Classifier Ensembles for Vector Space Embedding of Graphs

Kaspar Riesen and Horst Bunke

Department of Computer Science, University of Bern,
Neubrückstrasse 10, CH-3012 Bern, Switzerland
{riesen,bunke}@iam.unibe.ch

Abstract. Classifier ensembles aim at a more accurate classification than single classifiers. Different approaches to building classifier ensembles have been proposed in the statistical pattern recognition literature. However, in structural pattern recognition, classifier ensembles have been rarely used. In this paper we introduce a general methodology for creating structural classifier ensembles. Our representation formalism is based on graphs and includes strings and trees as special cases. In the proposed approach we make use of graph embedding in real vector spaces by means of prototype selection. Since we use randomized prototype selection, it is possible to generate n different vector sets out of the same underlying graph set. Thus, one can train an individual base classifier for each vector set und combine the results of the classifiers in an appropriate way. We use extended support vector machines for classification and combine them by means of three different methods. In experiments on semi-artificial and real data we show that it is possible to outperform the classification accuracy obtained by single classifier systems in the original graph domain as well as in the embedding vector spaces.

1 Introduction

The key idea in multiple classifier systems is to combine several classifiers such that the resulting combined system achieves a higher classification accuracy than the original classifiers individually [1]. In the case of statistical patterns, that is, patterns represented by feature vectors, a large number of methods for the creation and combination of classifiers have been developed over the past few years. Bagging, for instance, creates classifiers by randomly selecting the set of training examples to be used for each classifier [2]. A similar idea is that of random feature subset selection [3]. In this method, one randomly selects the features (dimensions) to be used for each feature vector to create a group of classifiers. A third prominent example of classifier creation methods is boosting, where classifiers are created sequentially out of a single base classifier by giving successively higher weights to those training samples that have been misclassified [4].

Structural pattern recognition is characterized by the use of symbolic data structures, such as strings, trees, or graphs, for pattern representation. Such representations have a number of advantages over feature vectors used in the statistical

M. Haindl, J. Kittler, and F. Roli (Eds.): MCS 2007, LNCS 4472, pp. 220–230, 2007.

approach. For example, a string may consist of an arbitrary number of symbols. This is in contrast to a feature vector where one is confined to always using the same number of features, regardless of the size or the complexity of the individual patterns to be represented. Furthermore, if one uses graphs, structural relationships between individual pattern components can be conveniently represented. There is no direct way to represent such relations in a feature vector. In fact, there are many applications in pattern recognition and related areas, including computational biology and chemistry, where such representations are essential [5].

One disadvantage of the structural approach is that the space of strings, trees or graphs has very little mathematical structure. This means that elementary operations, such as computing the average, the covariance, or the product of two object representations, do not exist. Therefore, up to a few exceptions [6,7,8], mostly classifiers of the nearest neighbor type have been applied to structural pattern representations. Consequently, there has been little work on multiple classifier systems based on structural pattern representations. A pioneering paper is [9] where it was shown that by the use of statistical and structural classifiers in a multiple classifier system the accuracy of a fingerprint recognition system can be improved. In [10] an approach has been proposed where several graph representations of the same pattern are derived and merged into a single representation format. Then the single graph resulting from the merging operation is input to a nearest neighbor classifier based on graph edit distance. In [11], random node selection on graphs has been used in order to derive ensembles of graph-based classifiers. But still, only classifiers of the nearest neighbor type are applied in this work.

In the current paper we propose a new method that is based on two fundamental ideas. The first idea is the embedding of graphs into the n-dimensional real space by means of prototype selection and edit distance computation. Such a procedure has been originally proposed in [12] in order to embed feature vectors in a dissimilarity space. In subsequent work a similar procedure has been used for the embedding of strings and graphs [13,14]. By means of this procedure any set of graphs can be mapped to a set of feature vectors. Consequently, any pattern recognition method that has ever been developed for feature vector representations becomes applicable to graphs. The second fundamental idea in this paper is based on the observation that mapping a population of graphs into a vector space is controlled by a set of prototypes. One possible procedure to actually get these prototypes is by random selection from the given training set of graph. Obviously, if we repeat the process of random selection a number of times, we can derive different graph sets that all can be used in order to train a classifier. As a result, we get a classifier ensemble for structural input data. The classifier we adopted for the work described in this paper is Support Vector Machine (SVM). However, any other type of classifier can be used as well.

