Some results about the use of tree/string edit distances in a nearest neighbour classification task

Juan Ramón Rico-Juan and Luisa Micó *

Dept. Lenguajes y Sistemas Informáticos
Universdad de Alicante, E-03071 Alicante, Spain,
{juanra, mico}@dlsi.ua.es

Abstract In pattern recognition there is a variety of applications where the patterns are classified using edit distance. In this paper we present some results comparing the use of tree and string edit distances in a handwritten character recognition task. Some experiments with different number of classes and of classifiers are done.

Keywords: nearest neighbour, handwritten character recognition, edit-distance; metric space.

1 Introduction

One of the most useful and simplest techniques in Statistical Pattern Recognition that can be used in a wide range of applications of computer science and technology is the Nearest Neighbour (NN) rule. In this rule, an input pattern is assigned to the class of the nearest prototype pattern. Many times, each class is a set of prototype patterns and a k-NN rule is used: the input pattern is assigned to the class containing the larger fraction of the k nearest prototypes.

A variety of applications can be developed using the NN rule. Some of them are directly related with Pattern Recognition (as the handwritten recognition task), but also in data compression [1], data mining [2] or information retrieval [3].

When patterns may be represented as strings or trees, conventional methods based on a vector representation can not be used. In this case methods that only use a distance (and the metric properties of the distance) and an adequate data structure can be used to perform the classification. Some algorithms as `AESA` [4] and `LAESA` are focused on the reduction in the number of distance computations [5][1]. Others such as Fukunaga [6] are focused on the reduction of the temporal overhead using a tree structure. Recently, a new algorithm based on approaching spatially the searched objects and called sa-tree (spatial approximation tree) was proposed [7].

* Work partially supported by the spanish CICYT TIC2000-1599-C02 and TIC2000-1703-CO3-02

[1] These methods are adequate when the computational cost of the distance is very expensive

Given a particular representation, the edit distance between two objects is defined as the number of insertions, deletions and substitutions needed to transform one representation into the other. In the case of a string representation, insertions, deletions and substitutions are made on the individual symbols of the strings. In the case of a tree representation, insertions, deletions and substitutions are made on the nodes of the tree.

In previous works as [8] the experiments were done using digits (10 classes). In this work, some additional experiments are done using characters (26 classes) to have a better knowledge of the behaviour of two fast search algorithms (`AESA` and `LAESA`) when two different (string and tree) edit distances are used in a handwritten character recognition task.

2 String and tree representation of characters

Two different representations of handwritten characters are done. In both cases, the mathematical morphology opening transformation are used to avoid noisy pixel and to smooth the shapes of characters.

