One-Class Learning with Multi-Objective Genetic Programming*

Robert Curry and Malcolm Heywood†

August 22, 2007

Abstract

One-class classification naturally only provides one class of exemplars on which to construct the classification model. In this work, multi-objective genetic programming (GP) allows the one-class learning problem to be decomposed by multiple GP classifiers, each attempting to identify only a subset of the target data to classify. In order for GP to identify appropriate subsets of the one-class data, artificial outclass data is generated in and around the provided inclass data. A local Gaussian wrapper is employed where this reinforces a novelty detection as opposed to a discrimination approach to classification. Furthermore, a hierarchical subset selection strategy is used to deal with the necessarily large number of generated outclass exemplars. The proposed approach is demonstrated on three one-class classification datasets and was found to be competitive with a one-class SVM classifier and a binary SVM classifier.

1 Introduction

One-class classification, where the training dataset contains exemplars from only one class (the target class), has the potential to provide solutions to a range of application domains otherwise inappropriate for classical two class models. For example, in domains such as intrusion detection, medical diagnosis and fault detection data might only be available to support a single class of behavior, the 'cost' of acquiring missing classes appearing too high.

The principle distinction between one-class learners and the more typical binary approach to classification is that the learner must operate as a novelty detector as opposed to a discriminator [10, 11]. In order to ensure such a behavior under the GP paradigm we make use of a local wrapper (Gaussian) rather

*Published in Proceedings of the 2007 IEEE Systems, Man and Cybernetics (SMC) conference, Montreal, Canada, 10/07/2007

†R. Curry and M. Heywood are with the Faculty of Computer Science, Dalhousie University, 6050 University Avenue, Halifax, NS, Canada, B3H-1W5 {rcurry,mheywood}@cs.dal.ca

than a global wrapper (sigmoid). In doing so, a classifier only responds to target exemplar subsets explicitly encountered during training. We then adopt a one-class learning model in which data for the unseen class is created artificially, with the objective of learning a mapping able to maximize the distinction between target and artificial outlier data [17]. Finally, problem decomposition is facilitated, by way of an evolutionary multi-objective optimization (EMOO) approach as developed under different data partitions identified using active learning. This enables us to provide multiple one-class classifiers focusing on different parts of the problem, such that detection rate is maximized, but false positive rates are exceptionally low. The emphasis on minimizing the false positive rate effectively helps to guarantee predictable behavior under multi-class conditions.

Empirical evaluation is conducted against one-class and binary classifiers devised under the SVM paradigm [14, 15]. The proposed approach is able to better the SVM algorithms under two of three datasets considered and provides competitive results under the third. Moreover, equivalent performance is only attained under the easier of the three datasets considered.

2 Related Work

The problem of one-class learning or 'novelty detection' presents a unique set of requirements from that typically encountered in classification. For example, the discriminatory models of classification often utilize a 'global' decision function, such as a sigmoid operator, to distinguish between two classes. As such it is difficult to identify how the model will behave under unseen data. Conversely, the one class learning problem requires that any model explicitly identify when data differs from the target class appearing under training conditions.

Machine learning algorithms employed under the one-class domain therefore need to address the discrimination-detection problem directly. In this work we will concentrate on the kernel or Support Vector Machine (SVM) approach [14, 17, 18] (for a wider survey see [10, 11]). The one-class SVM model of Schölkopf relies on the correct identification of "relaxation parameters" to separate exemplars from the origin (representing the second unseen class) [14]. Unfortunately, the values for such parameters vary as a function of the data set. However, a recent work proposed a kernel autoassociator for one-class classification [18]. In this case the kernel feature space is used to provide the required non-linear encoding, this time in a very high dimensional space (as opposed to the MLP approach to the encoding problem). A linear mapping is then performed to provide the original features as the output. Finally, the work of Tax again uses a kernel based one-class classifier, however, the approach is distinct in that data is artificially generated to aid the identification of the most concise hypersphere describing the in-class data [17]. Such a framework builds on the original support vector data description model, whilst reducing the significance of specific parameter selections. The principle drawback, however, is that tens or even hundreds of thousands of artificially generated data is required to build a

suitably accurate model [17]. The work proposed in this paper uses the artificial data generation model of Tax, but specifically addresses the training overhead by employing an active learning algorithm. Moreover, the GP paradigm provides the opportunity to solve the problem using an explicitly multiple objective model, where this will be shown to aid problem decomposition.

3 Methodology

The proposed multiobjective one-class genetic programming classifier is outlined in Fig. 1. Although the one-class learning algorithm presented here is not restricted to any one form of GP, dynamic page-based linear genetic programming is used as the underlying learner, Section 3.1. In order to train binary GP classifiers, artificial or outlier exemplars are generated in and around the provided target or inclass data, Section 3.2. To ensure that the target class exemplars will be surrounded in all feature directions by artificial outlier exemplars it is necessary to generate a sufficient number of artificial exemplars. The *balanced block* active learning algorithm, Section 3.3, is then utilized in order to scale GP classifiers to deal with the resulting large datasets. Finally, as opposed to having GP produce a single program as the classifier solution, multiple programs are encouraged to decompose the classification problem and participate in the solution by using multiobjective Pareto ranking, Section 3.5.