2 Graph Embedding in Real Vector Spaces

In [15] an approach to graph embedding in vector spaces has been introduced. This method is based on algebraic graph theory and utilizes spectral matrix

decomposition. Another approach for graph embedding has been proposed in [16]. It makes use of the relationship between the Laplace-Beltrami operator and the graph Laplacian to embed a graph onto a Riemannian manifold. Our embedding method makes explicitly use of graph edit distance [17,18]. The key idea of graph edit distance is to define the dissimilarity, or distance, of graphs by the amount of distortion that is needed to transform one graph into another. A sequence of edit operations that transforms a graph g_1 into another graph g_2 is called an *edit path* between g_1 and g_2. Costs are assigned to each edit path, representing the strength of the distortions of this edit sequence. Consequently, the *edit distance* of two graphs is defined as the minimum cost, taken over all edit paths between two graphs under consideration. Typically, the edit distance is used to classify an input graph by computing its distance to a number of training graphs and feeding the resulting distance values into a nearest-neighbor classifier. In our approach we make use of edit distances to construct a vectorial description of a given graph. Our method was originally developed for the problem of embedding sets of feature vectors in a dissimilarity space [12,19]. The embedding of strings and graphs has been studied in [13,14]. Assume we have a labeled set of training graphs, $T = \{g_1, \ldots, g_t\}$, and a dissimilarity measure $d(g_i, g_j)$. After having selected a set $P = \{p_1, \ldots, p_m\}$ of $m < t$ prototypes from T, i.e. $P \subseteq T$, we compute the dissimilarity of a graph $g \in T$ to each prototype $p \in P$. This leads to m dissimilarities, $d_1 = d(g, p_1), \ldots, d_m = d(g, p_m)$, which can be interpreted as an m-dimensional vector $(d_1, \ldots, d_m)$. In this way we can transform any graph from the training set, as well as any other graph from a validation or testing set, into a vector of real numbers. Note that whenever a graph from the training set, which has been choosen as a prototype before, is transformed into a vector $\mathbf{x} = (x_1, \ldots, x_m)$ one of the vector components is zero.

Different methods for selecting the m prototypes needed for embedding have been proposed in the literature [12,13,14]. The intention of all methods is the same, that is, finding a selection of m prototypes that lead to a good performance

Algorithm 1. Generating n prototype sets out of one graph set.

Input: Training graphs $T = \{g_1, \ldots, g_t\}$, number of required prototype sets n, and dimensionality of the resulting feature vectors m
Output: Set $PROTO$ consisting of n different prototype sets of size m each

```
initialize TABU to the empty set {}
initialize PROTO to the empty set {}
for i = {1, ..., n} do
    P_i = {}
    for j = {1, ..., m} do
        if |TABU| == t then
            reset TABU to the empty set {}
        else
            select p randomly out of T \ TABU
            P_i = P_i ∪ {p}
            TABU = TABU ∪ {p}
        end if
    end for
    PROTO = PROTO ∪ {P_i}
end for
return PROTO
```

of the resulting classifier in the vector space. Since in this paper we want to generate not only one, but a number of vector sets out of the same graph set, we make use of the random method described in Algorithm 1. Output of this procedure is a set consisting of n different prototype sets of size m each. To build such a set, the method described in Alg. 1 randomly picks n times m graphs from the training set T. After picking a graph from T, the selected graph becomes temporarily unavailable for further selection. Once all training graphs from T have been selected, all training graphs become available again. This procedure is very interesting in that it naturally lends itself to a method for the automatic generation of classifier ensembles.

