Extraction of Discriminant Features from Image Fractal Encoding

Matteo Baldoni, Cristina Baroglio,
Davide Cavagnino, Giuseppe Lo Bello

Dipartimento di Informatica — Università degli Studi di Torino
Corso Svizzera, 185 — I-10149 Torino (Italy)
Tel. +39 11 74 29 111, Fax +39 11 75 16 03

E-mail: {baldoni,baroglio,davide,lobello}@di.unito.it
URL: http://www.di.unito.it/~baldoni/fractals/

Abstract. In this paper we face the problem of finding characteristic information about images of different objects, showing that the fractal encoding based on *Iterated Function Systems*, besides allowing very high compression rates, can be successfully applied *also* for capturing discriminatory features that can be exploited for *non-fractal* image classification. An original feature extraction algorithm was developed and applied to encode the hand-written digits data set. Then, different learning algorithms were applied and their performances were compared both to those obtained using a general purpose fractal encoder (*enc* by Fisher) and to the work done in the StatLog project on the same data set.

Keywords: Machine learning, feature extraction, fractal encoding.

1 Introduction

In this paper we tackle the problem of classifying visual representations of objects w.r.t. a set of classes given a priori: a central problem in many artificial vision tasks, e.g., optical character recognition [13], face recognition [24] for security systems, and object recognition for assembly. The approach we present is divided in three main steps: (1) finding a way that allows to *extract*, in an automatic way, a small set of *features* characterizing the instances of the target classes; (2) learning a *classifier* that maps the extracted features on the target classes; (3) using the obtained classifier for recognition purposes. Differently than what can be found in the literature, this approach does not use any model of the objects at issue (as, instead, [6, 9, 14]) nor it needs a set of templates to be defined in advance (as, instead, [22, 2]). On the contrary, it simply exploits the values of a set of *descriptive features* which characterize the *visual representations* of the objects. The ideal features we would like to identify are easy to extract, robust, and well suited to a fast and reliable interpretation. Moreover, since we deal with image recognition, they should be *invariant* w.r.t. the distance, keeping a high *discriminatory power*, i.e., be different for different classes.

In particular, in this work we investigated the use a new type of feature derived from a *fractal* description that is obtained by means of an *Iterated Function*

System (IFS). The motivation for studying fractal features is that they are *already* used for image transmission purposes; reconstructed images are impressive for their fidelity and greatly reduce the computational cost of encoding, allowing compression rates as high as $2,000:1$. Nevertheless, the focus of our attention was not to use IFSs to *reproduce* images but to see if they can be used to *extract discriminant information* about images. The question was: given that IFSs are a powerful encoding tool, do they also contain enough information to distinguish images belonging to different classes? This topic was already addressed in [8] but only in the case of (1) images with an intrinsic *fractal* structure where (2) feature extraction was hand-made. Here, instead, we have investigated the use of IFSs to capture discriminant information about *non-fractal* objects, on one hand, by developing an original *application-oriented* algorithm that allows to extract fractal discriminant features in an automatic way; on the other, we applied a well-known, general purpose fractal encoder (*enc* see [10]). This algorithm was applied to StatLog hand-written digits and the encoded samples were used to train different learning algorithms (heading to different approaches to learning, i.e., Genetic Algorithms, Neural Networks, and Symbolic Induction), whose performances are reported at the end of this paper.

The paper is organized as follows. In Section 2 the basic notions on IFSs are provided. Section 3 describes how to use IFSs to extract meaningful features. Experiment descriptions and results are reported in Section 4 together with a description of the algorithms used. Conclusions are drawn in Section 5.

2 IFSs theoretical background

This section is a summary of a few concepts about fractal theory that are particularly relevant to this work. More details can be found in [3, 4].

The basic idea underlying fractal generation is that a fractal contains, at any scale, smaller copies of itself, i.e., it is *self-similar*. One way of generating fractal images in the plane is to *iterate* simple geometrical transformations starting from a predefined shape. For instance, from a geometrical point of view, the *rule* that allows the curve of Figure 1(d) to be built consists in repeatedly *mapping* the initial segment AB (Figure 1(a)) to each of the segments AC, CE, ED and DB, respectively (Figure 1(b)) [11]. Each mapping (or *transformation*) consists of a *rotation* and a *contraction*[1] (the length of the segment reducing to one third of the original).

