Online and Offline Character Recognition Using Alignment to Prototypes

Jonathan Alon, Vassilis Athitsos, and Stan Sclaroff
Computer Science Department
Boston University
Boston, MA 02215

Abstract

Nearest neighbor classifiers are simple to implement, yet they can model complex non-parametric distributions, and provide state-of-the-art recognition accuracy in OCR databases. At the same time, they may be too slow for practical character recognition, especially when they rely on similarity measures that require computationally expensive pairwise alignments between characters. This paper proposes an efficient method for computing an approximate similarity score between two characters based on their exact alignment to a small number of prototypes. The proposed method is applied to both online and offline character recognition, where similarity is based on widely used and computationally expensive alignment methods, i.e., Dynamic Time Warping and the Hungarian method respectively. In both cases significant recognition speedup is obtained at the expense of only a minor increase in recognition error.

1 Introduction

Character recognition has been an active research area for the past three decades (see [12] for a comprehensive survey). The last decade has witnessed many successful systems for both online [2, 6, 13] and offline [3, 10] character recognition, with increasing recognition accuracy and decreasing time and memory resources. The simple nearest neighbor (NN) classifier can be applied to this problem and provide excellent recognition accuracy. For example, as we show in the experiments, a simple NN classifier using Dynamic Time Warping (DTW) [8] decreases the recognition error from 2.90% to 1.90% compared to the sophisticated CSDTW method [2] on the UNIPEN digits database[5]. As another example, a simple NN classifier using Shape Context (SC) [3] gave state-of-the-art recognition error of 0.63% on the MNIST database of handwritten digits [10].

The main drawback of the nearest neighbor classifiers is that they are often too slow for practical applications, especially when the underlying similarity measure is based on a computationally expensive alignment, that is superlinear in the number of object features. For example, DTW is quadratic in the number of object features, and the Hungarian method, which is used in the Shape Context distance, is cubic in the number of object features. Methods [2, 15] have been proposed to speedup recognition by avoiding the brute force computation of similarities between the test object and all database objects. These methods select prototypes using unsupervised clustering techniques and require only a few similarity score computations between the test object and those prototypes in order to recognize the test object.

This paper proposes a simple approximate NN method, which is almost as accurate as the simple NN method yet is more efficient. The basic idea underlying the method is depicted in Figure 1. In a nutshell, the method computes an approximate similarity score between two objects, based on their exact alignment to a small number of prototypes. For simplicity, assume we have a single prototype. Then, in the offline step all database objects are aligned with this prototype, so that after alignment all database objects are represented by vectors of some fixed length. In the online recognition step, the test object is aligned with the prototype and is represented by a vector of the same length. Approximate similarity scores between the test and database objects can be efficiently computed in linear time in the number of prototype features using the fixed length vector representation, and can then be used in the filter step of filter and refine retrieval [7] to obtain significantly faster and almost as accurate recognition compared to the simple NN method.

The proposed method can be viewed as an extension of Lipschitz embeddings [4]. In Lipschitz embeddings an object is represented by a vector of distances to a fixed number of prototypes. Our key observation is that computing the distance between an object and a prototype requires a precomputation of the alignment between them. Therefore, since the alignment is readily available and contains richer information than only the distance, it is advantageous to encode the alignment information in the object representation. The experiments confirm that using the richer representation indeed results in increased recognition performance.

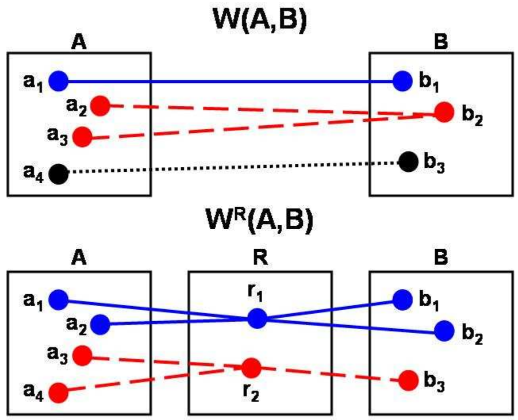

Figure 1. Top row: exact alignment $W(A, B)$ between two objects A and B. Bottom row: approximate alignment $W^R(A, B)$ between the same two objects A and B via a prototype R. Corresponding features in the two objects are connected by lines with the same line pattern.

