Statistical vs. Structural Pattern Recognition

A Survey

Roman Bögli

University of Applied Science and Arts Northwestern Switzerland (FHNW),
4600 Olten, Switzerland

December 24, 2019

Abstract. This paper surveys the different approaches in pattern recognition (PR). After the fundamental idea of PR is stated, a taxonomy landscape is presented which divides into three families, namely statistical, structural, and hybrids. The first represents a well-researched topic in PR which engendered popular and efficient supervised and unsupervised pattern discovery algorithms. The second family addresses techniques to find patterns in structurally represented data using graphs that allow capturing the information of relationships among objects. Thirdly, the hybridization of the prior two families will be discussed. This includes the elaboration of transformation methods that allow to embed a graph into a vector space using graph kernels or graph embedding.

Keywords: Statistical Pattern Recognition · Structural Pattern Recognition · Graph Kernels · Graph Embedding

1 Introduction

PR methods aim to find patterns in data which are hard or even infeasible to discover for humans. Although there are many different approaches to accomplish this, it always requires the derivation of a similarity or dissimilarity indicator among the data objects. Based on this measurement, arrangements can be built that group similar data objects together and distance dissimilar data objects from themselves. In classification disciplines, these groups are commonly referred to as *classes*, in clustering disciplines *clusters*.

The science of recognizing patterns in data started to emerge in the late sixties. A popular paper by Fu (1980) summarized the early developments of PR. Nowadays, PR has become an established and easily accessible tool which can be observed in many popular applications. For instance, recognition of faces (Hazim Barnouti, Sameer Mahmood Al-Dabbagh, & Esam Matti, 2016), creation of customer behaviour patterns (Koudehi, Rajeh, Farazmand, & Mohamad, 2014),

detection of cancer cells (Ozdemir & Gunduz-Demir, 2013), or forecasting time series data (Aghabozorgi, Seyed Shirkhorshidi, & Ying Wah, 2015).

A crucial part of PR is the representation of the underlying information in order to be processed by a computational algorithm. In general, one distinguishes between *statistical* and *structural* data. Either case has advantages and disadvantages regarding the task of emerging patterns in this data. Furthermore, the applicable algorithmic tool repository differs as well. This survey constitutes popular PR methods in both data representations and reviews ideas to link them together.

The remaining part of this survey is organized as follows: Section 2 examines the statistical representation and the two subcategories of supervised and unsupervised pattern discovery. Section 3 addresses the structural representation. Here, a brief introduction to graph theory will be provided, followed by the elaboration of exact and inexact graph matching algorithms. Section 4 addresses emerged unification concepts, namely graph kernels and graph embeddings. In the end, the most important findings will be concluded.

1.1 Pattern Recognition Families

In order to group similar data objects while separating them from dissimilar data objects, a distance measurement must be established. Large distances indicate low similarity or high dissimilarity while small distances indicate high similarity or low dissimilarity. Since these two distance notions are interchangeable, only the term *similarity* will be used for the rest of this survey for sake of convenience.

Before the distances can be evaluated, the underlying data must be brought into a computer-readable representation. There are two types of representations, namely a statistical representation or a structural representation. Depending on the data situation, a processable representation is either natively present or must first be derived by applying certain methods. Besides the representation, also the available PR tools differ in these two data situations which is why one distinguishes them in two families. The focus of this survey lies on these two prominent PR families.

Furthermore, a third type will be addressed which represents the unification of structural distance derivations and statistical pattern discovery techniques. Such hybrid forms firstly examine structural relationships of the underlying data with the goal to perform the actual pattern formation statistically. This can be reached by means of embedding data objects explicitly or implicitly in a multidimensional vector space (Bunke & Riesen, 2012).

In the past years, the development of a fourth family named *End-To-End Learning* (EEL) has grown in importance. It includes the automated integration of several subtasks which usually must be conducted separately or manually. *Convolutional Neural Networks* represent an implementation of EEL. Feature

detection and extraction from raw data is one of these subtasks, to name an example. In fields where the majority of input data is present in an unstructured binary form [1], demand such a PR approach. Autonomous driving systems (Bojarski et al., 2016) or speech recognition and interpretation software (Amodei et al., 2015) deliver ideal use cases for this promising classification solution. However, this survey will not examine this PR family any further since it represents a fundamentally different paradigm.

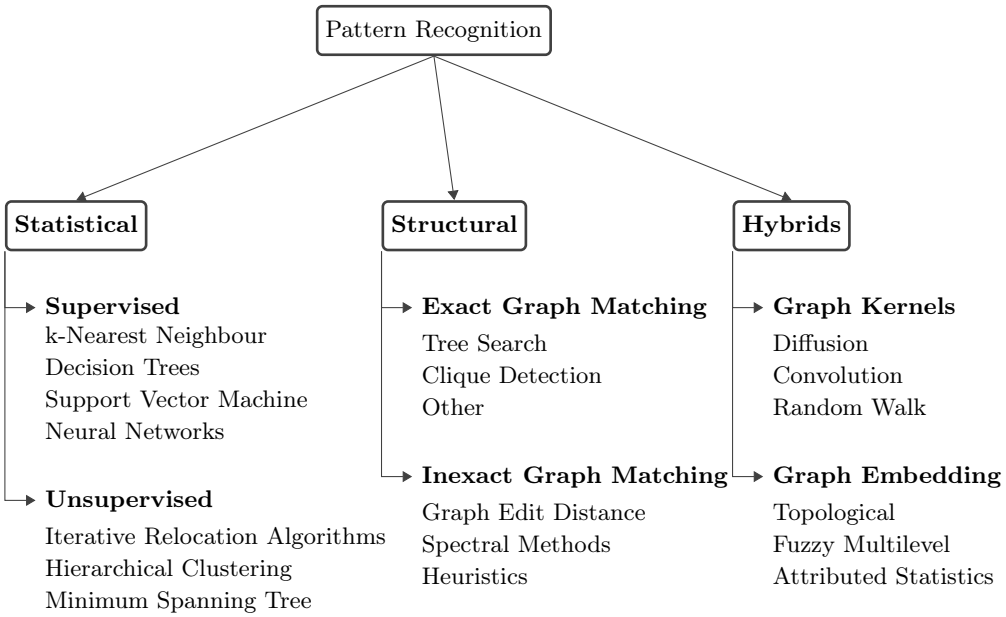

Fig. 1. Taxonomy tree outlining the three PR families statistical, structural, and hybrids followed by its subcategories. The leaves either express concrete algorithms or different classes of algorithms.

2 Statistical Representation

When a set of data objects can be described by the same set of features, the data objects are statistically represented. An example would be a list of people indicating their age, height, weight, and hair color. This allows to represent each person as a vector of fixed length in a vector space where the axes correspond to the attributes.

