Finding significant points for a handwritten classification task

Juan Ramón Rico-Juan and Luisa Micó[⋆]

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

Abstract When objects are represented by curves in a plane, highly useful information is conveyed by significant points. In this paper, we compare the use of different mobile windows to extract dominant points of handwritten characters. The error rate and classification time using an edit distance based nearest neighbour search algorithm are compared for two different cases: string and tree representation.
Keywords: feature extraction, nearest neighbour, handwritten character recognition, 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. Examples of application of NN rule include handwritten/speech recognition, data compression [1], data mining [2] and information retrieval [3].

If patterns can be coded in a vector space, methods based on the coordinates of the representation can be applied. However, this is not the general case and often, only methods that use a distance (and the metric properties of the distance) can be applied to perform the classification.

A popular distance used in general metric spaces is the edit distance. The edit distance between two objects is defined as the number of basic operations (insertion, deletion and substitution) needed to transform one representation into another. Depending on the type of representation (for instance, strings or trees) basic operations are differently defined. Each basic operation has associated a weight, usually identical for insertion and deletion ($W_I = W_D$), and a third weight for substitution (W_S) that fulfils the following relationship:

$$W_I + W_D \geq W_S$$

Different algorithms allow to obtain a good code representation of planar objects [4,5,6,7]. These algorithms extract points from a figure that help us to obtain the features to represent it. Some of them obtain a set of approximately equidistant points (EP) [8].

⋆ Work partially supported by the Spanish CICYT under contract and TIC2003-08496-CO4

Others algorithms obtain a set of dominant points with an irregular distribution (IP) [4,5,7].

Classification using the edit distance and IP methods gives poor results (tested in [9]). Improving classification requires the definition of a more complex distance that takes into account the geometric distance between adjacent significant points.

In [8] two different representations of handwritten characters have been used: the contour string and a tree of significant points obtained after using a modified thinning algorithm [10]. The set of points obtained with the second method are almost equidistant because each new point is obtained when a square window is moved a fixed distance in the original image.

In this work we use a mobile circular window to obtain equidistant points for strings and trees as representation of handwritten characters. The number of points in the representation depends on the radius of the window. In section 2 we describe the method to obtain the string and tree code. The edit distance used to compare these representations is introduced in section 3. In section 4, the obtained results are listed while, in section 5, the concluding remarks are offered.

2 String and tree representation of characters

Here, two different representations of handwritten characters have been used. In both cases, the mathematical morphology opening transformation reference are used to avoid noisy pixels and to smooth the shapes of the characters.

2.1 Tree code

The Nagendraprasad-Wang-Gupta thinning algorithm modified as in [10] was applied (figure 1b). The result image is transformed into a tree representation using the following steps:

1. The radius R is selected.
2. The first up and left pixel, r, is marked and assigned the tree root with a special label “0”. Two empty pixel sets C, G and a pair pixel set T are created.
3. $C \leftarrow \{r\}$
4. While $C \neq \emptyset$ repeat steps 5-7.
5. For all elements $t \in C$, collect in set G every unmarked pixels in the circumference of radius R centred in the pixel associate to t. Follow and mark connected pixels until a pixel, g, is found with one of the following properties:
   (a) the branch has the maximum radius R (see figure 4);
   (b) the pixel has no unmarked neighbours (terminal pixel);
   (c) the pixel has more than one unmarked neighbour (intersection pixel).
6. Add to T the new branches $(t,g) : g \in G$. A label is assigned to the branch depending on the final pixel (g), relative position to the starting one[1] (t).
7. $C \leftarrow \{g\} : (t,g) \in G$. Erase all elements from G.
8. end repeat.

[1] The 2D space is divided in 8 regions (figure 3).

9. return T.

An example showing this feature extraction with character 'F' is presented in figure 1.

2.2 String code

After the mathematical morphology opening transform using n pixels [2] is applied the following algorithm is used to extract the external contour of the character .

1. A radius R is selected. String s is empty.
2. The first black pixel, r, is searched with a left-to-right scan starting from the top.
3. From r and clockwise, the contour of the character is followed until a new black pixel, t, is found. This pixel is the intersection between the contour and the circumference centred in r with radius R. Add to the string s the code of the direction[3] (r,t).
4. If $(t \neq \text{last pixel}) r \longleftarrow t$ and go step 3.
5. return s.

