

# Introduction and Basic Concepts

# 1

The real power of human thinking is based on recognizing patterns.

---

Ray Kurzweil

## 1.1 Pattern Recognition

Pattern recognition describes the act of determining to which category, referred to as *class*, a given pattern belongs and taking an action according to the class of the recognized pattern. The notion of a *pattern* thereby describes an observation in the real world. Due to the fact that pattern recognition has been essential for our survival, evolution has led to highly sophisticated neural and cognitive systems in humans for solving pattern recognition tasks over tens of millions of years [1]. Summarizing, recognizing patterns is one of the most crucial capabilities of human beings.

Each individual is faced with a huge amount of various pattern recognition problems in every day life [2]. Examples of such tasks include the recognition of letters in a book, the face of a friend in a crowd, a spoken word embedded in noise, the chart of a presentation, the proper key to the locked door, the smell of coffee in the cafeteria, the importance of a certain message in the mail folder, and many more. These simple examples illustrate the essence of pattern recognition. In the world there exist classes of patterns we distinguish according to certain knowledge that we have learned before [3].

Most pattern recognition tasks encountered by humans can be solved intuitively without explicitly defining a certain method or specifying an exact

algorithm. Yet, formulating a pattern recognition problem in an algorithmic way provides us with the possibility to delegate the task to a machine. This can be particularly interesting for very complex as well as for cumbersome tasks in both science and industry. Examples are the prediction of the properties of a certain molecule based on its structure, which is known to be very difficult, or the reading of handwritten payment orders, which might become quite tedious when their quantity reaches several hundreds.

Such examples have evoked a growing interest in adequate modeling of the human pattern recognition ability, which in turn led to the establishment of the research area of pattern recognition and related fields, such as machine learning, data mining, and artificial intelligence [4]. The ultimate goal of pattern recognition as a scientific discipline is to develop methods that mimic the human capacity of perception and intelligence. More precisely, pattern recognition as computer science discipline aims at defining mathematical foundations, models and methods that automate the process of recognizing patterns of diverse nature.

However, it soon turned out that many of the most interesting problems in pattern recognition and related fields are extremely complex, often making it difficult, or even impossible, to specify an explicit programmed solution. For instance, we are not able to write an analytical program to recognize, say, a face in a photo [5]. In order to overcome this problem, pattern recognition commonly employs the so called *learning methodology*. In contrast to the *theory driven* approach, where precise specifications of the algorithm are required in order to solve the task analytically, in this approach the machine is meant to learn itself the concept of a class, identify objects, and discriminate between them.

Typically, a machine is fed with training data, coming from a certain problem domain, whereon it tries to detect significant rules in order to solve the given pattern recognition task [5]. Based on this training set of samples and particularly the inferred rules, the machine becomes able to make predictions about new, i.e. unseen, data. In other words, the machine acquires generalization power by learning. This approach is highly inspired by the human ability to recognize, for instance, what a dog is, given just a few examples of dogs. Thus, the basic idea of the learning methodology is that a few examples are sufficient to extract important knowledge about their respective class [4]. Consequently, employing this approach requires computer scientists to provide mathematical foundations to a machine allowing it to learn from examples.

Pattern recognition and related fields have become an immensely im-

portant discipline in computer science. After decades of research, reliable and accurate pattern recognition by machines is now possible in many formerly very difficult problem domains. Prominent examples are mail sorting [6, 7], e-mail filtering [8, 9], text categorization [10–12], handwritten text recognition [13–15], web retrieval [16, 17], writer verification [18, 19], person identification by fingerprints [20–23], gene detection [24, 25], activity predictions for molecular compounds [26, 27], and others. However, the indispensable necessity of further research in automated pattern recognition systems becomes obvious when we face new applications, challenges, and problems, as for instance the search for important information in the huge amount of data which is nowadays available, or the complete understanding of highly complex data which has been made accessible just recently. Therefore, the major role of pattern recognition will definitely be strengthened in the next decades in science, engineering, and industry.