[1] e.g. signal data

Such a representation has two advantages. Firstly, features can be correlated in order to derive a new feature, for instance, a body mass index. The possibility of conducting arithmetic conversions allows to assess the data set more diversely. This freedom does not directly exist in structural representations. The second advantage lies in the fact that the pure statistical information itself already serves as a distance measure since it distinguishes the data objects from each other natively. Thus, separating tall and small people can be performed effortlessly.

Fukunaga (1990) broadly formulates statistical PR as a density estimation problem of objects in a high-dimensional space and the question of how to partition this space into regions so that they correspond to observed categories. In general, such pattern discovery algorithms are either based on supervised learning or unsupervised learning (Duda, Hart, & Stork, 2001). Both categories are elaborated in this chapter.

2.1 Supervised

In supervised PR techniques, the classification model or *classifier* is derived from a pre-labelled data set. This data set is commonly referred to as *training set* since it trains the classifier in order to separate the data objects in an optimal way based on their known class memberships. The goal is to create a model that eventually will classify new, unlabelled data objects likewise. Such a model acts as a discriminant function which takes the features of a data object as input and outputs the class allocation (Fukunaga, 1990). The training set calibrated this function.

This rather trivial approach of assigning new data objects to pre-existing classes based on the data object that are already in these classes can be applied to many ordinary problems. Creditworthiness analysis serves as an example (Turkson, Baagyere, & Wenya, 2016). In this example, the authors illustrate different techniques to show whether or not a customer pays back its loan in future by means of feature such as age, balance limit, and payment history.

The rest of this chapter surveys three prominent supervised classification techniques, namely the k-Nearest-Neighbour, decision trees, and Support Vector Machines. These fundamental techniques have gained importance in PR since their generalization is often used in more sophisticated PR algorithms.

k-Nearest-Neighbour (kNN) is a prominent technique to solve classification problems due to its simplicity (Webb & Copsey, 2011). Here, an unclassified data object receives its classification based on its location in the vector space. The data objects from the training set serve as orientation points. Using the *Euclidean Distance*, the k closest orientation points are determined. "The number of neighbours k is selected as a trade-off between choosing a value large enough

to reduce the sensitivity to noise, and choosing a value small enough that the neighbourhood does not extend to the domain of other classes" (Webb & Copsey, 2011, p. 154).

The kNN classifier poses a challenge when the identified k neighbours belonging to k or $k-n$ different classes where $n < k-1$. Such a tie can be resolved by either rejecting the classification decision or making a random choice. Dudani (1976) introduces a version of the kNN, which allows to weight closer neighbours more heavily in order to make a more veridical classification. *Fuzzy* kNN is another way to address this problem. Here, k class memberships are proportionally represented in sample vectors (Keller, Gray, & Givens, 1985).

Decision trees represent another form of supervised classification. A tree is derived from the training set where each node represents a feature and each edge a decision function. The features that are most salient are preferably represented first in the decision tree (Jain, Duin, & Jianchang Mao, 2000). *Hunt's algorithm* serves as basis for multiple tree construction algorithms, such as CART (Breiman, Friedman, Olshen, & Stone, 1984), C4.5 (Quinlan, 1993), or ID3 (Quinlan, 1986). Once the tree has been constructed, unclassified data objects can be fed into it. After they traversed through the appropriate nodes, they eventually end up in a leaf and therewith in class.

According to Jain et al. (2000), there are two advantages of decision trees. One is its low computational complexity. Another beneficial property is the interpretation of decision rules based on the individual features. On the other hand, the authors mention the risk of obtaining over-trained or *over-fitted* trees as a disadvantage. The better a decision tree classifies the data object in the training set, the higher the risk of losing generality and gaining specificity. However, this risk can be diminished by applying pruning (Gelfand, Ravishankar, & Delp, 1991), a technique to remove very specific sections of the tree in order to remain general.

Breiman (2001) introduced the *Random Forest* classification approach. By randomly choosing data object from the training set, several new data sets of equal size are created. This process is also known as *bootstrap aggregation* or *bagging* (Breiman, 1996). Each of this bootstrapped data set is used to construct a decision tree. New data object are then fed into all of these trees resulting in a proportional class assignments.

Support Vector Machines (SVMs) constitutes a third widely used classification technique due to its versatile implementation potential. The concept was initially introduced by Vapnik (1982). Here, data objects are projected into a higher dimensional feature space in order to find a hyperplane that separates the patterns in an optimal way from each other (Webb & Copsey, 2011). The

hyperplane serves as a linear discriminant function which allows to classify new data objects accordingly in this new vector space. Byun and Lee (2002, p. 214) comprehensively exemplify this as follows: "Such an optimal hyperplane is the one with the maximum margin of separation between the two classes, where the margin is the sum of the distances from the hyperplane to the closest data points of each of the two classes. These closest data points are called Support Vectors (SVs)".

SVMs are especially useful when the underlying data objects cannot directly be classified by linear separation function, which is usually the case in real-world problems. Using non-linear transformation functions, the data objects are mapped into the new vector space. Thus, the data landscape is bent which eventually allows separating patterns linearly again. This process is known as *kernalization* or the *kernel trick*, since the functions used to perform the transformation from the original vector space to the new higher dimensional vector space are called *kernel functions* (Webb & Copsey, 2011).

Although SVMs are designed for binary classification problems, they can be applied to multi-class problems using one of two techniques. One technique suggests a pairwise comparison of one class with all other classes (Trafalis & Oladunni, 2005). Another technique compares the data objects of one class in the training set to a virtual class representing the data object of all remaining classes (Zhao, Liu, & Xia, 2000).

Byun and Lee mention disadvantages that arise in SVMs applications. This includes the difficult choice of the optimal kernel function, the high computational complexity, and the right selection of the SVs. The latter is especially important when the training set includes noise or expresses non-separable patterns.

Other techniques in supervised classification include *Neural Networks* and *Naive Bayes algorithms*. Neural Networks consists of several linear discrimination functions, also known as *perceptrons*, chained together in order to form a multi-layer network. The training set is then used to orchestrate the weights in the individual discrimination functions which simultaneously trains the classification model itself (Duda et al., 2001).

Naive Bayes algorithms classify new data objects based on the likelihoods observed in the training set (Zhang, 2004). This probabilistic classifier model, however, assumes complete independence among the features. Although this is often not the case in reality, the results are competitive. Common applications of it can be found in fields where high dimensional data sets are present which is usually the case in text classification problems (Aggarwal & Zhai, 2012).

