An empirical method to improve edit-distance parameters for a nearest-neighbor-based classification task

Eva Gómez-Ballester, Mikel L. Forcada, María Luisa Micó

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

Abstract

This paper describes a method for automatically improving the parameters of the edit distance used in a given nearest-neighbor pattern classification task. The method uses a simple self-consistent procedure which assigns costs to the basic editing operations according to their *discriminative power*. The discriminative powers of operations are defined based on the statistics of their use when a prototype set is used to classify itself. The parameters obtained are not optimal but may be obtained easily and the experimental evidence —a handwritten numeral classification task— shows that the performance of the classifier may be improved easily by using the proposed method on the prototype set.

Keywords: Distance method; hand-written character recognition; edit-distance optimisation

1 Introduction

The issue of pattern classification is central to many applications of computer science and technology. In most of them, it may occur that a given pattern, in general, does not exactly match any of the prototypes used to represent the classes. In these cases, approximate matching techniques which use a conveniently defined distance may be used to classify an unknown pattern: nearest-neighbor technique assigns the pattern to the class that contains the prototype to which the pattern has the least distance.

When patterns may be represented as strings, and classes are represented by sets of prototype strings, it is customary to use the so-called *edit* or Levenshtein distance [1], defined as the minimum number of *edit operations* (symbol substitutions, insertions and deletions) needed to transform one string into another. The edit distance is usually computed using a dynamic programming algorithm [2]. One may use a *weighted* Levenshtein distance, where the contribution of symbol deletions, symbol insertions, and each possible symbol substitution is different [3]. The optimum values of the weighting parameters may vary from task to task. The purpose of this work is to propose a simple empirical method to obtain an improved set of weighting parameters (costs) for a given pattern classification task, as assessed by monitoring the variation of the error rate when classifying a known pattern sample.

The objective of the proposed method is to obtain a set of weighting parameters (or costs) that improves classification. To that end, a "discriminating power" is defined for each edit distance operation. When the operation has a high discriminating power, that is, when it appears more frequently in the edit traces[1] of pairs of strings belonging to different classes and less frequently in the edit traces of pairs of strings belonging to the same class, it is reasonable to assign a high cost to that operation. Conversely, when an operation has a low discriminative power, that is, when it appears more frequently in the edit traces of pairs of strings belonging to the same class and less frequently in the edit traces of pairs of strings belonging to different classes, it is reasonable to assign a low cost to the operation. When costs are assigned in this way, inter-class distances tend to increase and intra-class distances tend to decrease, so that classes tend to be more clearly separated. In some sense, the representation space "warps" due to the changes in the metric (distance) used [4] [5] [6].

However, modifying edit costs results in changes in the actual edit traces obtained when distances are computed again. Edit operation statistics change, and the need for new adjustments to the costs arises.

[1] The edit trace is the sequence of edit operations that transform one string into another

We propose a simple empirical procedure to assign values to costs based on discriminating powers defined on inter-class and intra-class edit operation statistics. The procedure is repeated until the edit costs and the observed statistics are consistent. In this paper, the only edit costs that are modified are (symmetric) substitution costs, which are allowed to vary between 0.0 and 2.0; insertion and deletion costs are fixed to 1.0. This ensures that the edit distance remains a true distance satisfying the three conditions:

- the edit distance of a string to itself is zero, that is $d(u, u) = 0\ \ \forall u$
- edit distances are symmetric, $d(u, v) = d(v, u)\ \ \forall u, v$, and
- edit distances satisfy the triangular inequality: $d(u, v) + d(v, w) \geq d(u, w)$, $\forall u, v, w$

and that a substitution is always less costly than a deletion followed by an insertion or vice versa.

The next section describes the method in detail, and section 3 describes its application to a handwritten digit classification task. Finally, the results are discussed, and conclusions are drawn, in section 4.

2 Method

Let us consider a nearest-neighbor classification task involving a prototype set P partitioned in N_c classes. Let a_i and a_j be symbols in the alphabet Σ from which symbols are drawn to build the strings. Let p_{ij} be the cost of substituting symbol a_i by symbol a_j. Costs are required to be symmetric, $p_{ij} = p_{ji}$. A superscript will be used to denote the iteration number: $p_{ij}^{[t]}$ is the cost of substituting a_i by a_j in iteration t.