## 1.2 Learning Methodology

The key task in pattern recognition is the analysis and the classification of patterns [28]. As discussed above, the learning paradigm is usually employed in pattern recognition. The learning paradigm states that a machine tries to infer classification and analysis rules from a sample set of training data. In pattern recognition several learning approaches are distinguished. This section goes into the taxonomy of *supervised*, *unsupervised*, and the recently emerged *semi-supervised* learning. All of these learning methodologies have in common that they incorporate important information captured in training samples into a mathematical model.

**Supervised Learning** In the supervised learning approach each training sample has an associated class label, i.e. each training sample belongs to one and only one class from a finite set of classes. A class contains similar objects, whereas objects from different classes are dissimilar. The key task in supervised learning is *classification*. Classification refers to the process of assigning an unknown input object to one out of a given set of classes. Hence, supervised learning aims at capturing the relevant criteria from the training samples for the discrimination of different classes. Typical classification problems can be found in biometric person identification [29], optical character recognition [30], medical diagnosis [31], and many other domains.

Formally, in the supervised learning approach, we are dealing with a

pattern space  $\mathcal{X}$ , and a space of class labels  $\Omega$ . All patterns  $x \in \mathcal{X}$  are potential candidates to be recognized, and  $\mathcal{X}$  can be any kind of space (e.g. the real vector space  $\mathbb{R}^n$ , or a finite or infinite set of symbolic data structures<sup>1</sup>). For *binary classification* problems the space of class labels is usually defined as  $\Omega = \{-1, +1\}$ . If the training data is labeled as belonging to one of  $k$  classes, the space of class labels  $\Omega = \{\omega_1, \dots, \omega_k\}$  consists of a finite set of discrete symbols, representing the  $k$  classes under consideration. This task is then referred to as *multiclass classification*. Given a set of  $N$  labeled training samples  $\{(x_i, \omega_i)\}_{i=1, \dots, N} \subset \mathcal{X} \times \Omega$  the aim is to derive a prediction function  $f : \mathcal{X} \rightarrow \Omega$ , assigning patterns  $x \in \mathcal{X}$  to classes  $\omega_i \in \Omega$ , i.e. *classifying* the patterns from  $\mathcal{X}$ . The prediction function  $f$  is commonly referred to as *classifier*. Hence, supervised learning employs some algorithmic procedures in order to define a powerful and accurate prediction function<sup>2</sup>.

Obviously, an overly complex classifier system  $f : \mathcal{X} \rightarrow \Omega$  may allow perfect classification of all training samples  $\{x_i\}_{i=1, \dots, N}$ . Such a system, however, might perform poorly on unseen data  $x \in \mathcal{X} \setminus \{x_i\}_{i=1, \dots, N}$ . In this particular case, which is referred to as *overfitting*, the classifier is too strongly adapted to the training set. Conversely, *underfitting* occurs when the classifier is unable to model the class boundaries with a sufficient degree of precision. In the best case, a classifier integrates the trade-off between underfitting and overfitting in its training algorithm. Consequently, the overall aim is to derive a classifier from the training samples  $\{x_i\}_{i=1, \dots, N}$  that is able to correctly classify a majority of the unseen patterns  $x$  coming from the same pattern space  $\mathcal{X}$ . This ability of a classifier is generally referred to as *generalization power*. The underlying assumption for generalization is that the training samples  $\{x_i\}_{i=1, \dots, N}$  are sufficiently representative for the whole pattern space  $\mathcal{X}$ .

**Unsupervised Learning** In unsupervised learning, as opposed to supervised learning, there is no labeled training set whereon the class concept is learned. In this case the important information needs to be extracted from the patterns without the information provided by the class

<sup>1</sup>We will revisit the problem of adequate pattern spaces in the next section.

<sup>2</sup>The supervised learning approach can be formulated in a more general way to include other recognition tasks than classification, such as *regression*. Regression refers to the case of supervised pattern recognition in which rather than a class  $\omega_i \in \Omega$ , an unknown real-valued feature  $y \in \mathbb{R}$  has to be predicted. In this case, the training sample consists of pairs  $\{(x_i, y_i)\}_{i=1, \dots, N} \subset \mathcal{X} \times \mathbb{R}$ . However, in this book considerations in supervised learning are restricted to pattern classification problems.