2.2 Unsupervised

In unsupervised PR techniques, no pre-labelled data objects exist from which a model could be derived. Patterns are purely emerged by comparing the values that define a data object (Webb & Copsey, 2011). There are many application scenarios in practice, such as clustering clients into different personas or grouping documents. The latter may result in map where informal and formal documents appear in different clusters due to sentence length, writing style, or the (in)existence of document building blocks such as tables or cross references.

Other than in supervised problems, where training data supervises the creation of the classifier, clusters are purely emerged by examining each data objects in the context of the completeness of the entire data set. "A good clustering method will produce high superiority clusters with high intra-class similarity and low inter-class similarity" (Mann & Kaur, 2013, p. 42). In general, the number of clusters obtained is not known in advance. However, some methods allow to specify the desired amount of cluster in advance or afterwards.

Madhulatha (2012) divides the algorithms used in clustering problems into different classes. In *Partitioning Clustering* (PC), clusters are evolved by slicing the data set into different regions or partitions. *Hierarchical Clustering* (HC) creates clusters by either dividing or aggregating the data objects into smaller or bigger subsets respectively. The result of HC algorithms are usually *hard* clusters, meaning that each data object only belongs to one cluster. PC algorithms, however, allow so-called *fuzzy* clustering (Äyrämö & Kärkkäinen, 2006) by expressing the cluster assignments for data object in probabilities. Other classes are *density-based* (Kriegel, Kröger, Sander, & Zimek, 2011) and *grid-based* (Amini, Wah, Saybani, & Yazdi, 2011) algorithms. Together with PC algorithms, they belong into the group of non-hierarchical clustering methods. These two classes, however, will not be revisited in further progress of this survey.

Iterative relocation algorithms (IRAs) belong to the most popular techniques used to establish clusters. IRAs partition the data landscape into regions. These regions represent the clusters. A user-defined number of points (k) are randomly determined in the data's vector space representing the centres of the initial clusters. Afterwards, all the data objects become assigned to their nearest cluster directly followed by a recalculations of the cluster centres. Consequently, the centres are relocated. This process is repeated until a stopping criteria occurs, such as the convergence to a certain error margin threshold or a almost stationary cluster center (Webb & Copsey, 2011).

The *k-Means* algorithm represents a famous implementation of an IRA due to its computational simplicity (Theodoridis & Koutroumbas, 2009). It outputs *hard* cluster allocations, meaning that every data object is associated with only one class. The cluster centres are calculated using the mean of all distances

between the data objects belonging to this cluster and the cluster's center point. This reasons its sensitivity to outliers and noise (Theodoridis & Koutroumbas, 2009).

Meanwhile, there are several optimized versions of k-Means developed. Zhang (2000) introduces the *k-Harmonic-Means* algorithms which can be used beforehand in order to determine better initial cluster centres. Although the speed of k-Means is already seen as an advantage, Na, Xumin, and Yong (2010) further optimized the efficiency of calculating the distances by caching some information during the process.

Another IRA implementation is called *k-Medoids*. As opposed to its precursor k-Means, k-Medoids is robust to outliers and noise due to the fact of choosing a specific data object as cluster centre (Theodoridis & Koutroumbas, 2009). Therefore also discrete values can be processed which is not the case in k-Means since the mean values could be out of domain. On the other hand, the determination of the medoids is known to be computationally more demanding compared to averaging. In addition, the centres do not have any statistical meaning (Theodoridis & Koutroumbas, 2009).

For large data sets, Kaufman and Rousseeuw (2005) propose the *Clustering for Large Applications* (CLARA) algorithm which builds on samplewise clustering. In 2002, Ng and Han introduced another solution called *Clustering Large Applications based on Randomized Search* (CLARANS). Here, the computational effort is further reduced using randomized search.

Hierarchical Clustering (HC) is another popular class of clustering algorithms. It is extended by two subclasses, namely *agglomerative* and *divisive* clustering. In agglomerative algorithms [2], every data object forms its own cluster at the beginning. Then, the clusters are gradually combined until the entire data set represents one single cluster. Hence, it is also referred to as bottom-up approach.

Divisive algorithms [3] make use of the inverted process. Here, one starts with one cluster encompassing all data objects which eventually are divided into more granular clusters. Thus, one refers to it as top-down approach.

The result of both approaches can be visualized by means of a tree-like diagram called *dendrogram*. The leaves of a dendrogram correspond to the individual data objects. Each node level represents one iteration of the HC algorithm. This allows a user-defined level of granularity regarding the number of clusters which is seen as an advantage (Mann & Kaur, 2013). The shape of the dendrogram, however, looks significantly different depending on the distance linkage method used.

[2] also known as *agglomerative nesting* (AGNES)
[3] also known as *devise analysis* (DIANA)

In general, the distance between two clusters can be denoted by the pairwise distance of the data objects in two clusters (Kassambara, 2017). Consequently, *single linkage* refers to the minimum, *complete linkage* to the maximum, and *average linkage* to the average distance between the data objects of two clusters. Single linkage produces stepwise clusters and complete linkage produces more compact clusters with rather high-level grouping.

On the other hand, Mann and Kaur (2013) state that most HC algorithms do not revise their result in order to find optimizations which they see as a disadvantage. Finding an optimal cut point in the hierarchy tree, which eventually defines the number of clusters, can also be challenging.

Minimum Spanning Tree (MST) inspires a third class of algorithms to discover clusters. Here, the data objects are represented by means of a graph, where the distances represent the weights of the edges. A tree has to be found that touches or spans over all vertices without circuits (Gower & Ross, 1969). The tree with the lowest total weight is called MST. Kruskal (1956) and Prim (1957) proposed popular algorithms to accomplish this.

The edges of the MST are sorted in descending order according to their weights. This allows to start clustering the data by successively removing the heaviest edge from this stack (Grygorash, Zhou, & Jorgensen, 2006). Thanks to the antecedent sorting, the MST is divided into subgraphs with maximized inter-distance. Each subgraph corresponds to a new cluster.

Wang, Wang, and Mitchell Wilkes (2009) introduce an approach to optimize the construction of a MST in terms of computational complexity by applying a divide and conquer strategy. Päivinen (2005) presents an algorithm which aims to build a *scale-free* MST. In scale-free graphs, the vertices degree distribution is subject to a power-law (Newman, 2018). In other words, the majority of vertices are adjacent to a few highly connected ones what qualifies them as potential cluster centres.

3 Structural Representation

More sophisticated data situations such as network systems, cannot directly be embodied in a statistical representation. Instead, the data objects are primarily represented in a structural manner to determine a similarity index. Fu (1974) conducted fundamental research in this topic. In order to accomplish this, the underlying data is encoded by means of a graph. This graph describes the data's associated attributes and its relations to each other which allows to preserve the structural information for the pattern discovery process.

