Learning a Classifier with Very Few Examples: Analogy Based and Knowledge Based Generation of New Examples for Character Recognition

S. Bayoudh[1], H. Mouchère[2], L. Miclet[1], and E. Anquetil[2]

[1] IRISA-ENSSAT
{bayoudh,miclet}@enssat.fr
[2] IRISA-INSA
{mouchere,anquetil}@irisa.fr

Abstract. This paper is basically concerned with a practical problem: the on-the-fly quick learning of handwritten character recognition systems. More generally, it explores the problem of *generating new learning examples*, especially from very scarce (2 to 5 per class) original learning data. It presents two different methods. The first one is based on applying distortions on original characters using *knowledge* on handwriting properties like speed, curvature etc. The second one consists in generation based on the notion of *analogical dissimilarity* which quantifies the analogical relation "A is to B almost as C is to D". We give an algorithm to compute the k-least dissimilar objects D, hence generating k new objects from three examples A, B and C. Finally, we experimentally prove the efficiency of both methods, especially when used in conjunction.

Keywords: instance generation, sequence generation by analogy, knowledge-based generation, handwritten character recognition.

1 Introduction

In a number of Pattern Recognition systems, the acquisition of labeled data is user unfriendly. Still, the classification system has to be retrained with as many examples as possible. A way to overcome this difficulty is to generate artificial new examples. It can be done either in the feature space or on the raw representation of the data.

The first technique has already been studied and experimented. [1] combines feature selection and generation of "corrupted" examples to avoid overfitting when boosting on small samples. [2] analyzes the regularization performed by adding noise on the learning data. In the framework of neural networks, the effect of generating more examples is known as a good way to avoid overfitting.

For on-line character recognition systems, several image distortions have been used [3]: slanting, shrinking, ink erosion and ink dilatation; [4] extends the learning data base by elastic distortions before training a neural network. Text lines can also be distorted as in [5] before training a Hidden Markov Model.

J.N. Kok et al. (Eds.): ECML 2007, LNAI 4701, pp. 527–534, 2007.

Bishop [6] gives no theoretical coverage about example generation, but draws a pragmatic conclusion: " (...) the addition of random noise to the inputs (...) has been shown to improve generalization in appropriate circumstances".

In this paper, we deal with the quick tuning of a handwritten character recognition to a new user, when only a very small set of examples is available. We describe in section 2 our system and the representation used to generate new examples. This operation is realized along two different axes, before transforming the data into the feature space. Firstly, we consider the inputs as sequences of a combination of Freeman chain-codes and of special symbols (*anchorage points*) describing pen up and downs, extrema, etc. Then, in section 3, we use an original method of generating: from any three examples, we produce as many examples as required, provided that they are in weak *analogical dissimilarity* from the three basic patterns. Secondly, as described in section 4, we extract high-level information from the characters : slant, writing speed, etc. and we randomly vary these parameters to generate new examples. To finish, we combine the two methods and we describe in section 5 our experimental protocol. We show in particular in this section that generating hundreds of fake examples from a very small number of real samples per class is a better learning strategy than using 10 real samples per class and comparable to use 30 real samples per class.

2 On-Line Handwriting Signal Description

The on-line characters used in this paper have been gathered on a PDA using a specialized pen-based human-computer interface. The characters are isolated. They are acquired without any context, in a random order, and on-line.

When acquired on-line, a character is considered as a function $p(t)$ describing the pen positions. Each point $p(t)$ is defined by its x and y geometric position.

Contrary to the knowledge-based generation, the generation based on Analogical Proportion (AP) requires a slightly more elaborated representation of the signal. Hence, before the AP generation, the signal is re-sampled in space and basically transformed into a Freeman chain-code. In this paper we use the extended 16-Freeman code $\Sigma = \{0, 1, .., 16\}$, where the direction 0 corresponds to an isolated point in the original signal. The AP generation produces characters in Freeman chain-code. Then they are decoded into an on-line signal by following the direction sequence using the same inter-point distance.

For the needs of the AP generation we have extended the Freeman code with a set of 8 capital letters $\{C, D, E, H, K, L, M, N\}$ each one corresponding to a special point in a handwritten character. They are called *anchorage points* and they lead to a better AP generation. These anchorage points come from an analysis of the stable handwriting properties, as defined in [7]: pen-up/down, y-extrema, angular points and in-loop y-extrema.