A particular kind of transformation is the *affine contractive transformation*, which, in words, allows to reproduce a smaller copy of an initial pattern, keeping its original shape. More formally, let us consider the n-dimensional space $\mathbb{R}^n$ and let $\mathbf{z}$ be a point in $\mathbb{R}^n$: a mapping $M : \mathbb{R}^n \to \mathbb{R}^n$ is an *affine transformation* if it has the form $M(\mathbf{z}) = T(\mathbf{z}) + \mathbf{w}$, where T is a linear transformation on $\mathbb{R}^n$ and $\mathbf{w}$ is a vector in $\mathbb{R}^n$. In other words, an *affine transformation* is a combination

[1] Given a closed subset E of $\mathbb{R}^n$, a mapping $M : E \to E$ is a *contraction* on E if there is a number c, with $0 < c < 1$, such that $|M(x) - M(y)| \leq c|x - y|$ for all $x, y \in E$. We will consider the modulo norm in E, i.e. the Euclidean distance.

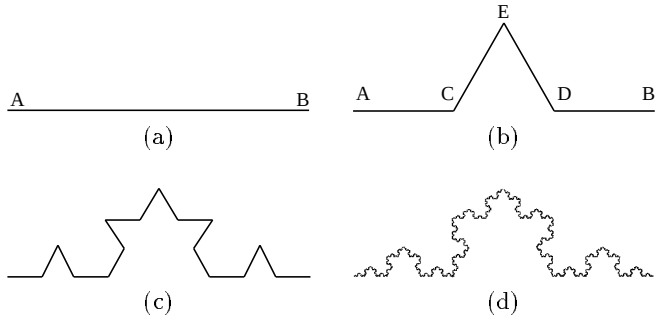
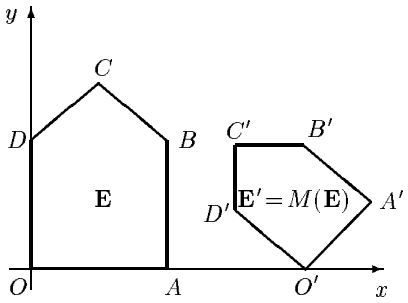


Fig. 1. Successive stages in the construction of the von Koch curve [11]. At each step, the middle third of each existing segment is removed and replaced by the two sides of an equilateral triangle. (a) Initial segment. (b) Basic transformation. (c) Curve after 2 steps. (d) The von Koch curve.

of *rotations*, *scalings* and *translations* of the coordinates. For instance, a two dimensional affine transformation has the general matrix form

$$M(\begin{bmatrix} x \\ y \end{bmatrix}) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} r\cos\theta & -s\sin\phi \\ r\sin\theta & s\cos\phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \qquad (1)$$

where θ and ϕ are rotations and r and s are scalings on the x and y components, respectively; an example is shown in Figure 2.



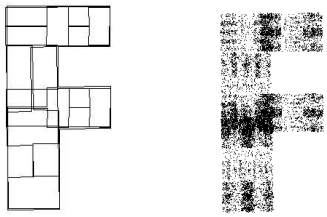
$$M(\begin{bmatrix} x \\ y \end{bmatrix}) = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

Fig. 2. Example of affine transformation: figure $\mathbf{E}$ is mapped on $\mathbf{E}'$ by means of M.

An affine transformation for which the *contractive* property holds is said to be *contractive*. Affine contractive transformations are fundamental in this work; in fact, although, in general, an IFS is *any* collection of contractions, here we focus on collections of affine contractive transformations $\{M_1, \ldots, M_m\}$.

For every IFS in $\mathbb{R}^n$ there exists a set F such that $F = \bigcup_{i=1}^{m} M_i(F)$. F is called *invariant set*; it is *unique* and *non-empty* and can be a *fractal*. Moreover, due to the *Collage Theorem* [3, 11], given any subset E of $\mathbb{R}^n$ and an arbitrary precision of approximation, it is always possible to find an IFS, whose invariant set approximates E as finely as desired. In the plane the Collage Theorem helps in reconstructing an image by giving a method for doing it, ensuring at the same time the accuracy of the reconstruction. In particular, an image can be reconstructed by covering it with a set of contracted affine copies of itself (see Figure 3). Other characteristics of the IFS are that it is *robust* and *stable*, i.e., small changes in the transformations produce small changes in its invariant set; hence, varying the coefficients in a continuous way, the shape of the invariant set also changes in a continuous manner.