label [5]. Metaphorically speaking, in this learning approach no teacher is available defining which class a certain pattern belongs to, but only the patterns themselves. More concretely, in the case of unsupervised learning, the overall problem is to partition a given collection of unlabeled patterns  $\{x_i\}_{i=1,\dots,N}$  into  $k$  meaningful groups  $C_1, \dots, C_k$ . These groups are commonly referred to as *clusters*, and the process of finding a natural division of the data into homogeneous groups is referred to as *clustering*. The *clustering algorithm*, or *clusterer*, is a function mapping each pattern  $\{x_i\}_{i=1,\dots,N}$  to a cluster  $C_j$ . Note that there are also *Fuzzy clustering* algorithms available, allowing a pattern to be assigned to several clusters at a time. Yet, in the present book only *hard clusterings*, i.e. clusterings where patterns are assigned to exactly one cluster, are considered.

Clustering is particularly suitable for the exploration of interrelationships among individual patterns [32, 33]. That is, clustering algorithms are mainly used as data exploratory and data analysis tool. The risk in using clustering methods is that rather than finding a natural structure in the underlying data, we are imposing an arbitrary and artificial structure [3]. For instance, for many of the clustering algorithms the number of clusters  $k$  to be found in the data set has to be set by the user in advance. Moreover, given a particular set of patterns, different clustering algorithms, or even the same algorithm randomly initialized, might lead to completely different clusters. An open question is in which scenarios to employ a clustering approach at all [34].

An answer can be found in the concept of a cluster. Although both concepts, class and cluster, seem to be quite similar, their subtle difference is crucial. In contrast to the concept of a class label, the assignment of a pattern to a certain cluster is not intrinsic. Changing a single feature of a pattern, or changing the distance measurement between individual patterns, might change the partitioning of the data, and therefore the patterns' cluster membership. Conversely, in a supervised learning task the class membership of the patterns of the labeled training set never changes. Hence, the objective of clustering is not primarily the classification of the data, but an evaluation and exploration of the underlying distance measurement, the representation formalism, and the distribution of the patterns.

**Semi-supervised Learning** Semi-supervised learning is halfway between supervised and unsupervised learning [35]. As the name of this approach indicates, both labeled and unlabeled data are provided to the learning algorithm. An important requirement for semi-supervised learning

approaches to work properly is that the distribution of the underlying patterns, which the unlabeled data will help to learn, is relevant for the given classification problem. Given this assumption, there is evidence that a classifier can lead to more accurate predictions by also taking into account the unlabeled patterns. In the present book, however, this learning approach is not explicitly considered, but only purely supervised and unsupervised learning methods, i.e. classification and clustering algorithms.

### 1.3 Statistical and Structural Pattern Recognition

The question how to represent patterns in a formal way such that a machine can learn to discriminate between different classes of objects, or find clusters in the data, is very important. Consider, for instance, a pattern classification task where images of dogs have to be distinguished from images of birds. Obviously, a person might come up with the classification rule that birds have two legs, two wings, plumes, and a pecker. Dogs, on the other hand, have four legs, no wings, a fur, and no pecker. Based on this knowledge every human will be able to assign images showing dogs and birds to one of the two categories.

Yet, for a machine the concepts of *leg*, *fur*, or *wing* have no predefined meaning. That is, the knowledge about the difference between dogs and birds established above, has no relevance for machines. Consequently, the first and very important step in any pattern recognition system is the task of defining adequate data structures to represent the patterns under consideration. That is, the patterns have to be encoded in a machine readable way. There are two major ways to tackle this crucial step: the *statistical* and the *structural* approach<sup>3</sup>. In this section we examine both approaches and discuss drawbacks and advantages of them.

**Statistical Pattern Recognition** In statistical pattern recognition objects or patterns are represented by feature vectors. That is, a pattern is formally represented as vector of  $n$  measurements, i.e.  $n$  numerical features. Hence, an object can be understood as a point in the  $n$ -dimensional real space, i.e.  $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ . As pointed out in [4], in a rigorous mathematical sense, *points* and *vectors* are not the same. That is, points

<sup>3</sup>Note that there exists also the *syntactical* approach where the patterns of a certain class are encoded as strings of a formal language [36]. Language analysis can then be used in order to discriminate between different classes of objects. The present book, however, is restricted to statistical and structural pattern recognition.

are defined by a fixed set of coordinates in  $\mathbb{R}^n$ , while vectors are defined by differences between points. However, in statistical pattern recognition both terminologies are used interchangeably. Patterns are represented as points in  $\mathbb{R}^n$ , but for the sake of convenience, they are treated as vectors so that all vector space operations are well defined.

