Graph-Based Multiple Classifier Systems A Data Level Fusion Approach*

Michel Neuhaus and Horst Bunke

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

Abstract. The combination of multiple classifiers has been successful in improving classification accuracy in many pattern recognition problems. For graph matching, the fusion of classifiers is normally restricted to the decision level. In this paper we propose a novel fusion method for graph patterns. Our method detects common parts in graphs in an error-tolerant way using graph edit distance and constructs graphs representing the common parts only. In experiments, we demonstrate on two datasets that the method is able to improve the classification of graphs.

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 fusion of classifiers have been developed over the past few years, ranging from the combination of individual classification results to the fusion of feature vectors. However, for structural patterns, and attributed graph patterns in particular, the fusion of classifiers has mainly been constrained to the decision level [2,3,4], that is, to the combination of the individual classifiers' results. The integration of knowledge about the structure of graphs into a multiple classifier system has not been considered until now.

In the present paper we propose a method to merge several graphs representing the same underlying pattern into a graph that is more robust against noise and hence a better structural representative of the pattern. We proceed by identifying common parts in two or more graphs and derive a graph representing the common structure. The structural matching is performed by means of graph edit distance computation. The graph fusion method can be applied when several graph representations of a pattern are given.

In Section 2, graph edit distance is briefly introduced. The proposed method for the fusion of graphs is presented in Section 3. An experimental evaluation follows in Section 4, and Section 5 offers some concluding remarks.

* Supported by the Swiss National Science Foundation NCCR program "Interactive Multimodal Information Management (IM)2" in the Individual Project "Multimedia Information Access and Content Protection".

F. Roli and S. Vitulano (Eds.): ICIAP 2005, LNCS 3617, pp. 479–486, 2005.

2 Graph Edit Distance

Structural pattern recognition is generally performed by transforming patterns into strings or graphs and matching the resulting structures. Using an attributed graph representation of patterns allows for a powerful representation of complex strucured objects that is better suited to certain pattern recognition problems than a statistical feature-based approach. In recent years a large number of approaches to graph matching have been proposed [5].

One of the most common error-tolerant graph matching methods is based on graph edit distance [6]. Graph edit distance is a general dissimilarity measure on attributed graphs. The key idea of edit distance is to define the dissimilarity of graphs by the amount of edit operations, reflecting small structural distortions, needed to transform one graph into another. To allow for graph distance measures that are tailored to specific applications, it is common to define for each edit operation an edit cost reflecting the strength of the corresponding structural distortion. From these edit costs, the edit distance of two graphs can be defined by the minimum cost sequence of edit operations transforming one graph into the other.

The result of an edit distance computation is a minimum cost edit path from the first to the second graph and its associated edit costs. Nodes and edges of the first graph that are substituted by nodes and edges of the second graph, according to the optimal edit path, can be regarded as locally corresponding parts of the two graphs. Conversely, inserted and deleted nodes and edges can be seen as the non-matching parts. In traditional graph matching methods, the edit distance value is used in the context of a nearest-neighbor classifier, while the optimal node and edge correspondences given by the edit path are not taken into account any further. In this paper, we propose to use the substitutions of the optimal edit path for an error-tolerant detection of the common parts of two or more graphs. The following section describes how several graph patterns can be merged into a single graph by means of edit distance.

3 Data Level Fusion of Graphs

Multiple classifier systems have successfully been used to improve graph matching systems [2,3,4]. Graph classifier fusion, however, is usually performed at the decision level. That is, each single classifier votes for a single class or reports a confidence measure for each class, and all votes or confidence measures are then combined into an overall classification result. The fusion approach we propose in this paper is based on graph fusion at the data level. We assume that each pattern is initially represented by several graphs. In practice, this is the case when several graph extraction methods have been developed based on the same key pattern characteristics, or when the same graph extraction process is carried out several times with different parameters. A crucial requirement of our method is that all those graph representations are compatible, meaning that the same attributes are used in all graphs. The basic idea is to detect common

