On Graph-Associated Matrices and Their Eigenvalues for Optical Character Recognition

Miriam Schmidt, Günther Palm, and Friedhelm Schwenker

Institute of Neural Information Processing,
University of Ulm, 89069 Ulm, Germany
{miriam.k.schmidt,guenther.palm,friedhelm.schwenker}@uni-ulm.de
http://www.uni-ulm.de/in/neuroinformatik.html

Abstract. In this paper, the classification power of the eigenvalues of six graph-associated matrices is investigated and evaluated on a benchmark dataset for optical character recognition. The extracted eigenvalues were utilized as feature vectors for multi-class classification using support vector machines. Each graph-associated matrix contains a certain type of geometric/spacial information, which may be important for the classification process. Classification results are presented for all six feature types, as well as for classifier combinations at decision level. For the decision level combination probabilistic output support vector machines have been applied. The eigenvalues of the weighted adjacency matrix provided the best classification rate of 89.9 %. Here, almost half of the misclassified letters are confusion pairs, such as *I*-*L* and *N*-*Z*. This classification performance can be increased by decision fusion, using the sum rule, to 92.4 %.

Keywords: graph classification, weighted adjacency matrix, spectrum, support vector machine.

1 Introduction

Spectral graph theory is an important branch in the area of graph classification. Matrices associated with graphs, e.g. the adjacency matrices, contain essential information about the graph's connectivity [1]. This information is also included in the eigenvalues of the adjacency matrix which build the so-called spectrum. The spectrum of a graph exhibits some important properties, which make them ideal candidates for classification tasks [2],[3].

First, the spectrum is invariant with respect to the labeling of the nodes. Two graphs, which only differ in the labeling, are called isomorph to each other and the graph isomorphism problem (the computation, if two graphs are isomorph) belongs to the NP-complete problems. These problems build a subset of NP problems, which are defined as decision problems whose solutions can be verified in polynomial time, but the time required to solve the problems increases quickly [4]. The graph isomorphism problem occurs, if one has to match the nodes of two graphs to calculate, for example, the graph edit distance [5],[6].

N. Mana, F. Schwenker, and E. Trentin (Eds.): ANNPR 2012, LNAI 7477, pp. 104–114, 2012.

Hereby, a distance between two graphs is calculated, taking the number and corresponding costs of operations (deleting or inserting nodes or edges, relabeling) into account which are needed to transform one graph into the other. By using the spectrum as feature of the graph, it is not necessary to deal with this matching problem. However, if two graphs are just flipped or rotated, the spectrum does not provide the ability to distinguish these isomorph graphs.

Furthermore, if the underlying matrix is real and symmetric, the eigenvalues are also real. Hence, the eigenvalues can be used to map the graphs in a coordinate system and use well known clustering or classification algorithms. This method is called spectral embedding [7],[8].

In some applications, graphical representations of different orders have to be considered. In the underlying data set (see Sect. 4) are graphs with one node up to graphs with eight nodes. This can be easily accomplished by using only a distinct subset of the eigenvalues, e.g. the first three eigenvalues [9].

But one big restriction to the adjacency matrix is that it includes no information about the length of the edges in the graph. It contains just the binary information, if there is an edge or not. If two graphs have the same edges, but differ in the stretching of the graph, it is impossible to distinguish them by using the adjacency matrices, because they are exactly the same. Thus, to include the information about the stretching, the weighted adjacency matrices (with the lengths of the edges as labels) can be used [10]. In this case, the binary values in the matrices are replaced by the labels of the corresponding edges.

In this paper, the power of the principal eigenvalues of six different graph-associated matrices for the spectral classification is investigated. Each matrix ((weighted) adjacency matrix, (weighted) Laplacian matrix and (weighted) adjacency matrix of the complement graph) includes different information, which can be crucial for the classification or needless. Classification results are presented for all six feature types and by investigating the miss-classified samples, we were able to differentiate the qualities of the different matrices. To improve the classification performance, we also utilized classifier combinations at decision level. For the decision level combination probabilistic output support vector machines habe been applied.

As application data set, we used the capital letter data set of the *IAM Graph Database Repository*[1] because of its publicity, its complexity and its accessibility. Our classification results are below the best performance on this dataset [11], but the aim was not to outperform the existing classification approaches but to investigate different spectra for classification problems.

