

Chapter 1

Introduction

Graphs are a popular concept for the representation of structured information. On a very basic level, a graph can be defined by a set of entities and a set of connections between these entities. Due to their universal definition, graphs have found widespread application for many years as tools for the representation of complex relations. For instance, in software engineering, graphs are used for the visualization of dependencies and interactions of components in large software systems. In relational databases, information is stored by defining relations between individual entities by means of unique identifiers, which is essentially equivalent to a graph structure. Flow-chart diagrams can be used to decompose complex processes into concise systems of states, rules, and transitions, which can be interpreted in a straightforward manner as nodes and edges in a graph. The world-wide web is another example of a graph, where webpages are regarded as documents and hyperlinks as connections between related documents. The general concept of representing and processing information in terms of binary relations, that is, links between two objects at a time, seems to be applicable to a wide range of problems. In the context of this book, graphs are used for the matching of structured data.

The key task in pattern recognition is the analysis and classification of objects [Friedman and Kandel (1999); Duda *et al.* (2000)]. In principle these objects, or patterns, can be of any kind, including images taken with a digital camera, spoken words captured by a microphone, or handwritten texts obtained by means of a digital pen. The detection and extraction of individual objects in images, for instance, can be referred to as a pattern analysis, whereas the recognition of cursive handwriting corresponds to pattern classification. The first step in any pattern recognition system consists of the representation of patterns by data structures that can be interpreted

by the recognition algorithm to be employed. For example, an image captured with a digital camera can be regarded as a pixel matrix of grayscale values, and recorded speech can be represented by a time-amplitude signal. In statistical pattern recognition [Duda *et al.* (2000)], patterns are represented by feature vectors of a fixed dimension. The basic idea is to define a set of features describing properties of the underlying patterns that are relevant for the recognition task under consideration. Such feature extraction procedures can be developed for patterns from completely different domains, which makes the statistical pattern recognition approach widely applicable. The main advantage of statistical pattern recognition is that feature vectors belong to a mathematically rich vector space. If a nearest-neighbor method is used for classification, the Euclidean distance between vectors can easily be computed; if principal component analysis is used for dimensionality reduction, vectors can be projected onto the maximal-variance subspace; if the K-means algorithm is used for clustering, the mean of a set of patterns can easily be computed; and if Gaussian mixture modelling is used for novelty detection, the mean and covariance matrix can be estimated empirically from a sample set of vectors.

Yet, in some cases, representing patterns uniformly by feature vectors of a fixed dimension is not appropriate. Especially if structure plays an important role, graphs offer a versatile alternative to feature vectors. If a feature vector is regarded as an unary attributed relation, graphs can analogously be interpreted as a set of attributed unary and binary relations of arbitrary size. The key advantage of graphs is that their representational power is clearly higher than that of feature vectors, since nodes and edges can be labeled with feature vectors themselves, and the number of nodes and edges can be adapted to the complexity of the pattern under consideration. For instance, if objects are to be detected and extracted from images, graphs are a natural choice of representation, since they allow us to represent extracted objects by nodes and object neighborhood relations by edges. A major difficulty of graph based pattern recognition, however, is the definition and application of algorithms for pattern analysis and classification in the space of graphs. While the representational power of graphs is higher than that of feature vectors, the sparse space of graphs contains almost no mathematical structure, in contrast to feature vector spaces. Basic mathematical concepts that are well-defined in vector spaces, such as the Euclidean distance or linear combinations of feature vectors, are often required for standard algorithms to be applicable. For graphs, these basic 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.

In recent years, a large number of graph matching methods based on various matching paradigms have been proposed [Conte *et al.* (2004)], ranging from the spectral decomposition of graph matrices to the training of artificial neural networks and from continuous optimization algorithms to optimal tree search procedures. Most graph matching algorithms are either restricted to a certain class of graphs or only applicable to weakly distorted data. For instance, a number of graph matching problems can be solved more efficiently for planar graphs than for general graphs. On the other hand, spectral graph matching methods are in principle applicable to arbitrary unlabeled graphs, but appear to be rather sensitive to noise. In view of these limitations, graph edit distance is considered one of the most powerful methods for graph matching [Bunke and Allermann (1983); Sanfeliu and Fu (1983)]. Graph edit distance is an error-tolerant dissimilarity measure for arbitrarily structured and arbitrarily labeled graphs. The basic idea is to define the dissimilarity of two graphs by the minimum amount of distortion that is needed to transform one graph into the other. The edit distance provides us with a general dissimilarity measure in the space of graphs, but this is not sufficient for most standard pattern recognition algorithms. In fact, edit distance based graph matching is basically limited to nearest-neighbor classification and K-median clustering. Unfortunately, for nearest-neighbor classifiers to be successful, a large number of training patterns covering a substantial part of the pattern space are required. In vector spaces, where the Euclidean distance to a large number of patterns can easily be computed, such a procedure is usually feasible. The edit distance algorithm, on the other hand, is computationally inefficient, and edit distance based nearest-neighbor classification is therefore restricted to rather small training sets.