The method starts by assigning initial values to all substitution costs, $p_{ij}^{[0]}$. A reasonable choice is to give all costs the value 1.0, or a random value from a small interval around 1.0 ($p_{ij}^{[0]} \in [1.0 - \epsilon, 1.0 + \epsilon]$). This last choice is made to reduce the probability of the fortuitous case in which, for a set of edit costs, there exists more than one edit trace that has the minimum cost.

In each iteration t the set of costs obtained in the previous iteration ($p_{ij}^{[t-1]}$) is used to compute the edit distances for all possible pairs of strings in the prototype set. The statistics of edit operations is used to compute $n_{ij,CD}^{[t-1]}$, the number of times the substitution ij appears in operations involving classes labeled C and D in iteration $t-1$. Then, the discriminating power $Q_{ij}^{[t-1]}$ of operation ij in iteration $t-1$ is computed as follows:

$$Q_{ij}^{[t-1]} = \frac{\sum_{D>C}^{N_c} \sum_{C=1}^{N_c} n_{ij,CD}^{[t-1]}}{\sum_{D>C}^{N_c} \sum_{C=1}^{N_c} n_{CD}^{[t-1]}} \left(\frac{\sum_{C=1}^{N_c} n_{ij,CC}^{[t-1]}}{\sum_{C=1}^{N_c} n_{CC}^{[t-1]}} \right)^{-1}, \tag{1}$$

where

$$n_{CD}^{[t-1]} = \sum_{\substack{k, l \in \Sigma \\ k \neq l}} \sum_{k \in \Sigma} n_{kl,CD}^{[t-1]} \tag{2}$$

When substitution ij appears very often between strings in different classes and seldom between string in the same class, Q_{ij} is larger than 1, and smaller than 1 in the opposite case. If the operation appears as frequently between different classes as it does inside the same class, Q_{ij} equals one.

The values of $Q_{ij}^{[t-1]}$ are used to obtain the new values of the edit costs using the following empirical formula:

$$p_{ij}^{[t]} = (1 - \lambda) p_{ij}^{[t-1]} + \lambda f_a(Q_{ij}^{[t-1]}), \tag{3}$$

where $\lambda \in [0, 1]$ is a parameter used to control the convergence of the iterative process and $f_a(x) = 2x^a/(1 + x^a)$ with $a > 0$. Function f has been chosen to ensure the following properties:

- $f_a(0) = 0$,
- $f_a(1) = 1$, and
- $\lim_{x\to\infty} f_a(x) = 2$,

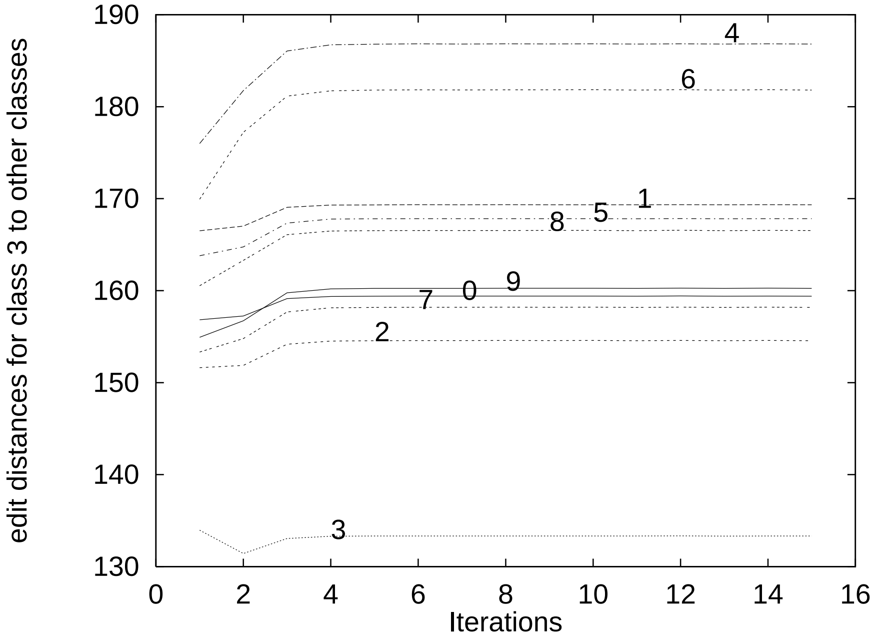

Figure 1: Average edit distance from strings in class 3 to strings in the remaining classes using a training set of 500 prototypes at each iteration of the proposed method.

but other choices would be possible.