Riesen (2015) declares the two major disadvantages which are present in statistical PR methods. One is the fixed length of all vectors based on the predefined

set of attributes. The other lies in the missing possibility to describe relations between the individual vectors. Structural PR overcomes both of these limitations, however, at cost of higher computational complexity. The reason for this lies in the fact that common parts must be identified under the consideration of all their subgraphs (Riesen, Jiang, & Bunke, 2010).

Practical examples of such a graph-based approach can be found in pictorial PR problems. Technically, images can be represented in a vector space which encodes the color and the brightness of each pixel. This would allow to perform pattern discovery using the tools shown in Section 2. Such a representation, however, would be too detailed for further processing (Pavlidis, 1977). In order to describe an image structurally, one could use the segmentation boundaries which evolve after partitioning the pixel landscape into maximised regions that are consistent in color and light intensity.

Though the field of structural PR can also be divided into the two categories supervised and unsupervised, this survey focuses on the matching process of subgraphs since it forms an essential step in order to derive a similarity measure between graphs. After a brief introduction into graph theory, *exact* and *inexact* graph matching algorithms are discussed.

3.1 Graph Theory

Graph Theory (GT) closely relates to the mathematical subfield of *Operational Research* (OR). OR strives to model problems in order to find suitable answers among many possibilities using different techniques. This includes prominent disciplines such as linear, non-linear, dynamic, or network programming (Taha, 2017). What they generally have in common is that solutions are found algorithmically rather than by means of closed formulas. Especially network programming is important as it frames the fundamental concept of structural PR. Hence, this survey provides a condensed introduction to graph theory.

The explanations in this paragraph are sourced from Rosen (2019). A graph $G(V, E)$ consists of V vertices [4] and E edges. The edges are associated with either one or more vertices. When an edges starts and ends at the same vertex, it represents a loop. In *directed* graphs, an edge (u, v) starts at vertex u and ends at v. When the order is ignored, the graph is considered *undirected*. One can traverse a graph using a sequence of edges. The traversed distance within a graph is commonly referred to as path. The degree of a vertex describes the number of edges incident with it. Neglecting loops, this measure expresses how well a vertex is connected in a graph. In weighted graphs the edges are associated with values representing its weight. When the vertices of two different sets are solely connected with vertices of the other set, one speaks of a *bipartite* graph.

[4] Also referred to as nodes

When vertices and edges are associated with additional information, one speaks of an *Attributed Relational Graph* (ARG).

The field of OR provides interesting problems for which a decent answer can be found using GT and the tools that come with it. A social media network, for instance, could be modelled as an undirected graph. Vertices represent the persons while the edges identify the friendship relations among them. This allows to find answers to questions such as how well an individual is connected or through how many friends two strangers could know each other. The graph is considered an ARG, for instance, when person's demographic information is associated to each vertex.

Bitcoin and the underlying public blockchain ledger on the other hand serves as an example for a directed graph. Here the addresses can be seen as vertices while the transactions among them define the edges. Additionally the edges could be associated by the amount of Bitcoins transferred as well as the transaction's time stamp.

3.2 Graph Matching

Graph Matching (GM) algorithms aim to compute a value that expresses how similar two graphs are (Riesen et al., 2010). When two graphs are structurally identical, they are *isomorphic* to each other. An isomorphism can also be seen as a function that is able to distinctively map two objects using a bijection that preserves adjacencies. "In other words, when two simple graphs are isomorphic, there is a one-to-one correspondence between vertices of the two graphs that preserves the adjacency relationship" (Rosen, 2019, p. 706).

Subgraph isomorphism detection describes the task of finding identical parts of a graph G in a graph H (Bunke & Messmer, 1997). To transform this binary distinction into a more meaningful expression, one compares several parts of two graphs with each other. This allows the derivation of a similarity measure.

Finding isomorphic subgraphs is deemed an *NP-complete* problem (Larrosa & Valiente, 2002). This means that so far no algorithm exists that is able to solve this problem in polynomial time complexity. However, a given solution can be verified in polynomial time (Garey & Johnson, 2009).

Riesen et al. (2010) outline two paradigms in GM. One is the *exact* GM which identifies strict accordance between two graphs. The other approach accepts some level of inaccuracy, hence it is referred to as *inexact* GM. Many popular approaches in both strategies build on tree search algorithms (Conte, Foggia, Sansone, & Vento, 2004).

Exact GM algorithms assess the similarity of two graphs based on the number of identical subgraphs. The *depth-first* tree search algorithm represents a fundamental building block in this field (Messmer & Bunke, 1999). It aims to traverse

through a graph as far as possible until either a leaf node is reached or the trail points to an already visited vertex. Then the algorithm backtracks upwards along the tree until the next vertex which allows it to follow an unvisited path downwards again.

Ullmann's Algorithm is an extension of this basic version, proposed by Ullmann in 1976. Here the computational effort is reduced by the so-called *Apriori method* which eliminates the rest of an unvisited path after a mismatch occurred. Ullmann names this *refinement process*, however, the term *pruning* is more common nowadays. Although pruning leads to an increase in efficiency, it may prevent finding isomorphic subgraphs which are located in the depth of the tree. Čibej and Mihelič (2015) proposed further improvements of Ullmann's Algorithm by means of neighbourhood filtering, a redesigned pruning process, and the application of heuristics.

Another exact matching technique is based on *clique detection*. A clique describes a subset of vertices C of graph G in which every pair of vertices in C is adjacent in G. The subgraph C represents a *complete* graph. More interesting are maximum cliques. A clique is maximized when there is no other complete subgraph in G that contains C (Bron & Kerbosch, 1973). Since the maximum cliques are insightful descriptive characteristics of a graph, it serves well as a comparison feature to determine distances. Larrosa and Valiente (2002) embedded the clique detection task into the more general definition of constraint satisfaction problems. This induced them to use heuristics as it is often the case in OR problems.

The detection of *Maximum Common Induced Subgraphs* (MCIS) also serves as a tool for GM. Looking at two graphs G and T, a MCIS describes a subgraph of G or T that is present in both of them and maximized in the number of vertices. Vismara and Valery (2008) introduce a slightly different matching approach by drawing attention to *Maximum Common Connected Induced Subgraphs* (MCCIS).

Messmer and Bunke (1999) proposed another technique of exact GM which makes use of a decision tree that is preprocessed from a graph library beforehand. This allows to match a new graph in a quadratic time regardless of the library's size. The same authors frist proposed an inexact version of this technique in 1998. *Nauty* (McKay, 1981) represents also a technique that is not based on tree search algorithms but group theory. "The algorithm deals only with the isomorphism problem, and is regarded by many authors as the fastest isomorphism algorithm available today." (Conte et al., 2004, p. 271).

