, that is, such indices that assign the same rank to each feature. This is the aforementioned case of the ANOVA F-test and the Calinski–Harabasz index and the pair Gamma and G(amma)+ indices. Apart from these two pairs, the highest similarity is encountered for the pair of two classifiers: SVM and *k*-NN. With regard to similarity of a classifier and an index, relatively low scores, less than 6000 for the SVM classifier and less than 7000 for the *k*-NN classifier, are encountered for the pair of perfectly consistent ANOVA F-measure and Calinski–Harabasz index and then for the GDI-41, the MCR, and the PBM indices. The point-biserial index is an example of an index that exhibits high DR score values. This index (among a few others) is very inconsistent with other indices.

The Distance by Segments Cardinality (*DSC*)

The second criterion used for measuring similarity is called the *distance by segments cardinality* (*DSC*). This score is based on consecutive groups of top features in ranks created by two compared indices. Then, cardinalities of an intersection of corresponding sets of features are summed. The following formula defines the DSC:

$$\mathrm{DSC}(R_1, R_2) = \sum_{i=1}^{\lfloor M/r \rfloor} \mathrm{card}\left(\bigcup_{j=1}^{r \cdot i} \left\{ R_1^{-1}(j) \right\} \cap \bigcup_{j=1}^{r \cdot i} \left\{ R_2^{-1}(j) \right\} \right) \tag{1.47}$$

where R_1 and R_2 are ranks based on two indices, R_1^{-1} and R_2^{-1} are inverse mappings of ranks, and r is the segment length parameter. Explicitly, $R_1^{-1}(j)$ is the feature X_i if and only if $R_1(X_i) = j$, and $\mathrm{card}\left(\bigcup_{j=1}^{r \cdot i} \left\{ R_1^{-1}(j) \right\} \cap \bigcup_{j=1}^{r \cdot i} \left\{ R_2^{-1}(j) \right\} \right)$ is the number of common features in initial segments of length $r \cdot i$ in both ranks. Hence, this criterion operates on initial segments of both ranks, namely, the segments of length $r, 2r, 3r, \ldots, \lfloor M/r \rfloor \cdot r$. From (1.47) we can easily conclude that the first segments

TABLE 1.4 Features ranking with *distance by rank*: compared are all pairs of classifiers and indices, the scores less than 6000 for the SVM index and less than 7000 for the *k*-NN index are bolded

	SVM	*k*-NN (*k* = 1)	ANOVA F-meas.	Calinski–Harabasz	Gamma	Gamma+	GDI-41	McClain–Rao	PBM	Point biserial
SVM	0	**2783**	**5,283**	**5,283**	8862	8862	**5214**	**4,723**	**5408**	10,560
k-NN (*k* = 1)	**2,783**	0	**6,550**	**6,550**	9437	9437	**5341**	**6,042**	**6749**	9,727
ANOVA F-meas.	**5,283**	**6550**	0	0	5913	5913	6135	1,638	1923	11,551
Calinski–Harabasz	**5,283**	**6550**	0	0	5913	5913	6135	1,638	1923	11,551
Gamma	8,862	9437	5,913	5,913	0	0	9698	5,327	5418	8,888
Gamma+	8,862	9437	5,913	5,913	0	0	9698	5,327	5418	8,888
GDI-41	**5,214**	**5341**	6,135	6,135	9698	9698	0	6,607	6792	9,340
McClain–Rao	**4,723**	**6042**	1,638	1,638	5327	5327	6607	0	2199	11,721
PBM	**5,408**	**6749**	1,923	1,923	5418	5418	6792	2,199	0	11,274
Point biserial	10,560	9727	11,551	11,551	8888	8888	9340	11,721	11274	0

Note that the SVM-*k*-NN score is also low. The score was computed according to (1.46) for M=159 features.

(of length r) are counted $\lfloor M/r \rfloor$ times, the second segments (of length i) are counted $\lfloor M/r \rfloor - 1$, and so on. In this way, DSC score gives priority to shorter segments. Finally, the higher the score for a pair of indices is, the better the consistency of indices is.

The results of the DSC applied to selected indices are outlined in Table 1.5. The SVM classifier is compared with the k-NN classifier and with selected indices. The parameter r is set to 10, so then segments of length tens are considered, that is, in the first row, we have cardinalities of intersection of top 10 features in the SVM rank and in ranks of k-NN and consecutive indices. In the second row, we have cardinalities of top 20 features in the SVM rank, in the ranks of k-NN and consecutive indices, and so on. Specifically, top 10 features in SVM and k-NN ranks have 6 common features, top 10 features in SVM and ANOVA ranks have 3 common features, and so on. Let us notice that two pairs of indices (the pair ANOVA–Calinski–Harabasz and the pair Gamma – Gamma+) give the same results due to their perfect consistency being observed earlier.

The final comparison to the SVM score is shown in Table 1.5 in the row labeled "SVM DSC". In the row labeled "k-NN DSC", we give the final comparison to the k-NN ($k = 1$) score, detailed data are dropped. Notice that the value in the column labeled "k-NN" concerns a comparison for the k-NN rank with itself, and hence it is maximal. In both rows with the final score, we highlight with the bold font scores greater than an (arbitrarily) set threshold 800.

TABLE 1.5 Features ranking with *distance by segments cardinality*: compared are initial segments of ranks created by classifiers and an index

Segment—Index	k-NN	ANOVA	C-H	G	G+	GDI	M-R	PBM	P-B
SVM—[1–10]	6	3	3	0	0	1	3	3	0
SVM—[1–20]	11	8	8	0	0	4	7	6	0
SVM—[1–30]	15	13	13	3	3	8	15	14	0
SVM—[1–40]	24	17	17	6	6	12	20	18	0
SVM—[1–50]	34	24	24	10	10	19	27	26	2
SVM—[1–60]	44	33	33	17	17	30	37	36	3
SVM—[1–70]	53	41	41	27	27	41	46	44	7
SVM—[1–80]	65	51	51	37	37	51	54	53	22
SVM—[1–90]	77	63	63	49	49	64	65	61	37
SVM—[1–100]	88	77	77	59	59	81	79	75	52
SVM—[1–110]	102	89	89	74	74	95	92	87	65
SVM—[1–120]	114	105	105	90	90	111	105	100	83
SVM—[1–130]	126	120	120	106	106	126	122	117	102
SVM—[1–140]	138	137	137	121	121	135	136	133	121
SVM—[1–150]	149	149	149	141	141	143	150	150	141
SVM DSC	**1046**	**930**	**930**	740	740	**921**	**958**	**923**	635
k-NN DSC	**1200**	**860**	**860**	715	715	**917**	**886**	**849**	687

SVM DSC denotes scores of distance by segments cardinality for the SVM classifier and given indices. k-NN DSC denotes scores for k-NN ($k = 1$) and given indices. Indices are taken from Table 1.4, and their abbreviations are in the first row. The bold font is used to highlight relatively high scores.

