KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2007
Cluj-Napoca (Romania), June 6–8, 2007, pp. 169–176

GENETIC PROGRAMMING WITH HISTOGRAMS FOR HANDWRITTEN RECOGNITION

OANA MUNTEAN(1)

Abstract. Handwritten recognition is a popular problem which requires special Artificial Intelligent techniques for solving it. In this paper we use Genetic Programming (GP) for addressing the off-line variant of the handwritten digit recognition. We propose a new type of input representation for GP: histograms. This kind of representation is very simple and can be adapted very easily for the GP requirements. Several numerical experiments with GP are performed by using several large datasets taken from the well-known MNIST benchmarking set. Numerical experiments show that GP performs very well for the considered test problems.

1. Introduction

Handwriting recognition concerns the conversion of the analog signal of handwriting into a digital symbolic representation. The analog signal can be in the form of either a two-dimensional scanned image of paper or a temporal one-dimensional signal captured from a device such as a tablet or PDA.

It is widely accepted that a perfect recognition of digits requires intelligence. This is why Artificial Intelligence technique have been intensively used for solving this problem. Among them, Artificial Neural Networks are the most popular.

In the recent years several techniques inspired from nature have emerged as good alternatives to the standard mathematical methods. One of them is Genetic Programming (GP) [6] whose aim is to generate computer programs based on a high level description of the problem to be solved.

In this paper we use GP for handwritten digit recognition. The novelty consists in the representation of the input. Because the matrices, encoding the images involved in the numerical experiments, are too big to be used as direct input for GP we have to extract and use only some partial information. This is why we have constructed the horizontal and vertical histograms [3] and this information was used as input for GP.

2000 *Mathematics Subject Classification.* 68T10, 68T20.
Key words and phrases. Genetic Programming, Handwritten recognition, Classification.

We have performed several numerical experiments by using a well-known benchmarking data set: MNIST [9]. Results have shown a more than 90% classification accuracy for most of the cases. For some tests the accuracy was more than 99%.

The paper is organized as follows: GP technique is briefly surveyed in Section 2. The handwritten digit recognition problem is defined in Section 3. Related work in the field of handwritten recognition is briefly reviewed in Section 4. The proposed approach is deeply presented in Section 5. Test problems are given in Section 6. The results of numerical experiments are given in Section 7. Strengths and limitations of the proposed technique are discussed in Section 8. Conclusions and future work directions are given in Section 9.

2. Genetic Programming

Genetic Programming (GP) technique provides a framework for automatically creating a working computer program from a high-level problem statement of the problem [6, 7]. Genetic programming achieves this goal by genetically breeding a population of computer programs using the principles of Darwinian natural selection and biologically inspired operations.

Five major preparatory steps [6] must be specified in order to apply a GP technique to a particular problem:

(1) the set T of terminals (e.g., the independent variables of the problem, zero-argument functions and random constants)
(2) the set F of primitive functions,
(3) the fitness measure (for explicitly or implicitly measuring the quality of individuals in the population),
(4) certain parameters for controlling the run
(5) the termination criterion and the method for designating the result of the run.

These preparatory steps are problem dependent so they must be specified, by a human user, for each particular problem.

3. Handwritten recognition problem

The recognition of handwritten characters by a machine is a very tough task. It takes years to come to a commercially valuable result in this area - years of research, years of development and years of turning theory into software that would do at least a small piece of an evident thing: turning human handwritten digital curves into letters and words. This work involves identifying a correspondence between the pixels of the image representing the sample to be analyzed (recognized) and the abstract definition of that character.

However, a lot of work has been done in this field lately and this happened due to the fact that it has an important role in many applications such as bank checks and tax forms reading, postal addresses interpretation, etc.

Another important thing to be considered about handwritten recognition methodologies is the manner in which the data is collected. There are two methods for accomplish this: on-line [11, 13] or off-line [10]. On-line handwritten data is collected using a digitizer or an instrumented pen to capture the pen-tip position $(x_t,\ y_t)$ as a function of time. Contrary to on-line, off-line handwritten data is collected using a scanner resulting in the generation of the signal as an image.