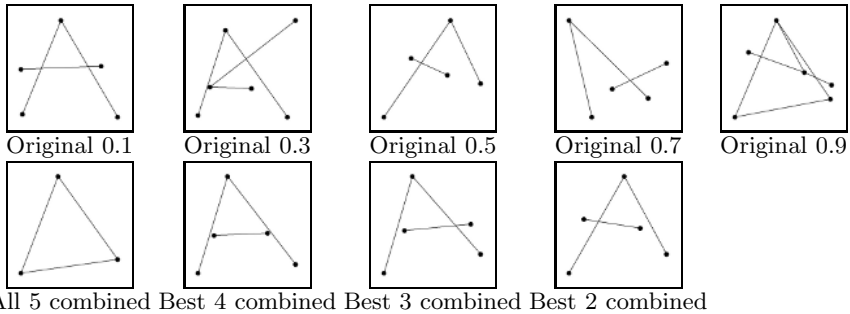

Fig. 1. Graphs from five databases (Original 0.1–0.9) and their *emcs* combinations consisting of all five graphs (All 5) as well as the best matching graphs only (Best 4–2)

structures occurring in all graphs representing a single pattern, and use these common parts in the successive matching process. The common parts typically correspond to structures that are robust against noise and distortions and thus allow for a more reliable classification.

In order to detect common structures, we use the concept of maximum common subgraph. The maximum common subgraph of two graphs is defined as the largest parts of two graphs that are isomorphic to each other [7]. To detect the common parts of two graphs in an error-tolerant way, we first compute the edit distance of the two graphs and use the substitutions of the optimal edit path as a description of the common parts. We then proceed by merging the nodes and edges of the common parts to obtain a new graph representing the structure existing in both graphs. We call this graph the error-tolerant maximum common subgraph (*emcs* graph) of the two graphs. Note that maximum common subgraph, as described in [7], is a special case of *emcs* under the condition that only identical substitutions occur.

If more than two graphs are to be merged, we compute all mutual distances and merge the two graphs with the smallest distance first. We then proceed by merging the current *emcs* graph and the remaining graphs until all graphs have been merged into a single *emcs* graph. The motivation for this procedure arises from the observation that two very different graphs will most likely lead to a small or even an empty *emcs* graph, while two very similar graphs will lead to a large graph that represents the common parts of the graphs very well. This merging procedure also has the advantage that the *emcs* graph computation can be stopped at different stages of the merging process, for instance using the *emcs* graph of the two most similar graphs only instead of the *emcs* graph of all graphs, therefore eliminating the effect of outlier graphs. The result of *emcs* merging is illustrated in Fig. 1. First, five graph instances of a letter *A* line drawing with various degrees of distortion are given. The next graph shows the complete *emcs* graph consisting of all five graphs, and the remaining graphs correspond to the *emcs* graph of the four, three, and two most similar graphs. The original graphs (see the upper row of Fig. 1) exhibit quite a significant amount of distortion in terms of added and displaced nodes and edges, while the

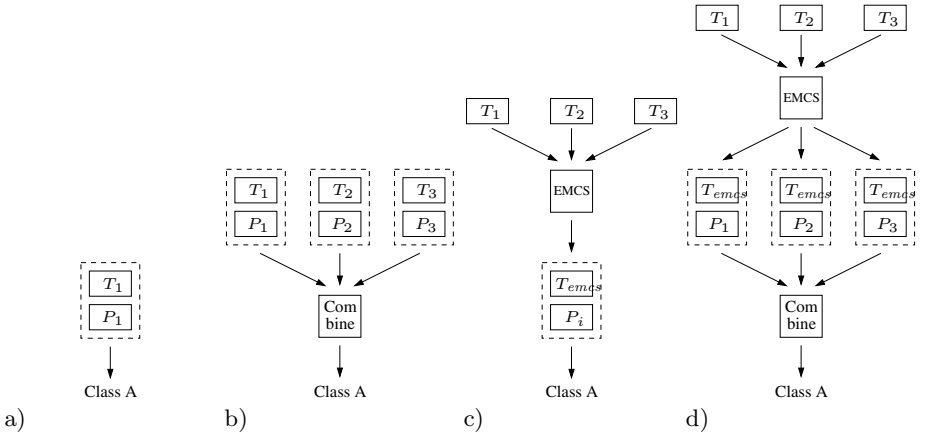

Fig. 2. a) Individual classifier, b) decision level classifier fusion, c) data level pattern fusion, and d) combination of data level and decision level fusion