Finally, the analysis of the DR and the DSC scores allows to recommend five indices for dealing with features selection: the ANOVA F-measure, the Calinski–Harabasz, GDI-41, the MCR, and the PBM indices. Notice that both the DR and the DSC scores recommend the same indices to be consistent with classifiers. Since the ANOVA F-measure and the Calinski–Harabasz index are perfectly consistent, we decided to drop the Calinski–Harabasz index and leave four others.

1.4.7 Selection of Subsets of Features

Needless to say, a successful classification process in most cases requires many features, not only a single one. Therefore, we need to select a number of them from a wider spectrum. One may be tempted to expect that selecting a number of features with the highest individual evaluation will guarantee the best choice. Unfortunately, evaluation of single features is not the best indicator of the predictive power of a model based on more than one feature. The aforementioned feature interactions influence a design of the classifier.

Let us reiterate that a subset of features, say, k features, with the best individual evaluation does not guarantee the best evaluation within other subsets with the same cardinality. Namely, it is necessary to test all subsets including k features out of M in order to have the best set of k features. Since the number of all k subsets of cardinality k out of M features, given in (1.48), is factorial, roughly estimating the complexity of such a method is exponential, so then such method is useless in practice for problems of higher dimensionality:

$$\binom{M}{k} = \frac{M!}{(M-k)!k!} \tag{1.48}$$

In the next section, we discuss some approximation methods that can be used in selecting the best subset of features out of a wider range of them. Such approximation methods, in fact, do not guarantee the best set to be identified, but we can expect that a set close to the optimal one could be selected.

Index-Based Feature Selection Methods

The most straightforward method for feature selection, conceptually speaking, is to generate subsets of features based on a full set of features, evaluate all generated subsets using some index, and select a subset with the highest score. As a quality evaluation index, we can use any of the clustering indices discussed before. Naturally, they are fit to evaluate not only a single feature but also a set of features. This straightforward procedure is outlined in Algorithm 1.4.

Algorithm 1.4

Index-based evaluation of a given set of features

Data:	Set of Features for Evaluation
	Learning Set of Patterns
	Index for Feature Set Evaluation
Algorithm:	**evaluate** the Set of Features **using** Index for Feature Set Evaluation
Results:	quality score for the given Set of Features

Wrapper-Based Feature Selection Methods

Wrapper-based methods rely on comparing the quality of multiple classifiers built on different sets of features. In the following text we formulate two algorithms to be used in the wrapper-based evaluation of sets of features. These algorithms are executed multiple times to form a wide collection of models, out of which the best one is selected.

Algorithm 1.5

Classifier-based evaluation of a given set of features

Data: Set of Features
Learning Set of Patterns
Classification Method
Classifier Evaluation Method

Algorithm: split Learning Set of Patterns into Training and Test Sets
build classifier for given
Classification Method **and** Training Set
for Training and Test Sets **do**
evaluate constructed **classifier**
using given Classifier Evaluation Method
use evaluation of **classifier** as evaluation of features

Results: quality score of the given Set of Features

This algorithm works as follows. First, we construct a classifier using a given classification method and the training set of patterns. Once the classifier has been built, the classifier evaluation method produces scores on the training set and the test set. Finally, both scores are involved in evaluation of the set of features.

This algorithm is usually extended with a cross-validation technique, which allows to average classifier construction. We rewrite Algorithm 1.5 with a small update in order to underline that the cross-validation concerns the training set and that the test set is used only for classifier evaluation. Again, we skip details of cross-validation techniques in order to keep clarity of narration. It is worth noting that Algorithm 1.5 is a special case of Algorithm 1.6 with the number of cross-validation folds equal to 1.

Algorithm 1.6

Classifier-based evaluation of a set of features with cross-validation

Data: Set of Features
Learning Set of Patterns
Classification Method
Classifier Evaluation Method
r—number of cross-validation folds

Algorithm: split Learning Set of Patterns into Training and Test Set
repeat *r* times
begin
use Training Set to build a fold
build a classifier **using** given Classification Method
and the current fold
end

build the final classifier **based on** obtained *r* results
evaluate constructed **classifier**
using given Classifier Evaluation Method
and Training and Test Sets
use evaluation of **classifier** as evaluation of features
Results: quality score of the given Set of Features

Algorithms 1.5 and 1.6 provide methods for evaluation of feature sets using a classifier of choice. Typically, they could be employed as a component of the wrapper-based feature selection method that generates sets of features and then employs evaluation algorithms for feature sets to select the best one. Of course, saying *the best set of features*, we mean that such a set of features gets the best evaluation on the given learning set of patterns. We also hope that this set of features will allow to construct such a classifier that will occur to be the best one among the others in future applications.

In the addressed schemes the classifier is built based on a training set (a subset of the learning set used for model construction). Next, we evaluate classifier performance on a test set. The test set, in other words, is a holdout set of patterns unseen at the stage of classifier construction. Finally, the scores obtained on train and test sets can be combined in any way to get an evaluation of the set of features. For the sake of clarity, we do not discuss details such as proportion between cardinalities of the training and test sets, quality evaluation method, relation between evaluation of the training set and the test set of patterns, and so on. Among early researches on wrapper-based feature selection, we may refer to Kohavi and John (1997).

1.4.8 Feature Subsets Generation

As mentioned before, building classifiers for multiple sets of features requires long computation time, and, of course, it is definitely useless for systematic searching in large spaces of features. Therefore, instead of classifiers, we discuss employment of indices to evaluate the quality of sets of features. We collate the results with *classical* wrapper-based methods relying on classifiers only. We place a strong emphasis on comparison between wrapper-based methods, and therefore this discussion is placed in the section devoted to wrapper-based feature search methods.

However, no matter if we use a classifier or an index, still one problem remains unsolved: how to generate subsets of features that need to be checked.

In subsequent sections let us discuss several methods that could be used in generating an optimal set of features. Saying *an optimal set of features*, we mean that performance of such set will be as good as possible. It is important to underline that such a set of features may not guarantee the best performance among other sets. This is for several reasons such as cost reduction of the features selection process. There is no one universal and objective method that could be used to find the best set of features.