Therefore, costs are updated toward a maximum value of 2 when the corresponding operations have a high discriminating power and toward a minimum value of 0 when they have a low discriminating power. The parameter a controls the steepness of the function around $x = 0$: for high values of a, costs are pushed hard towards the extreme values of 0 and 2, whereas small values of a yield values of weights around 1 unless the discriminating powers are either very high or very low.

The method is expected, although not guaranteed, to converge to a consistent set of discriminating powers and edit costs (values of λ below 1.0 are used to avoid oscillations where possible), and the error rate of classifying each one of the prototypes among all the classes should decrease as this point of consistency is approached.

The final error rate will depend on the actual choice of a; therefore, this parameter may be used to "tune" the method for optimum performance. To avoid *overfitting* the parameters to irrelevant details of the prototype set, we propose to use a cross-validation technique: the prototype set is partitioned in two sets: the *training* set, used to estimate the edit costs, and the *validation* set, used to monitor the error rate. The best set of costs is the one that lowers the error rate of both the training and the validation sets.

3 Experimental results

All experiments have been made using strings over an alphabet of eight direction vectors as representation of handwritten digits. These vectors represent the contour of the digits in an image of 128×128 pixels [7]. Initial values of substitution costs were fixed to 1, although using random initial values (see section 2) resulted in similar behaviour.

The first group of experiments was made for studying the behavior of the proposed method with different values of parameters a and λ. Experiments show that the convergence of edit cost, edit distance and success rate is faster and more stable using $\lambda = 1$. Using $\lambda = 1$, an additional set of experiments was carried out to study the behaviour when changing the parameter a. These experiments show that the best results of the success rate and earlier convergence are obtained when $a = 1$. For larger values of a the method shows oscillations. With these two values of parameters a and λ, experiments were made using sets of 500 prototypes. The resulting edit costs were used with validation sets of the same size.

Results show that edit distances between different classes increase in the earlier iterations of the

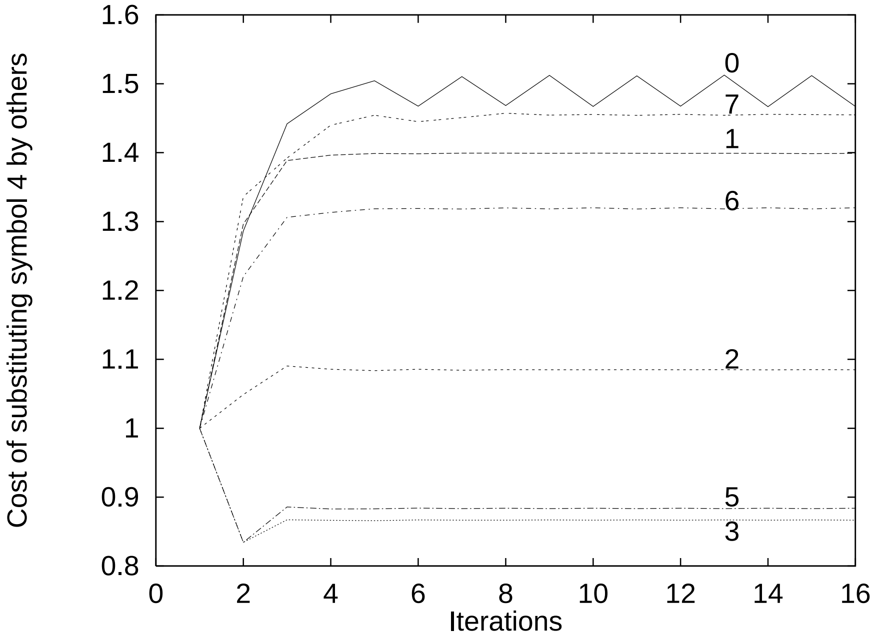

Figure 2: Substitution cost from symbol 4 to all other directions using a training set of 500 prototypes at each iteration of the proposed method.