3 Extended Support Vector Machine

In Sect. 2 we have introduced a general methodology for embedding graphs in real vector spaces. Clearly, one can build arbitrarily many different vector sets out of the same graph set. Assume a given graph set has been embedded n times with different prototypes. Hence, we have n different vector sets available all representing the same graphs. Obviously, it is possible to train a classifier on each vector set seperately. Therefore, we obtain n classifers $L_1, \ldots, L_n$ whose results can be combined to one output. In the following we assume that we deal with a problem involving k different classes $\{C_1, \ldots, C_k\}$. As any classifier combination method necessarily depends on the type of the n underlying classifiers, we distinguish three types of classifiers:

- **Type-1 classifiers:** Output of these classifiers is exactly one class C_i.
- **Type-2 classifers:** Output is a ranking list, i.e. an ordered list $(C_{i1}, \ldots, C_{ik})$ including all classes, where C_{i1} is the class with the highest and C_{ik} the class with the lowest plausibility.
- **Type-3 classifiers:** Output is a plausibility value $p(C_i)$ for each class C_i. This plausibility value corresponds with the probability that a test element under consideration belongs to the respective class. Thus, each classifer L_j outputs a vector of size k, $\{(p_j(C_1), \ldots, p_j(C_k))\}_{1 \leqslant j \leqslant n}$.

Pattern classification by means of support vector machines (SVMs) has become very popular recently [20,21]. The basic idea of SVM is to seperate classes of patterns by hyperplanes. Intuitively, one would choose a hyperplane such that its distance to the closest pattern of either class is maximal. Such hyperplanes are expected to perform best on an independent test set. SVMs are able to find such optimal hyperplanes. Originally, SVMs have been developed to handle two class problems. To generalize SVM to problems with more than two classes one can use the *1-to-1* method. In this approach all pairs of classes $(C_i, C_j)_{1 \leqslant i < j \leqslant k}$ are considered seperately, and for each pair an individual SVM is trained. This leads to $k(k-1)/2$ different SVMs. An unseen test element is assigned to the class C_i that occurs the most frequently among the $k(k-1)/2$ SVM decisions. Output of a traditional SVM is one class and thus SVMs are typically type-1 classifiers. Since we want to use not only combiners depending on type-1 but also on

type-2 and type-3 classifers, one has to generalize SVM appropriately. The first generalization, which leads to a type-2 classifier, is simple and straightforward. Instead of returning only the most frequent class C_i among the $k(k-1)/2$ SVM decisions, one can extend the i-th SVM to return an ordered list $(C_{i1}, \ldots, C_{ik})$. C_{i1} stands for the most frequent class and C_{ik} represents the class that has won the fewest SVM decisions. To get a type-3 classifier out of a standard SVM one can use the information about the distance f of a test sample to the hyperplane of the current SVM. These distances f are used to obtain pairwise class probabilities by feeding them into a *sigmoid* function:

$$r_{ij} = \frac{1}{1+exp(\alpha f+\beta)},$$

where α and β are parameters that have to be estimated. To obtain one probability $p(C_i)$ per class out of the r_{ij} values one has to to solve an optimization problem. The probabilistic SVM resulting from this procedure is described in more detail in [22,23].

4 Classifier Ensembles

Let us summarize the whole procedure discussed so far. Starting point are patterns given by graph-based representations. With random prototype selection and graph edit distance computation, we embed these graphs in real vector spaces. Since we use randomized prototype selection, this step leads to n different vectorial descriptions of the same graph set. Based on these n vector sets one can train n different SVMs. Hence, from an unknown test pattern, we get n different classification results. Output of i-th SVM is either a single class C_{i1} (type-1 classifier), a vector with all possible classes $(C_{i1}, \ldots, C_{ik})$ ordered by the frequency of all SVM decisions (type-2 classifier), or a list with plausibility values $(p_j(C_1), \ldots, p_j(C_k))$, where $p_j(C_i)$ is derived from the distances of the test elements to the hyperplanes of the individual SVMs (type-3 classifier). Based on these three different output formats of the n SVMs, one can use different combination strategies to obtain the final result. In this work we use a *Voting* algorithm for type-1 SVMs, a ranking sum method for type-2 SVMs (*Borda count*) and *Bayes' combination* using the plausibility values obtained by type-3 SVMs.