Inexact GM algorithms allow to detect more rough patterns in graphs since they accept a certain level of errors in the comparison process of subgraphs. Hence they are also referred to as *error-tolerant* GM algorithms (Riesen et al.,

2010). The development of a less rigid approach was motivated by the high computational effort which is needed to find exact matches. Other than in exact GM algorithms, the edge-preservation requirement is not mandatory, however, it influences a penalty score. Inexact GM algorithms try to minimize this score (Conte et al., 2004).

One technique is based on the *graph edit distance* (GED). This distance measure, developed by Sanfeliu and Fu (1983), counts the minimal number of operations required to transform a graph G into a graph T. Inserting, deleting, or substituting vertices and edges are considered as edit operations (Riesen et al., 2010). Tsai and Fu (1979) introduced an algorithm which makes use of the GED in order to find isomorphic subgraphs by means of substituting vertices and edges. Since the other edit operations insertion and deletion are omitted in this algorithm, the candidate graphs must be structurally identical (Conte et al., 2004).

Relaxation labelling is another inexact GM technique. Here, the discrete graph problem is transformed into a statistical problem. This allows to find solutions with the help of continuous optimization. This inherits two advantages, namely a greater variety of suitable algorithms to choose from and the computational complexity is decreased to polynomial-time (Conte et al., 2004).

Another approach that allows to process GM in polynomial time is based on *spectral methods*. These methods describe a graph by means of its eigenvalue and eigenvector. When the adjacency matrices of two graphs indicate the same eigenvalues and eigenvectors, they are isomorphic (Conte et al., 2004). However, this rule does not hold for the opposite. Xu and King (2001) proposed a method which makes use of this property in combination with continuous optimization.

The application of heuristics is also a tool used in inexact GM. Suganthan (2002) recognizes patterns in an ARG by applying a *genetic algorithm* (GA). GAs are based on heuristics and inspired of Darwin's evolutionary theory. At the beginning, a set of possible solutions is randomly generated. The best solutions in this set are used to breed a new set of solutions with the hope of bequeathing the superior parts of the parent solutions to the new generation.

4 Hybrids

As shown in Section 2, the statistical approach in PR offers a well-elaborated set of tools which allows to find patterns efficiently and with good results. Unfortunately, the required data representation is often not derivable from real-world problems without losing a significant amount of information. Structural PR, described in Section 3, overcomes this limitation using a graph-based representation. This enables to incorporate the relations that data objects share among each other. However, there exists a smaller variety of tools to find patterns in

graphs. The ones that exist are limited in arithmetic operations and often computationally expensive.

This reasons the motivation behind the attempts of merging the two PR families. One of the first papers introducing this idea was published by Fu (1986). The general strategy includes the transformation of a graph into a vector space. This transformation process is also called *embedding*. Embedding in this context refers to the mapping of complex representation into a simple representation with minimized loss of information. There are two different approaches to accomplish such a projection of a graph into a vector space. One relies on *graph kernels*, the other on *graph embeddings* (Bunke & Riesen, 2012).

4.1 Graph Kernels

Graph kernels are kernel functions that allow to map or embed a graph into a higher-dimensional vector space (Bunke & Riesen, 2012). This vector space, however, remains typically unknown which is why one names this process *implicit* embedding. Formally, a graph kernel is a function K that expresses the similarity of two graphs g_i and g_j in form of scalar or dot product $K(g_i, g_j) = x \in \mathbb{R}$ such that $x = \langle \varphi(g_i), \varphi(g_j) \rangle$. The function φ is an unknown embedding function used to establish a mapping from a graph domain G into a vector space $\mathbb{R}^n$, formally $\varphi : G \to \mathbb{R}^n$. In other words, graph kernels allow to describe a structurally represented data set using a matrix of dot products (Shawe-Taylor & Cristianini, 2004). This extends the set of applicable PR tools, although the variety of vectorial operations is still limited when embedding implicitly (Conte et al., 2013).

Classification or clustering algorithms that accept such matrices as input are generally called *kernel machines*. As stated in Section 2.1, SVMs are also deemed as kernel machines with the difference that the mapping is performed from one vector space to another vector space. Thus, SVMs are not directly applicable in a graph scenario. Graph kernels, on the other hand, first compute a similarity measure in the graph space and then embed the object implicitly in a vector space [5] (Riesen, 2015). This allows to convert the graph-based PR problem from a discrete to a continuous problem which enables the application of statistical PR tools such as SVMs, IRAs, or kNN. "Hence, by means of kernel functions one can benefit from both the high representational power of graphs and the large repository of algorithmic tools available for feature vector representations of objects" (Bunke & Riesen, 2007, p. 21).

Although there are many functions to compute a similarity measure between two structurally represented objects (see Section 3.2), not every function is considered to be a kernel function. Kernel function must satisfy the condition that

[5] This new vector space is a *Hilbert* space which extends the model of a linear vector space with further properties (Muscat, 2014).

the resulting matrix is *symmetric* and *positive semi-definite* (Shawe-Taylor & Cristianini, 2004). The rest of this chapter surveys three popular classes of graph kernels which generally describe a kernel's algorithmic strategy.

Diffusion Kernels represent a class of kernels introduced by Kondor and Lafferty (2002). They allow to use any similarity measure applicable to graphs in order to construct a similarity matrix (Bunke & Riesen, 2012). Given the fact that the used similarity measure is symmetric, diffusion kernels will transform this matrix into a valid kernel matrix (Neuhaus & Bunke, 2007).

Convolution Kernels describe another class introduced by Haussler (1999). Here, the similarity measure of two graphs is calculated by decomposing the graphs into smaller parts which implies less computational complexity (Neuhaus & Bunke, 2007). "Given the similarities between the simpler parts of the underlying objects, a convolution operation is applied in order to turn them into a kernel function" Bunke and Riesen, 2012, p. 817.

Random Walk Kernels are based on the number of matching paths or walks which were randomly traced in the candidate graphs (Neuhaus & Bunke, 2007). Gärtner, Flach, and Wrobel (2003) showed that this number can be derived from two graphs using the direct product. Since this supersedes the need for performing the individual walks, random walks of arbitrary length can be used (Bunke & Riesen, 2012).

4.2 Graph Embedding

In general, graph embedding divides into two classes, namely explicit or implicit embedding (Conte et al., 2013). However, this chapter only addresses the explicit embedding since the implicit one is commonly invoked using graph kernels which was discussed in Section 4.1.

Explicit embedding describes the act of mapping a graph into a vector space using an known embedding function $\varphi : G \mapsto \mathbb{R}^n$. The structural information of a graph g_i is translated into a vector $\varphi(g_i) \mapsto (x_1, ..., x_n) \in \mathbb{R}^n$ where each dimension or axis corresponds to an explicitly defined feature (Conte et al., 2013).