Figure 1 shows a global overview of the complete generation process of new examples using knowledge-based distortions and AD generation on Freeman encoding. After generation, the characters are transformed in a vector of 21 features. These features are the input of our handwriting recognition system. They

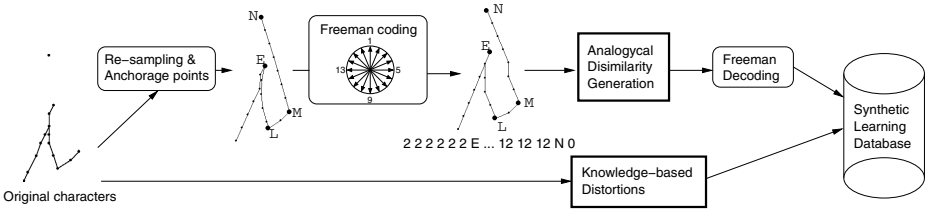

Fig. 1. Overview of the generation process of synthetic learning data base using knowledge-based distortions and AD generation on Freeman encoding.

are based on stable properties of handwriting like downstrokes [7]. The classification and test processes of the section 5 are done in this 21-dimension feature space.

3 Analogy Based Generation

In this section, we give general notions on analogy and explain how it can be used to generate new examples. Generally speaking, an *analogy*, or to be more specific, an *analogical proportion*, involves four objects (or terms) such that "*the second term is to the first as the fourth is to the third*" according to the seminal definition by Aristotle. Analogy has been widely used as a model of reasoning, in philosophy [8], Artificial Intelligence [9], computational linguistics [10].

3.1 Analogical Proportion

An *analogical proportion* (AP) between four objects A, B, C and D, in the same universe, is expressed by : "A `is to` B `as` C `is to` D". If an element x of the analogical equation is unknown, finding it is called resolving an *analogical equation*. For example, in "*fins* `is to` *fish* `as` *wings* `is to` x", the solution x would be *bird* in the semantic domain.

Definition 1. *An AP on a set* $\mathbf{X}$ *is a relation between four elements of* $\mathbf{X}$ *(i.e. a subset of* $\mathbf{X}^4$*). An element of this relation writes "a : b :: c : d" and reads "a* `is to` *b* `as` *c* `is to` *d", or "a, b, c and d are in AP". According to [10], an AP must fulfill the three properties:*

$$\begin{aligned} &\textit{Symmetry of the "as" relation:} && (a : b :: c : d) \Leftrightarrow (c : d :: a : b) \\ &\textit{Exchange of the means:} && (a : b :: c : d) \Leftrightarrow (a : c :: b : d) \\ &\textit{Determinism:} && (a : a :: b : x) \Rightarrow (x = b) \end{aligned}$$

How can we use this definition in pattern generation ? Suppose that the handwritten character "`a`" is encoded as a sequence and that we have 3 examples a_1, a_2 and a_3 of this character. We can generate more "a" characters by solving analogical equations on sequences like: "a_1 : a_2 :: a_3 : x" or "a_3 : a_2 :: a_1 : x". If we have n examples, the number of new "`a`" that we can generate is $n^2 \times (n-1)/2$. We can even go further, as we show in the following paragraph, by relaxing the definition of the solution to an analogical equation.

3.2 Analogical Dissimilarity Between Objects

We give in this section a precise definition of the approximate *analogical proportion* "a `is to` b `almost as` c `is to` d". For this purpose, we have introduced a quantity that reflects how far are four objects from being in AP. This measure is called *Analogical Dissimilarity* (AD) [11]. To coherently extend the AP, it has to verify the following properties:

1. $\forall u, v, w, x, AD(u, v, w, x) = 0 \Leftrightarrow u : v :: w : x$
2. $\forall u, v, w, x, AD(u, v, w, x) = AD(w, x, u, v) = AD(v, u, x, w)$
3. $\forall u, v, w, x, z, t, AD(u, v, z, t) \leq AD(u, v, w, x) + AD(w, x, z, t)$
4. In general, $\forall u, v, w, x, AD(u, v, w, x) \neq AD(v, u, w, x)$.