Representing objects or patterns by feature vectors  $\mathbf{x} \in \mathbb{R}^n$  offers a number of useful properties, in particular, the mathematical wealth of operations available in a vector space. That is, computing the sum, the product, the mean, or the distance of two entities is well defined in vector spaces, and moreover, can be efficiently computed. The convenience and low computational complexity of algorithms that use feature vectors as their input have eventually resulted in a rich repository of algorithmic tools for statistical pattern recognition and related fields [1, 5, 37].

However, the use of feature vectors implicates two limitations. First, as vectors always represent a predefined set of features, all vectors in a given application have to preserve the same length regardless of the size or complexity of the corresponding objects. Second, there is no direct possibility to describe binary relationships that might exist among different parts of a pattern. These two drawbacks are severe, particularly when the patterns under consideration are characterized by complex structural relationships rather than the statistical distribution of a fixed set of pattern features [38].

**Structural Pattern Recognition** Structural pattern recognition is based on symbolic data structures, such as *strings*, *trees*, or *graphs*. Graphs, which consist of a finite set of nodes connected by edges, are the most general data structure in computer science, and the other common data types are special cases of graphs. That is, from an algorithmic perspective both strings and trees are simple instances of graphs<sup>4</sup>. A string is a graph in which each node represents one character, and consecutive characters are connected by an edge [39]. A tree is a graph in which any two nodes are connected by exactly one path. In Fig. 1.1 an example of a string, a tree, and a graph are illustrated. In the present book graphs are used as representation formalism for structural pattern recognition. Although focusing on graphs, the reader should keep in mind that strings and trees are always included as special cases.

---

<sup>4</sup>Obviously, also a feature vector  $\mathbf{x} \in \mathbb{R}^n$  can be represented as a graph, whereas the contrary, i.e. finding a vectorial description for graphs, is highly non-trivial and one of the main objectives of this book.

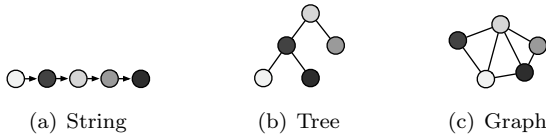


Fig. 1.1 Symbolic data structures.

The above mentioned drawbacks of feature vectors, namely the size constraint and the lacking ability of representing relationships, can be overcome by graph based representations [40]. In fact, graphs are not only able to describe properties of an object, but also binary relationships among different parts of the underlying object by means of edges. Note that these relationships can be of various nature, viz. spatial, temporal, or conceptual. Moreover, graphs are not constrained to a fixed size, i.e. the number of nodes and edges is not limited a priori and can be adapted to the size or the complexity of each individual object under consideration.

Due to the ability of graphs to represent properties of entities and binary relations at the same time, a growing interest in graph-based object representation can be observed [40]. That is, graphs have found widespread applications in science and engineering [41–43]. In the fields of bioinformatics and chemoinformatics, for instance, graph based representations have been intensively used [27, 39, 44–46]. Another field of research where graphs have been studied with emerging interest is that of web content mining [47, 48]. Image classification from various fields is a further area of research where graph based representations draw the attention [49–55]. In the fields of graphical symbol and character recognition, graph based structures have also been used [56–59]. Finally, it is worth to mention computer network analysis, where graphs have been used to detect network anomalies and predict abnormal events [60–62].

One drawback of graphs, when compared to feature vectors, is the significant increase of the complexity of many algorithms. For example, the comparison of two feature vectors for identity can be accomplished in linear time with respect to the length of the two vectors. For the analogous operation on general graphs, i.e. testing two graphs for isomorphism, only exponential algorithms are known today. Another serious limitation in the use of graphs for pattern recognition tasks arises from the fact that there is little mathematical structure in the domain of graphs. For example, computing the (weighted) sum or the product of a pair of entities (which are elementary operations required in many classification and clustering algo-

gorithms) is not possible in the domain of graphs, or is at least not defined in a standardized way. Due to these general problems in the graph domain, we observe a lack of algorithmic tools for graph based pattern recognition.