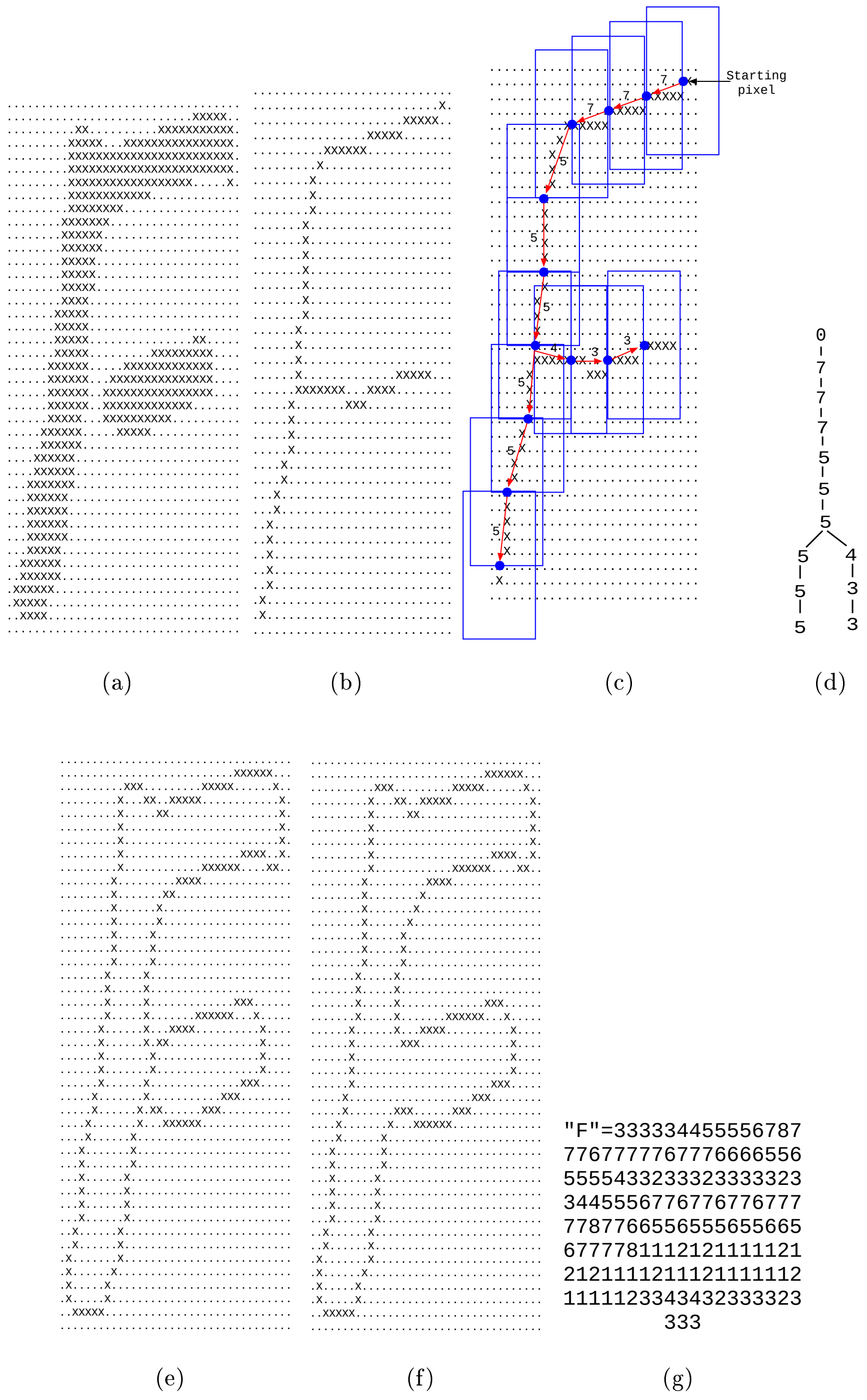
2.1 Tree code

The Nagendraprasad-Wang-Gupta thinning algorithm modified as in [9] was applied (figure 1b). The result image is transformed into a tree representation using the following steps:

1. The first up and left pixel, r, is marked and assigned the tree root with a special label "0". Two empty pixel sets C and G and created.
2. $C \Leftarrow \{r\}$
3. If $C = \emptyset$ go to the end (step 8).
4. For all elements $t \in C$ collect in set G every unmarked pixels into the window (size 11) centred in the pixel associate to t (figure 1c). Follow connected pixels until a below criteria was true:
   (a) the branch has the maximum fixed parameter size (see figure 1b);
   (b) the pixel has no unmarked neighbours (terminal pixel);
   (c) the pixel has more than one unmarked neighbour (intersection pixel).
5. Create the new branches: branch(t, g) : $g \in G$. The label is assigned to the branch depending on the final pixel, g, relative position to the starting one[2], t.
6. $C \Leftarrow G$ and erase all elements from G.
7. Go to step 3.
8. End.

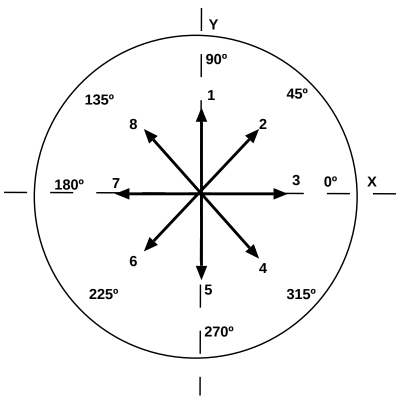
A complete process showing this feature extraction with character 'F' is presented in figure 1.

[2] The 2D space is divided in 8 regions (figure 2) .

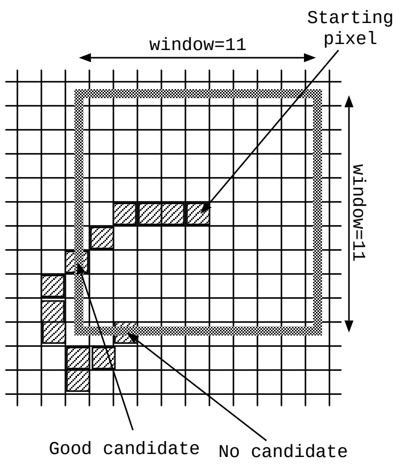


```
"F"=333334455556787
7767777767776666556
5555433233323333323
3445556776776776777
7787766556555655665
6777781112121111121
2121111211121111112
1111123343432333323
        333
```

Figure 1. Example of character “F” (a) original image; (b) thinned image; (c) tree labelling process; (d) final labelled as a tree; (e) problem image to extract the contour string; (f) image right formed to extract contour string; (g) coded string.



(a)



(b)

Figure 2. (a) 2D labelled regions; (b) example to get next candidates to create branches in structured tree extraction.

