Pareto-coevolutionary Genetic Programming Classifier

Michal Lemczyk
Dalhousie University
Halifax, NS
B3H 1W5, Canada,
lemczyk@cs.dal.ca

Malcolm Heywood
Dalhousie University
Halifax, NS
B3H 1W5, Canada,
mheywood@cs.dal.ca

ABSTRACT

The conversion and extension of the Incremental Pareto-Coevolution Archive algorithm (IPCA) into the domain of Genetic Programming classifier evolution is presented. In order to accomplish efficiency in regards to classifier evaluation on training data, the coevolutionary aspect of the IPCA algorithm is utilized to simultaneously evolve a subset of the training data that provides distinctions between candidate classifiers. The algorithm is compared in terms of classification "score" (equal weight to detection rate, and 1 - false positive rate), and run-time against a traditional GP classifier using the entirety of the training data for evaluation, and a GP classifier which performs Dynamic Subset Selection. The results indicate that the presented algorithm outperforms the subset selection algorithm in terms of classification score, and outperforms the traditional classifier while requiring roughly $\frac{1}{430}$ of the wall-clock time.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Performance

Keywords

Evolutionary Computation, Genetic Programming, Supervised Learning, Subset Selection, Co-evolution

1. INTRODUCTION

Binary classification problems within the context of a supervised learning paradigm provide the basis for a wide range of application areas under machine learning. In this work we concentrate on how the training process can be made more efficient by evaluating classifier fitness over some adaptive subset of the total training data. To date, the typical approach has been to utilize an active learning algorithm for this purpose, where the Dynamic Subset Selection (DSS) family represents one widely used approach [2],[4]. An alternative approach to the problem, however, would be to formulate the problem as a competition between two populations, one representing the classifiers, the other the data. Progress has recently been made using Genetic Algorithms based on a Pareto foundation of this coevolutionary approach, albeit within the context of player behaviours in gaming environments. In particular, this work is based on the "IPCA" algorithm, where this has been shown to address several potential problems with the competitive coevolutionary approach (relativism, focusing, disengagement, and intransitivity) [1]. The algorithm reported in this work, hereafter denoted the Pareto-coevolutionary GP Classifier (PGPC) is novel in the fact that it extends a Genetic Algorithm "game-playing" context into the domain of GP classification.

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
ACM 1-59593-186-4/06/0007.

1.1 The Pareto-coevolutionary GP Classifier Algorithm

In terms of the GP individuals (learners), a Tree structured representation is employed, whereas individuals in the point population(s) index the training data.

The PGPC algorithm utilizes four populations of individuals: (1) a fixed size learner population which provides the exploratory aspect of the learner evolution. (2) a learner archive which contains the pareto-front of learners, bound by a maximum size value. (3) a fixed size point population (point population << training exemplar count). (4) a point archive which describes the current subset of training points relevant to the learner archive, bound by a maximum size value. The PGPC algorithm consists of the following steps per each generation of evolution:

Generate points in the point population: Mutation is utilized to generate a completely new point population. Furthermore, to ensure class balance within the point population, each half of the population is randomly filled with points belonging to the same class.
Generate learners in the learner population: Using fitness proportionate selection (with fitness being calculated on the point population and archive). Mutation and crossover operators are used to generate offspring.
Compute the set of useful points regarding the learner population and archive: If a newly generated learner is dominated by the learner archive or contains equal values (evaluated over the point archive), and the addition of a new point into the evaluation set provides a distinction such that the generated learner is pareto-equivalent to the archive with no equal values, the point is inserted into the archive.
Compute the set of useful learners regarding the

point population and archive: Any generated learner that is pareto-equivalent to the archive with no equal values (again, evaluated over the point archive) enters the archive. Furthermore, if a generated learner is undefeated, and a generated point defeats it, they both enter their respective archives.
Remove duplicates in the learner and point archives and newly-dominated learners in the learner archive: As per IPCA [1].

To maintain the efficiency of the algorithm, a limit on the archive sizes may have to be placed. For pruning the learner archive, the proposed greedy approach consists of removing the learner with the worst performance against the point archive (with the measure being the number of incorrectly classified instances). For pruning the point archive, the proposed basis utilizes the genotypic information of the point co-ordinates to delete one of the two closest points, distance defined using the Euclidean metric, adhering to the following criteria: the two points must be of the same class, and that class must be over-represented in the point archive. This approach will promote class-balanced diversity in the point archive, while preserving the points which define boundaries between clusters of points.