3.3 Analogical Dissimilarity Between Sequences

Let $\Sigma = \{1, 2, \ldots, 16, 0, C, \ldots, N\}$ be the alphabet of the Freeman symbols code and of the anchorage points. A handwritten character is represented by a sequence of letters of Σ. We introduce a new letter $\backsim$ to Σ, to be added anywhere in a sequence without changing its semantics ("$35C4$" means the same as "$3 \backsim 5C \backsim\backsim 4$"). With this augmented alphabet, we can define an alignment between four sequences and the notion of analogical dissimilarity.

Alignment. An alignment of four sequences of different lengths is realized by inserting letters $\backsim$ so that all the four sequences have the same length. Once this is done, we consider in each column of the alignment the analogical dissimilarity in the augmented alphabet. For example $AD(1, 5, 2, 6) = 0$ because they are in exact AP, $AD(3, A, 4, A) = 1$ (because shifting 4 to 3, which are at distance 1, would give an exact AP).

The principle of our generation process is to align three sequences of characters "`h`" for example, and resolving column after column to generate the fourth sequence (see Figure 2). We define the cost of an alignment as the sum of the AD on columns, and an *optimal alignment* as one of minimal cost. The Analogical Dissimilarity between four Sequences (ADS) is the cost of an optimal alignment. It has all the properties given above, except the third point (the triangular inequality).

To generate the k best sequences regarding to ADS, we use the general search algorithm $A^\star$ [12] used to find the k shortest paths in a graph [13], with a heuristic value set to 0.

$h_1 =$ 9 $\backsim$ 9 $\backsim$ 9 9 9 9 $\backsim$ H 1 2 $\backsim$ 4 K 6 9 9 9
$h_2 =$ 1 K $\backsim$ 8 9 9 9 9 9 10 H $\backsim$ 2 2 4 K $\backsim$ 8 8 9
$h_3 =$ $\backsim$ $\backsim$ 9 8 9 9 9 9 9 10 H 2 2 3 3 K 8 9 9 $\backsim$
$x =$ 1 K $\backsim$ 8 9 9 9 9 9 10 H 2 2 3 3 K 8 8 8 $\backsim$

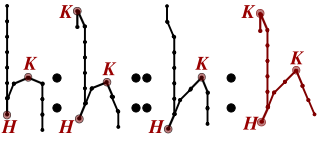

Fig. 2. Example of a resolution on Freeman direction sequences by AP and the corresponding characters representation

4 Knowledge Based Generation

In this section we present two classical image distortions and two new on-line distortions based on specificities of the on-line handwriting. Each distortion depends on one or more random parameters. To generate a synthetic character from an original one using a distortion, we first randomly choose a value for each corresponding parameter and then we apply it.

4.1 Generation by Image Distortions: Scaling and Slanting

To generate new learning examples, one has to choose distortions according to the data acquisition process. Indeed images captured with a camera or in a video can be perturbed by perspective, rotation, noise... while scanned handwriting is disturbed by ink thick variation only. In this paper we limit us to scaling and slanting deformations because of the nature of our data. Two random parameters correspond to the scale transformations, α_x and α_y which are the ratio of the corresponding scales. The slant allows to generate inclined handwriting. It depends of one random parameter α_s which represents the tangent of the slant.

4.2 Generation by On-Line Distortions

There are few works about using handwriting generation in order to increase on-line learning data. In [14] authors use works about handwriting generation [15] to increase the size of a learning database to train an off-line sentence writer-independent recognizer. The authors use a unique on-line handwriting model. This approach is unusable in our context because we need writer-dependent models. That is why we propose two simple distortions of on-line handwriting: *Speed Variation* and *Curvature Modification*.

Speed Variation. The aim of *Speed Variation* distortion is to modify the size of vertical and horizontal parts of the stroke, as shown in Figure 3, depending of a random parameter α_v. Indeed this straight parts of the writing can vary without changing the handwriting style. For this we modify the speed $\boldsymbol{V}(t) = (x(t+1) - x(t), y(t+1) - y(t))$ depending of its direction. If this vector is near one of the axes then it is increased or decreased by the ratio α_v. The new synthetic handwriting is defined by $p'(t)$:

$$p'(t) = p'(t-1) + \beta * \boldsymbol{V}(t-1), \text{ with } \beta = \begin{cases} 1 & \text{if } \arg(\boldsymbol{V}(t-1)) \left[\frac{\pi}{2}\right] \in [\frac{\pi}{8}, \frac{3\pi}{8}], \\ \alpha_v & \text{else.} \end{cases}$$