The rest of the paper is organized as follows: First, in Sect. 2 the essential notations in (spectral) graph theory are provided. The illustration of the support vector machine classifier in Sect. 3 is followed by the data description including the explanation of the feature extraction in Sect. 4. In Sect. 5 the experiments and the results are presented before the paper will be completed by a summary and conclusion.

[1] Databases of the Institute of Computer Science and Applied Mathematics in Bern, Switzerland. `http://www.iam.unibe.ch/fki/databases/iam-graph-database`

2 Graph Associated Matrices

This section provides a brief introduction in graph theory. The textbooks [12] and [13] are recommended for extensive information about these topics.

An *undirected graph* is defined as a pair $G = (V, E)$, consisting of a finite set of nodes $V(G) = \{v_1, ..., v_n\}$ and a set of edges $E(G) = \{e_1, ..., e_m\}$, which are unordered two-element subsets of $V(G)$ with $e = \{v_k, v_l\}$, v_k, v_l in $V(G)$. The order of a graph is described by $|V(G)| = n$, the size by $|E(G)| = m$. The degree of v_k is $d(v_k) = |\{v_l \in V(G) | \{v_k, v_l\} \in E(G)\}|$, $k, l = 1, ..., n$ (the number of edges connected with v_k).

A graph $G = (V, E, w)$ is called *weighted graph*, with a labeling function $w(e_i)$, which assigns a label to each edge e_i. Usually real numbers are used as labels (e.g. length, cost).

The most known and used graph associated matrices are the *adjacency matrix* and the *Laplacian matrix*. Two nodes v_k and v_l are adjacent to another if there exist an edge $e = \{v_k, v_l\}$. The entries $a_{k,l}$ of the adjacency matrix $A(G) = [a_{k,l}]_{n \times n}$ are 1, if there exists an edge $e = \{v_k, v_l\}$ (otherwise 0). The Laplacian matrix is defined as $L(G) = D(G) - A(G)$, where $D(G)[d_{k,l}]_{n \times n}$ is the degree matrix with $d_{k,k} = d(v_k)$ and $d_{k,l} = 0$, for all k unequal l.

The *weighted adjacency matrix* $A^\star(G) = [a^\star_{k,l}]_{n \times n}$ of a weighted graph G contains the labels associated with the edges, instead of just 1: $a^\star_{k,l} = w(\{v_k, v_l\})$ if $e = \{v_k, v_l\}$ in $E(G)$ (otherwise 0). Analogously, the *weighted Laplacian matrix* is defined as $L^\star(G) = D^\star(G) - A^\star(G)$, using the diagonal weighted degree matrix $D^\star(G)[d^\star_{k,l}]_{n \times n}$ with $d^\star_{k,k} = \sum_l a^\star_{k,l}$ (sum of all edge labels connected with the node v_k).

The *complement graph* $\overline{G} = (V, \overline{E}, \overline{w})$ of a graph $G = (V, E, w)$ incorporates the same node set V as G, and if two nodes are not adjacent in G, they are adjacent in $\overline{G}$. The *weighted adjacency matrix of the complement graph* will be named $\overline{A}(G)$ in the following.

3 Classification in Vector Spaces

There exist a huge amount of classification approaches to classify patterns in vector spaces [14],[15]. One of the most popular methods is the classification with support vector machines (SVMs) [16]. These supervised machine learning methods can be used for binary classification problems. During the training process, the SVM uses the given data points (training set) with the corresponding class labels (teachers) to optimize a hyperplane h, which separates the data points into the two given classes, whereby the two margins/gaps between the hyperplane and the data points should be as wide as possible. New data points are then mapped in the same space and the side of the hyperplane they belong to, decides their class labels.

We also applied a fuzzy output version of the SVMs. Therefore, the distance of a data point to the hyperplane is calculated (instead of just the algebraic sign)

and normalized with a logistic sigmoid function. Hence, the output of the SVM can be treated as a probability to belong to the first class.

If there are more than two classes to distinguish, as is shown in Figure 1, it is not possible to use just one hyperplane to separate the classes, e.g. a three-class problem: *circles*, *stars* and *squares*.

There are two possibilities to solve this problem:

1. **One-vs.-One Classifier:**
   all pairs of possible two-class SVMs (1-2, 1-3, 2-3) are trained and the decision is made by voting.