2.2 String code

The algorithm to extract the coded string from the image is detailed below:

1. Assign $i = 1$.
2. The mathematical morphology opening transformation with i pixels was applied and the algorithm to extract the external contour of the characters is used to obtain the patterns from the images.
3. If the new image contains pixels with 3 o more neighbours, as in figure 1e), the algorithm will have problems to follow the contour, so do $i = i + 1$ and go to step 2.
4. The first black pixel is searched from the left-to-right scan starting from the top. From this pixel going to the right, the border of the character is followed until this first pixel is reached again. During this route the algorithm builds a string with the directions that it follows to find the next pixel of the border[3].

[3] There are eight neighbouring pixels that can be found after a given pixel (figure 1f and 1g), therefore, only eight symbols can appear in this chain-code (see figure 2a)

3 Edit distances

3.1 The tree edit distance

A general tree edit distance is described in [10]. The distance between two ordered trees is considered to be the weighted number of edit operations (insertion, deletion and substitution) to transform one tree into another.

A dynamic programming algorithm is implemented to compute the distance between two trees, T_1 and T_2 whose complexity is in space $O(|T_1| \times |T_2|)$ and time $O(|T_1| \times |T_2| \times \min(\text{depth}(T_1), \text{ leaves}(T_1)) \times \min(\text{depth}(T_2), \text{ leaves}(T_2)))$.

Each basic operation has an associated weight. Substitution weights w_{ij} are $\min(|i-j|, 8-|i-j|)$. Both insertion and deletion have a weight $w_I = w_D = 2$. This distance is finally normalised with the sum of the number of nodes in each tree.

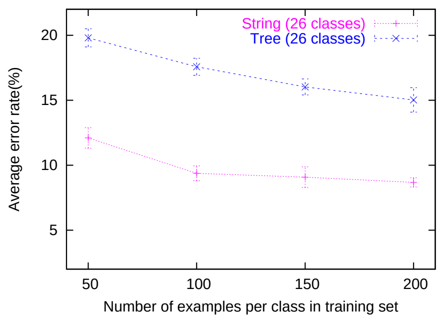
3.2 The string edit distance

The string edit distance is defined as the minimum-cost set of transformations to turn a string into the other. The basic transformations are deletion, insertion and substitution of a single symbol in the string. The cost values are equal as those used in tree edit distance. The string edit distance can be computed in time in $O(|x|, |y|)$ using a standard dynamic-programming technique [11]. As in the tree edit distance, this final measure is normalised, in this case by the sum of the lengths of the strings.

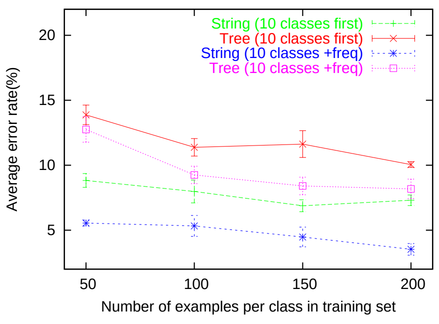
4 Experiments

Two fast approximating-eliminating search algorithms have been used in this work: `AESA` and `LAESA`. These algorithms has been applied in a handwritten character recognition task using the NIST SPECIAL DATABASE 3 of the National Institute of Standards and Technology. Some results using only digits from this data set have been presented in a recent work [8]. In this work new experiments are made using the upper handwritten characters. The increasing-size training samples for the experiments were built by taking 500 writers and selecting the samples randomly. The figures show the results averaged for all combinations.

A first set of experiments using AESA were made to compare the average error rate between the string and the tree edit distances. In these experiments (see figure 3), different number of classes have been used: one set with 26 classes representing all the alphabet, and two different sets of 10 classes (the first 10 characters of the alphabet and the 10 more frequently used characters). In all the cases the use of strings allow to have a better accuracy in the recognition task. However, as figure 4 shows, the average number of distance computations is higher that in the tree representation. Moreover, the computation of the string distance is more expensive than the tree edit distance in average, because the number of symbols in the strings is higher than the number of nodes in the tree.

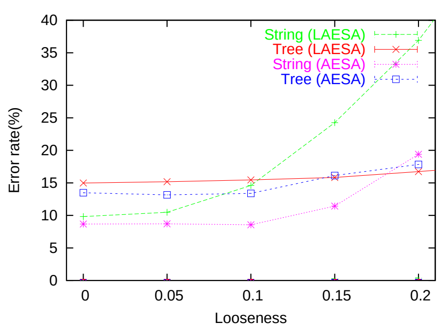


(a)

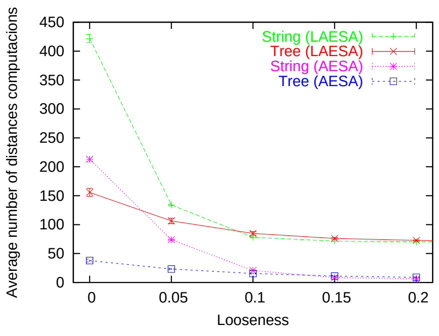


(b)

Figure 3. Average error rate as a function of the different training examples size: (a) 26 character classes (b) two different sets of 10 character classes.