Naïve (Brute Force) Selection

We begin this discussion with recalling the naïve (brute force) method. More specifically, we may use the naïve (brute force) selection, that is, investigation of all nonempty subsets of the set of features in order to select the best one. Of course, this

method, formulated in Algorithm 1.7, guarantees selection of the best set in terms of the learning set of patterns. But such a set of features may be less successful when it is used to process patterns not belonging to the learning set.

Algorithm 1.7

Naive (brute force) selection of the best subset of features

Data: Set of Features
Learning Set of Patterns
either Index for Feature Set evaluation
or Classification Method and Classifier Evaluation Method

Algorithm: **for** each nonempty subset of the Set of Features **do**
call Algorithm 4 or 5 or 6 to evaluate this subset of features
choose the subset with the best evaluation

Results: the subset of the Set of Features with the best evaluation

However, computational complexity of the naïve selection is exponential, and therefore it can be used only for a small set of features, that is, for a small M. Therefore, in practice, it is useless for more than 10 features. Saying it useless, we mean that although the running time of such a method is finite, it is so long that we will not get results in a reasonable time. In such cases, instead of the exact naïve method, different approximations are applied. In the following text we discuss four methods, three greedy ones and a method with a limited expansion.

Greedy Selection by Rank of Single Features

The first attempt to select an approximated best set of k features is just using the top k features from the rank provided by any quality measure of single features. We may use any measure to evaluate single features, for instance, those discussed in Section 1.4, that is, a classifier, the ANOVA F-test, the GDI-41, the MCR, and the PBM indices. Using single features ranking would be appropriate for a raw classification task, where the highest quality result is not the prime issue. Unfortunately, if high classification accuracy is a prime objective, this simple method is not sufficient. Therefore, we need to utilize more sophisticated selection in order to produce a set of features of better quality.

When we apply the single feature rank selection procedure and our objective is to choose k features; we select top k features. They were evaluated individually as the best, but we do not know how they *cooperate* with one another. In contrast, when we use greedy forward/backward selection (that will be described in the following two sections), we add/remove features one by one. We do not look at the quality of individual features. Instead, in each round of the selection procedure, we add/remove such one feature so that the new set of features becomes the best.

Greedy Forward Selection

We are looking for the approximately best set of features of given cardinality, say, k features out of all M ones. In this scheme, we start with an empty set, and then we keep adding one feature in each iteration. Assume that we have l features already selected. Then, for every feature f from the set of $M - l$ nonselected ones, evaluated is the set with feature f added, that is, the set of $l + 1$ features. After this, the set with the best evaluation is taken as the set of $l + 1$ selected features. Finally, the feature f included in

a newly created set is deleted from the set of nonselected ones. This method is formally described in Algorithm 1.8.

Algorithm 1.8

Greedy forward selection of the best subset of features

Data: SofF (Set of Features)
NofFtoS (Number of Features to Select)
Learning Set of Patterns
either Index for Feature Set Evaluation
or Classification Method and Classifier Evaluation Method

Algorithm:
initiate SofSF (Set of Selected Features) as the empty set
initiate SofRF (Set of Remaining Features) as SofF
while cardinality of SofSF is less than NofFtoS **do**
begin
  for every feature f **in** SofRF **do**
    evaluate the set SofSF $\cup\{f\}$ of features
      using Algorithm 1.4 or 1.5 or 1.6
  choose the feature $f_{\max}$ for which SofSF $\cup\{f\}$ gets the best evaluation
  add $f_{\max}$ to SofSF
  remove $f_{\max}$ from SofRF
end

Results: the subset of features of cardinality Number of Features to Select (NofFtoS)

Greedy forward selection is an iterative process running until the set of selected features reaches the required cardinality. In practice, this condition may be replaced with another one based on an intuitive expectation that the quality of the classification model will be increasing along with growing cardinality of the set of selected features. Based on this assumption, the stopping criterion would involve a very intuitive condition based on the quality of the classification model evaluated on the learning set, that is, the quality measured on the training set or on the testing set or on a combination of both measures. Once the required quality has been reached, the iterative process can be stopped.

Greedy Backward Selection

Algorithm 1.9 formulates greedy backward selection. Instead of starting from an empty set and then incrementally adding features, this method starts with a full set of features and iteratively reduces its size. At the beginning, we give the full set of M features. Then, in each turn, we evaluate all subsets obtained by removing one feature from the current set. After that, the current set of features is replaced with the subset with the best evaluation. This process is repeated until we skim the set of features to a desired size or until another stopping condition is satisfied. After each iteration we inspect the quality of features. When the quality starts to decrease substantially, the process should be halted without waiting for other stopping criteria to be met. Greedy backward selection can be used in a case when the initial full set of features is not very large. When we want to select a small subset of features out of a large set of all features, forward selection is less run time consuming than backward selection.

It will be shown later on that, in general, reducing a big set of features increases quality evaluation on the test set. This is a consequence of the phenomenon of overfitting. Therefore, the halting condition based on quality evaluation would be carefully defined, when applied to big sets of features.

Algorithm 1.9

Greedy backward selection of the best subset of features

Data: SofF (Set of Features)
NofFtoS (Number of Features to Select)
Learning Set of Patterns
either Feature Evaluation Method
or Classification Method with Classifier Evaluation Method

Algorithm: **initiate** SofSF (Set of Selected Features) as SofF (Set of Features)
while cardinality of SofSF is less than NofFtoS **do**
begin
  for every feature f **in** SofSF **do**
    evaluate the set SofSF $-\{f\}$ of features
      using Algorithm 1.4 or 1.5 or 1.6
  choose the feature $f_{\max}$ for which SofSF $-\{f\}$ gets the best evaluation
  remove $f_{\max}$ from SofSF
end

Results: the subset of features of cardinality Number of Features to Select (NofFtoS)

Greedy Forward Selection with Limited Expansion

Greedy forward selection with limited expansion extends simple forward selection. In each iteration of this algorithm, l the best sets of k features are processed, where l is the expansion width. Each such set is incremented with each available feature and then the l best sets are selected. Specifically, for each set of cardinality k, out of the l best ones, $M-k$ sets are created by inserting one not selected feature. In total, $l*(M-k)$ new sets of cardinality $k+1$ are obtained. Then, after deleting duplicated sets, new k best sets are selected. The details are presented in Algorithm 1.10.