2. **One-vs.-All Classifier:**
   every class is trained against all others (1-23, 2-13, 3-12) and the highest probability leads to the most likely class. Of course, this method just works for the fuzzy output SVMs.

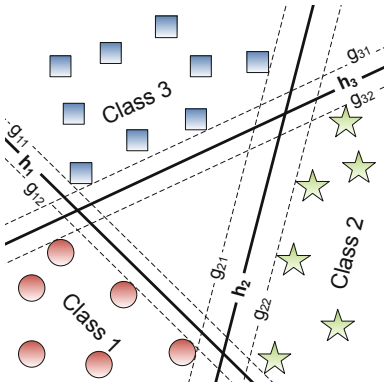

Fig. 1. Three-class problem, solved by a one-vs.-all classifier. The hyperplanes h_1, h_2 and h_3 separate the three classes.

Most often, the two classes are not linearly separable in their dimensional space. By applying a adequate function on the data points, they are mapped in a much higher dimensional space, where they are now linearly separable. The optimized linear hyperplane in this higher dimensional space is then transformed back in the lower dimensional space, where it becomes a non-linear hyperplane, but separates the classes. Because this method needs a lot of computation time, the so-called *kernel trick* is used. There are special functions, called *kernel functions*, which allow the optimization of the hyperplane without really transforming the data points. For more information on SVMs and RBFs see [16].

4 Data Description and Feature Extraction

The capital letter data set of the *IAM Graph Database Repository* [17] was utilized as application data set. It consists of graphs, representing capital letter drawings of all 15 "straight line" letters in the Roman alphabet (A, E, F, ...). In order to enforce differences in the samples, weak distortion operations, *shifting*, *removing* or *adding* lines to a prototype line drawing, were conducted. The nodes of the graphs represent the ending points of the lines in the drawing and the edges stand for the lines themselves. Finally, the nodes are labeled with their two-dimensional coordinates. The data set contains 2250 samples (150 for each class), equally divided in a test set, a validation set and a training set. Figure 2 shows examples of the graphical representation of the letter A with different kinds of distortion.

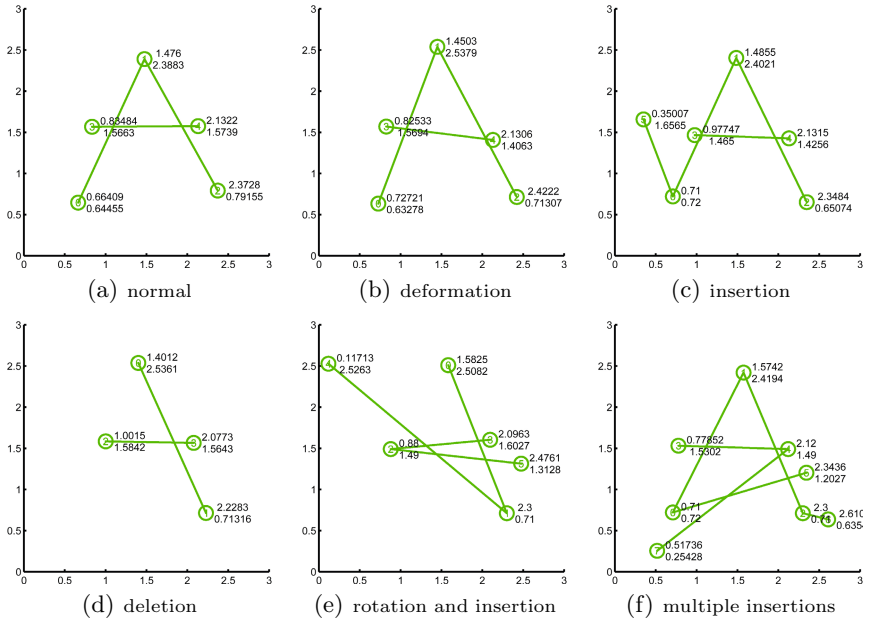

Fig. 2. Sample of the letter A with different kinds of distortion

In order to attain weighted graphs from the data, the following *labeling function* was used: $w : E(G) \rightarrow \mathbb{R}_0^+$. It attaches to each edge $e = \{v_k, v_l\}$ the Euclidean distance of the two nodes v_k and v_l.