Curvature Modification. The *Curvature Modification* distortion modify the curvature of the writing as shown in Figure 3. It allows to close or open the loops of handwriting. The curvature modification uses a random parameter α_c. The curvature at the point $p(t-1)$ is defined by the angle $\hat{\theta}(t-1) = (p(t-2), \widehat{p(t-1)}, p(t))$ in $]-\pi, \pi]$. In order to keep the structure of the character, we do not modify the straight lines and cusps. The following equation gives

the position of the point $p'(t)$ depending of the two previous points and of the original curvature $\hat{\theta}(t-1)$ modified by α_c:

$$(p'(t-2), \widehat{p'(t-1)}, p'(t)) = \hat{\theta}(t-1) - \alpha_c * 4 * \frac{|\hat{\theta}(t-1)|}{\pi} * (1 - \frac{|\hat{\theta}(t-1)|}{\pi}).$$

5 Experimentations

5.1 Experiment Protocol

Twelve different writers have written 40 times the 26 lowercase letters (1040 characters) on a PDA. Each writer database is randomly split in four parts with 10 characters per class. We use them in a 4-fold stratified cross validation: one fourth for $D10$ database (260 data) and three fourth for $D30$ database (780 data). Thus the experiment is done four times by switching these parts. The experimentations are composed of two phases in which three writer-dependent recognition systems are learned: a Radial Basis Function Network (RBFN) and a one-against-all Support Vector Machine (SVM).

Firstly, we compute two Reference recognition Rates without data generation : $RR10$ and $RR30$. For $RR10$, writer-dependent classifiers are learned for each writer on his $D10$ and evaluated on his $D30$ database for the four splits. For $RR30$, the classifiers are learned on $D30$ and tested on $D10$. Hence, $RR10$ is the recognition rate achievable with 10 original characters without character generation and $RR30$ gives an idea of achievable recognition rates with more original data. Practically speaking, in the context of on the fly learning phase we should not ask the user to input more than 10 characters per class.

Secondly the handwriting generation strategies are tested. For a given writer, one to ten characters per class are randomly chosen in his $D10$. Then 300 synthetic characters per class are generated to make a synthetic learning database. A classifier is learned with this base and tested on the database $D30$ of the writer. This experiment is done 3 times per cross validation split and per writer (12 times per user). Finally the means of the writer dependent mean recognition rate and the mean standard deviation are computed.

We study three different strategies for the generation of synthetic learning databases. The strategy *"Image Distortions"* chooses randomly for each generation one among the three image distortions. In the same way the strategy *"On-line and Image Distortions"* chooses randomly one distortion among the image distortions and on-line distortions. The *"Analogy and Distortions"* strategy generates two-thirds of the base with the previous strategy and the remaining third with AP generation.

5.2 Results

Figure 4 compares the recognition rates achieved by the three generation strategies for the three classifiers. Firstly we can note that the global behavior is the same for the two classifiers. Thus the following conclusions do not depend on the

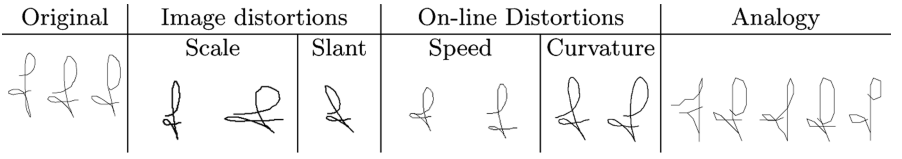

Fig. 3. Examples of synthetic characters generated by the three approaches

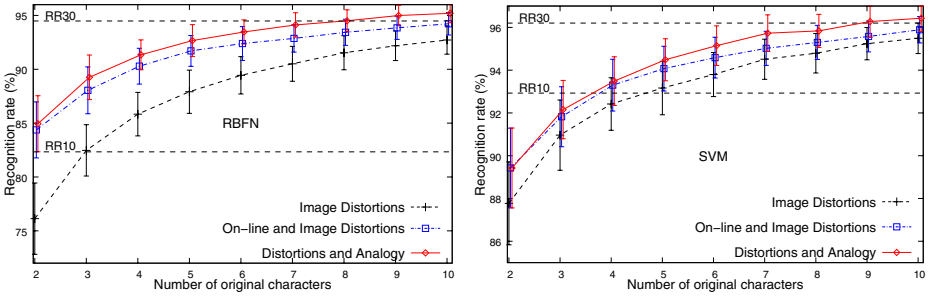

Fig. 4. Writer-dependent recognition rates and their standard deviation depending on the number of used original characters compared to reference rates using 10 or 30 characters per class for RBFN and SVM classifiers

