Handwritten character recognizers that using tree-edit-distance and fast nearest neighbour search

Juan Ramón Rico-Juan
Departament de Lenguatges i Sistemes Informàtics (Universitat d'Alacant)
E-03080 Alacant (Spain)
juanra@dlsi.ua.es

Jorge Calera-Rubio
Departament de Lenguatges i Sistemes Informàtics (Universitat d'Alacant)
E-03080 Alacant (Spain)
calera@dlsi.ua.es

Keywords: Classification, Nearest neighbour, Tree-edit-distance, Handwritten character recognition

Abstract: Although the rate of well classified prototypes using tree-edit-distance is satisfactory, the exhaustive classification is expensive. Some fast methods as AESA and LAESA have been proposed to find nearest neighbours in metric spaces. The average number of distances computed by these algorithms does not depend on the number of prototypes. In this paper we apply these classifiers algorithms to the task of handwritten character recognition and obtain a low average error rate (2%) and a fast classification.

1 Introduction

Pattern classification based on the nearest-neighbour (NN) decision rule is very popular in Pattern Recognition (Dasarathy 1991). Apart its simplicity and excellent properties, the NN rule allows us to design pattern classifiers that can directly manage any type of data. This advantage is of great interest since there are many Pattern Recognition problems involving data sets which cannot be adequately represented in a suitable vector space, though a distance function or metric is available to measure the dissimilarity between data points (Vidal 1986). In our case, the prototypes are labelled trees and we use the tree-edit-distance (Zhang and Shasha 1989). The high cost associated with the computation of distances make the NN searching process too expensive to implement practical efficient systems. Therefore, algorithms for fast NN searching in general metric spaces would help in this task.

In order to explore the application of this kind of algorithms to this problem, we apply some fast NN classifiers to the NIST SPECIAL DATABASE 3 of National Institute of Standards and Technology also used in Gómez et al. (1995), Micó and Oncina (1998), López and Piñaga (2000) and Pérez-Cortés et al. (2000).

2 Feature extraction

The Arcelli and di Baja (1985) thinning algorithm to eliminate redundant information was applied to all binary images used in the training set. These images are the start point of the process of the tree representation. We followed the similar process that López and Piñaga (2000):

1. The first left pixel is chosen and assigned the tree root with a special label "0".
2. Every tree node has so many descendents as neighbours has the selected pixel.
3. Each branch is extended following the neighbour until one of this criteria are true:
   (a) the branch has the maximum fixed parameter size *window*=8*;*
   (b) the pixel has no neighbours (terminal pixel);
   (c) the pixel has more than a neighbour (intersection pixels).
4. A new node is assigned to every pixel in the step 3. The node is labelled as show in the equation 1. Result of dividing 2D space in 8 regions; the north is labelled with **"**1**"** and clockwise, the rest is labelled. When the label of a segment has to be obtained, the starting pixel shall be placed at the

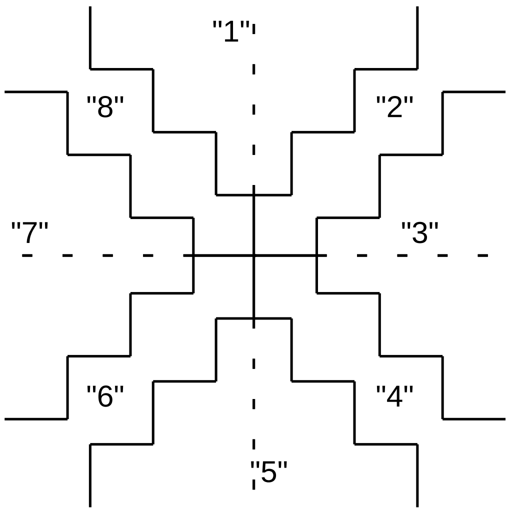

Figure 1: 2D labelled regions

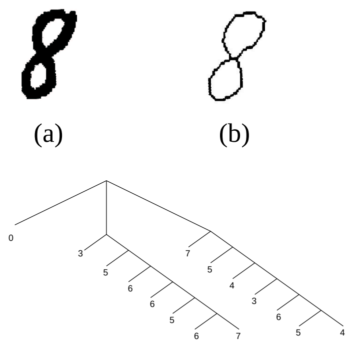

Figure 2: A digit example: (a) original image; (b) thinned image; (c) final labelled tree

origin of the axes; the labels is assigned to the final pixel depending on its relative situation to the starting one (figure 1). These labels are similar as codes used in Freeman (1961) except the special code "0" is assigned to the root label and the rest of 2D directions start at "1" until "8".