This paper presents an off-line implementation of the problem and focus on digits handwritten recognition; however it is possible to expand to full-character recognition.

4. Related work

A lot of work has been done in the field of handwritten recognition. The main drawback of this problem is the extremely variability of handwritten that produces a large number of features for different writing style. That is why there is no single method for solving handwritten recognition.

Artificial Neural Networks [5] are very popular techniques for solving this problem [1, 2, 11]. Although, GP [6] was suggested [12, 14] as an alternative method for attacking this problem.

In this section we consider the current state of research, the developments in the handwriting recognition field and present some related research work using GP algorithms. The basic idea of these algorithms is the classification of data in different classes, starting from different inputs and different way of representation.

One of the handwritten recognition approaches using GP uses Decision Trees to solve the problem [12]. Here two types of algorithms might be evolved: starting from bottom-up or top-down. The first one starts by dividing all digits into two different classes and applying the same pattern to both subtrees, while the second starts by classifying pairs of digits and continuing by a larger classification, i.e. from a binary classification, a classification with four classes can be performed.

Another GP approach for the problem focuses on the particular features of the characters using recursive cognition [14]. In this way the feature of an image containing a character is hierarchically divided into small sub-images and the process is applied recursively until an acceptance criteria is met (the character is recognized).

Similar to this, the fuzzy regional representation [10] is based on the features of the character, but here the image is divided into a grid from the beginning of the process. Each region of the grid is further included into a fuzzy vector, that will provide the particularities of the analyzed character.

5. Proposed approach

The proposed approach uses standard GP [6] as the main search mechanism.

What is different in our paper is the way in which the input is represented. GP ability to solve problems is highly related to the number of terminals [6]. If we have too many terminals it is more difficult for GP to select the good ones.

The original data sets consist in images represented as 20x20 matrices. If all pixels of this matrix are sent as input to GP we would have 400 terminals. This is a huge number for GP.

This is why we have tried to reduce the number of inputs. We can do that by extracting some information from the original 20x20 matrices. In this paper we have built the horizontal and vertical histograms and this information was actually sent as input to GP. This information is further processed by the GP classifier.

Histograms representation is very simple [3]. For each row and each column of the image we count the pixels containing ink (see Figure 1). The obtained number can give us some rough information about what digit we have there. Histograms have been used in the past for handwritten recognition [3]. However, this is the first time when they are used in conjunction with Genetic Programming.

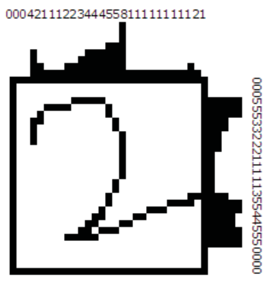

FIGURE 1. Histograms for digit 2. For each row and column we have counted the ink pixels.

6. Test data

The MNIST is a database [9] of handwritten digits having a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST [4]. The digits have been size-normalized (to a 20x20 matrix) and centered in a 28x28 fixed-size image. In our case we have extracted only the information from the central 20x20 matrix. By building the histograms we have obtained a dataset with 60000 examples, each of them having 40 attributes.

7. Numerical experiments

First of all we wanted to find out if the method has enough power to distinguish between pairs of digits. For instance we are interested to find a mathematical formula that can be applied in order to distinguish between 0 and 1.

In this case we deal with a classification problem with 2 classes. The fitness is computed as the number of incorrectly classified examples over the total number of cases. Thus, the fitness has to be minimized.

For this purpose we have extracted from MNIST all 45 distinct pairs of digits. Secondly we have built the histograms and then we have applied GP for all of them.

Parameters of the GP method used in all numerical experiments are given in Table 1.

Table 1. General parameters of the GP algorithm

Parameter	**Value**
Population size	200
Number of generations	51
Mutation probability	0.1
Crossover probability	0.9
Selection	Binary Tournament
Terminal set	Problem inputs (40 as many)
Function set	$F = \{+, -, *, \%\}$
Maximum GP tree height	10

7.1. **Results.** The classification accuracy obtained by running GP method for all test problems is given in Table 2. Because the method works with pseudo-random numbers, we performed more runs (30 runs in fact) and we have averaged the results. In all runs we have obtained a very good classification error.