**Bridging the Gap** A promising approach to overcome the lack of algorithmic tools for both graph classification and graph clustering is graph embedding in the real vector space. Basically, an embedding of graphs into vector spaces establishes access to the rich repository of algorithmic tools for pattern recognition. However, the problem of finding adequate vectorial representations for graphs is absolutely non-trivial. As a matter of fact, the representational power of graphs is clearly higher than that of feature vectors, i.e. while feature vectors can be regarded as unary attributed relations, graphs can be interpreted as a set of attributed unary and binary relations of arbitrary size. The overall objective of all graph embedding techniques is to condense the high representational power of graphs into a computationally efficient and mathematically convenient feature vector.

The goal of the present book is to define a new class of graph embedding procedures which can be applied to both directed and undirected graphs, as well as to graphs without and with arbitrary labels on their nodes and edges. Furthermore, the novel graph embedding framework is distinguished by its ability to handle structural errors. The presented approach for graph embedding is primarily based on the idea proposed in [4, 63–66] where the *dissimilarity representation* for pattern recognition in conjunction with feature vectors was first introduced. The key idea is to use the dissimilarities of an input graph  $g$  to a number of training graphs, termed *prototypes*, as a vectorial description of  $g$ . In other words, the dissimilarity representation rather than the original graph representation is used for pattern recognition.

## 1.4 Dissimilarity Representation for Pattern Recognition

Human perception has reached a mature level and we are able to recognize common characteristics of certain patterns immediately. Recognizing a face, for instance, means that we evaluate several characteristics at a time and make a decision within a fraction of a second whether or not we know the person standing *vis-a-vis*<sup>5</sup>. These discriminative characteristics of a pattern are known as *features*. As we have seen in the previous sections,

<sup>5</sup>There exists a disorder of face recognition, known as *prosopagnosia* or face blindness, where the ability to recognize faces is impaired, while the ability to recognize other objects may be relatively intact.

there are two major concepts in pattern recognition, differing in the way the features of a pattern are represented. However, whether we use real vectors or some symbolic data structure, the key question remains the same: How do we find good descriptors for the given patterns to solve the corresponding pattern recognition task.

The traditional goal of the *feature extraction* task is to characterize patterns to be recognized by measurements whose values are very similar for objects of the same class, and very dissimilar for objects of different classes [1]. Based on the descriptions of the patterns a similarity measure is usually defined, which in turn is employed to group together similar objects to form a class. Hence, the similarity or dissimilarity measure, which is drawn on the extracted features, is fundamental for the constitution of a class. Obviously, the concept of a class, the similarity measure, and the features of the pattern are closely related, and moreover, crucially depend on each other.

Due to this close relationship, the dissimilarity representation has emerged as a novel approach in pattern recognition [4, 63–68]. The basic idea is to represent objects by vectors of dissimilarities. The dissimilarity representation of objects is based on pairwise comparisons, and is therefore also referred to as *relative representation*. The intuition of this approach is that the notion of proximity, i.e. similarity or dissimilarity, is more fundamental than that of a feature or a class. Similarity groups objects together and, therefore, it should play a crucial role in the constitution of a class [69–71]. Using the notion of similarity, rather than features, renews the area of pattern recognition in one of its foundations, i.e. the representation of objects [72–74].

Given these general thoughts about pattern representation by means of dissimilarities, we revisit the key problem of structural pattern recognition, namely the lack of algorithmic tools, which is due to the sparse space of graphs containing almost no mathematical structure. Although for graphs basic mathematical operations cannot be defined in a standardized way, and the application-specific definition of graph operations often involves a tedious and time-consuming development process [28], a substantial number of graph matching paradigms have been proposed in the literature (for an extensive review of graph matching methods see [40]). Graph matching refers to the process of evaluating the structural similarity or dissimilarity of two graphs. That is, for the discrimination of structural patterns various proximity measures are available. Consequently, the dissimilarity representation for pattern recognition can be applied to graphs, which builds a

natural bridge for unifying structural and statistical pattern recognition.