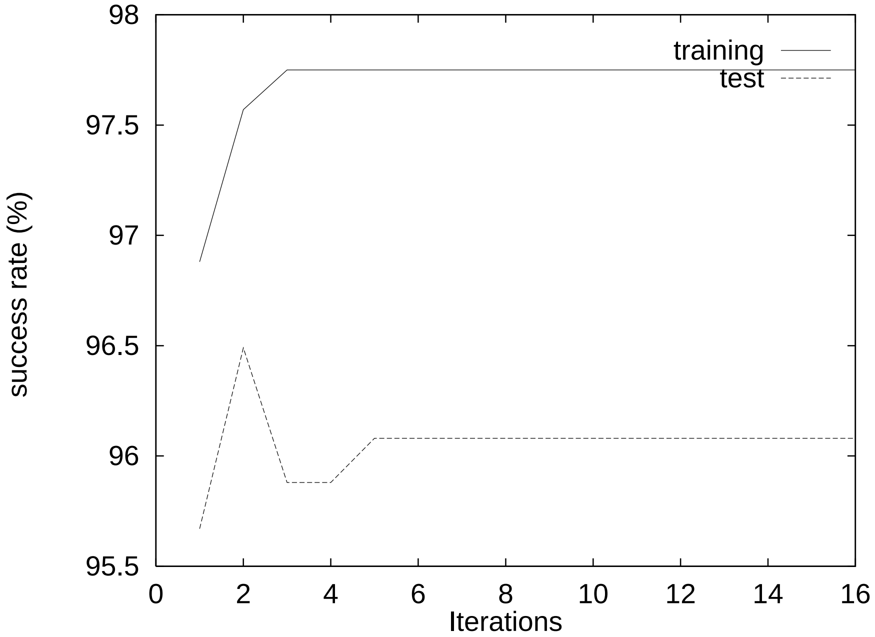

Figure 3: Success rate for both the training and the validation sets using sets of 500 prototypes at each iteration of the proposed method.

methods, and decrease (or do not increase) for the same class (see an example in figure 1). These values tend to a constant as the number of iterations grows. This behaviour is also observed for the edit costs (see figure 2), and the success rate (see figure 3). Figure 3 also shows the success rate for the validation set (test set) using the edit cost obtained in each iteration. The success rate for validation set increase in the same way as that for the training set.

4 Concluding remarks

The optimum edit cost set cannot be exactly reached since we have chosen a particular empirical function relating the statistics of edit operations and the edit costs, not the exact function that would correspond to the minimum of the success rate. We are currently working on methods to attain the minimum and plan to extract from these information on how to construct function f_a. Also, we have observed that the use of the improved edit costs reduces the computational cost (in terms of the number of computed distances) of a fast nearest neighbour classifier [8] [9].

Acknowledgements: The authors wish to thank the Comisión Interministerial de Ciencia y Tecnología of the Government of Spain for financial support through projects TIC93-0633-C02 and TIC95-0984-C02-01, and Rafael Carrasco and José Oncina for interesting suggestions.

References

1. V. Levenshtein: "Binary codes capable of correcting deletions, insertions, and reversals", *Soviet Phys. Dokl.* **6** (1966) 157–169.

2. P.H. Sellers: "The theory and computation of evolutionary distances: Pattern recognition", *J. Algorithms* **1** (1980) 359–373.

3. Andres Marzal, Enrique Vidal: "Computation of Normalized Edit Distance and Applications" *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15** (1993) 926–932.

4. H. Bunke, J. Csirik: "Inference of edit costs using parametric string matching", in Proc. 11th IAPR Internat. Conference on Pattern Recognition (The Hague, The Netherlands, 30 Aug.–3 Sep. 1992), vol. 2, conf. B: Pattern Recognition Methodology and Systems, 549–552.

5. H. Bunke, J. Csirik: "Parametric String Edit Distance and Its Application to Pattern Recognition", IEEE Transactions on Systems, Man, and Cybernetics **25** (1995) 202–206.

6. W.P. de Waard: "An optimised minimal edit distance for hand-written word recognition" *Pattern Recognition Letters* **16** (1995) 1091–1096.

7. Eva Gómez-Ballester, María Luisa Micó and José Oncina: "Testing the Linear Approximating Eliminating Search Algorithm in Handwritten Character Recognition Tasks", Actas del *VI Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes*, (Córdoba, 3–7 Abril 1995) 212–217.

8. Enrique Vidal: "New formulation and improvements of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA)" *Pattern Recognition Letters* **15** (1994) 1–7.

9. María Luisa Micó, Jose Oncina and Enrique Vidal: "A new version of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA) with linear preprocessing time and memory requirements" *Pattern Recognition Letters* **15** (1994) 9–17.