- **Voting:** The class C_{i1} output by classifier L_i $(1 \leqslant i \leqslant n)$ is regarded as one vote for $C_{i1} \in \{C_1, \ldots, C_k\}$. The class that receives the plurality of the votes is choosen by the combiner. This method is often termed *plurality voting* [1]. Of course, one can use more restrictive voting methods with rejection (e.g. *majority voting* [1]).
- **Borda Count:** Assume that each classifier L_i outputs an ordered list including all classes $\{C_j\}_{1 \leqslant j \leqslant k}$. To combine the results of type-2 classifiers one can introduce rank functions $r_i(C_{ij})$ for each classifer L_i. Function $r_i(C_{ij})$ delivers the position of the class C_{ij} in the ordered list given by classifier L_i, i.e. $r_i(C_{ij}) = j$. Hence, for each class $\{C_i\}_{1 \leqslant i \leqslant k}$ the sum of all ranks can

be computed, $R(C_i) = \sum_{j=1}^{n} r_j(C_i)$. Subsequently, the combiner chooses the class $\{C_i\}_{1 \leqslant i \leqslant k}$ with the minimum value of $R(C_i)$. This combination method is known as *Borda count*.

– **Bayes' Combination:** In this approach the individual plausibility values $\{p_j(C_i)\}_{1 \leqslant j \leqslant n}$ are combined to get one plausibility value P_i per C_i. Common strategies to combine the plausibility values are given below [24]:
  - $P_i = \max(p_1(C_i), \ldots, p_n(C_i))$
  - $P_i = \min(p_1(C_i), \ldots, p_n(C_i))$
  - $P_i = \frac{1}{n} \sum_{j=1}^{n} p_j(C_i)$
  - $P_i = \prod_{j=1}^{n} p_j(C_i)$

  Regardless which of these formulas is used, the ensemble eventually chooses the class C_i with the maximum value of the corresponding P_i. In the present paper we use the last approach based on the product, which is known as *Bayes' combination*.

A crucial question is how many classifiers should be included in an ensemble. With Alg. 1 we have the possibility to build n vector sets, and thus we have n classifiers available. To determine the size of the final ensemble we propose a sequential floating search selection according to Algorithm 2 [25]. First the best individual classifier in terms of classification accuracy is added to the ensemble in line 2. Then, the best fitting classifier, i.e. the classifier that complements the ensemble generated so far the best, is added incrementally (line 5 and 6). After each forward step a number of backward steps are applied as long as the resulting subsets are better than the evaluated ones at that level (line 11 and 12). Obviously, this procedure generates n subsets of the classfier set $L = \{L_1, \ldots, L_n\}$ with size 1 to n. The best performing subset, i.e. the ensemble E_i with the lowest classification error on an independent validation set is used as the final ensemble (line 19). This strategy is also known as *overproduce-and-select* [1].

5 Experimental Results

The purpose of the experiments described in this section is to compare the classification accuracy of the ensembles obtained by the proposed method with two reference systems. The first reference system is a traditional nearest-neighbor classifier in the graph domain, while a single SVM in the vector domain is used as the second reference system. The first reference system, the nearest-neighbor classifier, has proved to be suitable for the classification task in graph domains for many different applications. Basically, this classifier assigns the label of the nearest neighbor in a training set in terms of edit distance to an unknown test-element. Note that as of today – up to few exceptions, e.g. [6] – there exist no other classifiers for general graphs that can be directly applied in the graph domain. The second reference system is obtained through picking the best individual classfier L_i out of L, i.e. the classifier that leads to the best classification accuracy on the validation set. In each of our experiments we make use of three

Algorithm 2. Determine the best performing classifier ensemble.

Input: A set of n classifiers $L = \{L_1, \ldots, L_n\}$ sorted in order of their individual classification accuracy. (L_1 has the highest and L_n the lowest classification accuracy)
Output: The best performing classifier ensemble E_{max}

```
1: Initialize n empty ensembles E = {E_1, ..., E_n}
2: add the best individual classifier to the ensemble: E_1 = {L_1}
3: intitalize k := 1
4: while k < n do
5:    L^+ = argmax_{L_i ∈ L \ E_k} accuracy(E_k ∪ {L_i})
6:    add the classifier L^+ to the ensemble: E_{k+1} = E_k ∪ {L^+}
7:    k := k + 1
8:    initialize removed := false
9:    repeat
10:      removed := false
11:      L^- = argmax_{L_i ∈ E_k} accuracy(E_k \ {L_i})
12:      if accuracy(E_k \ {L_i}) > accuracy(E_{k-1}) then
13:         E_{k-1} = E_k \ {L_i}
14:         k := k - 1
15:         removed = true
16:      end if
17:   until removed = false
18: end while
19: return the best performing ensemble E_max = argmax_{E_i ∈ E} accuracy(E_i)
```