emcs graphs (see the lower row of Fig. 1) constitute an intuitively less distorted and hence better representation of letter A. The *emcs* graphs indicate that it may be advantegeous to merge more than only the two best fitting graphs (Best 2), but less than all five (All 5) to benefit from a robust representation while eliminating the influence of outlier graphs. Generally, if too many graphs are taken into account in the *emcs* computation, there may be a severe influence of outliers. On the other hand, if too few graphs are considered, the variation of the underlying population may not be covered well enough.

The strategies for the fusion of graph classifiers employed in this paper are illustrated in Fig. 2b–d. To explain our notation, we first give an illustration of a single classifier of the nearest-neighbor type in Fig. 2a, consisting of a set of labeled prototype graphs P_1. In a first step, the original patterns are converted into graphs. The set of graphs to be classified, or test set, is denoted by T_1. Each graph from test set T_1 is classified according to the class of its nearest-neighbor prototype in P_1. In Fig. 2b, the fusion of classifiers at the decision level is illustrated. Here, three different graph extraction procedures are used, which means that every original pattern is represented by three different graphs, one graph in T_1, one in T_2, and one in T_3. For each of the three test sets T_1, T_2, and T_3, a corresponding set of prototype graphs is developed, denoted by P_1, P_2, and P_3, respectively. Each one of the three graphs a pattern is represented by is first individually classified with the respective nearest-neighbor classifier. To obtain an overall classification, the individual results are combined at the class level.

In Fig. 2c, the fusion is performed at the data level. Before classification, the three graphs representing the same pattern — one from T_1, one from T_2, and one from T_3 — are merged into one *emcs* graph. The module denoted by EMCS represents the error-tolerant maximum common subgraph computation, and the test set T_{emcs} represents the new set of *emcs* graphs. In this scenario, one unmodified set of prototypes, say P_i, is chosen for the nearest-neighbor clas-

sification; $i \in \{1, 2, 3\}$. That is, every *emcs* graph in T_{emcs} is classified according to the class of its nearest-neighbor prototype graph in P_i. Finally, in Fig. 2d, a combination of data level fusion and decision level fusion is shown. Again, the three test sets T_1, T_2, and T_3 are merged into a single test set T_{emcs}. Every *emcs* graph is then individually classified according to the original prototypes, that is, once according to P_1, once according to P_2, and once according to P_3. The final classification is obtained by combining the results of the individual classifiers.

4 Experimental Results

In our first experiment we focus on the data level fusion of graphs (see Fig. 2c) using an artificially created database of letter drawings. An experimental evaluation of all fusion strategies illustrated in Fig. 2, applied to the difficult problem of fingerprint classification using real-world data, is described subsequently.

In the first experiment, a clean prototype drawing is manually constructed for 15 capital letters consisting of straight lines only. These prototype drawings are then repeatedly distorted to obtain sample drawings. To control the strength of applied distortions, a distortion parameter is used. By means of this procedure, we create five databases containing 1,500 distorted drawings each, with the distortion parameter ranging from 0.1 to 0.9. A sample drawing of a letter *A* from each database is shown in Fig. 1 (Original 0.1-0.9). The letter drawings are finally transformed into attributed graphs by representing line endings by nodes (containing a position attribute) and lines by edges (without attributes), resulting in graph datasets L_1, L_2, L_3, L_4, and L_5. Each dataset L_i, $i \in \{1, \ldots, 5\}$, is split into two subsets. The test set T_i is defined as the first half of dataset L_i, and the prototype set P_i is defined as the second half of dataset L_i. Using the previously described procedure, we obtain a merged test set T_{emcs} consisting of *emcs* graphs, each of which results from merging five graphs. Similarly, a merged prototype set P_{emcs} is obtained from the prototype sets $P_1, \ldots, P_5$. The results obtained with a nearest-neighbor classifier on each test set $T_1, \ldots, T_5$ individually according to Fig. 2a are shown in the first five rows of Table 1. Using the *emcs* method according to Fig. 2c with prototype set P_{emcs} results in an improvement of the classification rate of almost 10% compared to the best indi-