A trivial example of an explicit graph embedding would be a vector space where one axis represents the number of vertices and the other one the number of edges. However, this strategy may not be very insightful in regard of discovering patterns. Riesen and Bunke (2009) show a more sophisticated approach that involves a set of comparative graphs acting as prototypes. Using the GED as a similarity measure, all candidate graphs are successively compared with these

prototypes resulting in a new vector of fixed length. Hence, each dimension of the vector space embodies the graph similarity to a specific prototype.

Other than the embedding established using graph kernels, explicit embedding invokes no limitations regarding vectorial mathematics (Fu & Ma, 2013) which reasons its attractiveness. The remaining rest of this chapter explores further explicit embedding techniques.

Topological embedding transfers a graph into a vectorial representation using a lexicon which contains prominent graph topologies (Sidère, Héroux, & Ramel, 2009). The structural patterns are described by the number of how many times certain criteria are observable in the graph. The frequency of the lexicon's topologies present in the graph would be a simple criterion. A more sophisticated criterion, for instance, would be vertices of degree two connected to vertices of degree three.

Fuzzy Multilevel Graph Embedding (FMGE) represents another technique of explicit graph embedding, introduced by Luqman, Ramel, Lladós, and Brouard (2013). Here, an ARG is described from three different perspectives, namely holistically, structurally, and elementary with the goal of minimizing information loss (Conte et al., 2013). FMGE allows to transform this information into a low-dimensional numeric vector space using fuzzy logic (Luqman et al., 2013).

Spectral embedding describes a technique in which the graph adjacency matrix is used to calculate an eigenvector in order to derive several spectral properties represented as vectors (Luo, C. Wilson, & Hancock, 2003). The authors illustrate the embedding of these vectors using principal or independent components analysis and multidimensional scaling.

Other methods include, for instance, the technique proposed by Torsello and Hancock (2007) where a set of trees is embedded by means of a super-tree. This super-tree is constructed based on this set with the goal to minimize the total edit distance. The axes in the vector space correspond to the nodes of this super-tree. Gibert, Valveny, and Bunke (2012) introduce attributed statistics based embedding which conveys a graph with continuous vertex attributes in a vector space using a set of simple subgraphs as representatives.

5 Conclusion

The data objects in statistical PR problems are described by means of a fixed number of features which is why they can directly be represented in a multi-

dimensional vector space. Applicable pattern discovery algorithms in such a setting generally divide into the two classes supervised and unsupervised. While the classification model in supervised PR is learned using a pre-labelled training data set, unsupervised algorithms perform the data segmentation using the information among the different data objects. This survey reviewed popular algorithms in both classes.

A structural data representation by means of graphs allows to capture the relational information among the individual data objects. This outperforms the statistical PR in terms of representational power but at costs of higher computational complexity and a reduced repository of algorithmic pattern discovery tools. After stating some fundamental theoretic aspects regarding graphs, two classes of graph matching algorithms were reviewed, namely exact and inexact. The latter ones tolerate certain errors in the matching process and hence are less computationally expensive.

Unification strategies of these two prior PR families are motivated by the desire to benefit from both, the representational power of graphs and the access to a larger set of efficient pattern discovery algorithms. Such hybrid solutions are addressed as a third PR family in this survey. They transfer a graph either explicitly using graph embedding or implicitly using graph kernels into a vector space.

A totally different paradigm in the field of PR is introduced by End-To-End Learning. Although only marginally discussed by this survey, the development of such deep learning algorithms gains in importance since they automate several subtasks in PR beginning with the feature detection. Hence, End-To-End Learning seems to deliver promising advantages in application areas where the majority of data is captured in unstructured binary files.

References

Aggarwal, C. C., & Zhai, C. (2012). A Survey of Text Classification Algorithms. In C. C. Aggarwal & C. Zhai (Eds.), *Mining Text Data* (pp. 163–222). doi:10.1007/978-1-4614-3223-4_6

Aghabozorgi, S., Seyed Shirkhorshidi, A., & Ying Wah, T. (2015). Time-series clustering – A decade review. *Information Systems*, *53*, 16–38. doi:10.1016/j.is.2015.04.007

Amini, A., Wah, T. Y., Saybani, M. R., & Yazdi, S. R. A. S. (2011). A study of density-grid based clustering algorithms on data streams. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)* (Vol. 3, pp. 1652–1656). Kuala Lumpur, Malaysia. doi:10.1109/FSKD.2011.6019867

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., ... Zhu, Z. (2015). *Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin*. Baidu Research – Silicon Valley AI Lab.

Äyrämö, S., & Kärkkäinen, T. (2006). *Introduction to partitioning-based clustering methods with a robust example* (tech. rep. No. C. 1/2006). University of Jyväskylä, Department of Mathematical Information Technology.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... Zieba, K. (2016). End to End Learning for Self-Driving Cars. arXiv:1604.07316.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *24*(2), 123–140. doi:10.1023/A:1018054314350

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. doi:10.1023/A:1010933404324

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Chapman and Hall/CRC.

Bron, C., & Kerbosch, J. (1973). Algorithm 457: Finding All Cliques of an Undirected Graph. *Communications of the ACM*, *16*(9), 575–577. doi:10.1145/362342.362367

Bunke, H., & Messmer, B. T. (1997). Recent Advances in Graph Matching. *International Journal of Pattern Recognition and Artificial Intelligence*, *11*(01), 169–203. doi:10.1142/S0218001497000081

Bunke, H., & Riesen, K. (2007). A Family of Novel Graph Kernels for Structural Pattern Recognition. In L. Rueda, D. Mery, & J. Kittler (Eds.), *Progress in Pattern Recognition, Image Analysis and Applications* (pp. 20–31). Lecture Notes in Computer Science. doi:10.1007/978-3-540-76725-1_3

Bunke, H., & Riesen, K. (2012). Towards the unification of structural and statistical pattern recognition. *Pattern Recognition Letters*. Special Issue on Awards from ICPR 2010, *33*(7), 811–825. doi:10.1016/j.patrec.2011.04.017

Byun, H., & Lee, S.-W. (2002). Applications of Support Vector Machines for Pattern Recognition: A Survey. In *Lecture Notes in Computer Science* (pp. 213–236). Lecture Notes in Computer Science. doi:10.1007/3-540-45665-1_17

Čibej, U., & Mihelič, J. (2015). Improvements to Ullmann's Algorithm for the Subgraph Isomorphism Problem. *International Journal of Pattern Recognition and Artificial Intelligence*, *29*(07), 1550025. doi:10.1142/S0218001415500251

Conte, D., Foggia, P., Sansone, C., & Vento, M. (2004). THIRTY YEARS OF GRAPH MATCHING IN PATTERN RECOGNITION. *International Journal of Pattern Recognition and Artificial Intelligence*, *18*(03), 265–298. doi:10.1142/S0218001404003228

Conte, D., Ramel, J.-Y., Sidère, N., Luqman, M. M., Gaüzère, B., Gibert, J., ... Vento, M. (2013). A Comparison of Explicit and Implicit Graph Embed-

ding Methods for Pattern Recognition. In W. G. Kropatsch, N. M. Artner, Y. Haxhimusa, & X. Jiang (Eds.), *Graph-Based Representations in Pattern Recognition* (pp. 81–90). Lecture Notes in Computer Science. Berlin, Heidelberg. doi:10.1007/978-3-642-38221-5_9

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification* (2nd ed). New York: Wiley.