Algorithm 1.10

Greedy forward selection with limited expansion

Data: SofF (Set of Features)
NofFtoS (Number of Features to Select)
l—Width of Limited Expansion
Stop Condition
Learning Set of Patterns
either Feature Evaluation Method
or Classification Method with Classifier Evaluation Method

Algorithm: **for** i = 1 to l **do**
  Initialize SofSF$_i$ (i-th Set of Selected Features) **as** empty set
while Stop Condition is not satisfied **do**
begin

initialize PLofSofF (Pending List of Sets of Features) **as** empty list
for i = 1 to l **do**
begin
initialize SofRF (Set of Remaining Features) **as** SofF – SofSF_i
for each feature f **in** SofRF **do**
add $\text{SofSF}_i \cup \{f\}$ to the PLofSofF
end
delete duplicates from the PLofSofF
evaluate sets of features from the PLofSofF
using Algorithm 1.4 or 1.5 or 1.6
for i = 1 to l **do**
replace SofSF_i (i-th Set of Selected Features) **by**
i-th top set from PLofSofF
end

Results: the l best subset of selected features

Notice that for the parameter $l = 1$, the greedy forward search with limited expansion turns into a simple greedy forward search. On the other hand, when at each turn the parameter l is equal to $M - k$—that is, it is equal to the number of features not used yet in the current set of selected features—this algorithm turns into the naïve selection.

Notes on Computational Complexity

As mentioned earlier, computational complexity of the presented search algorithms significantly differs, which determines their usefulness in practical applications. Let us take a closer look at this problem. Evaluation of created subsets of features is the dominant operation in methods formulated in Algorithms 1.7–1.10. Therefore, run time of such an algorithm is proportional to the number of executions of this operation. Evaluation of a set of features is performed by Algorithm 1.4 (index-based) or Algorithm 1.5 (classifier, no cross-validation) or Algorithm 1.6 (classifier with cross-validation). Execution of Algorithm 1.6 is, roughly speaking, about r times longer than execution of Algorithm 1.5, where r is the number of cross-validation folds.

Having this in mind, let us estimate the number of executions of different subset search algorithms that in fact is equivalent to asymptotical complexity:

- Greedy selection by rank of single features requires each feature to be evaluated once. Hence complexity is of rank M, that is, $O(M)$.
- Naïve (brute force) selection requires evaluation of each subset of the set of features, which makes complexity to be exponential, that is, $O(2^M)$.
- In greedy forward selection and greedy backward selection in each turn consecutively $M, M-1, M-2, M-3, \ldots$, sets of features are evaluated, which raises (pessimistic) square complexity: $O(M^2)$.
- Greedy forward selection with limited expansion requires evaluation of k times more sets than simple greedy forward selection for constant k and $k << M$, so then the rank of complexity is $O(kM^2)$, where k is width of limited expansion.

Therefore, from the perspective of complexity, any of the previously mentioned methods is a reasonable choice except the naïve method, which may be applied only for small sets of features.

Illustrative Experiment

In this experiment, Algorithm 1.10 was employed to test methods for features selection. We tested the run time for different evaluation methods and then the quality of a classifier built on selected features. The experiment was carried out on the data set of handwritten digits (LeCun *et al*., 1998).

Besides the previously mentioned asymptotical complexity estimation, in practice important are run time of a dominant operation and proportion between the run time of the dominant and all operations. In our case, the evaluation of a set of features is the dominant operation. We used two types of evaluators: classifiers and clustering indices. In Table 1.6, the run times of Algorithm 1.10 are given. This algorithm was executed for different classification models and evaluation methods for the purpose of the characteristics outlined in Figures 1.10 and 1.11, that is, in each case (evaluation method) selected were sets of features of consecutive cardinality 1, 2, 3, 4, …, 100. In the case of the k-NN method, we took 1 neighbor and there was no cross-validation involved. In the case of the SVM classifier, the following parameters were set: Gaussian kernel function, $\gamma = 0.0625$ and $C = 1$, with 10-fold cross-validation, cf. Section 2.3.

In practice, the final set of features is of cardinality 20–40, much less than the set of 100 features, considered according to the prepared characteristics given in Figures 1.10 and 1.11. Therefore, the computation time of forward selection methods should be divided by a proper factor, say, 5 for the SVM selector and 2.5 for other selectors. Anyway, it is worth drawing attention to the fact that the range of time is huge even if these factors are taken into account. These ranges begin with one/a few hours (ANOVA F-test), 15 or so hours (k-NN and indices), and up to 15 or so days (SVM). Hence, we can conclude that performance time is an important factor that should be taken into consideration. Of course, the choice of an evaluation method determines not only its run time but also the quality of the final classifier constructed with selected features.

In Figure 1.9 we present values of indices for the best sets of features for three values of expansion limit and for cardinality of features sets ranging from 1 to 100. Results concern a dataset of handwritten digits. Displayed are values of ANOVA F-test, PBM index, GDI-41 index and accuracy values of SVM classifier built at

TABLE 1.6 Performance time of Algorithm 1.10 on the MNIST dataset (LeCun *et al.*, 1998) for three clustering indices and two classifiers

	ANOVA	MCR	PBM	k-NN	SVM
$k = 1$	1.5	15	17	7	135
$k = 3$	4.5	47	50	20	400
$k = 5$	7.5	70	80	33	650

Given is performance time in hours needed to compute data for characteristics outlined in Figures 1.10 and 1.11.

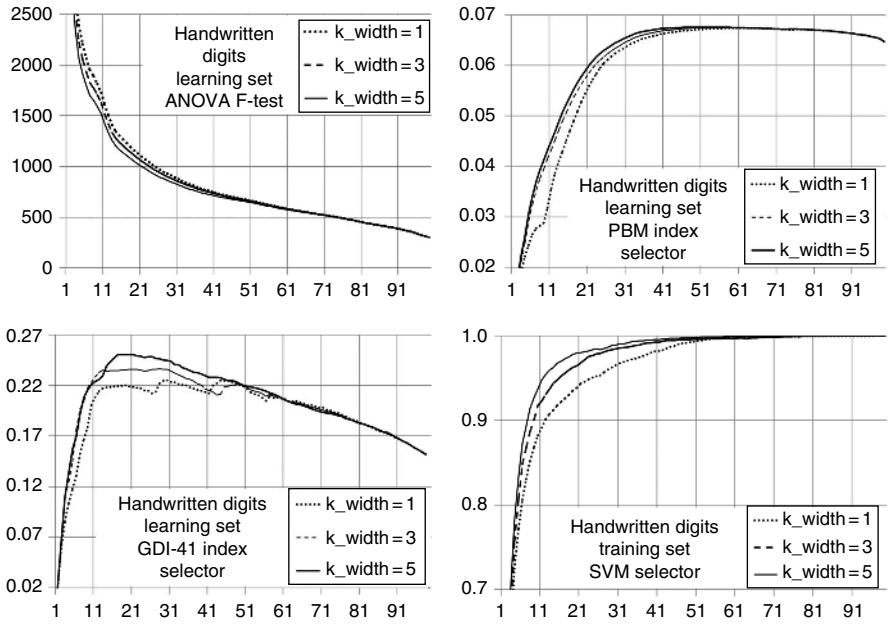

Figure 1.9 Evaluation of sets of features with ANOVA F-test, PBM index, GDI-41 index, and SVM classifier. Sets of features were selected with greedy forward with expansion limited to 1, 3, and 5. Displayed are scores of indices at the whole learning set of patterns and SVM classifier accuracy at the training set of patterns (vertical axis) as a function of cardinality of features sets ranging from 1 to 100 (horizontal axis). Results concern a dataset of handwritten digits.

the best sets of features selected with three values of expansion limit (1, 3, and 5) (cf. Algorithm 1.10).