3 Edit distances

A general tree edit distance is described in [11]. A dynamic programming algorithm is implemented to compute the distance between two trees, T_1 and T_2 whose complexity is in time $O(|T_1| \times |T_2| \times \min(\text{depth}(T_1), \text{leaves}(T_1)) \times \min(\text{depth}(T_2), \text{leaves}(T_2)))$ and in space $O(|T_1| \times |T_2|)$.

Each basic operation has an associated weight with the following values, used in [6]: substitution $w_{ij} = \min(|i-j|, 8-|i-j|)$ and insertion and deletion $w_I = w_D 2$. This distance is finally normalised with the sum of the number of nodes in each tree.

The cost values on the string edit distance are 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 [12]. As in the tree edit distance, this final measure is normalised, in this case by the sum of the lengths of the two strings.

4 Experiments

A classification task using the NIST SPECIAL DATABASE 3 of the National Institute of Standards and Technology has been done in this work. Only the 26 uppercase handwritten characters were used. The increasing-size training samples for the experiments were built by taking 500 writers and selecting the samples randomly. To perform the NN search, the Approximating Eliminating Search Algorithm, `AESA`, has been used in this work.

[2] n is the smallest positive integer allow to have a external contour where all the pixels have two neighbours.

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

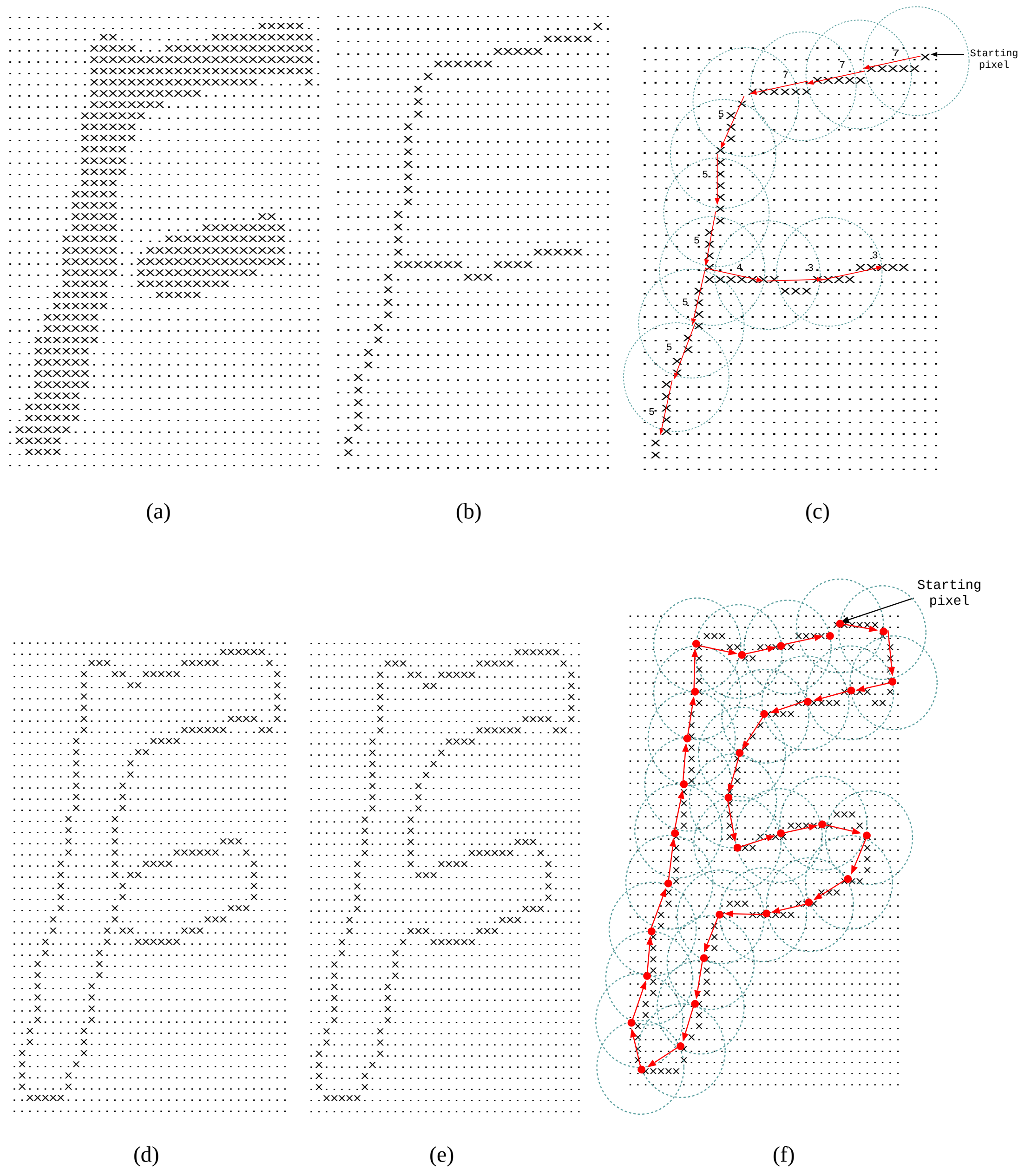