In the literature, many methods for automatically finding an IFS that encodes a given image, such as [5], can be found. One of them, derived from the works by Jacquin and Fisher (see [16] and [10]), was used in some of the experiments.



(a) (b)

a	b	c	d	e	f
-0.03	-0.37	-0.52	0.02	0.17	-0.50
-0.03	-0.37	-0.52	0.02	0.39	4.38
0.50	-0.06	0.00	-0.80	1.23	-2.48
-0.50	-0.02	0.01	-0.80	1.69	-2.56
0.01	-0.37	0.52	-0.02	0.15	-0.99
0.01	-0.37	0.52	-0.02	0.38	3.93

(c)

Fig. 3. (a) It is possible to cover a capital "F" with copies of itself in many ways. One possibility is shown here (from [20]): the "F" is covered with six contracted copies of itself, each "bar" being covered with two contracted copies placed side by side. (b) The corresponding invariant set. (c) Its IFS code.

3 IFSs as Descriptive Features

In this work, the Collage Theorem was applied to extract the meaningful aspects of an object, which are, then, encoded by an IFS. More precisely, given an image ξ in 256 grey levels, let $M(\xi) = \{M_i | 1 \leq i \leq r\}$ be the set of affine transformations which cover the image according to the Collage Theorem. The *idea* is to use this set of transformations as a set of *descriptive features*. This is done by associating to ξ the vector $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$ of the parameters of the r transformations. The number of features n necessary to characterize an object will, then, be equal to the number of parameters defining each transformation (e.g., in (1), they are the six coefficients a, b, c, d, e, and f) times the number r of

transformations used (see Figure 3(c)). Since the image is not to be reconstructed but only *distinguished* from others, the number of transformations needed is by far lower than that required to reproduce an image with high fidelity.

One way of using descriptive features is to define a set of prototypes for each target class, using a distance measure to find which of them are the most similar to the instances to be classified. However, defining such prototypes is not trivial. In this work we solved the problem by applying different learning algorithms, whose performances are reported in the following section.

4 Experimental results

All experiments were done on the *ten digits* test-bed, also used in StatLog [18]. Briefly, this data set is a collection of hand-written samples of the ten digits (collected by the German Federal Post), acquired with a resolution of 16×16 pixels with 256 grey levels (some examples are reported in Figure 6(a)). In some experiments the samples were binarized, i.e., all pixels whose value was higher than a certain threshold were considered as elements of the digit otherwise they were considered as background.

First of all, we tried to understand, in a short time, if IFSs capture characteristic information about (non-fractal) images; to this purpose, the samples were hand-coded in the following way: given the binarized image of a digit, we extracted the fractal features by covering it by means of four self-affine transformations, mapping the whole digit to parts of itself (see Figure 4). Due to the

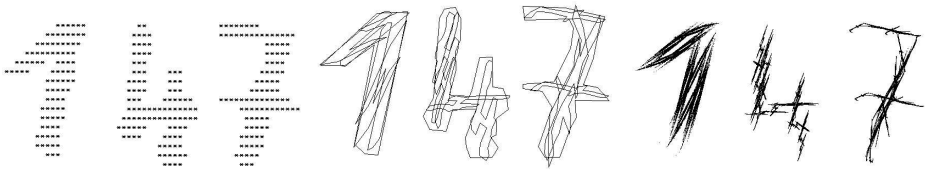


Fig. 4. (a) Binarized instances of the digits "1", "4" and "7". (b) Digit covering with four self-affine transformations. (c) Digit fractal reconstruction.

long time required by sample hand-coding, only digits "1", "4", and "7" were taken into account because of their similarity in hand-writing. A three-layered Multi Layer Perceptron (MLP) with 24 input units (4 transformations $\times$ 6 parameters each, see Section 2), 80 hidden neurons and 3 output neurons was used. The learning set was made of 30 examples (10 per class). The test set consisted of 120 examples (40 per class). Results show a 100 % recognition rate both on the learning and on the test sets.

