To appear in *Materialy II Krajowej Konferencji Naukowo-Szkoleniowej 'Komputerowe Systemy Rozpoznawania' KOSYR'2001* (accepted)

GENETIC PROGRAMMING USING PARTIAL ORDER OF SOLUTIONS FOR PATTERN RECOGNITION TASKS

Krzysztof Krawiec[1]

[1]Institute of Computing Science, Poznan University of Technology
Piotrowo 3A, 60-965 Poznan, krawiec@cs.put.poznan.pl

ABSTRACT

This paper investigates the use of genetic programming (GP) for learning of pattern recognition programs. The central topic here is the introduction of GP incorporating *partial* order of solutions as opposed to the standard *complete* (linear) order imposed by the scalar fitness function. We claim that such an extension protects the 'interesting', however worse w.r.t. the value of the fitness function, solutions from being discarded in the selection process, and thus increases the diversity of the population. That hypothesis is verified on a real-world case study concerning the recognition of handwritten characters.
Keywords: machine learning, visual learning, genetic programming, handwritten character recognition.

INTRODUCTION

The motivation for this paper comes from the following observation. Let us assume that we perform a search in a discrete space of solutions using some search algorithm and an evaluation function. In the case of common search algorithms, like tabu search, simulated annealing or evolutionary computation, it is a usual assumption, that the evaluation function returns a scalar numerical value as the measure of the 'fitness' (quality) of a particular solution. An advantage of such an approach is the clear and simple interface between the search engine and the evaluation function, what makes their replacement easy.

However, there is a price we pay for this simplification, which consists in imposing a *complete* (linear) order of solutions onto the search space. As a consequence, it is assumed that solutions are always *comparable* and that, given a pair of them, we are always able to point the better one, unless they have the same value of the evaluation function.

The primary claim of this paper is that in fact solutions may be *incomparable* and that it is possible to commit a serious oversimplification when comparing them in the complete order framework. This statement is widely accepted in the multiple objective optimization (see, for instance, [13], or, for review, [14]), where the solutions are evaluated w.r.t. their different features. However, here we show that incomparability of solutions may have different origins than the presence of multiple, explicitly defined, objectives. In particular, we focus here on the case when evaluation of a particular solution is based on a *set* of some entities and aggregates somehow the behavior of the solution on particular elements of this set. This setting is characteristic for machine

learning, where solutions are hypotheses, the aforementioned set contains training cases, and the evaluation function is usually the accuracy of classification. This paper focuses on this perspective.

PARTIAL ORDER OF SOLUTIONS

Let us start with an illustrative example embedded in the machine learning environment, where the above-mentioned search is controlled by the so-called *inducer* (e.g. decision tree inducer) and takes place in the space of hypotheses (decision trees, respectively) evaluated on the set T of training examples [10]. For a particular hypothesis h, the evaluation function f returns its accuracy of classification on the training set, the simplest and the most widely used measure of hypothesis quality. Suppose there are three hypotheses (solutions), h_1, h_2 and h_3, characterized by subsets of correctly classified examples H_1, H_2 and H_3, respectively. Thus, for instance $f(h_1)=|H_1|/|T|$. Let us assume, that $|H_1| > |H_2| = |H_3|$. Then, with respect to f, hypotheses h_2 and h_3 are of the same quality and are both worse than h_1.

However, having a closer look at the subsets H_1, H_2 and H_3 and their mutual relations illustrated in Fig. 1 should incline us to revise some of the above statements. As $H_2 \subset H_1$, the superiority of h_1 to h_2 is still well founded, but what about the relation between h_1 and h_3? Although h_1 classifies correctly more examples than h_3, there is a remarkable subset of examples ($H_3 \setminus H_1$), which it doesn't cope with, while they are successfully classified by h_3. Thus, superiority of h_1 to h_3 is doubtful and, as the same applies when considering the superiority of h_3 to h_1, the question concerning their mutual relation should probably remain without answer, leading us to the concept of the solution *incomparability*. Then, incomparability implies a *partial* order in the solution space (as opposed to the *complete* order imposed by scalar evaluation function f) and, in particular, the possibility of simultaneous existence of many 'best' solutions (even with different values of f).

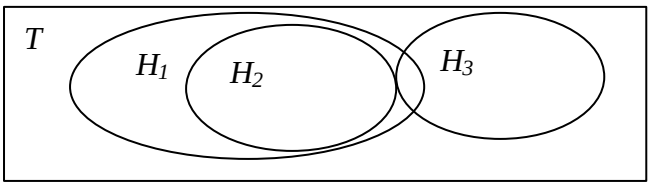

Fig. 1. An example of comparable and incomparable hypotheses.