2 Recognition by Prototypes

This section first describes how to represent objects using their exact alignment to a set of prototypes, and defines the induced approximate alignment (via prototypes) between any pair of objects, and the corresponding approximate distance. The following subsection then describes how to select the set of prototypes in an optimal way.

2.1 Approximate Alignment

We will assume that objects are represented by sets of observations. Let X be a space of such objects, and $A, B \in X$ be two objects of size n_A and n_B respectively, such that $A = a_1, \ldots, a_{n_A}$, and $B = b_1, \ldots, b_{n_B}$. Each a_i and b_j is an observation belonging to a feature space G, with distance measure D_G.

We assume that we have an alignment algorithm (e.g., DTW and the Hungarian method) that establishes correspondences between elements of A and elements of B. Given A and B, this algorithm produces an alignment $W(A, B)$, where the k^{th} element of $W(A, B)$, denoted as $w_k = (a_i, b_j)$, indicates that observation a_i of A corresponds to observation b_j of B. We further assume that each a_i and b_j occur in at least one w_k in $W(A, B)$. The exact distance $D_X(A, B)$ between objects A and B can often be defined using the underlying alignment as follows:

$$D_X(A, B) = \sum_k D_G(\underset{i,j:w_k=(a_i,b_j)}{a_i, b_j}) \tag{1}$$

That is, the distance between A and B is the sum of distances between each a_i and b_j of corresponding observations that appear in the alignment $W(A, B)$.

Suppose now that $R \in X$ is a prototype. Using alignment $W(A, R)$, we can define an embedding $F^R : X \to G^{n_R}$, where n_R is the size of prototype R. In other words, F^R maps any object A into an ordered set of n_R features

$$F^R(A) = (a'_1, a'_2, \ldots, a'_{n_R}), \text{where} \tag{2}$$

$$a'_j = f(\{a_i \mid (a_i, r_j) \in W(A, R)\}), \tag{3}$$

The aggregate function $f(\cdot)$ takes as input all features a_i in A that are aligned with the single feature r_j of the prototype R. The aggregate function $f(\cdot)$ is a design choice that depends on the alignment method under consideration. For example, in DTW the aggregate function for the 2D position features is the arithmetic mean, and the aggregate function for the direction feature is the tangent angle of the vector connecting the first and last position features. Shape Context gives a 1-1 correspondence so the aggregate function is the identity function, i.e., $a'_j = a_i$.

The embeddings $F^R(A) = (a'_1, \ldots, a'_{n_R})$ and $F^R(B) = (b'_1, \ldots, b'_{n_R})$ of objects A and B induce an approximate alignment $W^R(A, B)$, whose k^{th} element is denoted as $w^R_k = (a'_k, b'_k)$. Naturally, if we choose d prototypes R_i, we can define an embedding F by simply concatenating embeddings F^{R_i}:

$$F(A) = (F^{R_1}(A), \ldots, F^{R_d}(A)). \tag{4}$$

F maps objects into $G^{n'}$, where $n' = \sum_{i=1}^{d} n_{R_i}$ is the sum of the sizes of the R_i's. Now, if $F(A) = (a'_1, \ldots, a'_{n'})$ and $F(B) = (b'_1, \ldots, b'_{n'})$, we can define the distance D' between them as:

$$D'(F(A), F(B)) = \sum_{l=1}^{n'} D_G(a'_l, b'_l)\,. \tag{5}$$

Intuitively, the computationally expensive way to compare A and B is to first find their alignment $W(A, B)$ (which is assumed to be time consuming) and then evaluate $D_X(A, B)$ based on this alignment. The embedding F maps objects into ordered sets of features, which are naturally aligned to each other: each a'_l of $F(A)$ maps to b'_l of $F(B)$. The induced alignment is only an approximation of the exact alignment $W(A, B)$, and it is based on the fact that some features in A and some features in B were aligned to the same feature of some prototype R_i.

