Use of Structured Pattern Representations for Combining Classifiers

Raisa Socorro[1] and Luisa Micó[2]

[1] Instituto Superior Politécnico Jose Antonio Echevarría
La Habana, Cuba
[2] Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
P.O. box 99, E-03080 Alicante, Spain

Abstract. In Pattern Recognition, there are problems where distinct representations can be obtained for the same pattern, and depending on the type of classifiers (statistical or structural) one type of representation is preferred versus the others. In the last years, different approaches to combining classifiers have been proposed to improve the performance of individual classifiers. However, few works investigated the use of structured pattern representations. In this paper combination of classifiers has been applied using tree pattern representation in combination with strings and vectors for a handwritten character classification task. In order to save computational cost, some proposals based on the use of both embedding structured data and cascade classifiers are provided.

Key words: combining classifiers, tree data representation, string data representation, edit distance, k-NN rule

1 Introduction

Multiple classifier systems have received much attention in the recent years. Its success is in the use of different decisions to obtain a more accurate classification than single classifiers. Classifier ensemble has mainly two advantages: on the one hand, the recognition performance is higher in opposite of single classifiers. On the other hand, the behaviour of a multiple classifier system is more robust due to the fact that a final decision is taken on the basis of several decisions (critical individual decision can be avoided) [1].

As stated in a recent work by Duin [2], a consistent set of different classifiers should be used, but also they should be comparable. In the same work different sets of consistent combined classifiers are presented (using different initializations, parameters choice, classification schemes, etc) obtaining the best results when different feature sets have been used.

While regular vector-space represented data are widely used for combining classifiers, structured data have been rarely used. The main reason is the wide range of different classification methods that can be applied when patterns are

represented as vectors. However, the number of classifiers to use is drastically reduced for structural pattern representations. For example, in the case of structured data (strings, trees,graphs ...), if a dissimilarity measure is defined, a k nearest neighbour (k-NN) classifier can be applied. Also, recently, the use of kernel functions based on edit distances [3] or the use of embedding for structured data in real vector spaces [4] have enabled the use of alternative statistical methods for classification.

Structural representation of data have been successfully used in many recognition tasks. In fact, it is the more natural representation of data in many applications, as for example in computational biology. The main advantages of using a structural representation is that the number of features is not fixed and the relationship between individual feature components appears explicitly in the representation (in contrast with the feature vector representation).

In this work we analyse how the combination of more than a structural representation can improve the classification error rate. As well as strings and vectors have been used previously in classifier ensembles, trees have not been used ever. The experiments performed show that the performance of the classification improves when trees are used in combination with vectors and strings. Moreover, experiments performed using embedding strings and vectors and a *"filter and refine"* classifier show that a significant speedup can be achieved at the expense of a decrease in the rate of success. In all the experiments, the classifier we adopted is the k-NN. As different representations are used, different dissimilarity measures are needed, associated to the type of data. For that reason, in this work the Euclidean distance, the string-edit-distance and the tree-edit-distance have been used.

In the following sections, feature extraction methods for data and ensemble schemes used in this work are described. Then, the results for the ensembles are presented for binary images of handwritten characters from the NIST 3 Database National Institute of Standards and Technology [5]. Finally, the conclusions drawn from the results are discussed, pointing the research to further work lines.

2 Feature extraction of the data set and distances

Three different feature extraction methods have been used in this work, two of them are structural representations (trees and strings) and the third is a representation in a real vector space. Given a pattern image, the key idea is that each representation gives some type of information about the pattern that it is not given by the others. For example, strings represent the contour of the images, trees represent the skeleton, and each dimension in a n-dimensional vector represents the ratio of black pixels over the total in a region of the image.