Table 1. Performance of a nearest-neighbor classifier on five letter graph datasets (T_1–T_5) and the corresponding combined *emcs* dataset (T_{emcs})

Test set	Prototype set	Classification rate
T_1	P_1	54.8
T_2	P_2	54.933
T_3	P_3	51.733
T_4	P_4	70.0
T_5	P_5	76.0
T_{emcs}	P_{emcs}	85.067

Table 2. Performance of five individual fingerprint classifiers and classifier combinations using a) majority voting, b) maximum confidence, and c) confidence voting

Test set	Prototype set	Classification rate	
T_1	P_1	80.85	
T_2	P_2	74.6	
T_3	P_3	73.4	
T_4	P_4	75.5	
T_5	P_5	74.625	
$T_1, \ldots, T_5$	$P_1, \ldots, P_5$	77.35	a)
$T_1, \ldots, T_5$	$P_1, \ldots, P_5$	86.8	b)
$T_1, \ldots, T_5$	$P_1, \ldots, P_5$	83.2	c)

vidual classifier. In this particular case, we find that the *emcs* graphs constitute a representation of the line drawing patterns that is more robust against noise than the original patterns.

We proceed by investigating the applicability of the *emcs* method to fingerprint classification [8]. The method we use extracts graphs from fingerprint images by detecting characteristic signatures in fingerprints and converting these into attributed graphs [9]. Our experiments are based on five graph fingerprint classifiers using similar graph extraction procedures with different parameters. Hence, by employing five slightly different graph extraction procedures, we obtain five different graph representations for each fingerprint image. The five graph datasets $T_1, \ldots, T_5$ have been constructed from the NIST-4 database of fingerprints consisting of 4,000 fingerprint images classified according to the Henry system [8]. In a manual construction process, typically around 30 graph prototypes have been developed for each of the five datasets. The set of graph prototypes belonging to test set T_i is denoted by P_i. The fingerprint graphs from T_i are then classified according to the nearest-neighbor among all prototypes in P_i.

To obtain a measure of the reliability of each classification result, we introduce a confidence measure for distance based nearest-neighbor classifiers. The confidence measure is defined as the ratio of the distance to the nearest neighbor in relation to the distance to the nearest neighbor of the second-closest class. For a single input fingerprint, we thus obtain per classifier the resulting class along with a confidence measure. The results of the five classifiers can then be combined with majority voting (in this case the confidence values are not needed), by selecting the result of the maximum confidence classifier, or another combination rule [10].

The results of the five individual classifiers and combinations at the decision level are given in Table 2. Note that the first five rows correspond to the individual classifier scheme illustrated in Fig. 2a, and the last three rows correspond to the decision level fusion scenario illustrated in Fig. 2b. From the results, we find that among the traditional decision level combination schemes the maximum confidence rule is very effective in improving the classification accuracy. Next, we merge the five databases T_1, T_2, T_3, T_4, and T_5 into an *emcs* database T_{emcs}. The performance of the individual classifiers and the decision

Table 3. Performance of fingerprint classifier combinations using *emcs* data fusion and a) majority voting, b) maximum confidence, and c) confidence voting

Test set	Prototype set	All 5	Best 4	Best 3	Best 2	
T_{emcs}	P_1	79.625	79.925	80.75	80.45	
T_{emcs}	P_2	79.325	79.6	80.375	79.925	
T_{emcs}	P_3	78.15	78.575	79.225	78.825	
T_{emcs}	P_4	63.2	63.25	63.75	63.55	
T_{emcs}	P_5	64.725	65.15	65.625	65.925	
$T_{emcs}, \ldots, T_{emcs}$	$P_1, \ldots, P_5$	73.275	74.725	74.85	75.2	a)
$T_{emcs}, \ldots, T_{emcs}$	$P_1, \ldots, P_5$	81.45	87.3	88.275	87.8	b)
$T_{emcs}, \ldots, T_{emcs}$	$P_1, \ldots, P_5$	81.35	82.35	82.575	82.8	c)

level fusion using T_{emcs} is presented in Table 3. Note that the first five rows correspond to the data level fusion illustrated in Fig. 2c, and the last three rows correspond to the combination of data level and decision level fusion illustrated in Fig. 2d. Comparing the first five rows in Table 3 with the first five rows in Table 2, we observe that the individual classifiers on T_{emcs} do not generally lead to an improvement compared to the original classifiers. Using a fusion at the decision level, however, outperforms all other classification rates, as can be seen in rows six to eight in Table 3. Merging only two or three graphs instead of all five graphs leads to an additional gain in recognition accuracy. Again, the maximum confidence rule turns out to be the most reliable combination method.