Now, assume that all database objects are embedded using F in the offline step. Then, in the online recognition step, instead of aligning the test object to all database objects, it is aligned to a few prototype objects, and is also embedded using F. Since the test object and each database object are automatically aligned to each other through the embeddings, computing the approximate distances D' between them is straightforward and takes time linear in n', as opposed to the more expensive exact distance measure D_X that is superlinear in n. Therefore, D' is a computationally efficient distance measure that can be used in the filter step of filter-and-refine retrieval to filter out database objects that are too far from the test object, and to retrieve only the P most promising candidates. Then, in the refine step the exact distance D_X can be used to rerank only the P objects retrieved in the filter step. Finally, the best matches after the refine step vote for the class of the test object.

2.2 Prototype Selection

The previous section described how to embed unaligned objects using their correspondences to multiple prototypes. This section describes how to select the prototypes in an optimal way. Finding the optimal set of d prototypes out of N training objects requires $O(N^d)$ time. The Sequential Forward Search [11], which is widely used in machine learning for feature selection, can be used to find an approximate solution in $O(Nd)$ time: we pick the first prototype to be the one that gives the best performance on the training set when used by itself, and we pick the i^{th} prototype to be the one that gives the best performance when combined with $R_1, \ldots, R_{i-1}$. The performance measure used to evaluate each of the prototypes is the nearest neighbor classification accuracy. This process is repeated until a prespecified number of prototypes are selected, or until the classification accuracy obtained on a separate validation set starts decreasing, which is a sign of overfitting.

3 Experiments

In order to evaluate our classification method we conducted experiments with two OCR databases; the UNIPEN online handwriting database [5], and the MNIST database [10], using respectively DTW and the Hungarian method for alignment. The tradeoff between recognition speed and recognition error is used to evaluate the performance of the proposed method.

3.1 Experiment 1: UNIPEN Database

In this experiment we used the isolated digits benchmark (category 1a) of the UNIPEN Train-R01/V07 online handwriting database [5], which consists of $15,953$ digit examples. Data preprocessing and feature extraction is carried out exactly as described in [2] Section 2. Each observation $a_i = (\tilde{x}_i, \tilde{y}_i, \theta_i)$ is represented by three features: 2D normalized location $(\tilde{x}_i, \tilde{y}_i)$ and the tangent angle between two consecutive observations θ_i. Figure 2 shows an example digit “seven” before and after preprocessing.

For this experiment the digits were randomly and disjointly divided into training and test sets with a 2:1 ratio (or 10,630 : 5,323 examples). The distance measure D_{DTW} used for classification is Dynamic Time Warping [8].

In the filter step, we precomputed the alignment between all database (training) digits and 10 prototypes, which were selected as described in Section 2.2. Thus, each database digit was embedded into a 1,101 dimensional space. The total sequence length of the 10 reference digits is 367, and each sample consists of 3 features $(\tilde{x}, \tilde{y}, \theta)$. During the online step, a test digit was embedded into the 1,101 dimensional space in a similar way. The L_1 distance in this vector space was used for approximating the true DTW distance, and for filtering out database digits that were too far from the test digit. The L_1 distance was used because in preliminary experiments it outperformed the L_2 distance.

Given a test digit and using the approximate measure we retrieved the P nearest database digits. In the refine step we computed the exact DTW measure between the test digit and the P retrieved digits to obtain the final results. The number of exact DTW computations in Table 1 for the proposed method is $10 + P$, where 10 is the number of exact DTW computations in the filter step and P is the number of exact DTW computations in the refine step. $P = 0$ indicates that the test digit is recognized using only the filter step.

Table 1 shows the test error as a function of exact distance computations. Not shown in the table is the exact DTW experiment. In that experiment we used brute-force search, i.e. we aligned a test digit with all the training digits, and classified it using the nearest neighbor rule. 10,630 DTW distance computations were needed for the exact DTW experiment, and the test error was 1.90%. In summary, the results in Table 1 indicate that as expected the test error decreases as the number of exact DTW computation increases. In addition, given a test error of 2.80% the method requires about half the number of exact DTW computations required by BoostMap [1], and about 10% of the number of exact DTW computations required by CSDTW [2]. The best error rate of CSDTW is 2.90% using 150 exact D_{DTW} computations.