classifier type. Secondly the three generation strategies are complementary because using *"On-line and Image Distortions"* is better than *"Image Distortions"* alone and *"Analogy and Distortions"* is better than using distortions. We proved statistically the signigicance of the improvement between the three generation strategies using the t-test and the *sign* test [16]. Thus, for each classifier and each number of original characters the difference between the strategies is due to chance has a probability less than 10^{-3} using the t-test and less than 10^{-30} using the *sign*-test.

Therefore, using only four original character with the complete generation strategy is better than the $RR10$. The $RR30$ is achieved by using 9 or 10 original characters. Thus we can conclude that using our generation strategies allows to learn classifier with very few original data as efficiently as using original data from a long input phase: we need about three times fewer original data to achieve the same recognition rate.

6 Conclusion

In this paper, we have shown how to generate synthetic examples for a quick tuning of a handwritten character classifier to a new writer. We have presented two complementary ways of generation, the first being based on the prior knowledge and the second on analogy on sequences. Both methods in conjunction have led to efficient results on writer-dependent recognition.

Although empirical, these experiments lead to the conclusion that it may be efficient to generate new samples, provided that the variation on few learning samples is not made in the feature decision space, but rather in a representation space close to the raw data. Knowledge-based and analogy-based generation, although of very different nature, have proved in this case to be both efficient.

References

1. Wolf, L., Martin, I.: Robust boosting for learning from few examples. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society Press, Los Alamitos (2005)
2. Bishop, C.: Training with noise is equivalent to Tikhonov regularization. Neural Computation 7(1), 108–116 (1995)
3. Cano, J., Pérez-Cortes, J.C., Arlandis, J., Llobet, R.: Training set expansion in handwritten character recognition. In: Proc. of the 9th Int. Workshop on Structural and Syntactic Pattern Recognition, pp. 548–556 (2002)
4. Simard, P., Steinkraus, D., Platt, J.C.: Best practice for convolutional neural network applied to visual analysis. In: Proc. of the 7th Int. Conf. on Document Analysis and Recognition (2003)
5. Varga, T., Bunke, H.: Generation of synthetic data for an HMM-based handwriting recognition system. In: Proc. of 7th Int. Conf. on Document Analysis and Recognition, pp. 618–622 (2003)
6. Bishop, C.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2007)
7. Anquetil, E., Lorette, G.: Perceptual model of handwriting drawing application to the handwriting segmentation problem. In: Proc. of the 4th Int. Conf. on Document Analysis and Recognition, pp. 112–117 (1997)
8. Hesse, M.: Aristotle's logic of analogy. The Philosophical Quarterly 15(61), 328–340 (1965)
9. Gentner, D., Holyoak, K.J., Kokinov, B.: The analogical mind: Perspectives from cognitive science. MIT Press, Cambridge (2001)
10. Lepage, Y.: Solving analogies on words: an algorithm. In: Proc. of COLING-ACL'98, vol. 1, pp. 728–735 (1998)
11. Bayoudh, S., Miclet, L., Delhay, A.: Learning by analogy: a classification rule for binary and nominal data. In: Proc. of the Int. Joint Conf. on Artificial Intelligence, vol. 20, pp. 678–683 (2007)
12. Nilsson, N.: Principles of Artificial Intelligence. Tioga Publishing Company (1980)
13. Eppstein, D.: Finding the k shortest paths. SIAM J. Computing 28(2), 652–673 (1998)
14. Varga, T., Kilchhofer, D., Bunke, H.: Template-based synthetic handwriting generation for the training of recognition systems. In: Proc. of 12th Conf. of the International Graphonomics Society, pp. 206–211 (2005)
15. Plamondon, R., Guerfali, W.: The generation of handwriting with delta-lognormal synergies. Biological Cybernetics 78, 119–132 (1998)
16. Gillick, L., Cox, S.J.: Some statistical issues in the comparison of speech recognition algorithms. In: IEEE (ed.) Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, Glasgow, Scotland, pp. 532–535. IEEE Computer Society Press, Los Alamitos (1989)