Afterward, different matrices were extracted from the graph and their eigenvalues were computed. The eigenvalues $\{\lambda_1, ..., \lambda_p\}$ of the adjacency matrix $A(G)$ of a graph G build the so-called spectrum $SP(G) = [\lambda_1, ..., \lambda_p]$ and represent the zeros of the characteristic polynomial $|\lambda I - A(G)|$. For the eigenvalues and the corresponding eigenvectors $\{x_1, ...x_p\}$, the following equation (1) holds:

$$A(G)x_i = \lambda_i x_i, \forall\, i = 1, ..., p\,. \tag{1}$$

Within one spectrum the eigenvalues were sorted in ascending order. To handle the different orders of the graphs, and therefore the different sizes of the spectra, we filled the too short spectra with zero values. The following spectra were computed:

SP_A	Spectrum of the adjacency matrix
SP_L	Spectrum of the Laplacian matrix
$SP_{A^\star}$	Spectrum of the weighted adjacency matrix
$SP_{L^\star}$	Spectrum of the weighted Laplacian matrix
$SP_{\overline{A^\star}}$	Spectrum of the weighted adjacency matrix of $\overline{G}$
$SP_{F^\star}$	Spectrum of the filled weighted adjacency matrix (in this case, all distances were included, even if there were no edges, i.e. distance matrix of the nodes).

5 Experimental Results

In this section, the classification results for the six different feature types (as defined in Sect. 4) are presented and the performances are compared. We conducted experiments with One-vs.-One SVMs as well as with One-vs.-All SVMs. As kernel functions *RBF kernels*, *linear kernels*, *quadratic kernels* and *polynomial kernels* were investigated. The validation set allows us to find a suitable standard deviation σ for the RBF functions, which provided the best classification performance.

The decisions of the SVMs have to be calculated in different ways: For the hard results (class labels) we conducted a decision voting of the 105 One-vs.-One SVMs (One-vs.-One - hard). Because each SVM emit a probability for the first class (see Sect. 3), it is possible to compute the average rate for each class too (One-vs.-One - fuzzy). The fuzzy decision for the One-vs.-All SVMs can be obtained by choosing the maximum probability of the single classes. Table 1 shows the classification rates for the different feature types and the different SVM models with RBF kernel functions (exemplary with $\sigma = 0.8$).

Table 1. Classification results for the spectra SP_A, SP_L, $SP_{A^\star}$, $SP_{L^\star}$, $SP_{\overline{A^\star}}$ and $SP_{F^\star}$. For all SVMs, radial basis function kernels were used with a specific standard derivations $\sigma = 0.8$. Decisions for the classes by majority vote (One-vs.-One - hard), by computing the average classification rate for the classes (One-vs.-One - fuzzy) and by selecting the class with the maximum probability (One-vs.-All - fuzzy).

Feature type	**One-vs.-One**		**One-vs.-All**
	hard	**fuzzy**	**fuzzy**
SP_A	0.491	0.557	0.179
SP_L	0.232	0.475	0.192
$SP_{A^\star}$	0.645	0.867	**0.897**
$SP_{L^\star}$	0.493	0.803	0.799
$SP_{\overline{A^\star}}$	0.292	0.825	0.845
$SP_{F^\star}$	0.364	0.745	0.760

The hard decisions of the One-vs.-One SVMs lead to the lowest classification rates. It is unlikely that all 14 SVMs, which are responsible for one letter, emit this letter but it frequently happens that a few wrong decisions lead to a wrong winner. The emitted probabilities of the SVMs revealed, that the correct letter is often the second or the third winner. By including the probabilities in the calculation (fuzzy decisions) we could keep this information. The fuzzy decisions leads to a significant ($p = 0.03$) improvement of the classification results. Although, the One-vs.-All SVMs achieved the highest classification fuzzy rate (89.7 % for

$SP_{A^\star}$), the average fuzzy rate (for all spectra) of the One-vs.-One classifiers with 71.2 % is higher than the average of the One-vs.-All classifiers with 61.2 %.

Table 2 shows the best models for the six feature types, including the most suitable assignment for the standard deviation σ of the RBFs (range of σ between 0.5 and 3.0).