Figure 3 shows the test error as a function of the number of prototypes. As expected the test error decreases as the number of prototypes increases. More surprisingly, using only a single prototype overall (a single 8 example) we obtain a relatively low test error of 3.76%. This indicates that a prototype of one class can supply useful information for recognizing an object of a different class.

3.2 Experiment 2: MNIST Database

In this experiment we evaluate our method on the MNIST database [10], using 20,000 database digits and

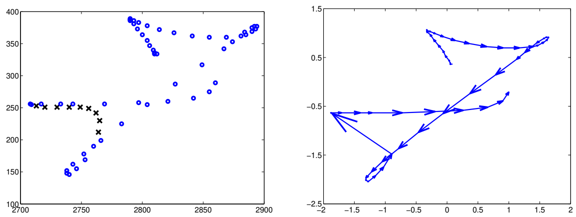

Figure 2. Left: Example of a “seven”. Circles denote “pen-down” locations, x’s denote “pen-up” locations. Right: The same example, after preprocessing.

NN test error as a function of exact distance computations for the UNIPEN digits database			
# D_{DTW}	P	Proposed Method	[1]
30	20	2.14%	2.34%
20	10	2.25%	3.00%
10	0	2.80%	6.23%

Table 1. Comparison of proposed method with [1] on the UNIPEN digit database, in terms of test error as a function of exact distance computations. The test error obtained with the CSDTW method [2] is 2.90% using 150 D_{DTW} distances.

10,000 test digits. In this database each digit is represented by an 28×28 grayscale image. Example images can be seen in Figure 4.

The distance measure D_{SC} used for classification is the Shape Context (SC) distance [3]. As in [3] we resample each image so that its size becomes 70×70. The original shape context matching function consists of a weighted combination of three terms: a shape context feature matching term, an image matching term, and a bending enregy term. In particular, local features called “shape context features” are extracted at each of 100 feature locations in each image. Given two images, using the Hungarian algorithm [9] a 1-1 correspondence between the SC features of those images is established. Using the 1-1 correspondences, one image is warped into the other image. The image matching term is then obtained by measuring the sum-of-squared-differences error between subwindows corresponding to matching feature locations, and the bending energy term is the cost of the warping.

Given a 1-1 correspondence between the SC features of an image and the SC features of the prototype image, we record, for each prototype feature location, the following information about the corresponding image feature location (after warping the image to the prototype): the 2D image location, the local contour orientation at that location, the shape context feature (which is a 60-dimensional vector, as in [3]) obtained at that location, and the intensity values of a local 9x9 subwindow centered around that location.

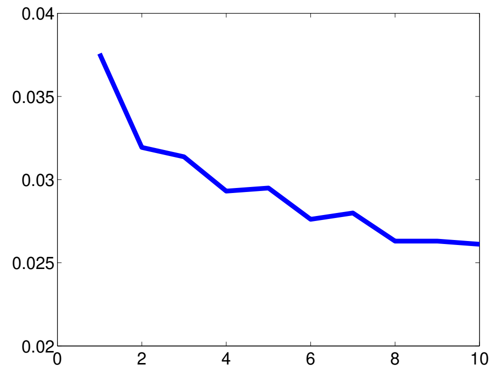

Figure 3. Test error as a function of the number of prototypes, using only the filter step ($P = 0$).

We choose ten prototypes randomly from the training set. We compute the embedding of each test and training object according to alignment to those ten prototypes. Because of the size of the shape context and subwindow features, it is impossible to store the embeddings of all training objects in physical memory: it would require storing $10 \times 100 \times (2+1+60+81)$ numbers per object; these numbers are, respectively, the number of prototypes, the number of feature locations per prototype, and the numbers needed to represent point locations, orientations, shape context features and subwindow features, i.e., 144,000 numbers per object. In order to keep the storage requirements manageable, and to improve the efficiency of the filter step, we perform feature selection using the AdaBoost algorithm [14].

We apply this feature selection algorithm to first choose point location features and orientation features, since those are very compact and cheap to store. Then we allow the training algorithm to choose a small number (in our experiments, 30 out of the possible 2,000) of shape context and subwindow features. The distance in this lower dimensional feature space was used for approximating the true SC distance. The filter and refine retrieval-based recognition was implemented in the same way as for the online character recognition experiment.