Ignoring this issue and forcing the evaluation procedure to impose a complete order of solutions may result in an undesirable behavior of the search algorithm, which discards solutions inferior (even slightly) w.r.t. accuracy of classification (h_2 and h_3), although some of them (h_3) discover some new knowledge from the training data. To prevent the search algorithm from losing such 'interesting' solutions, we should redefine the interface between search engine and evaluation module, replacing the scalar evaluation function by *pairwise comparison of solutions*. Such a redefinition is generally not straightforward, as most metaheuristcs rely on numerical evaluation and complete order of solutions. In this paper, the above-mentioned idea is being embedded in the metaheuristics of evolutionary computation [2], or, more specifically, genetic programming [4]. This choice is mostly due to the fact that GP is reported to be very effective in solving a broad scope of problems, including the search for pattern recognition programs [2][11], what was also subject of our former research [5][6][7].

To make the idea work, we have to define formally the *outranking relation*, denoted thereafter by '≥', between two solutions (hypotheses) h_1 and h_2, given the sets of examples (H_1 and H_2, respectively) properly classified by these hypotheses. According to the definition of outranking [12], $h_1 \geq h_2$ should express the fact that h_1 is *at least as good* as h_2. To keep the approach as simple as possible, we decided to define the outranking in a very intuitive way: $h_1 \geq h_2$ iff $H_2 \subseteq H_1$ [1]. Note that, as a consequence, even a single training example may disable the outranking ($|H_2 \setminus H_1|=1$), what implies some vulnerability of this relation. This is however the price we decided to pay for keeping the approach simple and non-parametric. Finally, four cases are possible: h_1 is better than h_2 (when $h_1 \geq h_2$ and not $h_2 \geq h_1$), h_2 is better than h_1 (analogously), h_1 is indiscernible with h_2 ($h_1 \geq h_2$ and $h_2 \geq h_1$) or h_1 and h_2 are incomparable (neither $h_1 \geq h_2$ nor $h_2 \geq h_1$).

Provided the pairwise comparison, we have to decide how to build up the mating pool, i.e. how to select the best solutions from the population P_t in t-th generation of GP run taking into account the potential presence of incomparability. In the preliminary research, we tried to extend for this purpose the popular tournament selection scheme [4]. Unfortunately, that approach did not yield satisfactory results in experimental evaluation, probably due to the fact, that the presence of incomparability generally decreases the selection pressure (some tournaments remain unsettled). Thus, the approach presented in this paper takes another way and consists in selecting the 'non-outranked' solutions, i.e. such solutions $h \in P_t$ that $\neg\exists\, h' \in P_t$: $h' \geq h$. However, as there are usually relatively few such solutions, the missing part of the mating pool is filled up with solutions obtained by means of the standard tournament selection.

CASE STUDY: LEARNING HANDWRITTEN DIGIT DISCRIMINATION

The proposed idea has been adopted in genetic programming-based *visual learning*, which was the subject of our previous research [5][6]. Here, the candidate programs (solutions) performing image analysis and recognition are evaluated on a set of training cases (images), called *fitness cases* in the GP terminology. GP searches the space of pattern recognition procedures formulated in a specialized language called GPVIS [7]. GPVIS is an image analysis-oriented language encompassing a set of operators responsible for simple feature extraction, region-of-interest selection, and numerical and logical operations. Despite its simplicity, it allows for formulating a complete pattern recognition program without the need for external machine learning classifier, what is usually when the processing is split into the feature extraction module and the reasoning module. Figures 2 and 3 show an example of image recognition program formulated in GPVIS (see [7] for details on GPVIS syntax and GP search using this representation of solutions).

As the experimental test bed for the approach, we chose the problem of handwritten character recognition, which is often referred to due to its wide scope of real-world applications. The solutions proposed in literature incorporate statistics, structural/syntactic methodology, sophisticated neural networks, or ad hoc feature extraction procedures, to mention only a few (for review, see [9]). The approach presented in this paper cannot be univocally classified into any of these categories, and

[1] To be more precise, this condition must hold simultaneously and separately for examples representing particular decision classes.

combines the *elasticity* and *learning* capability of adaptive systems with *comprehensibility* of symbolic approaches.

The data source was the MNIST database of handwritten digits [9]. It consists of two subsets, training and testing, containing together 70,000 digits written by approx. 250 persons (students and clerks), each represented by a 28×28 matrix of gray level pixels (Fig. 4). Characters are centered and scaled with respect to their horizontal and vertical dimensions, however, no 'deskewing' has been carried out.

```
(or (and (poutside (shift (absRoiN 19 8 2 4) (absPoint 12 0))(absPoint
15 16))(> (x (absPoint 21 14))(y (absPoint 25 4))))(routside (shift
(adjust (absRoiN 7 23 10 18)) (absPoint 7 16))(shift (shift (absRoiN
13 26 3 10)(absPoint 15 4))(absPoint 14 20))))
```

Fig. 2. A LISP-like representation of an exemplary solution in GPVIS language.