Table 2. Highest achieved classification rates for the different spectra SP_A, SP_L, $SP_{A^\star}$, $SP_{L^\star}$, $SP_{\overline{A^\star}}$ and $SP_{F^\star}$, the responsible model and the most suitable assignment for σ

Feature type	**Classification method**	σ	**Classification rate**
SP_A	One-vs.-One - fuzzy	0.8	0.557
SP_L	One-vs.-One - fuzzy	0.8	0.474
$SP_{A^\star}$	One-vs.-All - fuzzy	1.0	**0.899**
$SP_{L^\star}$	One-vs.-One - fuzzy	0.6	0.813
$SP_{\overline{A^\star}}$	One-vs.-All - fuzzy	1.0	0.847
$SP_{F^\star}$	One-vs.-One - fuzzy	1.5	0.777

The spectra of the adjacency matrices (SP_A) and the Laplacian matrices (SP_L), with no further information about the shape of the graph, provide the lowest classification rates. This is clear if one imagines the following simple example: the letter M and the letter W have the same number of nodes and the same edges between them. W is just a flipped version of M. Because the sorted eigenvalues and therefore the spectrum of these two matrices are invariant with respect to the labeling of the nodes, they are exactly the same.

By including all distances ($SP_{F^\star}$), the information about the characteristic shape of the letter gets lost. Letters with the same number of nodes only differ in the pairwise distances of the nodes but not in the number of non-zero entries in the matrices.

With the spectra of the Laplacian matrix ($SP_{L^\star}$) and the spectra of the adjacency matrix of the complement graph ($SP_{\overline{A^\star}}$), classification rates over 80 % could be achieved, but the spectra of the weighted adjacency matrices ($SP_{A^\star}$) provided the highest classification rate with 89.9 % (see Table 2). The information about the characteristic shape of the letters is included in all these features, i.e. where the edges are and also the lengths of the edges.

The difference between the recognition rates of the adjacency matrices (SP_A and $SP_{A^\star}$) and the Laplacian matrices (SP_L and $SP_{L^\star}$) is not significant ($p = 0.23$), however within this experiment, the matrices with the zeros on the diagonal A and $A^\star$ lead to a slightly higher performance.

Table 3 shows the confusion matrix of the 15 One-vs.-All SVMs for the spectrum of the weighted adjacency matrix $SP_{A^\star}$. The columns refer to the actual

Table 3. Confusion matrix of the 15 One-vs.-All SVMs for the feature $SP_{A^\star}$ (spectrum of the weighted adjacency matrix). The columns refer to the actual classes and the rows to the identified classes.

		actual class														
		A	E	F	H	I	K	L	M	N	T	V	W	X	Y	Z
identified class	A	46	3	0	0	0	0	2	0	0	0	0	0	0	0	0
	E	4	43	3	2	0	0	0	0	0	0	0	0	0	0	0
	F	0	2	47	1	0	4	0	0	0	0	0	0	0	0	0
	H	0	2	0	45	0	2	0	0	0	0	0	0	0	0	0
	I	0	0	0	0	41	0	3	0	0	0	0	0	0	0	0
	K	0	0	0	2	0	42	0	0	0	0	0	0	0	0	0
	L	0	0	0	0	8	1	43	1	0	0	0	0	0	0	0
	M	0	0	0	0	0	1	2	47	0	0	0	3	1	0	0
	N	0	0	0	0	0	0	0	0	47	0	2	1	0	0	6
	T	0	0	0	0	0	0	0	0	1	49	0	0	0	3	0
	V	0	0	0	0	0	0	0	0	1	0	46	1	1	0	0
	W	0	0	0	0	0	0	0	0	1	1	0	42	2	0	1
	X	0	0	0	0	0	0	0	0	0	0	0	0	46	0	0
	Y	0	0	0	0	0	0	1	2	0	0	0	3	0	47	0
	Z	0	0	0	0	1	0	1	0	0	0	2	0	0	0	43
		0.92	0.86	0.94	0.90	0.82	0.84	0.86	0.94	0.94	0.98	0.92	0.84	0.92	0.94	0.86

classes and the rows to the identified classes. The entries of the matrix are the number of letters (each letter appears 50 times in the data set). The last row contains the classification rates for each class.

