Increasing the Throughput of Expensive Evaluations Through a Vector Based Genetic Programming Framework

Jason Zutty, Daniel Long, Gregory Rohling
Electro-Optical Systems Laboratory
Georgia Tech Research Institute
Atlanta, GA 30332
jason.zutty@gtri.gatech.edu
daniel.long@gtri.gatech.edu
greg.rohling@gtri.gatech.edu

ABSTRACT

Traditional genetic programming only supports the use of arithmetic and logical operators on scalar features. The GTMOEP (Georgia Tech Multiple Objective Evolutionary Programming) framework builds upon this by also handling feature vectors, allowing the use of signal processing and machine learning functions as primitives, in addition to the more conventional operators [6]. GTMOEP is a novel method for automated, data-driven algorithm creation, capable of outperforming human derived solutions.

A challenge in this field is working with both large datasets and expensive primitive functions. This paper outlines some of the innovations Zutty et al. have introduced into the GTMOEP framework in order to more efficiently evaluate individuals and tackle new problems. These innovations include: Working with non-feature data, tiered datasets, subtree caches, and initial population creation.

CCS Concepts

•**Computing methodologies** → **Genetic algorithms; Genetic programming;** *Supervised learning by classification;*

Keywords

Genetic Programming, Computationally Expensive Evaluations, Vector Based

1. INTRODUCTION

The Georgia Tach Multiple Objective Evolutionary Programming (GTMOEP) framework introduced a new manner of using strongly typed genetic programming for automatically creating classification or prediction algorithms from feature data. This was achieved by creating a special data object that was passed through a tree structure that paired training and testing data together, and wrapping signal processing and machine learning algorithms to act on these pairs. Using this paradigm, high quality solutions were quickly reached, and human derived solutions were dominated. [6]

In this abstract, we present techniques used to expand the capabilities of GTMOEP to operate on non-feature data, including a new data structure for maintaining separate feature and continuous data sources, tiered datasets and subtree caching for faster computation, and seeding with hierarchy to decrease start up time and increase solution quality.

2. WORKING WITH CONTINUOUS DATA

There are many examples of genetic programming [2, 3, 5] that work with continuous data, however most do so in a linear fashion. First, features are constructed or extracted from continuous data, then they are either fed directly to a learner or a second layer of optimization is done to evolve a learner from these features. GTMOEP presents a shift towards wrapping both steps in to one optimization, where feature extraction and learning can occur in cycles and build from each other.

To support a wider set of application problems, the GTMOEP framework was modified to use a dual representation for maintaining features along with continuous data. When we use the terms continuous data or stream data we are referring to data where not only the contained information is important, but also the order in which it appears.

The representation consists of a list of instances (or observations) each instance contains a feature data object and a stream data object. This requires modifying all signal processing functions to consume a parameter to determine whether they would be acting on the feature data or stream data directly, or be creating features from the stream data and adding them to the feature set. Conversely, machine learning functions in the primitive set operate only on the feature space in this framework.

3. TIERED DATASETS FOR SPEEDING UP EVALUATIONS

Some of the challenges in working with machine learning functions include the memory and processing time required in order to preform classification or prediction. Both resources also increase exponentially with the size of the datasets. It follows then, that evaluating hundreds of thou-

GECCO'16 Companion July 20-24, 2016, Denver, CO, USA
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4323-7/16/07.
DOI: http://dx.doi.org/10.1145/2908961.2931641

sands of individuals is an intensive task for even a powerful cluster of computers. To this end, GTMOEP was designed to increase the size of the datasets used as an individual passes through stages. A program can first be tested with a very small set of data; all the individuals that meet a threshold requirement are analyzed, and the top tier (strong Pareto strength) are then sent on to the next largest data set.

Individuals are graduated from dataset to dataset as follows. Evaluation begins on the smallest dataset. Once fitnesses are returned, a subpopulation N is to be selected for transitioning to the next level. Then, using thresholds, the subpopulation is broken up in to two pieces, specialists (which are below the threshold in one objective) and well rounded individuals (which are below the thresholds in multiple objectives). Each piece contributes a portion of the subplopulation using SPEA2 selection.