Figure 1. Example feature extraction (a) original image; (b) thinned image; (c) tree labelling process; (d) image with problems to extract a contour string; (e) image right formed to extract a contour string; (f) string labelled process.

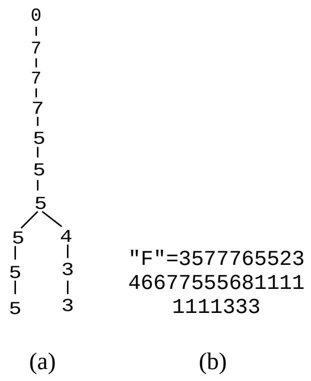


Figure 2. Example of extracted features from character in figure 1 (a) tree; (b) string.

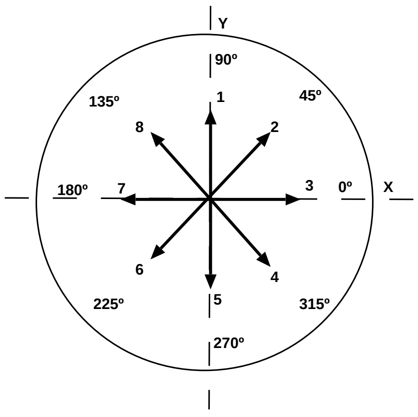


Figure 3. 2D labelled regions

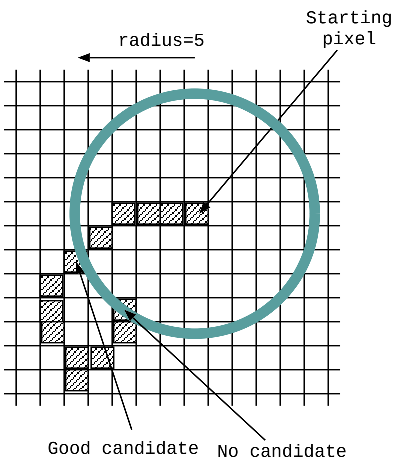


Figure 4. Example to get next candidates to create branches in structured tree extraction.

Figure 5 shows the comparison between the error rate in a classification task is evaluated for different sizes, R, of the two types of windows: the square window used in previous work[8] and a circular window. The figure shows the average error rate using a training set of 200 samples per class. This experiment shows that the error rate grows linearly with the radius of the circular window. However, for relatively small windows, the error rate is smaller using a circular window than a square window.

Figures 6 and 7 shows the behaviour of the error rate and the classification time when a circular window is used. In this case, different radius of the window ($R = 1, 2, 4, 8$) with different sizes of the training set have been used for handwritten characters represented as strings and trees.

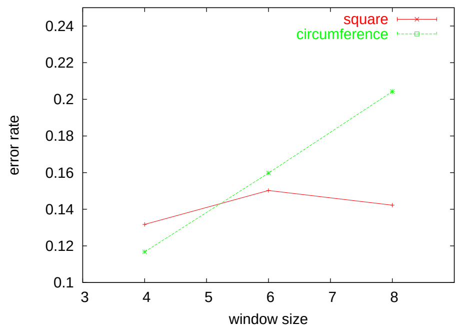
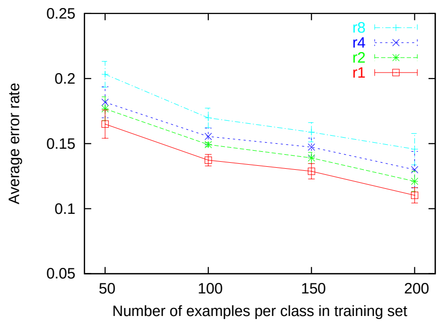
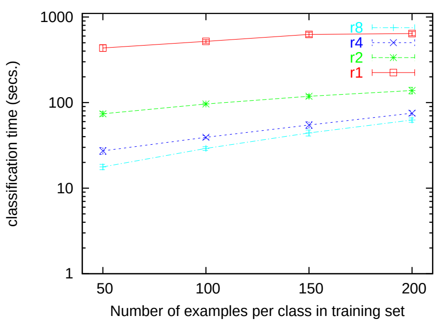