The confusion matrix shows, that there are some letters, i.e. pairs of letters, which seem more difficult to distinguish than others. Almost 15 % (11 out of 76) of the misclassified letters are confusions of the letters I-L. The other eye-catching pairs are A-E with almost 10 %, N-Z with almost 8 % and E-F with more than 6 % of the misclassification rate. These results are not really surprising because the pair building letters are similar to each other, e.g. one edge more or rotated.

The experiments show, it is essential to choose an underlying matrix, which contains the important information to maintain the differences between the classes. In some cases, the crucial information may be the number of edges, then the adjacency matrix can be appropriate. In this case, the graphs vary in their shape, which depends on the existence of the edges and even more on their length and hence, it was necessary to include this relevant information in the features.

5.1 Decision Fusion of the Different Outputs

To probably enhance the classification performance, we computed combinations of the One.-vs.-All - fuzzy classification outputs of the different feature types. In some cases, the highest probability leads to a wrong class, but the second highest would have been the correct class. Therefore, we added the probabilities of the different SVMs class-wise, before we chose the maximum to get the identified class. The best three combinations with their overall classification are (for the individual classification performances, see Table 1):

1.	$SP_{A^\star} + SP_{\overline{A^\star}} + SP_{F^\star}$	92.4 %
2.	$SP_{A^\star} + SP_{L^\star} + SP_{\overline{A^\star}} + SP_{F^\star}$	92.1 %
3.	$SP_{A} + SP_{A^\star} + SP_{L^\star} + SP_{\overline{A^\star}} + SP_{F^\star}$	92.0 %

The results show that through combination of the individual classifiers on decision level, the misclassification rate can be decreased by 24 %. The combination with the SVMs, trained with $SP_{A^\star}$, $SP_{\overline{A^\star}}$ and $SP_{F^\star}$ has the highest overall classification rate.

5.2 Comparison with Related Work

To compare our results with other approaches, we selected papers where the same data set was utilized. In [18], bipartite graph matching for computing the edit distance was used and the distance measure was then utilized to classify the graphs with a 5-nearest-neighbor classifier. The highest classification rate with 84 % was achieved with their own method (Munkres' Algorithm). As reference method they implemented the optimal tree-search algorithm for computing the graph edit distance, introduced in [19] and [5], which achieved a recognition rate of 82.7 %. However, these are completely different methods, since these methods are graph-based and not feature-based.

As mentioned in the introduction, the actually best performance with 98.9 % on this data was achieved by [11]. In this paper, they "aim at bridging the gap between the domain of feature based and graph based object representations". They calculate the graph edit distances between one graph and m chosen prototypes and utilize this m-dimensional dissimilarity measure to classify the graph in a vector space using a SVM classifier. Although this method achieved a high classification performance, it is depending on a lot of variables and time-consuming calculations: the calculation of the graph edit distance is strongly connected to the order of the graphs (which makes it inappropriate for large graphs) and a good selection of the prototypes may be crucial for the classification process.

6 Discussion and Future Work

Using the spectrum of a graph as feature for classification is a well known method. The eigenvalues of graph-associated matrices can be utilized to

embed the graphs in a vector space and use standardized methods, e.g. SVMs, for the classification. There are many matrices which can be extracted from of a graph and we investigated the power of six of them for classification: spectrum of the adjacency matrix (SP_A), spectrum of the Laplacian matrix (SP_L), spectrum of the weighted adjacency matrix ($SP_{A^\star}$), spectrum of the weighted Laplacian matrix ($SP_{L^\star}$), spectrum of the weighted adjacency matrix of $\overline{G}$ ($SP_{\overline{A^\star}}$) and the spectrum of the filled weighted adjacency matrix ($SP_{F^\star}$). As label for each edge, the Euclidean distances between the two connected nodes were utilized.

As data set, we used the capital letter data set of the *IAM Graph Database Repository* (see Sect. 4) because is has become a benchmark data set for graph classification and our results can easily be compared to other approaches.

The weighted versions ($SP_{A^\star}$,$SP_{L^\star}$ and also $SP_{\overline{A^\star}}$) include more information than the others and generate classification rates between 80 % and 90 %. The misclassified letters build confusion pairs, that means, there are some couples which are difficult to distinguish. The following pairs *I-L*, *A-E*, *N-Z* and *E-F* cause almost 40 % of the misclassified letters. One reason for the poor distinctness is, that the letters are similar to each other (e.g. just one edge more) and the other reason is that the spectrum is invariant with respect to the rotation of the graph, more precisely the labeling of the nodes (*N* ist just a rotated *Z*). In consideration of this fact, the classification performance is absolutely satisfying.