This concept follows the idea of extra-genetic information in nature, or in other words, knowledge acquired over the life span of a particular species. This feature of GTMOEP was developed to support the adult data set problem from the UCI Machine Learning repository [1]. This problem presents the challenge of having 48,842 instances of data. Each instance is comprised of fourteen features: age, workclass, fnlwgt, education, education-num, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week, and native-country. With a data set of this size, partitioning to a small and full data set allowed the ability for individuals to fail fast. This means that the most computer processing time was spent on individuals that had already been somewhat proven, rather than on fatal alleles (genotypes that resulted in poor performance). This is similar to the work done by Rohling in his thesis [4]. Rather than thresholding on time consuming objective functions, we are thresholding on time consuming and memory intensive datasets.

In practice, on the adult data set problem, we computed that out of 30,181 individuals 24,294 were not necessary to graduate to the larger datasets. At an average computation time of 258 seconds for an evaluation on the larger data set, this represents a savings of over 1741 hours of computation time. By contrast, on the smallest data set, only one percent of the total number of instances was used, and computation times averaged only 1.35 seconds. This means that our gain of approximately 1741 hours came at the expense of only 2.21 computational hours on the redundant computations required for those 5887 individuals that were graduated.

4. CACHING ML RESULTS

Zutty et al. previously showed that they maintained hashes of individuals [6]. Building on this, we have enabled GTMOEP to also hash results of machine learning training (in effect, subtrees). This is achieved as follows: before training a machine learning algorithm, the full input feature data matrix is hashed along with the learner type and parameters. This hash is used to explain a saved learner state saved on the filesystem. GTMOEP then checks to see if this hash has already been written. If it has, the trained machine learner is read in. Else, after training the machine learner, the results are written out to disk. This hashing method is significantly cheaper than retraining a learner on the same set of features. It is also of benefit because as subtrees are exchanged in mating, branches persist in multiple individuals.

5. STARTING WITH HIERARCHY

In Rohling's thesis he showed the benefits of seeding a genetic programming optimization with a set of topologies [4], rather than beginning randomly. Using this seeding, Rohling was able to converge to better solutions more rapidly. We have developed a similar paradigm for GTMOEP.

By parameterizing the primitive set of functions, we can randomly create hierarchies of feature selections, modifications, aggregations, transforms, and learning. We have had success with this method of trees of depth four and five. However, we have found that the deeper the trees the more challenging this becomes, as there is a higher probability of randomly introducing an invalid operation in the tree, creating a fatal allele.

6. CONCLUSIONS

Adding support for continuous data enables GTMOEP to process new types of prediction and classification problems such as signal and image processing tasks. This is an important capability toward our goal of automating the data scientist, to result in better algorithms and new techniques.

GTMOEP continues to change the paradigm for evolutionary computation where evaluations are significantly more computationally expensive than the processing required to produce a population. As such, the ideas presented result in more throughput of individuals despite any computational cost required for the techniques themselves.

Future work is required on the concepts presented here to more rigorously establish benchmarks and speed ups. Additionally we plan on altering some of these features to make them more robust to large problems, such as implementing caching rules for the hashing of subtrees, or smarter seeding such that with large trees there is still a strong probability of success.

7. REFERENCES

[1] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[2] L. Guo, D. Rivero, J. Dorado, C. R. Munteanu, and A. Pazos. Automatic feature extraction using genetic programming: An application to epileptic eeg classification. *Expert Systems with Applications*, 38(8):10425–10436, 2011.

[3] K. Holladay and K. Robbins. Evolution of signal processing algorithms using vector based genetic programming. In *Digital Signal Processing, 2007 15th International Conference on*, pages 503–506. IEEE, 2007.

[4] G. Rohling. *Multiple objective evolutionary algorithms for independent, computationally expensive objective evaluations*. PhD thesis, Georgia Institute of Technology, 2004.

[5] J. Streater. Genetic programming for the automatic construction of features in skin-lesion image classification. *Master's thesis, University of Edinburgh*, 2010.

[6] J. Zutty, D. Long, H. Adams, G. Bennett, and C. Baxter. Multiple objective vector-based genetic programming using human-derived primitives. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pages 1127–1134. ACM, 2015.