Afterwards, an original algorithm was developed so to make the above hand-made encoding *automatic*. This algorithm is application-oriented, i.e. it was developed to find IFS encodings for non-fractal, linear images. Finally, a *general-purpose* algorithm for fractal image compression, *enc*, was applied to the same data for comparing the performances.

The learner was implemented in many different ways for comparison purposes. In particular, we have used feed-forward neural networks (classical multilayer perceptron, a Conjugate Gradient method, SCG [23], and cascade correlation networks), CART [7], and genetic algorithms [12]. When not specified otherwise, the learning sets were made of 300 samples (30 samples per class) while the test sets were made of 700 samples (70 per class), all different than those shown during training. Cross-validation was used to check the independence of the results from the particular learning and test sets used: each experiment was repeated on three different learning/test set pairs; the percentages reported in Table 1 are the averages of the performances obtained in the various trials.

Table 1. Recognition rates obtained using: (a) CART; (b) GAs; (c) NNs (SCG); (d) NNs (Cascade-correlation); (e) NNs on features extracted by *enc*.

	(a) CART	*(b)* GA	*(c)* SCG-NN	*(d)* CC-NN	*(e)* NN/*enc*
"0"	75.7 %	85.7 %	85.0 %	85.7 %	79.6 %
"1"	92.9 %	90.0 %	96.7 %	95.7 %	78.0 %
"2"	84.3 %	75.7 %	91.9 %	94.3 %	71.6 %
"3"	54.3 %	74.3 %	84.2 %	75.7 %	60.8 %
"4"	70.0 %	44.3 %	81.4 %	72.9 %	65.6 %
"5"	75.7 %	77.1 %	86.7 %	75.7 %	81.6 %
"6"	68.6 %	77.1 %	90.2 %	82.9 %	72.4 %
"7"	82.9 %	84.3 %	92.4 %	90.0 %	69.2 %
"8"	60.0 %	41.4 %	62.4 %	45.7 %	26.0 %
"9"	60.0 %	58.6 %	79.0 %	75.7 %	47.2 %
avg.	72.4 %	70.9 %	85.0 %	79.4 %	65.2 %

Experiments with an automatic covering. The algorithm we developed is based on the following two observations. The first is that since we are looking for some discriminant features, we do not need to perfectly cover a given image: an approximation of it will be sufficient. The second is that, differently than what done in the first experiment, we cover an image by means of a set of auto-affine copies of a shape that is *different* (and simpler) than the one of the digit to cover. This can be done because, in the line of the first observation, this approach is sufficient to extract an information that is *intrinsic* to the covered

digit and does *not* depend on the particular objects used for covering. Moreover, using predefined simple shapes has also the advantage of simplifying the covering operation.

In particular, our algorithm is based on covering the *binarized* image of a digit by means of a *fixed* number of affine contractions of the *square box* containing it. In order to reduce the complexity of the computation, only a *limited number* of geometric transformations is allowed. With reference to equation (1), all rotations belong to the set $\{0°, 45°, 90°, 135°\}$, the scaling is fixed a priori and depends on the experiment, while the translations are not bounded. Consequently, we were able to use a more *compact* IFS representation; each transformation is encoded by means of only three numbers: the translations (e and f) plus a number t, between 0 and 3, which encodes the geometric transformation.

This procedure was implemented in a Constraint Logic Programming (CLP) language with constraints on finite domains [1]; it iteratively tries to cover a yet uncovered part of the digit with a contracted box; among all the possible transformations, it chooses the one that covers the greatest number of pixels belonging to the digit (Figure 5). CLP was used because it allows a fast prototyping and also because it allows to cut the search space in a very simple way.