2.1 Tree data representation

The first step before obtaining a tree data representation of patterns is the use of a thinning algorithm to eliminate redundant information in the image [6]. The

image is scanned from left to right starting from the top looking for the first black pixel. This first pixel is marked with label "0". Every tree node has so many descendents as unmarked neighbours has the selected and marked pixel; each of the new branches is extended following their neighbourhood until: a) the branch has a maximum size (parameter R of the algorithm [3]), b) is a terminal pixel or c) is an intersection pixel. After that, a new node is assigned to every end of branch. The labels of the nodes (except for the root) have values between "1" and "8" depending on the direction of the branch (see figure 1 on the left). More detailed information about this algorithm can be found in [7].

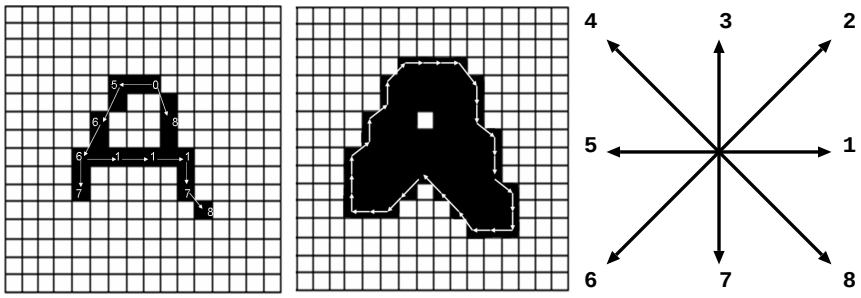
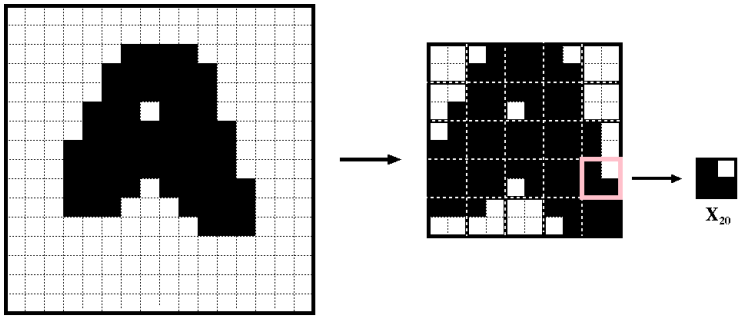


Fig. 1. On the left, example of a tree and a contour string. On the right, symbols used to encode strings and trees.



$$\vec{X} = (X_1, X_2, \ldots, X_{24}, X_{25})$$

Fig. 2. Example of a vectorial data representation.

[3] in this work the parameter R=2, that means that the maximum length of each branch of the tree is 2 pixels

A general tree edit distance is defined by Shasha and Zhang in [8]. Given trees T_1 and T_2, a dynamic programming algorithm with space complexity $O(|T_1| \times |T_2|)$ and $O(|T_1| \times |T_2| \times \min(depth(T_1); leaves(T_1)) \times \min(depth(T_2); leaves(T_2))$ in time complexity has been used in this work. For the edit basic operations (insertion, deletion and substitution of nodes) in the tree, the following weights were used:

- $w_i = w_d = 1$ (insertion and deletion weights)
- $w_{ij} = min(|i - j|; 8 - |i - j|)$ (substitution weights)

The distance $d_t(T_1, T_2)$, is finally normalized (to avoid the size invariance problem) with the maximum size of the two trees [4]:

$$d_N(T_1, T_2) = \frac{d_t(T_1,T_2)}{\max(|T_1|,|T_2|)}$$

2.2 String data representation

Following the same strategy than in the previous method, the image is scanned from left to right starting from the top, searching the first black pixel [5]. In this case, the external contour of the characters (the string) is obtained going to the right (in the clockwise sense) until the first pixel is reached again (see figure 1 in the middle). The symbols (directions) in the strings are the same than in the case of trees.

The edit distance between two strings S_1 and S_2, $d_s(S_1, S_2)$, is defined as the minimum-cost set of transformation that must be done to turn a string into the other. The edit basic operations are insertions, deletions and substitution of individual symbols in the string. The cost values for the operations are equal that those used in tree edit distance.