Taking into account the average values presented in Table 2 we can observe that the best error is obtained for classifying 6-7 digit pair. This means that is very easy to make distinction between digits 6 and 7. If we take into account the best values we can also see that the best classified pair of digits is again 6-7. The worst results are obtained for 3-5 pair and, then, for 5-8 pair. This means that is very difficult to make distinction between digits 3 and 5.

Note that the results can be further improved by using a larger population or running the search process for more generations.

7.2. **Comparison with other GP techniques.** In [14] GP with a special representation was used for digit classification. The reported error was between 2% and 5%. In [10] the error was between 3% and 9%.

In Table 3 we have compared our approach with another GP-based technique proposed in [8] on the same data set. The results show that our method outperformed the compared one in all cases.

Note that a perfect comparison cannot be made because the representations and the parameter settings are too different.

Table 2. The results (in percent) obtained by applying GP to the considered test problems. *Best/Worst* stands for the fitness of the best individual in the best/worst run. *Avg* and *StdDev* are the mean and the standard deviation of the quality for the best individual (in each run) over 30 runs. ds_{xy} means the dataset which contains only the images with digits x and y.

Data	*Best*	*Worst*	*Avg*	*StdDev*	*Data*	*Best*	*Worst*	*Avg*	*StdDev*
ds_{01}	3.32	5.29	4.36	0.61	ds_{29}	0.98	2.16	1.26	0.33
ds_{02}	2.89	6.16	4.04	0.88	ds_{34}	1.24	2.48	1.85	0.43
ds_{03}	3.73	5.21	4.44	0.50	ds_{35}	13.77	20.58	17.78	2.35
ds_{04}	0.84	1.43	1.12	0.19	ds_{36}	1.79	4.43	2.65	0.98
ds_{05}	5.89	14.28	8.33	2.60	ds_{37}	4.51	7.05	5.53	0.85
ds_{06}	1.63	3.02	2.08	0.43	ds_{38}	8.95	15.74	11.94	2.07
ds_{07}	1.18	2.57	1.83	0.43	ds_{39}	3.74	5.78	4.49	0.65
ds_{08}	5.23	6.85	5.91	0.50	ds_{45}	2.01	4.38	3.43	0.81
ds_{09}	1.06	2.15	1.61	0.38	ds_{46}	1.66	2.63	1.98	0.31
ds_{12}	4.52	9.42	7.42	2.17	ds_{47}	2.10	3.97	3.16	0.61
ds_{13}	7.13	13.09	9.77	2.07	ds_{48}	1.93	3.89	2.57	0.61
ds_{14}	1.03	6.32	2.12	1.68	ds_{49}	9.39	12.40	10.48	0.92
ds_{15}	5.91	10.95	8.63	1.76	ds_{56}	2.05	2.65	2.33	0.19
ds_{16}	2.18	4.90	2.71	0.79	ds_{57}	4.21	8.23	6.16	1.62
ds_{17}	1.30	6.30	2.09	1.50	ds_{58}	9.35	17.33	12.83	2.57
ds_{18}	11.10	14.80	13.06	0.97	ds_{59}	3.21	6.18	5.01	1.07
ds_{19}	1.85	3.21	2.52	0.51	ds_{67}	0.14	0.69	0.32	0.20
ds_{23}	6.86	11.20	8.23	1.23	ds_{68}	2.38	3.41	2.91	0.36
ds_{24}	1.48	2.37	1.77	0.23	ds_{69}	0.26	0.98	0.41	0.22
ds_{25}	9.00	11.58	10.21	0.91	ds_{78}	5.10	7.05	6.08	0.79
ds_{26}	6.27	9.88	7.82	1.20	ds_{79}	6.18	9.76	8.16	1.16
ds_{27}	1.36	3.58	1.97	0.64	ds_{89}	4.00	7.48	5.49	1.03
ds_{28}	3.02	5.81	4.29	0.87					

8. Strenghts and weaknesses

Genetic Programming strengths and weaknesses are already known to the research comunity. This is why we focus our attention to the advantages and disadvantages introduced by the representation with histograms.