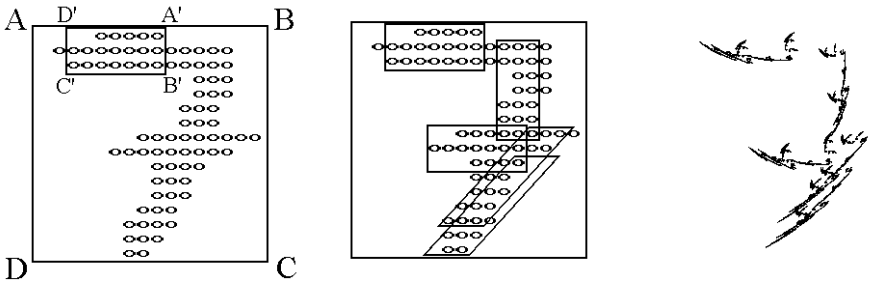


Fig. 5. (a) Sample digit inside its square box $ABCD$ ("o"s represent black pixels): application of the first affine contraction resulting in $A'B'C'D'$; (b) Final coverage using 5 contractions; (c) Fractal reconstruction: note how the digit structure is captured.

Different experiments were carried on: on one hand, we used different parameter values in order to find the encoding that gave the best results, while on the other we have applied different learning algorithms to compare their performances. In particular, the best results were obtained (alternatively) covering the data set by means of: 5 transformations where the 16×16 pixel frame square box is scaled to a 7×3 pixel rectangle; 6 transformations with a scaling to a 5×3 pixel rectangle; 6 transformations with a scaling to a 5×3 pixel rectangle, the difference with the previous case being that now the data were not binarized

any more, i.e., the 256 grey levels were maintained and the search for the best matching transformations took into account also the intensity level of the pixels.

The best results of this series of experiments are reported in Table 1. They were obtained using 6 transformations with a scaling to a 5×3 pixel rectangle. Columns (a), (b), (c), and (d) contain the average recognition rates obtained on the test sets digit per digit. Column (a) reports the results obtained using CART. Two different experiments were carried on: in the first, we used *symbolic* features only (this could be done because positions are discrete and vary in between 0 and 15 and the transformations allowed are only four), whereas in the second, the positions were considered *numerical* values. The best results were obtained in the second case.

Column (b) contains the results obtained by the Genetic Algorithms (here, all features are *symbolic*). They were obtained by means of system REGAL [19].

The greatest performances were obtained by neural networks (see columns (c) and (d)). Different network models were tried. Column (d) reports the results obtained by using Cascade-correlation; however, the model that gave the best and more stable (w.r.t. the parameters used during encoding) performances is the SCG network [23]. This network is full-connected and is characterized by a learning rule that exploits the second derivatives and converges faster than classical gradient descent methods. The nets we used always had 500 hidden neurons and 10 output neurons and were run for about 500 epochs.

Experiments with a general purpose encoder. At last, we tried a general purpose fractal image compression program for generalizing the method in order to apply it to *any* kind of image: *enc* by Fisher [10]. This algorithm exploits the Collage Theorem for *local IFSs* (see [5]): instead of covering the whole image with contracted affine copies of itself, it covers the image with contracted affine copies of *parts* of itself. A transformation can be specified by only *five numbers*:

(a) (b)

Fig. 6. (a) Some images we have used in our experiments. (b) Their reconstruction with 16 transformations.

a number t between 0 and 7 that codes the selected geometrical transformation (only the identity, 90°, 180° and 270° anticlockwise rotations, the reflection in the two axes — x and y — and in the two diagonals are allowed), two numbers, e and f, for the translation, and two real numbers, s and o, indicating the luminance transformation (these two numbers are necessary to compression because *enc* works on grey scale images, see for more details [16]).

In these experiments each digit was approximated by 16 transformations (each R_i being a 4×4 square), making 80 features in all (16 transformations $\times 5$ parameters). For instance, Figure 6(b) contains the reconstructions of the images of Figure 6(a). Due to the dimension of the input space, and also to the results obtained in the previous experiments, the learner was implemented by means of neural networks only. Three NNs had to be used: the first, having 250 hidden neurons (as well as the third one), was trained to recognize digits from "0" through "3" using as counterexamples also all digits from "4" through "9". The second had only 200 hidden neurons and was trained to classify digits "4", "5," and "6". The third learned to classify the remaining digits. The learning set was made of 500 digits (50 per class) while the test set was made of 2500 samples (250 per class). Results are reported in column (e) of Table 1.