Let us recall that the higher the value of indices/accuracy, the better the quality of the set of features is. In cases of PBM and GDI-41 Indices and SVM classifiers, quality is growing along with increasing expansion limit and cardinality of best sets of features and, amazingly, ANOVA F-test behaves in the opposite way: quality falls at the same time. In all four cases, around interval [20,30], there is an inflection region (ANOVA F-test, PBM index, and SVM) and maximum region (GDI-41 index).

In Figures 1.10 and 1.11, the quality of classifiers constructed on the basis of features sets selected with various methods is displayed. For each feature set an SVM classifier with parameters set to Gaussian kernel function, $\gamma = 0.0625$ and $C = 1$, with 10-fold cross-validation, was constructed. Results at the training set and at the test set are given for selected indices and for the SVM classifier used for feature sets evaluation.

Note that SVM parameters were the same for each tested set of features, that is, no parameters tuning was done. Parameters tuning is performed empirically. This in fact means that additional repetitions of model construction procedures would be

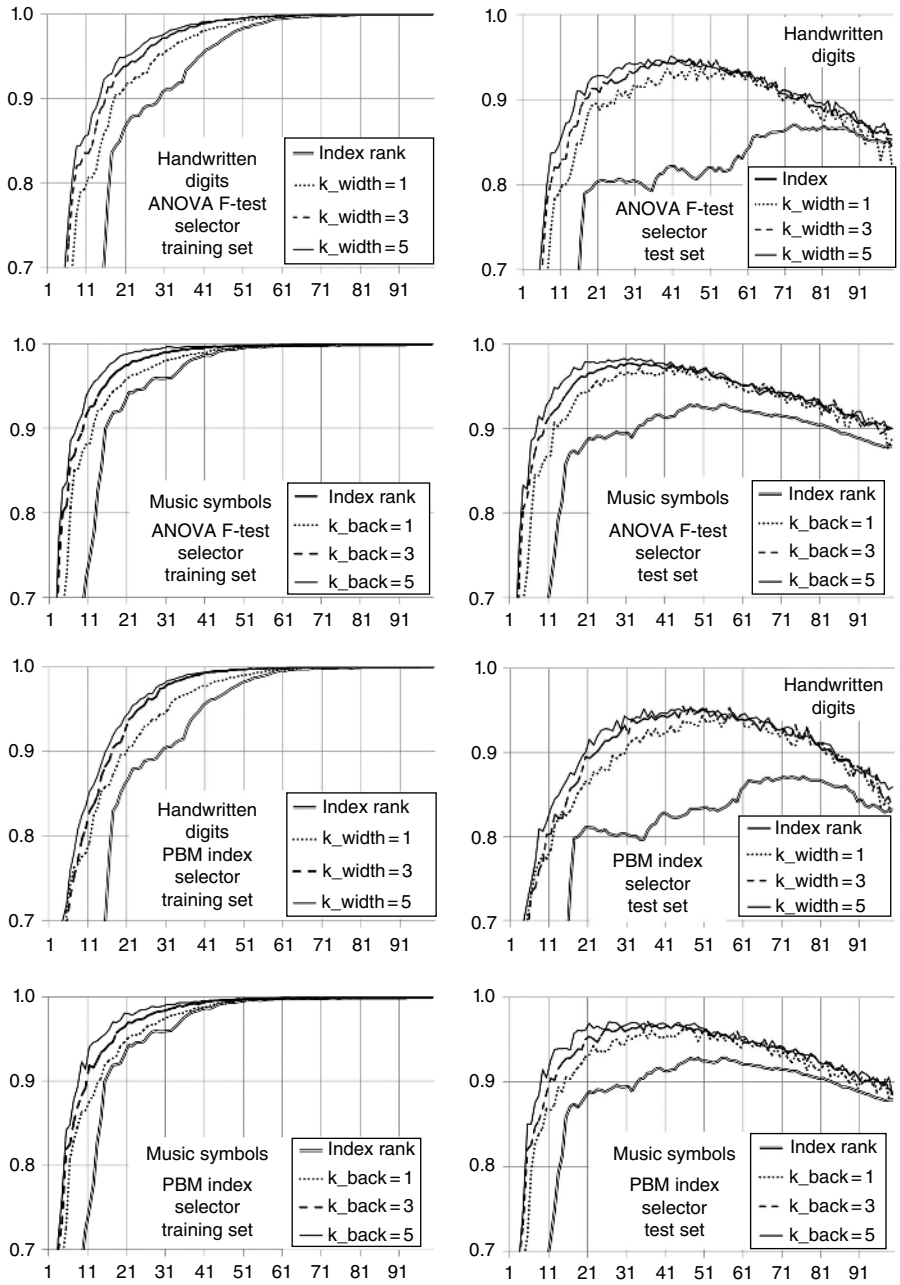

Figure 1.10 Quality of classification using SVM classifier constructed based on selected sets of features. Sets of features were selected using the ANOVA F-test and PBM index. Classification accuracy (vertical axis) is measured at the training set and at the test set for three values of expansion limit and for the best features in the individual index rank, cardinality of features sets ranging from 1 to 100 (horizontal axis). The first two rows of graphs concern F-ANOVA; the last two rows—PBM. Results concern handwritten digits and musical symbols datasets.