Dudani, S. A. (1976). The Distance-Weighted k-Nearest-Neighbor Rule. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-6*(4), 325–327. doi:10.1109/TSMC.1976.5408784

Fu, K. S. (1974). *Syntactic Methods in Pattern Recognition*. Mathematics in science and engineering. Academic Press.

Fu, K. S. (1980). Recent Developments in Pattern Recognition. *IEEE Transactions on Computers*, *10*, 845–854. doi:10.1109/TC.1980.1675467

Fu, K. S. (1986). A Step Towards Unification of Syntactic and Statistical Pattern Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *3*, 398–404. doi:10.1109/TPAMI.1986.4767800

Fu, Y., & Ma, Y. (Eds.). (2013). *Graph embedding for pattern analysis*. doi:10.1007/978-1-4614-4457-2

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition* (2nd ed). Computer science and scientific computing. Boston: Academic Press.

Garey, M. R., & Johnson, D. S. (2009). *COMPUTERS AND INTRACTABILITY: A Guide to the Theory of NP-Completeness*. A series of books in the mathematical sciences. Freeman.

Gärtner, T., Flach, P., & Wrobel, S. (2003). On Graph Kernels: Hardness Results and Efficient Alternatives. In B. Schölkopf & M. K. Warmuth (Eds.), *Learning Theory and Kernel Machines* (pp. 129–143). Lecture Notes in Computer Science. Berlin, Heidelberg. doi:10.1007/978-3-540-45167-9_11

Gelfand, S. B., Ravishankar, C. S., & Delp, E. J. (1991). An Iterative Growing and Pruning Algorithm for Classification Tree Design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *13*(2), 163–174. doi:10.1109/34.67645

Gibert, J., Valveny, E., & Bunke, H. (2012). Graph embedding in vector spaces by node attribute statistics. *Pattern Recognition*. Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011), *45*(9), 3072–3083. doi:10.1016/j.patcog.2012.01.009

Gower, J. C., & Ross, G. J. S. (1969). Minimum Spanning Trees and Single Linkage Cluster Analysis. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *18*(1), 54–64. doi:10.2307/2346439

Grygorash, O., Zhou, Y., & Jorgensen, Z. (2006). Minimum Spanning Tree Based Clustering Algorithms. In *2006 18th IEEE International Conference on*

Tools with Artificial Intelligence (ICTAI'06) (pp. 73–81). doi:10.1109/ICTAI.2006.83

Haussler, D. (1999). *Convolution Kernels on Discrete Structures* (Technical Report No. UCS-CRL-99-10). University of California.

Hazim Barnouti, N., Sameer Mahmood Al-Dabbagh, S., & Esam Matti, W. (2016). Face Recognition: A Literature Review. *International Journal of Applied Information Systems*, *11*(4), 21–31. doi:10.5120/ijais2016451597

Jain, A., Duin, P., & Jianchang Mao. (2000). Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(1), 4–37. doi:10.1109/34.824819

Kassambara, A. (2017). *Practical Guide to Cluster Analysis in R: Unsupervised Machine Learning* (Edition 1). Multivariate analysis. STHDA.

Kaufman, L., & Rousseeuw, P. J. (2005). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley series in probability and mathematical statistics. Wiley.

Keller, J. M., Gray, M. R., & Givens, J. A. (1985). A Fuzzy K-Nearest Neighbor Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-15*(4), 580–585. doi:10.1109/TSMC.1985.6313426

Kondor, R. I., & Lafferty, J. D. (2002). Diffusion Kernels on Graphs and Other Discrete Input Spaces. In *Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 315–322). ICML '02. San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Koudehi, F. A., Rajeh, S. M., Farazmand, R., & Mohamad, S. (2014). A Hybrid Segmentation Approach for Customer Value. *Interdisciplinary Journal of Contemporary Research in Business*, *6*(6), 11.

Kriegel, H.-P., Kröger, P., Sander, J., & Zimek, A. (2011). Density-based clustering. *WIREs Data Mining and Knowledge Discovery*, *1*(3), 231–240. doi:10.1002/widm.30

Kruskal, J. B. (1956). ON THE SHORTEST SPANNING SUBTREE OF A GRAPH AND THE TRAVELING SALESMAN PROBLEM. *Proceedings of the American Mathematical Society*, *7*(1), 48–50. doi:10.1090/S0002-9939-1956-0078686-7

Larrosa, J., & Valiente, G. (2002). Constraint satisfaction algorithms for graph pattern matching. *Mathematical Structures in Computer Science*, *12*(4), 403–422. doi:10.1017/S0960129501003577

Luo, B., C. Wilson, R., & Hancock, E. R. (2003). Spectral embedding of graphs. *Pattern Recognition*, *36*(10), 2213–2230. doi:10.1016/S0031-3203(03)00084-0

Luqman, M. M., Ramel, J.-Y., Lladós, J., & Brouard, T. (2013). Fuzzy multilevel graph embedding. *Pattern Recognition*, *46*(2), 551–565. doi:10.1016/j.patcog.2012.07.029

Madhulatha, T. S. (2012). An Overview on Clustering Methods. *IOSR Journal of Engineering*, *2*(4), 719–725.

Mann, A. K., & Kaur, N. (2013). Review Paper on Clustering Techniques. *Global Journal of Computer Science and Technology*, *13*(5), 42–48.

McKay, B. D. (1981). *Practical Graph Isomorphism*. Vanderbilt University.