In Fig. 1.2 an illustration of the procedure which is explored in the present book is given. The intuition behind this approach is that, if  $p_1, \dots, p_n$  are arbitrary prototype graphs from the underlying pattern space, important information about a given graph  $g$  can be obtained by means of the dissimilarities of  $g$  to  $p_1, \dots, p_n$ , i.e.  $d(g, p_1), \dots, d(g, p_n)$ . Moreover, while the flattening of the depth of information captured in a graph to a numerical vector in a standardized way is impossible, pairwise dissimilarities can be computed in a straightforward way. Through the arrangement of the dissimilarities in a tuple, we obtain a vectorial description of the graph under consideration. Moreover, by means of the integration of graph dissimilarities in the vector, a high discrimination power of the resulting representation formalism can be expected. Finally, and maybe most importantly, through the use of dissimilarity representations the lack of algorithmic tools for graph based pattern recognition is instantly eliminated.

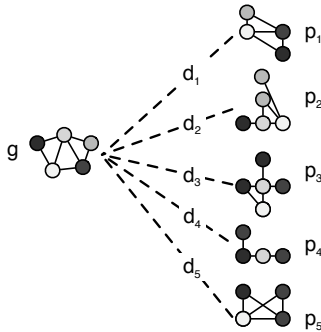


Fig. 1.2 The dissimilarity representation for graph based data structures. A graph  $g$  is formally represented as a vector of its dissimilarities  $(d_1, \dots, d_n)$  to  $n$  predefined graphs  $p_1, \dots, p_n$  (here  $n = 5$ ).

The concept of dissimilarity representation applied to graphs appears very elegant and straightforward in order to overcome the major drawback of graphs. However, the proposed approach stands and falls with the answer to the overall question, whether or not we are able to outperform traditional pattern recognition systems that are directly applied in the graph domain. That is, the essential and sole quality criterion to be applied to the novel approach is the degree of improvement in the recognition accuracy. More pessimistically one might ask, is there any improvement at all? As one

might guess from the size of the present book, in principle the answer is “yes”. However, a number of open issues have to be investigated to reach that answer. For instance, how do we select the prototype graphs, which serve us as reference points for the transformation of a given graph into a real vector? Obviously, the prototype graphs, and also their numbering, have a pivotal impact on the resulting vectors. Hence, a well defined set of prototype graphs seems to be crucial to succeed with the proposed graph embedding framework. The present book approaches this question and comes up with solutions to this and other problems arising with the dissimilarity approach for graph embedding.

## 1.5 Summary and Outline

In the present introduction, the importance of pattern recognition in our everyday life, industry, engineering, and in science is emphasized. Pattern recognition covers the automated process of analyzing and classifying patterns occurring in the real world. In order to cope with the immense complexity of most of the common pattern recognition problems (e.g. recognizing the properties of a molecular compound based on its structure), the learning approach is usually employed. The general idea of this approach is to endow a machine with mathematical methods such that it becomes able to infer general rules from a set of training samples.

Depending on whether the samples are labeled or not, supervised or unsupervised learning is applied. Supervised learning refers to the process where the class structure is learned on the training set such that the system becomes able to make predictions on unseen data coming from the same source. That is, a mathematical decision function is learned according to the class structure on the training set. In the case of unsupervised learning, the patterns under consideration are typically unlabeled. In this learning scheme, the structure of the patterns and especially the interrelationships between different patterns are analyzed and explored. This is done by means of clustering algorithms that partition the data set into meaningful groups, i.e. clusters. Based on the partitioning found, the distribution of the data can be conveniently analyzed. Note the essential difference between classification and clustering: The former aims at predicting a class label for unseen data, the latter performs an analysis on the data at hand.

There are two main directions in pattern recognition, viz. the statistical and the structural approach. Depending on how the patterns in a specific

problem domain are described, i.e. with feature vectors or with symbolic data structures, the former or the latter approach is employed. In the present book, structural pattern recognition is restricted to graph based representation. However, all other symbolic data structures, such as strings or trees, can be interpreted as simple instances of graphs.

Particularly, if structure plays an important role, graph based pattern representations offer a versatile alternative to feature vectors. A severe drawback of graphs, however, is the lack of algorithmic tools applicable in the sparse space of graphs. Feature vectors, on the other hand, benefit from the mathematically rich concept of a vector space, which in turn led to a huge amount of pattern recognition algorithms. The overall objective of the present book is to bridge the gap between structural pattern recognition (with its versatility in pattern description) and statistical pattern recognition (with its numerous pattern recognition algorithms at hand). To this end, the dissimilarity representation is established for graph structures.