This distance can be computed in time $O(|x| \times |y|)$ [9]. As in trees, the distance is finally normalized with the maximum of the length of the strings to avoid the size invariance problem:

$$d_N(S_1, S_2) = \frac{d_s(S_1,S_2)}{\max(|S_1|,|S_2|)}$$

2.3 Vectorial data representation

Firstly, to obtain a size invariance representation of characters, the smallest squared matrix that includes the character in the original image is selected (see Figure 2). The scaled image is divided in n fixed squared windows, where n will be the dimension of a vector, and each dimension of this vector represents the ratio of black pixels in relation to the total number of pixels in each window. After some preliminary experiments, the best results were obtained using 25-dimensional vectors.

[4] different normalizations have been applied, but in this work only results with the best behaviour are shown

[5] In order to reduce the length of the strings, all the images were scaled from 128×128 to 64×64.

3 Classifier ensembles

Designing a suitable method of decision combinations is a key point for the ensemble's performance. In this paper, two different combination schemes have been employed, one of them is based on a confidence voting scheme (*sum rule*) [10] and the second is a scheme based on the *filter-and-refine* framework defined in [11].

3.1 Confidence voting approach

In confidence voting methods, voters express the degree of their preference for a candidate. This is done by assigning a value (called the confidence value) to the candidate. The higher the confidence value, the more the candidate is preferred by the voter [1].

Given a test sample x and C classes, to measure the confidence of one classifier in each class, we have used:

$$C_i(x) = \frac{\sum_i \frac{1}{d_i(x)^2}}{\sum_{j=1}^{k} \frac{1}{d_j(x)^2}} \qquad i : 1 \dots C$$

where the k-nearest distances to the sample x have been used. In particular $\sum_i \frac{1}{d_i(x)^2}$ is the sum of the inverse squared distances of the nearest neighbours belonging to the class i among the k nearest neighbours.

Kittler et al. performed in [10] an experimental comparison of different classifier combination schemes. They show that the combination rule called *sum rule* outperforms others combination schemes when different pattern representation of data are used. For this reason in this work we have used this combination (even so, others combination schemes were also tested with worse results).

Sum rule: each voter has to give a confidence value for each candidate. Next, all confidence values are added for each candidate and the candidate (the class) with the highest value of the sum is selected.

3.2 Filter and refine schemes

In domains where the distance measure is computationally expensive, different approaches to accelerate the search have been proposed. For example, approximate cascade classifiers can been used [12]. In this scheme normally inaccurate but cheap classifiers are considered, followed by the more accurate and expensive, ie, the idea is to use at different levels different classifiers (starting by the less expensive). The number of levels depends on the problem and different possibilities can be build for the same problem.

On the other hand, embedding methods have been recently proposed for speeding up the search [11]. These methods embed the strings or trees (or any structural object for which a distance has been defined) in a vector space, so the distances of the embedded objects approximates the actual distances. Thus,

the search can be performed on the embedded objects using, for example, the Euclidean distance.

Embedding methods can be used in a "filter and refine" scheme to accelerate the search [12]. This method consists on:

- firstly, the sample in the embedding space is used as a *"filter"*, where a small set of candidates are selected (the most promising);
- secondly, a new classification in the original space is performed using only the selected set to *"refine"* the result [6].

In this work, two different two-level "filter and refine" schemes have been defined based on the use of these ideas.

4 Experiments

To perform the experiments, binary images of handwritten segmented uppercase characters from the NIST 3 Database National Institute of Standards and Technology were used (see some examples in figure 3):

a) number of classes in this task: 26
b) types of representation: 25-dimensional vectors, strings and trees
c) distances: Euclidean distance, string edit distance, and tree edit distance

Fig. 3. Some examples of upper handwritten characters from the NIST Database 3.

Each experiment were repeated 10 times using training and test sets with 1080 samples on average.