Messmer, B. T., & Bunke, H. (1998). A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* *20*(5), 493–504. doi:10.1109/34.682179

Messmer, B. T., & Bunke, H. (1999). A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, *32*(12), 1979–1998. doi:10.1016/S0031-3203(98)90142-X

Muscat, J. (2014). *Functional Analysis: An Introduction to Metric Spaces, Hilbert Spaces, and Banach Algebras*. doi:10.1007/978-3-319-06728-5

Na, S., Xumin, L., & Yong, G. (2010). Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm. In *2010 Third International Symposium on Intelligent Information Technology and Security Informatics* (pp. 63–67). doi:10.1109/IITSI.2010.74

Neuhaus, M., & Bunke, H. (2007). *BRIDGING THE GAP BETWEEN GRAPH EDIT DISTANCE AND KERNEL MACHINES*. Series in Machine Perception and Artificial Intelligence. OCLC: ocn166388136. World Scientific Publishing Co. Pte. Ltd.

Newman, M. (2018). *Networks* (Second edition). Oxford University Press.

Ng, R. T., & Han, J. (2002). CLARANS: A Method for Clustering Objects for Spatial Data Mining. *IEEE Transactions on Knowledge and Data Engineering*, *14*(5), 1003–1016. doi:10.1109/TKDE.2002.1033770

Ozdemir, E., & Gunduz-Demir, C. (2013). A Hybrid Classification Model for Digital Pathology Using Structural and Statistical Pattern Recognition. *IEEE Transactions on Medical Imaging*, *32*(2), 474–483. doi:10.1109/TMI.2012.2230186

Päivinen, N. (2005). Clustering with a minimum spanning tree of scale-free-like structure. *Pattern Recognition Letters*, *26*(7), 921–930. doi:10.1016/j.patrec.2004.09.039

Pavlidis, T. (1977). *Structural Pattern Recognition*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Prim, R. C. (1957). Shortest Connection Networks And Some Generalizations. *The Bell System Technical Journal*, *36*(6), 1389–1401. doi:10.1002/j.1538-7305.1957.tb01515.x

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. doi:10.1007/BF00116251

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. The Morgan Kaufmann series in machine learning. San Mateo, Califorina: Morgan Kaufmann Publishers.

Riesen, K. (2015). *Structural Pattern Recognition with Graph Edit Distance: Approximation Algorithms and Applications* (Springer International Publishing AG, Ed.). doi:10.1007/978-3-319-27252-8

Riesen, K., & Bunke, H. (2009). GRAPH CLASSIFICATION BASED ON VECTOR SPACE EMBEDDING. *International Journal of Pattern Recognition and Artificial Intelligence*, *23*(6), 1053–1081. doi:10.1142/S021800140900748X

Riesen, K., Jiang, X., & Bunke, H. (2010). Exact and Inexact Graph Matching: Methodology and Applications. In C. C. Aggarwal & H. Wang (Eds.), *Managing and Mining Graph Data* (Vol. 40, pp. 217–247). doi:10.1007/978-1-4419-6045-0_7

Rosen, K. H. (2019). *Discrete Mathematics and Its Applications* (Eighth edition). McGraw-Hill Education.

Sanfeliu, A., & Fu, K. S. (1983). A Distance Measure Between Attributed Relational Graphs for Pattern Recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-13*(3), 353–362. doi:10.1109/TSMC.1983.6313167

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. doi:10.1017/CBO9780511809682

Sidère, N., Héroux, P., & Ramel, J.-Y. (2009). Vector Representation of Graphs: Application to the Classification of Symbols and Letters. In *2009 10th International Conference on Document Analysis and Recognition* (pp. 681–685). ISSN: 2379-2140. Barcelona, Spain. doi:10.1109/ICDAR.2009.218

Suganthan, P. N. (2002). Structural pattern recognition using genetic algorithms. *Pattern Recognition*, *35*(9), 1883–1893. doi:10.1016/S0031-3203(01)00136-4

Taha, H. A. (2017). *Operations Research: An Introduction* (10th ed.). OCLC: 1085404371. Harlow, England: Pearson Education Limited.

Theodoridis, S., & Koutroumbas, K. (2009). *Pattern Recognition* (4th ed.). Elsevier Academic Press.

Torsello, A., & Hancock, E. R. (2007). Graph embedding using tree edit-union. *Pattern Recognition*, *40*(5), 1393–1405. doi:10.1016/j.patcog.2006.09.006

Trafalis, T., & Oladunni, O. (2005). Pairwise Multi-classification Support Vector Machines: Quadratic Programming (QP-PAMSVM) formulations. In *6th WSEAS International Conference on Neural Networks* (pp. 205–210). Lisbon, Portugal.

Tsai, W.-H., & Fu, K. S. (1979). Error-Correcting Isomorphisms of Attributed Relational Graphs for Pattern Analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, *9*(12), 757–768. doi:10.1109/TSMC.1979.4310127

Turkson, R. E., Baagyere, E. Y., & Wenya, G. E. (2016). A Machine Learning Approach for Predicting Bank Credit Worthiness. In *2016 Third International Conference on Artificial Intelligence and Pattern Recognition (AIPR)* (pp. 81–87). Lodz, Poland. doi:10.1109/ICAIPR.2016.7585216

Ullmann, J. R. (1976). An Algorithm for Subgraph Isomorphism. *Journal of the ACM*, *23*(1), 31–42. doi:10.1145/321921.321925

Vapnik, V. N. (1982). Estimation of Dependences Based on Empirical Data. New York: Springer-Verlag.

Vismara, P., & Valery, B. (2008). Finding Maximum Common Connected Subgraphs Using Clique Detection or Constraint Satisfaction Algorithms. In *Modelling, Computation and Optimization in Information Systems and Management Sciences* (pp. 358–368). doi:10.1007/978-3-540-87477-5_39

Wang, X., Wang, X., & Mitchell Wilkes, D. (2009). A Divide-and-Conquer Approach for Minimum Spanning Tree-Based Clustering. *IEEE Transactions on Knowledge and Data Engineering*, *21*(7), 945–958. doi:10.1109/TKDE.2009.37

Webb, A. R., & Copsey, K. D. (2011). *Statistical Pattern Recognition* (3rd ed.). Hoboken, NJ, USA: Wiley.

Xu, L., & King, I. (2001). A PCA Approach for Fast Retrieval of Structural Patterns in Attributed Graphs. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *31*(5), 812–817. doi:10.1109/3477.956043

Zhang, B. (2000). *Generalized K-Harmonic Means – Boosting in Unsupervised Learning* (tech. rep. No. HPL-2000-137). Hewlett Packard.

Zhang, H. (2004). The Optimality of Naïve Bayes. In *FLAIRS Conference 2004* (pp. 562–567). Florida, USA. AAAI Press.

Zhao, B., Liu, Y., & Xia, S.-W. (2000). Support Vector Machine and its Application in Handwritten Numeral Recognition. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000* (Vol. 2, pp. 720–723). ISSN: 1051-4651. doi:10.1109/ICPR.2000.906176