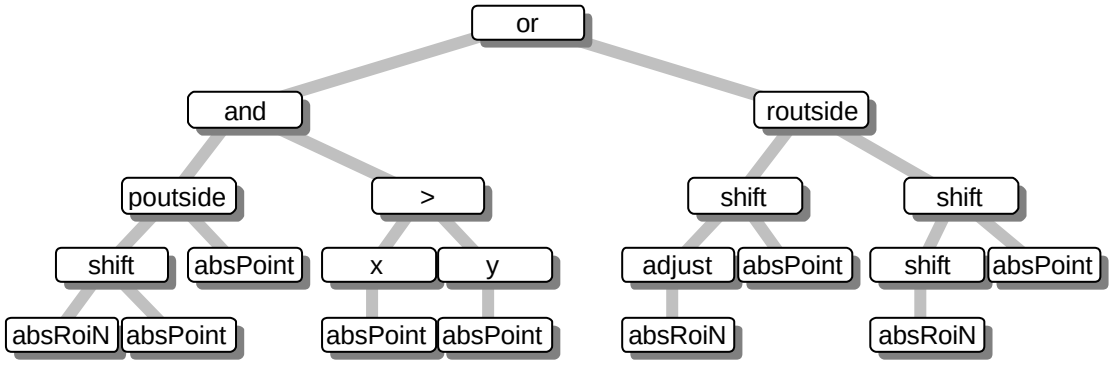

Fig. 3. The graphical representation of the solution from Fig. 3 (numerical values omitted).

COMPUTATIONAL EXPERIMENT

The experiment described in this section compares the efficiency of 'plain' GP and GP using partial order of solutions (GPPO), mostly w.r.t. the accuracy of classification of the resulting solution (image analysis and recognition program).

Programs formulated in GPVIS return logical value (true or false), so it is impossible to build the complete digit recognition system using it in a direct way. Therefore, we should decompose the ten-classes problem of digit recognition into binary classification tasks, where the decision can be computed by an expression written in GPVIS. Such decomposition may be done in several ways; for details related to this problem the reader should refer to the literature of the so-called *meta-classifiers* (e.g. [1]). However, as it was not the central topic here, in this experiment we focus on selected *pairs* of decision classes. To make the task more realistic, we used these pairs of classes, which are among the most difficult to discriminate for both humans and pattern recognition systems, i.e. (1,7), (2,7), (3,8), (4,9), and (8,9).

The genetic search was ran with the same values of parameters in both GP and GPPO cases: population size: 500; probability of mutation: 0.05; maximal depth of a randomly generated solution (initialization): 2 ('soft' limit); maximal depth of a randomly generated subexpression (mutation): 3; maximal number of generations: 50 (stopping condition); training set (set of fitness cases) size: 100 cases (50 images per class); tournament selection scheme with tournament size equal to 5. The solutions were modified in the common way. The mutation selects at random a term in the solution and

replaces it (and its antecessors) by a randomly generated subexpression. The crossover operator selects at random terms in the two parent solutions and exchanges them together with their antecessors. However, as the GPVIS language uses types, in these operations one should obey the so-called *strong typing* principle [4].

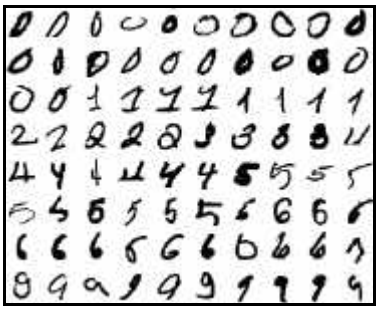

Fig. 4. Exemplary images from the MNIST database.

Table 1 presents the results obtained in GP runs for each of five aforementioned tasks and both methods (GP and GPPO). Each row shows the average of 10 runs (evolutions), each starting with a different starting point (initial population)[2]. The table presents the average profile of the best solution found, including accuracy of classification on the training (fitness) set and accuracy of classification on an independent test set (1600 cases, 800 images per class)[3]. For comparative purposes, the difference between the average result obtained by GPPO and GP and the false reject probability given by the paired, one-sided t-test are also given. Two last columns show the size of the best solution, defined as the number of GPVIS language terms it was composed of.