$$\begin{cases} \text{if } x \geq 0 & \begin{cases} y = \pm 2\lceil x \rceil \\ y = \pm \frac{\lfloor x \rfloor}{2} \end{cases} \\ \text{otherwise} & \begin{cases} y = \pm 2\lceil x-1 \rceil \\ y = \pm \frac{\lfloor x+1 \rfloor}{2} \end{cases} \end{cases} \tag{1}$$

5. For each node with neighbours, go to step 2.

The trees obtained with this process have labelled nodes. To transform these trees in skeletons without loss of information, the below operator is applied:

$$\begin{array}{l} \underset{\forall a \in \Sigma}{Sk(a) = a} \\ \underset{\forall t_1,\ldots,t_n \in \Sigma^T, \sigma \in \Sigma}{Sk\left(\sigma\left(t_1,\ldots,t_n\right)\right) = \alpha\left(\sigma, Sk\left(t_1\right),\ldots,Sk\left(t_n\right)\right)} \end{array} \tag{2}$$

Where Σ is the labels (from "0" until "8"), Σ^T is the set of all trees using Σ as node label, $\sigma\left(t_1,\ldots,t_n\right)$ is the label of this node with its n descendents and α is a special symbol which doesn't belong to Σ set. A feature example is showed in the figure 2.

3 The tree-edit-distance

A general tree-edit-distance is described in Zhang and Shasha (1989). In this paper the weights used by the distance have been adapted to obtain better results. The distance between two ordered trees is considered to be the weighted number of edit operations (insert, delete and modify) to transform one tree to another. A dynamic programming algorithm is implemented to compute the distance between T_1 and T_2 in time $\mathcal{O}\left(|T_1| \times |T_2| \times d_1 \times d_2\right)$ where $d_1 = \min\left(\text{depth}\left(T_1\right), \text{ leaves}\left(T_1\right)\right)$, $d_2 = \min\left(\text{depth}\left(T_2\right), \text{ leaves}\left(T_2\right)\right)$ and space $\mathcal{O}\left(|T_1| \times |T_2|\right)$.

Let us consider the three kinds of operations. Substituting n means changing the label on n (figure 3). When deleting a node n, the children of n become the children of the parent of n and n is removed (figure 4). Insertion is complementary of deletion. This means that inserting n as the child of n' will make n the parent of a consecutive subsequence of the current children of n' (figure 5).

Each basic operation has a associated weight. We use the weights proposed in Rico-Juan (1999). The insertion and deletion has a weight $w_I = w_D = 2$. Substitution weights w_{ij} are

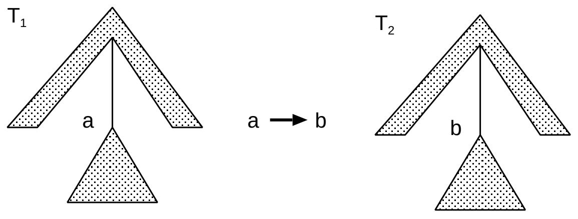

Figure 3: Substitution of node b by a

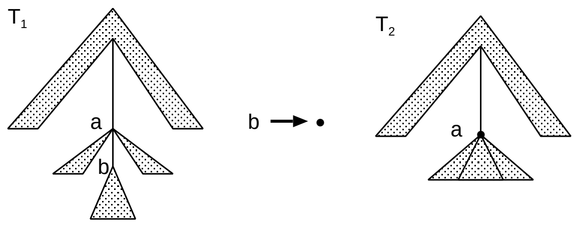

Figure 4: Deletion of node b.

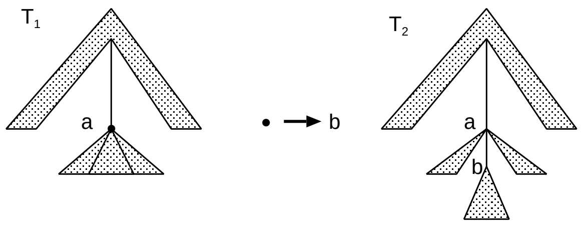

Figure 5: Insertion of node b.

$$w_{ij} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 3 & 2 & 1 \\ 1 & 0 & 1 & 2 & 3 & 4 & 3 & 2 \\ 2 & 1 & 0 & 1 & 2 & 3 & 4 & 3 \\ 3 & 2 & 1 & 0 & 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 & 0 & 1 & 2 & 3 \\ 3 & 4 & 3 & 2 & 1 & 0 & 1 & 2 \\ 2 & 3 & 4 & 3 & 2 & 1 & 0 & 1 \\ 1 & 2 & 3 & 4 & 3 & 2 & 1 & 0 \end{bmatrix}$$

Figure 6: Weights of substitution operation

$\min(|i-j|, 8-|i-j|)$. This equation assign smaller weights to close directions and larger weights to distant directions as shown in the figure 6.