disjoint graph sets, the *validation set*, the *test set* and the *training set*. The validation set is used to determine optimal parameter values for multiple graph embeddings and classification. The embedding parameters to be validated consist of the number of prototypes, i.e. the dimensionality of the resulting feature vector spaces, and the best performing ensemble, i.e. the best combination of ensemble members in terms of classification accuracy (see Alg. 1 and Alg. 2). Parameters for classification consist of different parameters for the SVM and depend on the kernel function [21,26]. The parameter values and the ensemble that result in the lowest classification error on the validation set are then applied to the independent test set.

5.1 Letter Database

The first database used in the experiments consists of graphs representing distorted letter drawings. In this experiment we consider the 15 capital letters of the Roman alphabet that consists of straight lines only (*A, E, F, ...)*. For each class, a prototype line drawing is manually constructed. To obtain aribtrarily large sample sets of drawings with arbitrarily strong distortions, distortion operators are applied to the prototype line drawings. This results in randomly shifted, removed, and added lines. These drawings are then converted into graphs in a simple manner by representing lines by edges and ending points of lines by nodes. Each node is labeled with a two-dimensional attribute giving its position. The graph database used in our experiments consists of a training set, a validation set, and a test set, each of size 750 for each of a total of five different distortion levels. The results of the experiments on the letter database are given in Table 1. First of all, the single SVM classifier in the vector domain improves the

classification accuracy compared to the reference system in the graph domain on all distortion levels. Note that two out of five improvements are statistically significant. Similar results have been reported in [14]. Despite these good results, the ensemble methods obtains further improvements. Especially on distortion levels 0.3, 0.5, 0.7 and 0.9, the ensemble methods achieve better results. Note that 8 out of 12 improvements compared to the single SVM are statistically significant. Compared to the first reference system, there are even 11 statistically significantly improvements. Only at distortion level 0.1 the single SVM is superior to the voting and Borda count method. The optimal parameters for this experiment, found on the validation set, are 150 prototypes per embedding and 12 ensemble members on average for voting, 13 ensemble members on average for Borda count and an ensemble of size 8 on average for Bayes' combiner.

Table 1. Letter Database: Classification accuracy in the graph and vector space

	Ref. Systems		Classifier Ensembles		
Distortion	k-NN (graph)	single SVM	Plurality Voting	Borda Count	Bayes' Combiner
0.1	98.27	98.53	98.27	98.13	98.67
0.3	97.60	98.00	98.27	98.53 ◎	98.67 ◎
0.5	94.00	96.53 ○	97.07 ○	96.93 ○	97.07 ○
0.7	94.27	94.53	96.00 ◎	95.87 ◎	96.00 ◎
0.9	90.13	93.33 ○	94.27 ◎	94.40 ◎	94.53 ◎

○ Statistically significant improvement over the first reference system (Z-test, $\alpha = 0.05$)
◎ Statistically significant improvement over both reference systems (Z-test, $\alpha = 0.05$)

5.2 Real World Data

For a more thorough evaluation of the proposed methods we additonally use three real world data sets. First we apply the proposed method to the problem of image classification. Images are converted in graphs by segmenting them into regions, eliminating regions that are irrelevant for classification, and representing the remaining regions by nodes and the adjacency of regions by edges [27]. The Le Saux image database consists of five classes (*city, countryside, people, streets, snowy*) and is split into a training set, a validation set and a test set of size 54 each. The classification accuracies obtained by the different methods are given in the first row of Table 2. Although the single SVM improves the accuracy by 5.6%, this improvement is not statistically significant. The further improved result achieved by the Borda count method – which actually corresponds to an improvement by 7.4% – obtains no statistical significance, either. Neither are the superior results of the reference systems compared to the other ensemble methods statistically significant. All this is due to the small size of the Le Saux database. We used 20 prototypes for embedding and 27 (voting), 7 (Borda count) and 31 (Bayes' combiner) ensemble members for classification. The second real world dataset is given by the NIST-4 fingerprint database [28]. We construct graphs from fingerprint images by extracting characteristic regions in fingerprints and converting the results into attributed graphs [29]. We use a validation set of size 300 and a test and training set of size 500 each. In this experiment we address