To perform a combination based on the confidence voting scheme, the results of the k-NN rule of individual classifiers with different representations of data has been used as input. In particular, the following combinations have been performed:

[6] in practice, this two steps can be interleaved

C1: strings and vectors
C2: strings and trees
C3: trees and vectors
C4: strings, trees and vectors

Table 1 shows the classification error rate (in %) when the k-NN classifier is used for the three types of representation. The first three columns show the result of the individual classifiers for each representation. In this case, the best results are obtained when the k-NN classifier is used with $k = 1$, and strings outperforms significantly trees and vectors. The four columns on the right of table show the classification error rate for the ensembles. In this case, it can be observed that the classification error rate is influenced by the number of nearest neighbours k, and depends on the combination. Some interesting results are obtained: firstly, depending on the combination, the best results were obtained for k values greater than 1; secondly, any combination of two representation were strings are represented (C1 and C2) improves the results of any individual classifier; thirdly, the best results are obtained using combination C4 with $k = 3$ (using the three representations).

Table 1. Error rate using the k-NN rule with contour strings, trees and vectors, and four combinations for a training set size of 1080 samples on average.

	individual classifiers			**classifier ensembles**			
k	**string**	**trees**	**vector**	**C1**	**C2**	**C3**	**C4**
1	**12.9**	26.4	27.4	19.6	19.4	26.5	13.3
3	13.8	27.4	28.9	11.0	11.5	16.7	**08.9**
5	15.9	30.4	32.4	10.5	10.9	15.8	09.1
7	18.0	33.2	35.6	10.5	11.1	16.2	09.3

As the use of edit distance with strings or trees is very time consuming, other alternatives have been analyzed trying to reduce it. In particular, experiments with two schemes based on refine and filter classifiers and embedding methods were done.

Embedding strings and trees. Strings and trees were embedded as vectors using the edit distance to a selected number of prototypes. That is, an object (string or tree) can be transformed into a vector by calculating the edit distance to all the selected objects, where each distance represents one vector component. Formally, given a set of objects P and a subset $B = \{b_1, \ldots, b_n\} \subseteq P$, the transformation $t_n^B : P \to \mathbb{R}^n$ is defined as a function where $t_n^B(x) = (d(x, b_1), \ldots, d(x, b_n))$, and where $d(x, b_i)$ is the edit distance between the objects x and b_i.

Spillman et al. proposed in [4] some prototype selection methods to use [7]. The method used in this work was the following: the first one, b_1, is randomly selected and then, for $i = 2, 3 \ldots n$

$$b_i = \text{argmax}_{p \in (P - B_i)} \min_{k=1}^{i-1} d(p, b_k),$$

where $B_i = \{b_1, \ldots, b_{i-1}\}$.

The edit distance from these n objects to the training set P are computed, and these n distances are used as the n coordinates of each object in a n-dimensional vector.

The experiment presented in Table 2 were performed to select the dimensionality in the transformed vector space.

Table 2. Classification error using different number of reference points in the embedding.

	dimension			
Type	**50**	**100**	**150**	**200**
strings	21.2	19.8	18.9	18.7
trees	31.1	30.1	30.3	30.0

The combination with the best behaviour in Table 1 (C4: combining strings, trees and vectors) were repeated using the embedding of trees and strings (for $n = 300$). The results can be shown in Table 3. Except for $k = 1$, for the other values of k we found a similar result using the new combination (called C5) than the better individual classifier with the difference that now the classification is faster that in the individual classifiers.

Table 3. Error rate using the k-NN rule with strings, trees and vectors, and combinations of embedded strings, trees with vectors.

	individual classifiers			**combining classifiers**
k	**Emb_string**	**Emb_trees**	**vector**	**C5**
1	22.2	31.4	27.4	18.0
3	21.4	30.5	28.9	12.6
5	21.2	30.0	32.4	**12.1**
7	21.5	30.3	35.6	12.4

Refine and filter classifiers. Two schemes were applied:

[7] these methods are similar than the used in some fast nearest search algorithms based on prototype selection [7]

EmbS-OrS: Firstly, a k-NN rule is used with the embedded strings in a 300-dimensional vector space using the Euclidean distance. Secondly, the 200 nearest samples to the test were used as training set for the second classifier. For this second classifier, the original strings using the string edit distance were used.