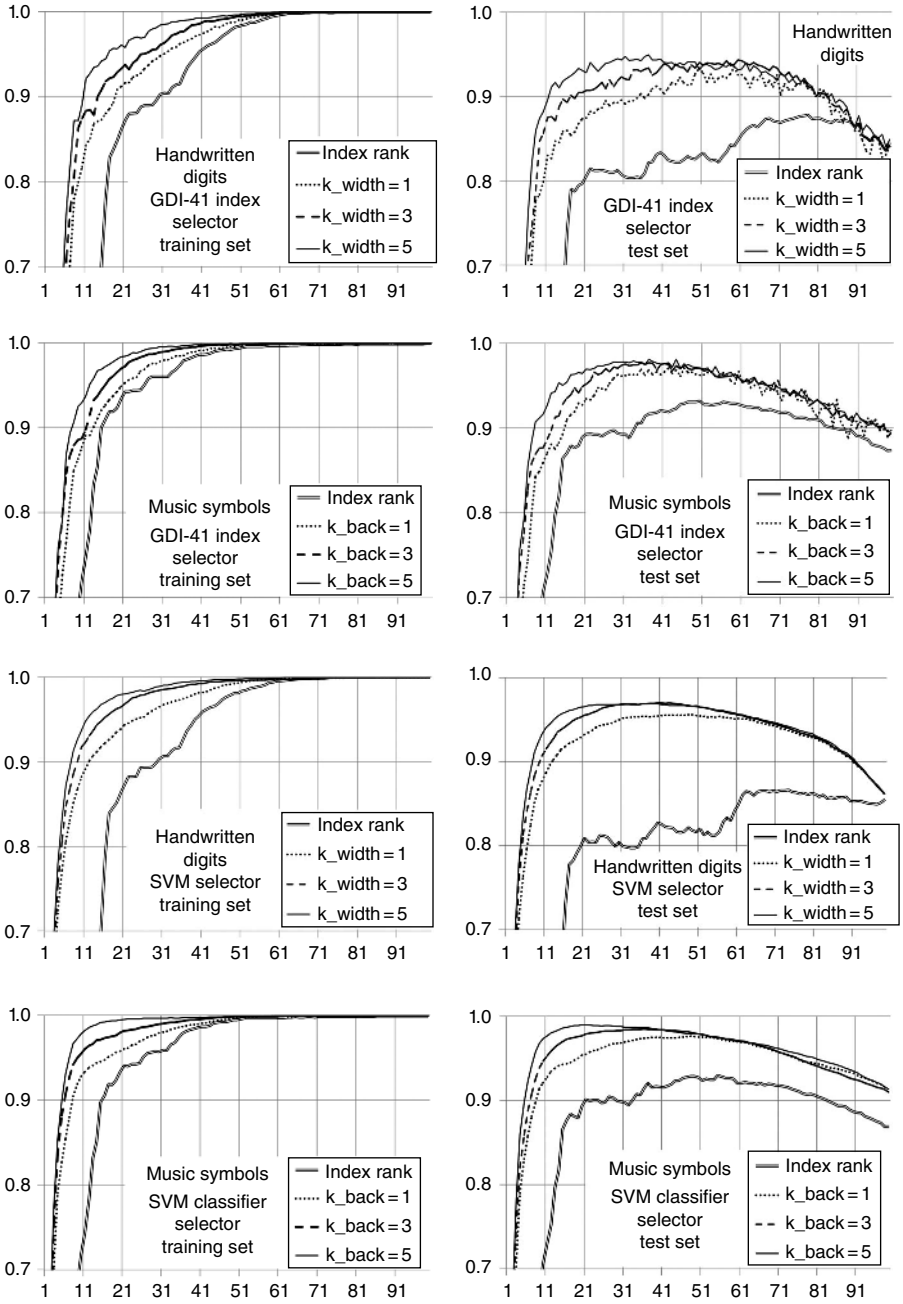

Figure 1.11 Quality of classification using the SVM classifier constructed based on selected sets of features. Sets of features were selected using the GDI-41 index and SVM classifier. Classification accuracy (vertical axis) is measured at the training set and at the test set for three values of expansion limit and for the best features in the individual index rank, cardinality of features sets ranging from 1 to 100 (horizontal axis). The first two rows of graphs concern F-ANOVA; the last two rows, PBM. Results concern handwritten digits and musical symbols datasets.

needed to select optimal parameters. This additional effort did not seem to be reasonable when the focus of this section was not on SVM itself, but on feature selection. Of course, tuned parameters may increase the quality of the final classifier, and it is recommended to perform parameters optimization for final model construction.

It is clearly visible that the SVM classifier based on features selected with the SVM classifier outperforms classifiers constructed based on indices. First, the maximal accuracy for the SVM selector overheads the other selectors by a few percent points. The second highlight is that the maximal accuracy is gained for smaller sets of features than in other cases. Third, characteristics for the SVM selector are more regular and smoother than for the other selectors, especially at the test set. However, time needed to execute a feature search with SVM as an evaluation method is huge. Observing characteristics displayed in Figures 1.10 and 1.11, we compare accuracy at the training set ranging from 0.9 to 0.98. What we can see is that cardinalities of optimal feature sets computed with the PBM and the ANOVA F-test are about 1.5 times greater than the cardinality of a set extracted with the SVM selector.

Also, the PBM and the ANOVA F-test-based models achieve maximal accuracy at the test set for feature sets that are twice as large as for the SVM selector. Having this in mind and referring to Table 1.6, we see that the run time of a procedure utilizing McClain and PMB is 10 times greater than a procedure based on the ANOVA F-test selector. Moreover, the run time of a procedure utilizing SVM is 50 (!) times greater than that of a procedure based on the ANOVA F-test selector. Therefore, we can conclude that the choice of features selector is a matter of a trade-off between model quality and its construction cost.

As an outcome of the experiments concerning greedy feature search, we have discovered suitable sets of features describing the datasets of handwritten digits and symbols of printed music notation. For the handwritten digits, we selected a set made up of 24 features, whereas for the musical symbols, they are described by 20 features. All these features are listed in Appendix 1.B. The order in which the features appear is according to the ranking realized by the SVM classifier used for evaluation of features.

1.5 CONCLUSIONS

It is needless to say that selecting proper features is an elementary and necessary assignment in any pattern recognition task. It determines the quality of models produced at later stages. In visual pattern recognition, feature selection is heavily determined by the kind of processed data. Here, we focus on characters and presented methodology that fits well in this category of problem. However, the discussed approaches may not be the best if we would wish to process substantially different data coming from a different domain, for instance, high-resolution colored scans obtained by high-end medical equipment. As it is for many other technical aspects of pattern recognition, the choice of methods is usually data driven.