Task	*Average accuracy on the training set*				*Average accuracy on an independent test set*				*Average size*	
	GP	*GPPO*	*Diff.*	*t-test*	*GP*	*GPPO*	*Diff.*	*t-test*	*GP*	*GPPO*
(1,7)	.937	.957	**.020**	.11	.912	.944	**.032**	.07	42	50
(2,7)	.920	.929	**.009**	.25	.879	.899	**.020**	.19	38	49
(3,8)	.843	.864	**.021**	.18	.722	.727	**.005**	.44	46	45
(4,9)	.824	.879	**.055**	.01	.704	.793	**.089**	.01	46	50
(8,9)	.874	.902	**.028**	.06	.770	.807	**.037**	.06	44	49

Table 1. The best solutions found in computational experiments (GP- 'plain' genetic programming, GPPO – genetic programming using partial order of solutions).

CONCLUSIONS

First of all, the above experiment gives evidence of the usefulness of GP in solving non-trivial, real-world pattern recognition tasks. Using this metaheuristics together with the GPVIS language, we are able to automatically induce complete (i.e. not requiring an extern classifier), comprehensible, and quite accurate image recognition programs.

[2] However, the experiments are paired, i.e. the corresponding GP and GPPO runs start from the same starting point.

[3] Note that the training (fitness) set and testing set are independent in a very strong sense, as they contain digits written by different people (see [9]).

The main qualitative result obtained in the experiment is that GPPO, i.e. evolutionary search taking into account the partial order of solutions, outperforms the 'plain' GP on average. When using GPPO, there is an increase of accuracy of classification in comparison to GP on both training and testing sets for all five tasks considered in the experiment. The statistical significance of these differences varies; nonetheless the t-test probability of false reject error is relatively small on average (0.125 and 0.155 for the training and testing set, respectively). Moreover, an additional result not shown in the table is that GPPO gave better accuracy in 36 runs on the training set and in 33 runs on the test set (per total number of 50 runs). For all five tasks, the best solution of all ten GPPO runs was not worse than that of GP (on the training set). Note also that these improvements have been obtained with solutions (programs) of very similar size.

The final conclusion of this work is that it seems to be worthwhile to protect the interesting, however sometimes worse w.r.t. the accuracy of classification, solutions from being discarded in the search process by means of an appropriate, incomparability-preserving, pairwise comparison relation. Further work on the topic may concern different aspects of the approach; some of them however are of special importance. In particular, it seems to be interesting to consider the more sophisticated definitions of hypothesis outranking, which should be less sensitive to the classification of particular examples. We expect even better results after introducing that modification.

ACKNOWLEDGEMENTS

The author would like to thank Yann LeCun for making the MNIST database of handwritten digits available to the public. This work was supported from the KBN research grant no. 8T11F 006 19.

REFERENCES

[1] Chan, P.K., Stolfo, S.J. Experiments on multistrategy learning by meta-learning. In: *Proceedings of the Second International Conference on Information and Knowledge Management*, 1993.

[2] Goldberg, D. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading 1989.

[3] Johnson, M.P. *Evolving Visual Routines*. M.Sc. Thesis, Massachusetts Institute of Technology 1995.

[4] Koza, J.R. *Genetic Programming - 2*. MIT Press, Cambridge 1994.

[5] Krawiec, K., Slowinski, R. Learning Discriminating Descriptions from Images. In: *Proc. of VI International Symposium 'Intelligent Information Systems'*, Zakopane 1997, pp. 118-127.

[6] Krawiec, K. *Constructive Induction in Picture-based Decision Support*. Ph.D. dissertation. Institute of Computing Science, Poznan University of Technology, Poznan 2000.

[7] Krawiec, K. Constructive induction in learning of image representation. Research Report RA-006/2000, Institute of Computing Science, Poznan University of Technology 2000.

[8] LeCun, Y., et al., Backpropagation applied to handwritten zip code recognition. *Neural Computation*, (1) 1989, pp. 541-551.

[9] LeCun, Y., et al., Comparison of learning algorithms for handwritten digit recognition. In: Fogelman, F., Gallinari, P. (eds.) *International Conference on Artificial Neural Networks*, Paris 1995, pp. 53-60.

[10] Mitchell, T.M. *Machine learning*, McGraw-Hill 1997.

[11] Poli, R. *Genetic Programming for Image Analysis*, Technical Report CSRP-96-1, The University of Birmingham 1996.

[12] Roy, B. *Wielokryterialne wspomaganie decyzji*. Wydawnictwa Naukowo-Techniczne, Warszawa 1990.

[13] Schaffer, J.D. Multiple objective optimization with vector evaluated genetic algorithms. In: Grefenstette, J.J.(ed.) *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*. Lawrence Erlbaum Associates, Hillsdale 1985.

[14] Van Veldhuizen, D.A. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio 1999.