(a)



(b)

Figure 4. Results applying AESA and LAESA algorithms as a function of looseness using 5200 prototypes belonging to 26 character classes: (a) average error rate; (b) average number of distance computations.

The application of the AESA and LAESA algorithms in previous works, as [12] and [13], shows that the “looseness” H in the triangle inequality can be used to reduce the number of distance computations[4].

The performance of both algorithms is compared evaluating the average error rate and the average number of distance computations as a function of the “looseness” [5] (see figure 4). This experiment reveals that the “looseness” is not a critical parameter when the tree representation is used. For any value of H between 0 and 0.2, the error rate in the classification and the average number of distance computations have a slight variation for the tree representation. However, in the case of the string representation there is a large variation. It will be necessary to use a higher value of H to reduce significantly the average number of distance computations in the tree case. The problem is that in this case the average error increases dramatically.

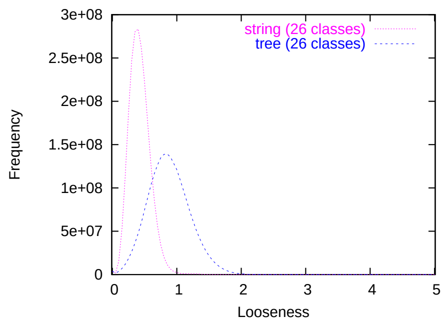


Figure 5. Histograms from looseness using normalised edit distance.

The histograms of the looseness can help to understand the last statement[6]. The figure 5 shows that the smallest looseness is observed for strings. For this reason, the error rate increases for smaller values of H for strings than for trees.

5 Conclusions

In this paper we have done some experiments comparing the performance and the accuracy of a handwritten recognition task using two different representa-

[4] Given a representation space E, the *looseness* is defined for each $x, y, z \in E$ as $h(x, y, z) = d(x, y) + d(y, z) - d(x, z)$. If a histogram of the distribution of $h(x, y, z)$ is computed, this histogram can be used to estimate of the probability that the triangle inequality is satisfied with a *looseness* smaller than H [14].

[5] The size of the set of base prototypes, B is selected to minimise the number of computed distances per sample, so is 70 and 140 for trees and strings, respectively.

[6] The triangle inequality is almost always satisfied for both representations and the distribution is reasonably normal-like.

tions. Our experiments show that the tree edit distance is a suitable choice as opossed to the string edit distance. Although the error rate is higher for the tree representation when no looseness is used, this difference dissapears when the looseness is applied to speed up the classification.

References

1. Allen Gersho and Robert M. Gray. *Vector quantization and signal compression.* Kluwer Academic Publishers, 1991.
2. T. Hastie and 1996. R. Tibshirani. Classification by pairwise coupling. Technical report, Stanford University and University of Toronto, 1996.
3. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw Hill, New York, 1983.
4. Enrique Vidal. New formulation and improvements of the Nearest-Neighbour approximating and eliminating search algorithm(AESA). *Pattern Recognition Letters*, 15(1):1–7, January 1994.
5. L. Micó, J. Oncina, and E. Vidal. A new version of the nearest-neighbour approximating and eliminating searh algorithm with linear preprocessing-time and memory requirements. *Pattern Recognition Letters*, 15:9–17, 1994.
6. K. Fukunaga and P. M. Narendra. A branch and bound algorithm for computing k-nearest neighbours. *IEEE Transactions on Computers*, 24(7):750–753, 1975.
7. Gonzalo Navarro. In *String Processing and Information Retrieval Symposium and International Workshop on Groupware*, pages 141–148. IEEE Press, 1999.
8. J. R. Rico-Juan and L. Micó. Comparison of AESA and LAESA search algorithms using string and tree edit distances. *Pattern Recognition Letters*, 24(9):1427–1436, 2003.
9. R. C. Carrasco and M. L. Forcada. A note on the Nagendraprasad-Wang-Gupta thinning algorithm. *Pattern Recognition Letters*, 16:539–541, 1995.
10. K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18:1245–1262, 1989.
11. R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. ACM*, 21:168–173, 1974.
12. E. Vidal and M. J. Lloret. Fast speaker independent DTW recognition of isolated words using a metric-space search algorithm (AESA). *Speech Communication*, 7:417–422, 1988.
13. L. Micó and J. Oncina. Comparison of fast nearest neighbour classifiers for handwritten character recognition. *Pattern Recognition Letters*, 19:351–356, 1998.
14. Enrique Vidal, Francisco Casacuberta, and H. Rulot. Is the DTW distance really a metric? an algorithm reducing the number of dtw comparisons in isolated words. *Speech Communication*, 4:333–344, 1985.