4 The algorithms

Classification by NN (Dasarathy 1991) consists in assigning every sample the same class as the nearest neighbour in a set of prototypes P.

In this paper three methods are compared:

1. The simply exhaustive search where a new sample t is compared with all database prototypes.
2. The Approximating and Eliminating Search Algorithm (AESA). This algorithm can be applied to general metric spaces and stands out as the fastest one in terms of distance computations required during NN search. Although it was originally developed using geometric arguments (Vidal 1986), an improved version which formally adheres the algorithmic strategy of (best-first) Branch & Bound has been proposed (Vidal 1994). According to the latter version, the AESA searches for the distance form a test sample to its NN prototype though a very tight Triangle-Inequality-based *lower bound* function, both for selecting prototypes which are gradually closer to the test sample (Approximation), and for pruning-out those prototypes whose lower bound estimates are not lesser than the smallest distance found form the test sample to the prototypes which have been already examined (Elimination). In addition to the accuracy of the estimates provided by this bounding function, it can be cheaply computed at the expense of preprocessing the matrix of pairwise distances between prototypes. As a result, for data sets involving affordable space requirements, empirical test has shown that the AESA performs NN search in constant average time.
3. Linear AESA (Micó et al. 1994, LAESA) is similar to previous algorithm. However, LAESA uses a linear array of distances computed in preprocessed time. The procedure starts by selecting from the given prototypes a relative small set, called “base prototypes” (BP), and computing the distances between them and the complete set of prototypes. One of the best strategies for selecting a subset of n base prototypes $B = \{b_1, ..., b_n\}$ is proposed in (Micó 1996). The first one, b_1, is randomly selected and then, for $i = 2, 3, ..., n$

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

# Prototypes	μ	D
1000	95.00	0.87
2000	96.25	0.72
3000	96.63	0.60
4000	97.26	0.49
5000	98.00	0.40

Table 1: Average classification error-rate (μ) and standard deviation (D).

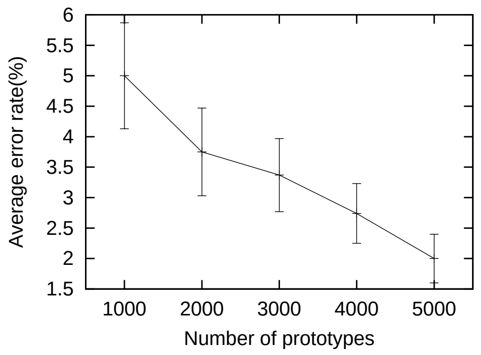

Figure 7: Average error rate obtained when classifying some sets of digits in the NIST database.

This strategy is slightly better than the proposed in the original paper (Micó et al. 1994). In this paper the BP prototypes never are pruning in Elimination conditions.

5 Results

The labelled trees used in the training set of experiments are the same as the classification proofs made in López and Piñaga (2000) where a edit distance between a tree and a deterministic tree automaton (López et al. 2000) was used to classify with a 20% error-rate.

We divided in 5 sets of different size 1000, 2000, 3000, 4000 y 5000, using from each subset 75% of the prototypes for training and the 25% as test.

The accuracy of NN search applied to this problem is shown in table 1 and graphically in figure 7. We can see as the error rate is always below 3%.

In figure 8, the average number of computed distances to classify a prototype is shown. While the distances computed by exhaustive algorithm grows linearly with respect to the size of training set, the number computed by AESA tends to constant (13), and those computed by LAESA tends to a higher constant (40).

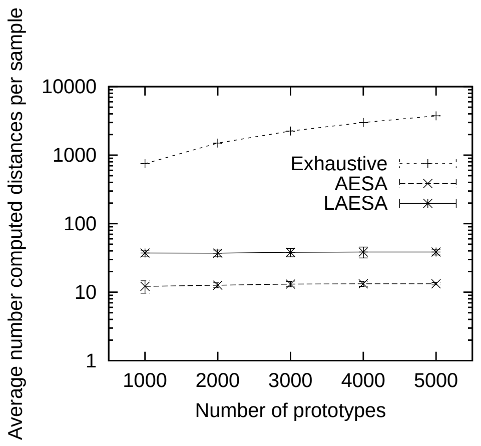

Figure 8: Average number of distances computed by the algorithms as a function of the numbers of prototypes $|P|$.