5 Conclusions

Since Mandelbrot [17] defined them, fractals have been widely addressed in the literature and, in strict connection with them, IFSs have been developed and used in various domains [21]. In this paper we tackled the problem of using *IFS encodings* for classifying images of objects which do *not* have a self-affine nature by, first, showing that such features *capture* discriminant information and, then, presenting an *algorithm* that allows to extract them in an *automatic way* for the specific application of the ten digits. Experiments show that the IFSs can, actually, be used for classification tasks, achieving, in the best case, an average 85% recognition rate. Even using an automatic fractal encoder that was designed for image compression and *not* for recognition purposes, a recognition rate greater than 70% was achieved for five classes out of ten.

The digits application was chosen because, on one hand, it is a *real-world* (non-fractal) image recognition problem and, on the other, it was used in the StatLog project, it is well-understood and was tackled by means of many different learning algorithms. Nonetheless, those experimental settings and results have been an important reference point and lead to the following conclusions. First, the use of IFSs allows to work on a *very small* input space by reducing both the number of space dimensions and the set of possible values for each input feature. Consequently, the learning time diminishes. For instance, let's consider the input space of column (b) in Table 1: each of the 6 transformations is determined by 3 parameters, two of them (the translations e and f) range between 0 and 15 (the image side size), and one, t, ranges between 0 and 3. The original images, instead, are represented by 256 pixels, each of which has a value in between 0 and 255 (the grey level).

Another appealing characteristic is that the number of features used for encoding is *independent* from the size of the image, being related to its intrinsic complexity only. These characteristics are extremely important because, as explained in [18], many learning algorithms cannot deal with applications having hundreds of dimensions, while those that can take a too long time before converging.

However, if we compare the *recognition rates* obtained in the experiments carried on during StatLog to ours we see that a lot of work is yet to be done: some of the methods used in this project achieve a 90–95% average recognition performance. Taking into account these percentages only, those methods seem better than ours. In order to better understand the meaningfulness of fractal features, we have, then, applied the neural network model that gave the best results on fractal features (the SCG model) to the 300 learning samples, now encoded as in StatLog. In the average, the net could recognize about 80% of the the 700 test samples. This difference of performance between the results obtained during StatLog and those obtained in this last experiment is due to the different number of learning examples used: in StatLog the classifiers were trained on 9,000 images while our classifiers are trained on 300 images only. Taking into account this result and also the fact that the encoder we developed is quite rough, we can reasonably suppose that finding more sophisticate encoding algorithms fractal features will allow to obtain (or even outperform) recognition rates that are very close to those obtained with different pre-processing methods, using less examples and training the classifiers in a shorter time.

Some interesting conclusion can be drawn also by comparing the performances of the different learning methods we applied. As can be observed, *numerical* methods (i.e. neural networks) gave the best results. Depending on the parameters of the encoding algorithm, SCG networks achieved a recognition rate in between 82% and 85%. Cascade-correlation networks (as an example of a different network model) achieved 79%. Symbolic methods, instead, achieved a maximum 72%. We believe that the reason for which neural nets (and, in particular, full-connected neural nets) perform better is that they can capture the structure of a sample, leaving out the particular order of the transformations in it. In different words, since digits are hand-written and binarized (i.e. they are affected by noise) and since the encoding algorithm selects first those transformations that cover more bits on the sample, if we take two samples of a same digit, say two "4", we cannot expect that the order of the transformations used to cover the two samples is the same. Let's suppose, for instance, that in the first "4" the vertical bar is strongly marked while in the second, the oblique line is strongly marked. Then, it is very likely that the first transformations generated to encode the first "4" cover the vertical line whereas those created to cover the second "4" cover the oblique line. This means that the two samples will be encoded by two sequences $T_{11}, ..., T_{1n}$ and $T_{21}, ..., T_{2n}$ where, for example, T_{11} covers the vertical bar in the first sample while T_{21} and T_{23} cover respectively the oblique and the vertical lines in the second sample. Note that the information contained in the two sequences will be the same, the only difference is the *order*

of the transformations. Now, our algorithm imposes a simple ordering of the transformations which is based on the position of the baricentre of the contractions; however, due to the non-regularity of hand-writing, the above mentioned problem can still arise (it is easy to imagine examples of instances of a same digit where such an ordering rule produces different sequentializations). Unfortunately, the symbolic methods we used are not able to generalize over different attributes, i.e., over information placed at different points in the sequences, e.g. T_{11} and T_{23} in the example. To overcome this limit, we mean to try *First Order* learning tools. Two alternatives are either to search for more sophisticate ordering rules that, for instance, take into account the relationships between the transformations, or to produce reacher learning sets that take into account any possible ordering of the transformations for any given sample.