Vec-OrS: Firstly, a k-NN rule is used with the 25-dimensional vector representation obtained in section 2.3 using the Euclidean distance. Secondly, the 200 nearest samples to the test were used as training set for the second classifier. For this second classifier, the original strings using the string edit distance were used.

Table 4 shown results with both schemes. It can be observed that the best results are obtained when the embedding method is used in the first classifier.

Table 4. Classification error for filter and refine framework

Second class.	EmbS-OrS	Vec-OrS	Contour string
1	13.2	13.6	12.9
3	12.3	14.5	13.8
5	12.4	16.5	15.9
7	12.9	18.5	18.0

In this work, the main goal for using refine and filter classifiers were the reduction of the time complexity. Table 5 shows this reduction (see the last two columns) were for the Vec-OrS scheme, slightly increases in error rate classification is obtained in relation to the contour string, but reducing up to 18.37% the classification time. However, if our main goal is to reduce the error rate, the combination C4 can be used. In our experiments, we have reduced the error from 12.9% to 8.9%.

Table 5. Classification error and time for all combinations. Contour string error rate and time is used as baseline.

	Contourn string	C4	C5	EmbS-OrS	Vec-OrS
Error (%)	12.9	**8.9**	12.1	12.3	13.6
Time (%)	100	158	**8**	13.9	18.37

5 Conclusions

In this work classifier ensembles using structural information of data have been applied in a handwritten character recognition task. Different ensembles have been evaluated, some of them to improve the classification rate and others to reduce the time complexity. The proposed schemes improve the classification rate or the time consuming in relation to the individual classifiers. To reduce the

time complexity a scheme based on the use of a refine and filter framework an embedding methods have been used, reducing this time until a 18% maintaining similar error rate that the best individual classifier. The combination of trees, strings and vectors reduce the error rate from 12.9% to 8.9% when a training set of 1080 samples on average is used (but at expense of consuming more time). In future work, new schemes using alternative embedding methods and fast approximated nearest neighbours algorithms will be analyzed to reduce both the error rate and the time complexity.

Acknowledgments

This work was supported by the Spanish project CICyT DPI2006-15542-C04-01 and the IST Programme of the European Community, under the Pascal Network of Excellence, IST-2002-506778.

References

1. Kuncheva, L.: Combining Pattern Classifiers: methods and algorithms. Wiley (2004)
2. Duin, R.: The combining classifier: to train or not to train? In: Proceedings of the International Conference on Pattern Recognition ICPR'2002. Volume II., Quebec Canada (2002) 765–770
3. Neuhaus, M., Bunke, H.: Edit distance-based kernel functions for structural pattern classification. Pattern Recogn. **39** (2006) 1852–1863
4. Spillmann, B., Neuhaus, M., Bunke, H., Pekalska, E., Duin, R.: Transforming strings to vector spaces using prototype selection. Lecture Notes in Computer Science **4109** (2006) 287–296
5. Grother, P.: Handprinted forms and character database. In: NIST Special Database 19. (2001)
6. Carrasco, R., Forcada, M.: A note on the Nagendraprasad-Wnag-Gupta thinning algorithm. Patter Recognition Letters **16** (2003) 539–541
7. Rico, J., Micó, L.: Comparison of aesa and laesa search algorithms using string and tree edit distances. Pattern Recognition Letters **24(9)** (2003) 1427–1436
8. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM J. Comput. **18** (1989) 1245–1262
9. Wagner, R., Fischer, M.: The string-to-string correction problem. Journal of the Association for Computing Machinery **211** (1974) 168–173
10. Kittler, J., Hatef, M., Duin, R., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 226–239
11. Hjaltason, G., Samet, H.: Properties of embedding methods for similarity searching in metric spaces. IEEE Transactions on Pattern Analysis and Machine Intelligence **25** (2003) 530–549
12. Athitsos, V., Alon, J., Sclaroff, S.: Efficient nearest neighbor classification using a cascade of approximate similarity measures. In: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1. (2005) 486–493