The basic limitation of graph matching is due to the lack of mathematical structure in the space of graphs. Kernel machines, a novel class of algorithms for pattern analysis and classification, offer an elegant solution to this problem [Schölkopf and Smola (2002); Shawe-Taylor and Cristianini (2004)]. The basic idea of kernel machines is to address a pattern recognition problem in a related vector space instead of the original pattern space. That is, rather than defining mathematical operations in the space of graphs, all graphs are mapped into a vector space where these operations are readily available. Obviously, the difficulty is to find a mapping that preserves the structural similarity of graphs, at least to a certain ex-

tent. In other words, if two graphs are structurally similar, the two vectors representing these graphs should be similar as well, since the objective is to obtain a vector space embedding that preserves the characteristics of the original space of graphs. A key result from the theory underlying kernel machines states that an explicit mapping from the pattern space into a vector space is not required. Instead, from the definition of a pattern similarity measure, or kernel function, it follows that there exists such a vector space embedding and that the kernel function can be used to extract the information from vectors that is relevant for recognition. In fact, the family of kernel machines consists of all algorithms that can be formulated in terms of such a kernel function, including standard methods for pattern analysis and classification such as principal component analysis and nearest-neighbor classification. Hence, from the definition of a graph similarity measure, we obtain an implicit embedding of the entire space of graphs into a vector space.

In statistical pattern recognition, where patterns are not represented by graphs, but feature vectors, standard algorithms can directly be applied to the original pattern space without mapping and without kernel function. Principal component analysis, to give an example, extracts dominant linear directions in vectorial data sets. If principal component analysis is applied in conjunction with a kernel function and an implicit mapping of patterns into another vector space, one obtains an extension of standard principal component analysis that is able to capture non-linear characteristics. Hence, the kernel machine framework can be used to extend, in a single step, linear algorithms to non-linear ones. Moreover, there is theoretical evidence that, under some weak conditions, mapping patterns into vector spaces of higher dimension may be of advantage for pattern analysis and classification. The most prominent kernel machine is undoubtedly the support vector machine (SVM). The training of SVMs is based on insights from statistical learning theory and is considered one of the most promising methods for estimating hidden class boundaries from a sample set of patterns. The basic idea of the SVM training is to strive for a balanced condition between overfitting and underfitting of the training data at hand, so that the resulting classifier is expected to perform well on unseen data from the same population. Such theoretical considerations and their integration in the kernel machine framework render SVMs extremely powerful for pattern recognition. For graph matching, in addition to these interesting properties, the most important advantage of kernel machines is that to make the large class of kernel machines applicable to graphs, the underly-

ing pattern space need not be endowed with any mathematical structure beyond the existence of a kernel function.

The objective of this book is to define graph kernel functions that are to a certain extent related to graph edit distance, either by deriving kernel functions from edit distance or by incorporating the edit operation distortion model into the definition of kernel functions. The rationale behind this idea is that the application of edit distance based graph kernels to kernel machines is expected to outperform traditional edit distance based nearest-neighbor classifiers.

This book is structured as follows. In Chap. 2, basic notations and graph definitions are introduced and pointers to graph matching applications are given. Exact graph matching algorithms, such as subgraph isomorphism, are briefly covered, and error-tolerant approaches to graph matching proposed in recent years are discussed. Chapter 3 is devoted to graph edit distance. The edit distance measure is introduced and exact and approximate edit distance algorithms are given. Edit distance based nearest-neighbor classification is briefly addressed, and an application of edit distance to the fusion of graphs is presented. Chapter 4 is concerned with theoretical considerations and properties of kernel machines. Kernel functions are formally introduced, and support vector machines, kernel principal component analysis, and Fisher discriminant analysis are discussed. In Chap. 5, the main contribution of this book, novel graph kernels related to edit distance are presented. An experimental evaluation of these kernels, in comparison to a traditional nearest-neighbor classifier, follows in Chap. 6. The performance of the graph kernels in terms of recognition accuracy and running time is evaluated on five graph data sets representing line drawings, pictures, microscopic images, fingerprints, and molecules. Finally, a summary and concluding remarks are given in Chap. 7.