A second and more challenging future research will consist in studying a fractal feature extraction method that is *domain-independent* and allows to characterize *any* kind of image. Actually, the experiment with *enc* was a first step towards this goal. *Enc*, was chosen because it *is* domain-independent; the problem is that it is too *application-dependent*, where the application, in this case, is compression. An idea we would like to explore is to try to change the range and domain shapes using a segmentation algorithm to drive the selection. Another possibility is to try to exploit the Hausdorff distance [15], used in fractal geometry to evaluate the distance between sets of points.

Acknowledgments. The authors are indebted to prof. Lorenza Saitta for the helpful discussions and suggestions and thank prof. Charles Taylor for all the information about the StatLog project.

References

1. *ECL^{i}PS^e 3.5 Extensions User Manual.* ECRC GmbH, 1995.
2. R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka. Analyzing images containing multiple sparse pattern with neural networks. In *Proc. of IJCAI-91*, Sidney, Australia, 1991.
3. M. Barnsley. *Fractals Everywhere.* Academic Press, San Diego, 1988.
4. M. Barnsley and S. Demko. Iterated function systems and the global construction of fractals. In *The Proceedings of the Royal Society of London*, volume A399, pages 243–275, 1985.
5. M. Barnsley and L. P. Hurd. *Fractal Image Compression.* AK Peters, Ltd., Wellesley, Massachusetts, 1993.
6. P. Besl and R. Jain. Three dimensional object recognition. *ACM Computing Surveys*, (17):75–154, 1985.
7. L. Breiman, J. Friedman, J. Ohlsen, and C. Stone. *Classification and Regression Trees.* Wadsworth & Brooks, Pacific Grove, CA, 1984.
8. G. Le Chiara and L. Saitta. Using fractals to learn image descriptions by means of artificial neural networks. In *IEEE International Conference on Neural Networks*, Orlando, USA, 1994.

9. R. Chin and C. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, (18):67–108, 1986.
10. Y. Fisher (Ed.). *Fractal Compression: Theory and Application to Digital Images.* Springer Verlag, New York, 1994.
11. K. Falconer. *Fractal Geometry, Mathematical Foundations and Applications.* John Wiley & Sons Ltd., Chichester, UK, 1990.
12. D.E. Goldberg. *Genetic Algorithms.* Addison-Wesley, Readings, MA, 1989.
13. V. K. Govindan and A. P. Shivaprasad. Character recogniton - A review. *Pattern Recognition*, 23(7):671–683, 1990.
14. W. Grimson, Lozano-Pérez, and D. Huttenlocher. *Recognition by Computer: The Role of Geometric Constraints.* The MIT Press, Cambridge, MA, 1990.
15. D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, (PAMI-15):850–863, 1993.
16. A. Jacquin. Image coding based on fractal theory of iterated contractive image transforms. In *Proc. of SPIE, Visual Communications and and Image Processing '90*, volume 1360, 1990.
17. B. Mandelbrot. *The Fractal Geometry of Nature.* Freeman & Co., San Francisco, CA, 1982.
18. D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine learning, neural and statistical classification.* Ellis Horwood series in artificial intelligence. Prentice Hall, 1994.
19. F. Neri and A. Giordana. A distributed genetic algorithm for concept learning. In *Int. Conf. on Genetic Algorithms*, pages 436–443, Pittsburgh, PA, 1995. Morgan Kaufmann.
20. D. Oliver. *Fractal Vision, Put Fractals to Work for You.* Sams Publishing, Indiana, USA, 1992.
21. A. P. Pentland. Fractal-Based Description of Natural Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):661–674, 1984.
22. A. Rosenfeld and A. Kak. *Digital Picture Processing.* Academic Press, New York, NY, 1982.
23. P. D. Wasserman. *Neural computing.* 1995.
24. C. J. Wu and J. S. Huang. Human face profile recognition by computer. *Pattern Recognition*, 23(3/4):255–259, 1990.

This article was processed using the LaTeX macro package with LLNCS style