In Table 2 we show the test error as a function of exact distance computations. Not shown in the table is the exact Shape Context experiment. In that experiment we used brute-force search, i.e. we aligned a test digit with all the training digits, and classified it using the nearest neighbor rule. 20,000 SC distance computations were needed for the exact SC experiment, and the test error was 0.84%. Using only 50 SC distances the test error obtained using the proposed method was 1.58%, which is significantly better than

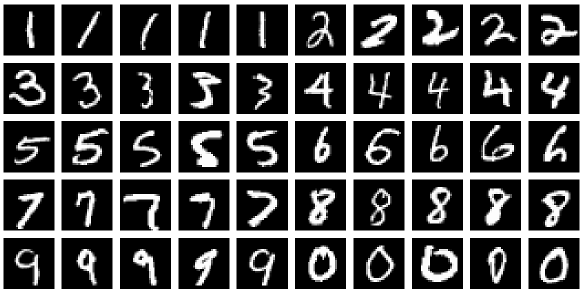

Figure 4. Some examples from the MNIST database of handwritten digit images.

NN test error as a function of exact distance computations for MNIST digits database			
# D_{SC}	P	Proposed Method	[1]
50	40	1.58%	1.89%
40	30	1.64%	2.15%
30	20	1.79%	2.68%
20	10	2.20%	4.38%
10	0	7.35%	12.66%

Table 2. Comparison of proposed method with two other methods on the MNIST database, in terms of test error as a function of exact distance computations. The test error obtained with the method used in [15] is 2.55% using 50 D_{SC} distances.

the 2.55% test error obtained by the method proposed in [15], which was specifically designed for the SC distance.

4 Conclusion

This paper presented a general method for computing an approximate similarity measure between a pair of objects based on their alignment to prototype objects. The proposed method is simple and can be used to design efficient approximate nearest neighbor recognition methods. The proposed method, although general, outperforms two state-of-the-art character recognition methods [2, 15] that were specifically designed to exploit their underlying similarity measures. Our method builds upon the Lipschitz embedding method, which represents objects using their distances to prototype objects, but in contrast to Lipschitz embeddings the proposed representation is richer and encodes the complete alignment information between an object and a prototype rather than only the distance between them. The superiority of the richer representation has been demonstrated in the experiments that have also shown that sufficiently accurate results may be obtained with as little as a single prototype.

Acknowledgments

We wish to thank Louis Vuurpijl for helping out with the UNIPEN database, and Claus Bahlmann and Eugene Ratzlaff for providing useful information about their algorithms. This research was supported in part through U.S. grants ONR N00014-03-1-0108, NSF IIS-0308213 and NSF EIA-0202067.

References

[1] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. BoostMap: A method for efficient approximate similarity rankings. In *CVPR*, 2004.

[2] C. Bahlmann and H. Burkhardt. The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. *PAMI*, 26(3):299–309, 2004.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(4):509–522, April 2002.

[4] J. Bourgain. On Lipschitz embeddings of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52:46–52, 1985.

[5] I. Guyon, L. Schomaker, and R. Plamondon. UNIPEN project of on-line data exchange and recognizer benchmarks. In *ICPR*, pages 29–33, 1994.

[6] J. Hebert, M. Parizeau, and N. Ghazzali. A new fuzzy geometric representation for on-line isolated character recognition. In *ICPR*, pages 1121–1123, 1998.

[7] G. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *PAMI*, 25(5):530–549, 2003.

[8] J. B. Kruskall and M. Liberman. The symmetric time warping algorithm: From continuous to discrete. In *Time Warps*. Addison-Wesley, 1983.

[9] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2*, pages 83–97, 1955.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[11] A. Miller. *Subset Selection in Regression*. Chapman and Hall, 1990.

[12] R. Plamondon and S. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *PAMI*, 22(1):63–84, January 2000.

[13] E. Ratzlaff. A scanning n-tuple classifier for online recognition of handwritten digits. In *ICDAR*, pages 18–22, 2001.

[14] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[15] H. Zhang and J. Malik. Learning a discriminative classifier using shape context distances. In *CVPR*, volume 1, pages 242–247, 2003.