APPENDIX 1.A

The list of 159 features used in experiments with handwritten digits and symbols of music notation recognition. Constant features were removed from the full list of 171 features

1 Projection V—Raw—Min—Value
2 Projection V—Raw—Min—Position
3 Projection V—Raw—Max—Value
4 Projection V—Raw—Max—Position
5 Projection V—Raw—Mean
6 Projection V—Raw—First moment
7 Projection V—Raw—Peaks count
8 Projection V—Differential—Min—Value
9 Projection V—Differential—Min—Position
10 Projection V—Differential—Max—Value
11 Projection V—Differential—Max—Position
12 Projection V—Differential—Mean
13 Projection V—Differential—First moment
14 Projection V—Differential—Peaks count
15 Projection H—Raw—Min—Value
16 Projection H—Raw—Min—Position
17 Projection H—Raw—Max—Value
18 Projection H—Raw—Max—Position
19 Projection H—Raw—Mean
20 Projection H—Raw—First moment
21 Projection H—Raw—Peaks count
22 Projection H—Differential—Min—Value
23 Projection H—Differential—Min—Position
24 Projection H—Differential—Max—Value
25 Projection H—Differential—Max—Position
26 Projection H—Differential—Mean
27 Projection H—Differential—First moment
28 Projection H—Differential—Peaks count
29 Histogram V—Raw—Min—Position
30 Histogram V—Raw—Max—Value
31 Histogram V—Raw—Max—Position
32 Histogram V—Raw—Mean
33 Histogram V—Raw—First moment
34 Histogram V—Raw—Peaks count
35 Histogram V—Differential—Min—Value
36 Histogram V—Differential—Min—Position
37 Histogram V—Differential—Max—Value
38 Histogram V—Differential—Max—Position
39 Histogram V—Differential—First moment
40 Histogram V—Differential—Peaks count
41 Histogram H—Raw—Min—Position
42 Histogram H—Raw—Max—Value
43 Histogram H—Raw—Max—Position
44 Histogram H—Raw—Mean
45 Histogram H—Raw—First moment
46 Histogram H—Raw—Peaks count
47 Histogram H—Differential—Min—Value
48 Histogram H—Differential—Min—Position
49 Histogram H—Differential—Max—Value
50 Histogram H—Differential—Max—Position
51 Histogram H—Differential—First moment
52 Histogram H—Differential—Peaks count
53 Cumulative Histogram V—Raw—Min—Value
54 Cumulative Histogram V—Raw—Max—Value
55 Cumulative Histogram V—Raw——Max—Position
56 Cumulative Histogram V—Raw—Mean
57 Cumulative Histogram V—Raw—First moment
58 Cumulative Histogram V—Raw—Peaks count
59 Cumulative Histogram H—Raw—Min—Value
60 Cumulative Histogram H—Raw—Max—Value
61 Cumulative Histogram H—Raw—Max—Position
62 Cumulative Histogram H—Raw—Mean
63 Cumulative Histogram H—Raw—First moment
64 Cumulative Histogram H—Raw—Peaks count
65 Transitions V—Raw—Min—Value
66 Transitions V—Raw—Min—Position
67 Transitions V—Raw—Max—Value
68 Transitions V—Raw—Max—Position
69 Transitions V—Raw—Mean
70 Transitions V—Raw—First moment

71 Transitions V—Differential—Min—Value
72 Transitions V—Differential—Min—Position
73 Transitions V—Differential—Max—Value
74 Transitions V—Differential—Max—Position
75 Transitions V—Differential—First moment
76 Transitions H—Raw—Min—Value
77 Transitions H—Raw—Min—Position
78 Transitions H—Raw—Max—Value
79 Transitions H—Raw—Max—Position
80 Transitions H—Raw—Mean
81 Transitions H—Raw—First moment
82 Transitions H—Differential—Min—Value
83 Transitions H—Differential—Min—Position
84 Transitions H—Differential—Max—Value
85 Transitions H—Differential—Max—Position
86 Transitions H—Differential—First moment
87 Offsets L—Raw—Min—Value
88 Offsets L—Raw—Min—Position
89 Offsets L—Raw—Max—Value
90 Offsets L—Raw—Max—Position
91 Offsets L—Raw—Mean
92 Offsets L—Raw—First moment
93 Offsets L—Raw—Peaks count
94 Offsets L—Differential—Min—Value
95 Offsets L—Differential—Min—Position
96 Offsets L—Differential—Max—Value
97 Offsets L—Differential—Max—Position
98 Offsets L—Differential—Mean
99 Offsets L—Differential—First moment
100 Offsets L—Differential—Peaks count
101 Offsets R—Raw—Min—Value
102 Offsets R—Raw—Min—Position
103 Offsets R—Raw—Max—Value
104 Offsets R—Raw—Max—Position
105 Offsets R—Raw—Mean
106 Offsets R—Raw—First moment
107 Offsets R—Raw—Peaks count
108 Offsets R—Differential—Min—Value
109 Offsets R—Differential—Min—Position
110 Offsets R—Differential—Max—Value
111 Offsets R—Differential—Max—Position
112 Offsets R—Differential—Mean
113 Offsets R—Differential—First moment
114 Offsets R—Differential—Peaks count
115 Offsets T—Raw—Min—Value
116 Offsets T—Raw—Min—Position
117 Offsets T—Raw—Max—Value
118 Offsets T—Raw—Max—Position
119 Offsets T—Raw—Mean
120 Offsets T—Raw—First moment
121 Offsets T—Raw—Peaks count
122 Offsets T—Differential—Min—Value
123 Offsets T—Differential—Min—Position
124 Offsets T—Differential—Max—Value
125 Offsets T—Differential—Max—Position
126 Offsets T—Differential—Mean
127 Offsets T—Differential—First moment
128 Offsets T—Differential—Peaks count
129 Offsets B—Raw—Min—Value
130 Offsets B—Raw—Min—Position
131 Offsets B—Raw—Max—Value
132 Offsets B—Raw—Max—Position
133 Offsets B—Raw—Mean
134 Offsets B—Raw—First moment
135 Offsets B—Raw—Peaks count
136 Offsets B—Differential—Min—Value
137 Offsets B—Differential—Min—Position
138 Offsets B—Differential—Max—Value
139 Offsets B—Differential—Max—Position
140 Offsets B—Differential—Mean
141 Offsets B—Differential—First moment
142 Offsets B—Differential—Peaks count
143 Directions—0
144 Directions—135
145 Directions—90
146 Directions—45
147 Directions—WE—Y
148 Directions—NS—X
149 Raw moments—First—m10
150 Raw moments—First—m01
151 Central moments—Second—m20
152 Central moments—Second—m11
153 Central moments—Second—m02
154 Height/width
155 Blackness level
156 Eccentricity
157 Euler number 4
158 Euler number 8
159 Euler number 6