For the sake of convenience, a summary of the experimental results is provided in Table 4. Since many fingerprint classification systems are evaluated on the second half of the NIST-4 database only instead of the full database, the recognition accuracy on the second half of NIST-4 is also given. The *emcs* data level fusion is particularly suitable in conjunction with the decision level fusion using the maximum confidence rule (see Fig. 2d). This fusion strategy is significantly better ($\alpha = 0.01$) than all individual classifiers and all other fusion strategies. We conclude that the *emcs* fusion of graph structures can improve nearest-neighbor based graph classifiers and outperform traditional fusion methods.

Table 4. Summary of classification rates obtained on NIST-4 database

Fingerprint classifier	NIST-4	Second half of NIST-4	Method
Best individual classifier	80.85	80.25	Fig. 2a
Best decision level fusion	86.8	86.95	Fig. 2b
Best data level fusion	80.75	80.55	Fig. 2c
Best data and decision level fusion	88.275	88.8	Fig. 2d

5 Conclusions

In the present paper we propose a method for data level fusion of graph patterns. Given two graphs to be merged, we first identify corresponding substructures in both graphs in an error-tolerant manner using graph edit distance. These local correspondences can then be used to construct a graph representing the common parts of the two original graphs. The merged graphs constitute robust representatives of the original graphs that are less prone to noise. By generalizing the concept to more than two graphs, we can use the graph fusion method to combine several datasets of the same underlying data into a single graph dataset. To demonstrate the usefulness of the proposed method, we apply the fusion method to artifically created line drawing graphs and the difficult problem of fingerprint classification. In both cases a significant improvement of the performance is obtained. In the future, we would like to investigate in greater detail what properties cost functions need to exhibit to be suitable for the graph fusion process.

References

1. Roli, F., Kittler, J., Windeatt, T., eds.: Multiple Classifier Systems, Proc. 5th Int. Workshop. LNCS 3077. Springer (2004)
2. Marcialis, G., Roli, F., Serrau, A.: Fusion of statistical and structural fingerprint classifiers. In Kittler, J., Nixon, M., eds.: 4th Int. Conf. Audio- and Video-Based Biometric Person Authentication. LNCS 2688 (2003) 310–317
3. Yao, Y., Marcialis, G., Pontil, M., Frasconi, P., Roli, F.: Combining flat and structured representations for fingerprint classification with recursive neural networks and support vector machines. Pattern Recognition **36** (2003) 397–406
4. Schenker, A., Bunke, H., Last, M., Kandel, A.: Building graph-based classifier ensembles by random node selection. In Roli, F., Kittler, J., Windeatt, T., eds.: Proc. 5th Int. Workshop on Multiple Classifier Systems. LNCS 3077, Springer (2004) 214–222
5. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. Journal of Pattern Recognition and Artificial Intelligence **18** (2004) 265–298
6. Sanfeliu, A., Fu, K.: A distance measure between attributed relational graphs for pattern recognition. IEEE Transactions on Systems, Man, and Cybernetics **13** (1983) 353–363
7. Fernandez, M.L., Valiente, G.: A graph distance metric combining maximum common subgraph and minimum common supergraph. Pattern Recognition Letters **22** (2001) 753–758
8. Maltoni, D., Maio, D., Jain, A., Prabhakar, S.: Handbook of Fingerprint Recognition. Springer (2003)
9. Neuhaus, M., Bunke, H.: A graph matching based approach to fingerprint classification using directional variance (2005) Submitted.
10. Kittler, J., Hatef, M., Duin, R., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 226–239