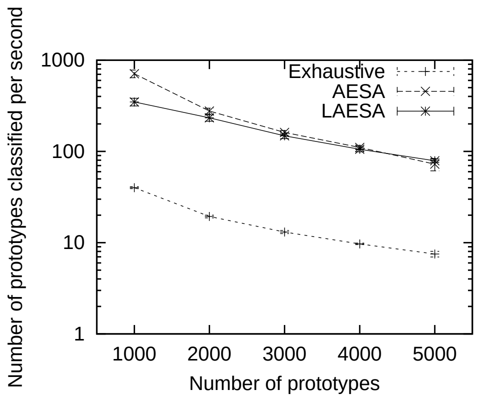

Figure 9: Average number of samples classified per second by the algorithms, as a function of the numbers of prototypes $|P|$.

Figure 9 shows the number of prototypes classified per second. They decrease when the training set size grows, but AESA and LAESA much higher rate (10 times exhaustive).

An AMD Athlon 1.2 GHz with Linux O. S. was used to perform the experiments.

6 Conclusions

This paper shows that NN search combined with tree-edit-distance is an efficient method in Handwritten Character Recognition. The AESA and LAESA algorithms compute very few distances in order to find NN in metric spaces where exhaustive search is prohibitive.

In previous works (Micó and Oncina 1998) a *looseness* was needed to compute less distances per prototype to make efficient the real task but the error rate grows. In this case the results obtained with AESA and LAESA are satisfactory even without the looseness.

In Gómez et al. (1995) the NIST prototypes are represented as chains of 2D codes but in this case the error rate is slightly better.

Acknowledgements

The authors wish to thank Rafael C. Carrasco, M. Luisa Micó and Paco Moreno for useful comments.

References

Arcelli, C. and di Baja, G. S. (1985). A width-independent fast thinning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:463–474.

Dasarathy, B. (1991). Nearest Neighbour (NN) norms: NN pattern classification techniques. *IEEE Society Press*.

Freeman, H. (1961). On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computer*, 10:260–268.

Gómez, E., Micó, M. L., and Oncina, J. (1995). Testing the linear approximating eliminating search algorithm in handwritten character recognition tasks. In *Actas del VI Simposium Nacional de Reconocimiento de Formas y Analisis de Imagenes*, pages 212–217, Córdoba (Spain).

López, D. and Piñaga, I. (2000). Syntactic pattern recognition by error correcting analysis on tree automata. In Ferri, F. J., Iñesta, J. M., Amin, A., and Pudil, P., editors, *Advances in Pattern Recognition*, volume 1876 of *Lecture Notes in Computer Science*, pages 133–142, Berlin. Springer-Verlag.

López, D., Sempere, J. M., and García, P. (2000). Error correcting analysis for tree languages. *International Journal of Pattern Recognition and Artificial Intelligence*, 14(3):357–368.

Micó, M. L. (1996). *Algoritmos de búsqueda de vecinos más próximos en espacios métricos*. PhD thesis, Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y Computación.

Micó, M. L. and Oncina, J. (1998). Comparison of fast nearest neigbour classifiers for handwritten character recognition. *Pattern Recognition Letters*, 19:351–356.

Micó, M. L., Oncina, J., and Vidal, E. (1994). 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.

Pérez-Cortés, J. C., Llobet, R., and Arlandis, J. (2000). Fast and Accurate Handwritten Character Recognition Using Approximate Nearest Neigbours Search on Large Databases. In Ferri, F. J., Iñesta, J. M., Amin, A., and Pudil, P., editors, *Advances in Pattern Recognition*, volume 1876 of *Lecture Notes in Computer Science*, pages 767–776, Berlin. Springer-Verlag.

Rico-Juan, J. R. (1999). Off-line cursive handwritten word recognition based on tree extraction and an optimized classification distance. In Torres, M. I. and Sanfeliu, A., editors, *Pattern Recognition and Image Analysis: Proceedings of the VII Symposium Nacional de Reconocimiento de Formas y Análisis de Imágenes*, volume II, pages 15–16. AERFAI, Geneve.

Vidal, E. (1986). An algorithm for finding nearest neighbours in (approximately) constant avarage time. *Pattern Recognition Letters*, 4:145–157.

Vidal, E. (1994). New fromulation and improvements of the Nearest-Neigbour approximating and eliminating search algorithm(AESA). *Pattern Recognition Letters*, 15(1):1–7.

Zhang, K. and Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18:1245–1262.