The dissimilarity representation aims at defining a vectorial description of a pattern based on its dissimilarities to prototypical patterns. Originally, the motivation for this approach is based on the argument that the notion of proximity (similarity or dissimilarity) is more fundamental than that of a feature or a class [4]. However, the major motivation for using the dissimilarity representation for graphs stems from the fact that though there is little mathematical foundation in the graph domain, numerous procedures for computing pairwise similarities or dissimilarities for graphs can be found in the literature [40]. The present book aims at defining a stringent foundation for graph based pattern recognition based on dissimilarity representations and ultimately pursues the question whether or not this approach is beneficial.

The remainder of this book is organized as follows (see also Fig. 1.3 for an overview). Next, in Chapter 2 the process of graph matching is introduced. Obviously, as the graph embedding framework employed in this book is based on dissimilarities of graphs, the concept of graph matching is essential. The graph matching paradigm actually used for measuring the dissimilarity of graphs is that of graph edit distance. This particular graph matching model, which is known to be very flexible, is discussed in Chapter 3. A repository of graph data sets, compiled and developed within this book and suitable for a wide spectrum of tasks in pattern recognition and machine learning is described in Chapter 4. The graph repository consists of ten graph sets with quite different characteristics. These graph sets are used throughout the book for various experimental evaluations. Kernel

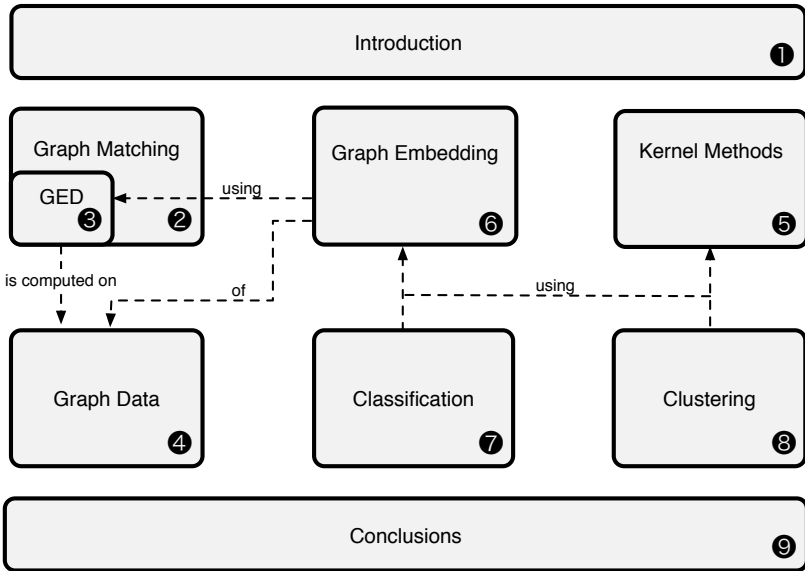


Fig. 1.3 Overview of the present book.

methods, a relatively novel approach in pattern recognition, are discussed in Chapter 5. Particularly, the support vector machine, kernel principal component analysis, and kernel  $k$ -means algorithm, are reviewed. These three kernel machines are employed in the present book for classification, transformation, and clustering of vector space embedded graphs, respectively. Moreover, the concept of graph kernel methods, which is closely related to the proposed approach of graph embedding, is discussed in detail in Chapter 5. The general procedure of our graph embedding framework is described in Chapter 6. In fact, the embedding procedure for graphs can be seen as a foundation for a novel class of graph kernels. However, as we will see in Chapter 6, the embedding procedure is even more powerful than most graph kernel methods. Furthermore, more specific problems, such as prototype selection and the close relationship of the presented embedding framework to *Lipschitz embeddings*, are discussed in this chapter. An experimental evaluation of classification and clustering of vector space embedded graphs is described in Chapters 7 and 8, respectively. For both scenarios exhaustive comparisons between the novel embedding framework and standard approaches are performed. Finally, the book is summarized and concluded in Chapter 9.