Finally, in order to attain a measure of classification performance on testing data at the completion of training, the learner pareto-front must be interpreted to provide one class prediction per testing point. To this end an "Average archive value" voting scheme ("AA") is used in which each pareto-front learner provides one vote for their class prediction of the input testing point. The class with the majority of the votes is selected as the system's prediction for the data point.

2. EXPERIMENTS

The following sections describe a set of proof-of concept experiments comparing the classification and run-time performance of PGPC against contemporary algorithms. To attain a measure of classification performance, the detection rate and 1 - false positive rate is averaged to produce a single classification "score". To measure efficiency, the run-time of the algorithms on a common machine under common conditions will be recorded. Due to the stochastic nature of the GP based algorithms, performance is reported over a total of 30 different population initializations per experiment.

The classification data sets used in the experiments consist of: Adult, and KDD99[1]. Each data set is considered a two-class problem, with the training/test partition for Adult being set at 75%, and the partition for KDD99 being specified by the separate training and testing data sets.

The relevant hardware of the test machine utilized for the run-time experiments consists of: Pentium 4, 2.60GHz HT, 800MHz FSB. 1GB DDR400 RAM. 36GB SATA 10K-RPM Hard Drive.

The base line comparison algorithm is a traditional tree structured GP classifier [3] (denoted as "Regular"), utilizing fitness proportionate selection, in conjunction with an absolute switching function wrapper to perform classification of points. The additional (active learning) comparison algorithm; Dynamic Subset selection, uses a limit on the number of training exemplars to provide a more accurate comparison for the PGPC classifier. The implementation utilized was based on the of the work described in [2], with the age-difficulty weighing being set to $Difficulty_p^{1.0} + Age_p^{3.5}$, as to mimic the values utilized in the referenced work.

The parameters common to all of the algorithms include the number of generations evolved at 500, the crossover probability set to 80%, the mutation probability at 20%, and the function set being {*, /, +, -, sin, cos, exp, sqrt}. The sizes of the populations and archives of the PGPC algorithm are all be set to a common value of 25. The comparison algorithms have their population sizes set to the sum of the PGPC population and archive sizes, that is a population size of 50 individuals, and a subset size of 50 exemplars.

Table 1 shows the resultant median score and run-time of the various algorithms on the input sets, with time measured in seconds. In terms of median classification score, the PGPC algorithm out-performed all of the comparison algorithms. This indicates that the proposed algorithm provided a mechanism for attaining a superior subset of training exemplars to perform the learner evolution upon. In terms of execution efficiency with regards to the subset size, the PGPC algorithm exhibited an average speed increase of approximately 430.31 times against the Regular algorithm, with the DSS algorithm exhibiting an average increase of only 320.23 times.

Table 1: Median scores and run-times in seconds of the various algorithms.

Median score	PGPC	Regular	DSS
Adult	0.736611	0.611569	0.526903
KDD99	0.918419	0.909291	0.827497
Median time	**PGPC**	**Regular**	**DSS**
Adult	41.38	1973.74	11.29
KDD99	56.97	40347.74	120.87

3. ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of CFI, NSERC, and MITACS grants during this work.

4. REFERENCES

[1] E. D. de Jong. The incremental pareto-coevolution archive. In *GECCO (1)*, volume 3102 of *Lecture Notes in Computer Science*, pages 525–536. Springer, 2004.
[2] C. Gathercole and P. Ross. Dynamic training subset selection for supervised learning in genetic programming. In *PPSN*, volume 866 of *Lecture Notes in Computer Science*, pages 312–321. Springer, 1994.
[3] J. R. Koza. *Genetic Programming II*. Cambridge, Mass.:MIT Press, 1994.
[4] C. Lasarczyk, P. Dittrich, and W. Banzhaf. Dynamic subset selection based on a fitness case topology. *Evolutionary Computation*, 12(2):223–242, 2004.

[1]Available at:
http://www.ics.uci.edu/∼mlearn/MLSummary.html [Adult]
http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html [KDD99]