The greatest benefit is the fact that we have been able to successfully apply GP for solving this problem. This is due to the reduced number of inputs which are sent to the GP individuals. Note that this was not possible if we send the entire

TABLE 3. Comparison between our approach and the one proposed in [8]. For each pair of digits we gave the errors obtained by both methods.

Pair	**Our approach**	**The approach proposed in** [8]
(1,7)	2.09	4.30
(2,7)	1.97	7.10
(3,8)	11.94	13.60
(4,9)	10.48	12.10
(8,9)	5.49	8.80

matrix to GP because the method cannot successfully handle such large amount of inputs.

The limitations of the proposed approach are mainly related to the limitations introduced by the histogram representation. There are several pairs of digits which are difficult to be distinguished when using this kind of representation. For instance the pairs 1-7, 3-8, 9-6 have very similar histograms and thus it is quite difficult to find a very good classification.

Luckily, our test data contains digits written by hand. In this case there is a more clear distinction between the digits from the previously enumerated pairs since the hand written characters have a huge number of possible representations. This fact has been shown by the results from Table 2 where the classification errors are quite good for pairs 1-7, 3-8 and 9-6.

Another weakness is that by using histograms we have reduced the amount of information which was sent to GP classifier. This could lead to some poor results in some cases.

9. Conclusions and further work

In this paper a new way of representing the input for solving handwritten recognition problems using GP has been suggested.

The proposed representation was tested on a well-known benchmarking data set. The results of the numerical experiments have shown very good classification accuracy.

Further efforts will be focused on the following directions:

- Testing the method for other difficult datasets (including characters).
- Reducing the size of the images. This will reduce the number of inputs for Genetic Programming.
- Using an extended set of functions for GP. This set may include the operators sin, exp, lg.

- Discovering classifiers which are able to make distinction between a particular digit and all other digits. This will increase the generalization ability of the method.

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[2] Y. L. Cun and (et al). Handwritten digit recognition with a back-propagation network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 396–404. Morgan Kaufmann, 1990.

[3] H.-C. Fu, H. Y. Chang, Y. Y. Xu, and H. T. Pao. User adaptive handwriting recognition by self-growing probabilistic decision-based neural networks. *IEEE-NN*, 11(6):1373–1384, 2000.

[4] M. D. Garris and (et al). NIST form-based handprint recognition system. In *National Institute of Standards and Technology (NIST) Intelligent Systems Division*, 1994.

[5] M. Hassoun. *Fundamentals of Artificial Neural Networks.* MIT Press, 1995.

[6] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[7] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, 1994.

[8] K. Krawiec. Genetic programming using partial order of solutions for pattern recognition tasks. In E. Puchala, editor, *Proceedings of the second National Conference on Computer Recognition Systems KOSYR-2001*, pages 427–433, Strona palacu w Milkowie, Karpacza, Poland, 28-31 May 2001.

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

[10] A. Lemieux, C. Gagne, and M. Parizeau. Genetical engineering of handwriting representations. In *Frontiers in Handwriting Recognition*, pages 145–150, 2002.

[11] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell*, 22(1):63–84, 2000.

[12] T. Tanigawa and Q. Zhao. A study on efficient generation of decision trees using genetic programming. In D. W. (et al), editor, *Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 1047–1052, Las Vegas, Nevada, USA, 2000. Morgan Kaufmann.

[13] A. Teredesai and (et al). On-line digit recognition using off-line features. In S. C. (et al), editor, *ICVGIP 2002, Proceedings of the Third Indian Conference on Computer Vision, Graphics & Image Processing, 2002.* Allied Publishers Private Limited, 2002.

[14] A. Teredesai, J. Park, and V. Govindaraju. Active handwritten character recognition using genetic programming. In J. F. M. (et al), editor, *Genetic Programming, Proceedings of EuroGP'2001*, volume 2038 of *LNCS*, pages 371–379. Springer-Verlag, 2001.

[1] Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Kogălniceanu 1, Cluj-Napoca, 400084, Romania

E-mail address: `oana_muntean85@yahoo.com`