APPENDIX 1.B

Lists of features selected for in experiments with handwritten digits recognition (left list) and symbols of music notation (right list)

1 Raw moments—First—m01
2 Central moments—Second—m02
3 Offsets L—Differential—Max—Position
4 Euler number 4
5 Offsets R—Raw—Min—Position
6 Offsets L—Differential—Min—Position
7 Directions—45
8 Offsets L—Differential—Max—Value
9 Offsets R—Differential—First moment
10 Offsets T—Differential—Max—Value
11 Raw Moments—first—m10
12 Height/width
13 Cumulative Histogram V—Raw—First moment
14 Offsets L—Differential—Min—Value
15 Offsets L—Differential—First moment
16 Central Moments—Second—m20
17 Offsets T—Raw—Min—Position
18 Projection V—Raw—Max—Value
19 Offsets R—Differential—Max—Value
20 Cumulative Histogram V—Raw—Max—Position
21 Projection H—Differential—Min —Value
22 Directions—90
23 Offsets B—Differential—Min—Position
24 Offsets L—Raw—Max—Value

1 Height/width
2 Offsets T—Raw—Min—Position
3 Central Moments—Second—m11
4 Projection V—Raw—Max—Value
5 Offsets R—Raw—First moment
6 Offsets R—Raw—Mean
7 Euler number 4
8 Central moments—Second—m02
9 Directions—135
10 Transitions H—Raw—Max—Value
11 Central moments—Second—m20
12 Offsets R—Raw—Min—Position
13 Directions—45
14 Offsets L—Raw—Max—Value
15 Projection V—Differential—Max—Value
16 Projection H—Differential—Peaks count
17 Raw moments—First—m01
18 Offsets L—Raw—Mean
19 Cumulative histogram V—Raw—first Moment
20 Cumulative histogram H—Raw—Peaks Count

REFERENCES

S. Bandyopadhyay, M. Pakhira, and U. Maulik, Validity index for crisp and fuzzy clusters, *Pattern Recognition* 37, 2004, 487–501.

H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, Speeded-up robust features (SURF), *Computer Vision and Image Understanding* 110(3), 2008, 346–359.

M. Charrad, NbClust: An R package for determining the relevant number of clusters in a data set, *Journal of Statistical Software* 61(6), 2014, 1–36.

B. Desgraupes, Clustering indices, *Report*, University Paris Ouest, Lab Modal'X, 2013.

B. Desgraupes, Package clusterCrit for R, *R Documentation*, 2016.

J. C. Dunn, A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters, *Journal of Cybernetics* 3(3), 1973, 32–57.

N. O. F. Elssied, O. Ibrahim, and A. H. Osman, A novel feature selection based on one-way ANOVA F-test for e-mail spam classification, *Research Journal of Applied Sciences* 7(3), 2014, 625–638.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, Gene selection for cancer classification using support vector machines, *Machine Learning* 46 (1–3), 2002, 389–422.

K. Haris, S. N. Efstratiadis, N. Maglaveras, and A. K. Katsaggelos, Hybrid image segmentation using watersheds and fast region merging, *IEEE Transactions on Image Processing* 7(12), 1998, 1684–1699.

W. Homenda, A. Jastrzebska, and W. Pedrycz, *The web page of the classification with rejection project*, 2017, http://classificationwithrejection.ibspan.waw.pl (accessed October 5, 2017).

W. Homenda, A. Jastrzebska, W. Pedrycz, and R. Piliszek, Classification with a limited space of features: Improving quality by rejecting misclassifications. In: *Proceedings of the 4th World Congress on Information and Communication Technologies* (*WICT 2014*), Malacca, Malaysia, December 8–11, 2014.

W. Homenda, M. Luckner, and W. Pedrycz, *Classification with rejection: concepts and formal evaluations, Knowledge, Information and Creativity Support Systems: Recent Trends*, Advances and Solutions, Advances in Intelligent Systems and Computing 364, Switzerland, Springer International Publishing, 2016, 413–425.

M. K. Hu, Visual pattern recognition by moment invariants, *IRE Transactions on Information Theory* IT-8, 1962, 179–187

C. L. Huang and C. J. Wang, A GA-based feature selection and parameters optimization for support vector machines, *Expert Systems with Applications* 31(2), 2006, 231–240.

R. Kohavi and G. H. John, Wrappers for feature subset selection, *Artificial Intelligence* 97(1–2), 1997, 273–324.

S. Krig, *Computer Vision Metrics. Survey, Taxonomy, and Analysis*, New York, Springer, 2014.

M. Kudo and J. Sklansky, Comparison of algorithms that select features for pattern classifiers, *Pattern Recognition* 33(1), 2000, 25–41.

Y. LeCun, C. Cortes, and C. J. C. Burges, *The MNIST database of handwritten digits*, 1998, http://yann.lecun.com/exdb/mnist/ (accessed October 5, 2017).

D. G. Lowe, Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* 60(2), 2004, 91–110.

B. S. Manjunath and W. Y. Ma, Texture features for browsing and retrieval of image data, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(8), 1996, 837–842.

J. O. McClain and V. R. Rao, CLUSTISZ: A program to test for the quality of clustering of a set of objects, *Journal of Marketing Research* 12(4), 1975, 456–460.

P. Mitra, C. A. Murthy, and S. K. Pal, Unsupervised feature selection using feature similarity, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(3), 2002, 301–312.

H. C. Peng, F. H. Long, and C. Ding, Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(8), 2005, 1226–1238.

Y. Saeys, I. Inza, and P. Larranaga, A review of feature selection techniques in bioinformatics, *Bioinformatics* 23(19), 2007, 2507–2517.

M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis and Machine Vision*, Pacific Grove, CA, PWS Publishing, 1998.

J. H. Sossa-Azuela, R. Santiago-Montero, M. Pérez-Cisneros, and E. Rubio-Espino, Computing the Euler number of a binary image based on a vertex codification, *Journal of Applied Research and Technology* 11(3), 2013, 360–370.

R. W. Swiniarski and A. Skowron, Rough set methods in feature selection and recognition, *Pattern Recognition Letters* 24(6), 2003, 833–849.

X. Tan and B. Triggs, Enhanced local texture feature sets for face recognition under difficult lighting conditions, *IEEE Transactions on Image Processing* 9(6), 2010, 1635–1650.

O. D. Trier, A. K. Jain, and T. Taxt, Feature extraction methods for character recognition – A survey, *Pattern Recognition* 29(4), 1996, 641–662.

M. Turk and A. Pentland, Eigenfaces for recognition, *Journal of Cognitive Neuroscience* 3(1), 1991, 71–86.