By combining the outputs depending on different features (decision fusion), the misclassification rate can be decreased by 24 %. Each of the underlying matrices contain information, which the others lack and the combination can benefit from all. With a classification rate of 92.4 % we are below the best classification performance on the data set with 98.9 % [11]. However, we did not aim at outperforming other approaches, but to investigate the power of different graph-associated matrices for classification. It is absolutely important to choose the graph-associated matrix which contains the diverse information needed for the classification task and in this case, the length between the nodes is highly relevant.

The next step is to utilize more complex data sets and investigate the eigenvalues of the weighted adjacency matrices (weighted with the Euclidean distances) in a more theoretical way. Is there a connection between a eventually existing distribution of the distances in the matrices and the eigenvalues of them? How big is the influence of the connectivity in the graph? We started our research with an artificial generated data set.

Acknowledgments. The work of Miriam Schmidt is supported by a scholarship of the Graduate School *Mathematical Analysis of Evolution, Information and Complexity* of the University of Ulm. The work of Günther Palm and Friedhelm Schwenker is supported by the Transregional Collaborative Research Center *SFB/ TRR 62 Companion-Technology for Cognitive Technical Systems*, funded by the German Research Foundation (DFG).

References

1. Cvetković, D.M., Doob, M., Sachs, H.: Spectra of Graphs: Theory and Applications, 3rd edn. Vch Verlagsgesellschaft Mbh (1998)
2. Chung, F.R.K.: Spectral Graph Theory. CBMS Regional Conference Series in Mathematics, vol. 92. Oxford University Press (1997)
3. Brouwer, A.E., Haermers, W.H.: Spectra of Graphs. Universitext. Springer (2012)
4. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co. (1990)
5. Sanfeliu, A., Fu, K.S.: A distance measure between attributed relational graphs for pattern recognition. IEEE Transactions on Systems, Man, and Cybernetics 13(3), 353–362 (1983)
6. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters 19(3-4), 255–259 (1998)
7. Luo, B., Wilson, R.C., Hancock, E.R.: Spectral embedding of graphs. Pattern Recognition 36(10), 2213–2230 (2003)
8. Wilson, R.C., Hancock, E.R., Luo, B.: Pattern vectors from algebraic graph theory. IEEE Trans. Pattern Anal. Mach. Intell. 27(7), 1112–1124 (2005)
9. Schmidt, M., Schwenker, F.: Classification of Graph Sequences Utilizing the Eigenvalues of the Distance Matrices and Hidden Markov Models. In: Jiang, X., Ferrer, M., Torsello, A. (eds.) GbRPR 2011. LNCS, vol. 6658, pp. 325–334. Springer, Heidelberg (2011)
10. Umeyama, S.: An eigendecomposition approach to weighted graph matching problems. IEEE Transactions on Pattern Analysis and Machine Intelligence 10(5), 695–703 (1988)
11. Riesen, K., Neuhaus, M., Bunke, H.: Graph Embedding in Vector Spaces by Means of Prototype Selection. In: Escolano, F., Vento, M. (eds.) GbRPR 2007. LNCS, vol. 4538, pp. 383–393. Springer, Heidelberg (2007)
12. Diestel, R.: Graph Theory, 3rd edn. Graduate Texts in Mathematics, vol. 173. Springer (2005)
13. Bollobás, B.: Modern Graph Theory, 2nd edn. Graduate Texts in Mathematics. Springer (2002)
14. Vapnik, V.N.: The nature of statistical learning theory, 2nd edn. Statistics for Engineering and Information Science. Springer (1999)
15. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer (2007)
16. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Adaptive Computation and Machine Learning. MIT Press (2002)
17. Riesen, K., Bunke, H.: IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) SSPR&SPR 2008. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008)
18. Riesen, K., Neuhaus, M., Bunke, H.: Bipartite Graph Matching for Computing the Edit Distance of Graphs. In: Escolano, F., Vento, M. (eds.) GbRPR 2007. LNCS, vol. 4538, pp. 1–12. Springer, Heidelberg (2007)
19. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters 1(4), 245–253 (1983)