Figure 5. Results applying NN classification algorithm with different types of window approximation



(a)



(b)

Figure 6. Results for NN classification with `AESA` search with tree representations of character obtained with different sizes of the window as a function of different training examples size belonging to 26 character classes: (a) average error rate; (b) average classification time.

In all cases the use of strings generates a lower error rate in the recognition task than the use of a tree representation, although the classification time is higher. However, as shown in figures 6 and 7 larger values of the radius of the window allow to reduce the classification time at a little increase in the error rate.

On the one hand, the use of a circular window with the string representation improves the classification error rate (compared to the tree representation) with a radius of the window less or equal than 4.

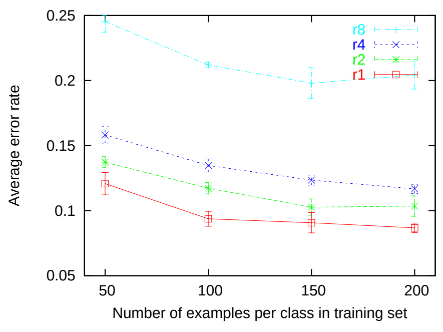
On the other hand, when the radius grows using a string code, the classification time tends to be similar than that using a tree code.

5 Conclusions

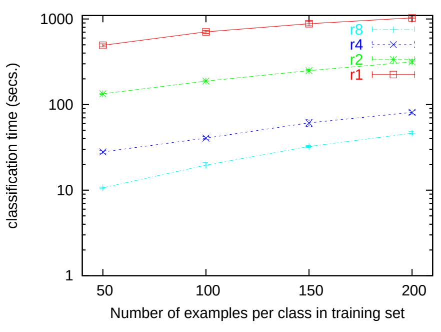
In this paper we have compared the performance and the accuracy of a handwritten recognition task using two different representations (strings and trees) obtained with a circular window. Our experiments show that better results in a classification task are obtained when a circular window with radius higher to one are used for a string representation of the handwritten characters.

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.



(a)



(b)

Figure 7. Results for NN classification with `AESA` search with string representations of character obtained with different sizes of the window as a function of different training examples size belonging to 26 character classes: (a) average error rate; (b) average classification time.

4. X. Li and D. Yeung. On-line handwritten alphanumeric character recognition using dominant points in strokes. *Pattern Recognition*, 30:31–34, 1997.
5. J. M. Iñesta, B. Mateo, and M. A. Sarti. Reliable polygonal approximations of imaged read objects though dominant point detection. *Pattern Recognition*, 31:685–697, 1998.
6. 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.
7. B. Sarkar, S. Roy, and D. Sarkar. Hierarchical representation of digitized curves though dominant point detection. *Patter Recognition Letters*, 24:2869–2882, December 2003.
8. J. R. Rico-Juan and L. Micó. Some results about the use of tree/string edit distances in a nearest neighbour classification task. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Pattern Recognition and Image Analysis*, number 2652 in Lecture Notes in Computer Science, pages 821–828, Puerto Andratx, Mallorca, Spain, june 2003. Springer.
9. J. R. Rico-Juan. Off-line cursive handwritten word recognition based on tree extraction and an optimized classification distance. In M. I. Torres and A. Sanfeliu, editors, *Pattern Recognition and Image Analysis: Proceedings of the VII Symposium Nacional de Reconocimiento de Formas y Análisis de Imágenes*, volume 3, pages 15–16, Bilbao (Spain), May 1999.
10. R. C. Carrasco and M. L. Forcada. A note on the Nagendraprasad-Wang-Gupta thinning algorithm. *Pattern Recognition Letters*, 16:539–541, 1995.
11. 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.
12. R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. ACM*, 21:168–173, 1974.