Table 2. Fingerprint-, Image- and Molecules Database: Classification accuracy in the graph and vector space

	Ref. Systems		Classifier Ensembles		
Database	k-NN (graph)	single SVM	Plurality Voting	Borda Count	Bayes' Combiner
Le Saux	57.4	63.0	61.1	64.8	55.6
NIST-4	82.6	84.8 ∘	85.2 ∘	85.0 ∘	84.8 ∘
Molecules	97.1	98.1 ∘	98.3 ◎	98.3 ◎	97.7

∘ Statistically significant improvement over the first reference system (Z-test, $\alpha = 0.05$)
◎ Statistically significant improvement over both reference systems (Z-test, $\alpha = 0.05$)

the 4-class problem (*arch, left-loop, right-loop, whorl*). The single SVM and all ensemble methods achieve statistically significantly better results than the first reference system. However, there is no significant difference between the single SVM and the proposed classifier ensembles. Nevertheless, note that the ensemble obtains two improvements and one equal result compared to the second reference system. On this data set a configuration with 100 prototypes for embedding and 11 (voting and Borda count) and 2 (Bayes' combiner) ensemble members obtains the best result on the validation set and is therefore used on the test set.

Finally, we apply the proposed method of graph embedding and subsequent SVM classification to the problem of molecule classification. To this end, we construct graphs from the AIDS Antiviral Screen Database of Active Compounds [30]. Our molecule database consists of two classes (*active, inactive*), which represent molecules with activity against HIV or not. We use a validation set of size 250, a test set of size 1500 and training set of size 250. Thus, there are 2000 elements totally (1600 inactive elements and 400 active elements). The molecules are converted into graphs in a straightforward manner by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled with the corresponding chemical symbol and edges by the valence of the linkage. Although the accuracy of the reference system in the graph domain is quite high, it can be statistically significantly improved by the single SVM. The voting and the Borda count methods outperform the reference system in the graph domain, too. Actually, the good result achieved in vector domain by a single SVM can be further improved by these ensemble methods with statistical significance. We used 150 prototypes for embedding, and 14 (voting and Borda count) and 23 (Bayes' combiner) ensemble members for classification.

6 Conclusions

While many methods for building classifier ensembles based on feature vector representations of the underlying data have been proposed, little work has been done for structural representations. In this paper we propose a general approach to graph based classifier ensembles. Our approach makes use of graph embedding in real vector spaces. The key idea is to map graphs to the m-dimensional real space by means of graph edit distance and prototype selection. To this end, we discuss a randomized prototype selector with the objective of finding n different prototype sets. With these sets, one can map a set of graphs n times to different

vector sets, such that we obtain n different vector sets all representing the same graph set, i.e. the same pattern elements. For each vector set an individual SVM is trained and thus one gets n different classifiers. Hence, a number of methods become available for combining the results of individual ensemble members. The proposed methods were tested on a number of graph datasets with different characteristics, comming from various application domains. From the results of our experiments, one can conclude that the classification accuracy can be enhanced by most ensemble methods on almost all data sets.

Acknowledgements

This work has been supported by the Swiss National Science Foundation (Project 200021-113198/1). Furthermore, we would like to thank R. Duin and E. Pekalska for valuable discussions and hints regarding the embedding method. Finally, we thank B. Le Saux for making the Le Saux database available to us.

References

1. L. Kuncheva. Combining Pattern Classifiers: Methods and Algorithms. John Wiley, 2004.
2. L. Breiman. Bagging predictors. Machine Learning, 24:123–140, 1996.
3. T.K. Ho. The random subspace method for constructing decision forests. IEEE Trans. on Pattern Analysis ans Machine Intelligence, 20(8):832–844, 1998.
4. Y. Freund and R.E. Shapire. A decision theoretic generalization of online learning and application to boosting. Journal of Computer and Systems Sciences, 55:119–139, 1997.
5. D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. Int. Journal of Pattern Recognition and Artificial Intelligence, 18(3):265–298, 2004.
6. M. Bianchini, M. Gori, L. Sarti, and F. Scarselli. Recursive processing of cyclic graphs. IEEE Transactions on Neural Networks, 17(1):10–18, January 2006.
7. M. Neuhaus and H. Bunke. Edit distance based kernel functions for structural pattern classification. In Pattern Recognition, pages 1852 – 1863, 2006.
8. T. Gärtner, J. Lloyd, and P. Flach. Kernels and distances for structured data. Machine Learning, 57(3):205–232, 2004.
9. G.L. Marcialis, F. Roli, and A. Serrau. Fusion of statistical and structural fingerprint classifiers. In J. Kittler and M.S. Nixon, editors, 4th Int. Conf. Audio- and Video-Based Biometric Person Authentication, LNCS 2688, pages 310–317. Springer, 2003.
10. M. Neuhaus and H. Bunke. Graph-based multiple classifier systems – a data level fusion approach. Lecture Notes in Computer Science, 3617:479–487, 2005.
11. A. Schenker, H. Bunke, M. Last, and A. Kandel. Building graph-based classifier ensembles by random node selection. In F. Roli, J. Kittler, and T. Windeatt, editors, Proc. 5th Int. Workshop on Multiple Classifier Systems, LNCS 3077, pages 214–222. Springer, 2004.
12. E. Pekalska, R. Duin, and P. Paclik. Prototype selection for dissimilarity-based classifiers. Pattern Recognition, 39(2):189–208, 2006.

13. B. Spillmann, M. Neuhaus, H. Bunke, E. Pekalska, and R. Duin. Transforming strings to vector spaces using prototype selection. In Proc. 11.th int. Workshop on Strucural and Syntactic Pattern Recognition, LNCS 4109, pages 287–296. Springer, 2006.
14. K. Riesen, M. Neuhaus, and H. Bunke. Graph embedding in vector spaces by means of prototype selection. Submitted.
15. R.C. Wilson, E.R. Hancock, and B. Luo. Pattern vectors from algebraic graph theory. IEEE Trans. on Pattern Analysis ans Machine Intelligence, 27(7):1112–1124, 2005.
16. A. Robles-Kelly and E.R. Hancock. A riemannian approach to graph embedding. Pattern Recognition, 40:1024–1056, 2007.
17. A. Sanfeliu and K.S. Fu. A distance measure between attributed relational graphs for pattern recognition. IEEE Transactions on Systems, Man, and Cybernetics (Part B), 13(3):353–363, 1983.
18. H. Bunke and G. Allermann. Inexact graph matching for structural pattern recognition. Pattern Recognition Letters, 1:245–253, 1983.
19. R. Duin and E. Pekalska. The Dissimilarity Representations for Pattern Recognition: Foundations and Applications. World Scientific, 2005.
20. C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):121–167, 1998.
21. J. Shawe-Taylor and N. Cristianini. Kernel Methods for Pattern Analysis. Cambridge University Press, 2004.
22. C.C. Chang and C.J. Lin. LIBSVM: A Library for Support Vector Machines, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
23. C.F.J. Wu, C.J. Lin, and R.C. Weng. Probability estimates for multi-class classification by pairwise coupling. Journal of Machine Learning Research, 5:975–1005, 2004.
24. J. Kittler, M. Hatef, R. Duin, and J. Matas. On combining classifiers. IEEE Trans. on Pattern Analysis ans Machine Intelligence, 20(3):226–239, 1998.
25. P. Pudil, J. Novovicova, and J. Kittler. Floating search methods in feature-selection. PRL, 15(11):1119–1125, November 1994.
26. B. Schölkopf and A. Smola. Learning with Kernels. MIT Press, 2002.
27. B. Le Saux and H. Bunke. Feature selection for graph-based image classifiers. In Proc. 2nd Iberian Conf. on Pattern Recognition and Image Analysis, LNCS 3523, pages 147–154. Springer, 2005.
28. C.I. Watson and C.L. Wilson. NIST special database 4, fingerprint database. National Institute of Standards and Technology, March 1992.
29. M. Neuhaus and H. Bunke. A graph matching based approach to fingerprint classification using directional variance. In Proc. 5th Int. Conf. on Audio- and Video-Based Biometric Person Authentication, LNCS 3546, pages 191–200. Springer, 2005.
30. Development Therapeutics Program DTP. Aids antiviral screen, 2004. `http://dtp.nci.nih.gov/docs/aids/aids_data.html`.