3.1 Dynamic Page-Based Linear GP

In this work dynamic page-based linear genetic programming (DPLGP) was employed as the underlying learner, where this implies a fixed length representation and a steady state tournament selection operator.

Representation. In linear genetic programming individuals are expressed as linear lists of instructions, executed sequentially, thereby mimicking the process of program execution normally associated with a simple register machine [1]. Instructions are defined in terms of an opcode and operand that modify the contents of general purpose registers $\{R[0], \ldots, R[k]\}$, memory and program counter [1]. For DPLGP in particular, an individual is described in terms of a number of pages, where each page consists of the same number of linear GP instructions.

Genetic Operators. DPLGP utilizes three types of genetic operators namely crossover and two forms of mutation: 'mutation' and 'swap'. Crossover in DPLGP is limited to the exchange of single pages between two parents. The motivation is to make the action of crossover in GP less destructive by making the location of crossover points constant, where this appears to result in concise solutions across a range of benchmark regression and classification problems (see [7]). The first form of mutation is the 'mutation' operator which randomly selects an instruction and replaces it with an alternative instruction (uniform

1. Create artificial outlier data;
2. Partition outlier and target classes (**level 0**);
3. Initialize population of programs;
4. **While** (Level 1 Termination == FALSE)
   (a) Use DSS to select a target and outlier partition to create a block (**level 1**) (combine an outlier and target class partition selected by DSS);
   (b) **While** (Level 2 Termination == FALSE)
      i. **If** (DSSiteration MOD (SubsetFrequency))
      Then Identify training subset from level 1 block using DSS (**level 2**);
      ii. Select tournament for fitness evaluation;
      iii. **While** (program < TournamentSize)
         A. **While** (pattern < NumSubsetPatterns)
         Run program on pattern; record GP_{out};
         B. Partition GP_{out} to create class-consistent regions;
         C. Determine μ, σ^2 and class separation distance for each partition;
         D. Determine best target partition (2);
         E. Assess multi-objective fitness;
      iv. Rank programs by ***Pareto dominance***;
      v. Select parents and apply search operators;
      vi. Update exemplar difficulty and age;
   (c) **While** (program < (Population ∪ Archive))
      i. Run program on block recording GP_{out};
      ii. Partition GP_{out} and determine best partition;
      iii. Determine the best cutoff for error function;
      iv. Assess multi-objective fitness;
   (d) Rank by ***Pareto dominance***;
   (e) Find ***Pareto front***;
   (f) Update partition archive;
   (g) Update partition difficulties and ages;
   (h) SubsetTermination = f(MaxSubsetIteration, PartitionErrors);
5. Post Processing;

Figure 1: **Multiobjective One-Class GP with Balanced Block Algorithm**

probability in both cases). The second form is the 'swap' operator where two instructions from the same individual are randomly selected with uniform probability and their positions exchanged. This provides sequence modification and is motivated by the fact that the order that instructions are executed within a program can have a significant effect on the outcome of that program. For more information on DPLGP see [7].

3.2 Artificial Data Generation

In a conventional binary classification problem the decision boundaries between classes are supported by the exemplars provided from each class. However, in an one-class scenario it is much more difficult to establish appropriate and concise decision boundaries since they are only supported by the target class. The approach we adopt for developing one-class classifiers is therefore to build the outlier class data artificially and train using a two class model with appropriate regularization constraints [17].

Therefore, in order to train the binary GP classifier on the target data, artificial outlier data is generated in and around the target exemplars, Fig. 1, Step 1. In particular, the outlier generation algorithm of Tax and Duin is utilized in this work [17]. Outliers are generated that are uniformly distributed in a unit hypersphere with as many dimensions as there are features in the target exemplars. The unit hypersphere is then shifted and rescaled to fit around the target exemplars. Due to the wider range of possible values for outlier data, and to ensure that the target class exemplars will be surrounded in all feature directions by outlier exemplars, the ensuing training dataset tends to be of the order of $10,000$ to $100,000$ exemplars as per Tax [17].

3.3 Balanced Block Dynamic Subset Selection

Datasets examined in this work contain < 200 target exemplars in their inclass training datasets. Increasing the size of the training datasets through the use of the artificial outlier exemplars has the potential to significantly increase the training overhead. In order to address this overhead an active learning algorithm is utilized that filters the initial training dataset in parallel with the learning process. This algorithm is termed the Balanced Block Algorithm (BBA) and is part of a family of hierarchical subset selection algorithms [5, 16]. The motivation of these algorithms is to focus GP training on the most difficult or least recently visited exemplars through a hierarchy of subset selections that mimic the concept of a memory hierarchy. The use of hierarchical subset selection algorithms was found to provide a training speed up of several orders of magnitude for GP on large datasets without negatively impacting the classification accuracy [5, 16].

The architecture is summarized in terms of three levels by Fig. 2. At Level 0 of the subset selection hierarchy the original dataset is first sorted by class with each class then being divided into partitions (Fig. 1, Step 2). A partition is then selected from each class in proportion to the partition difficulty and age

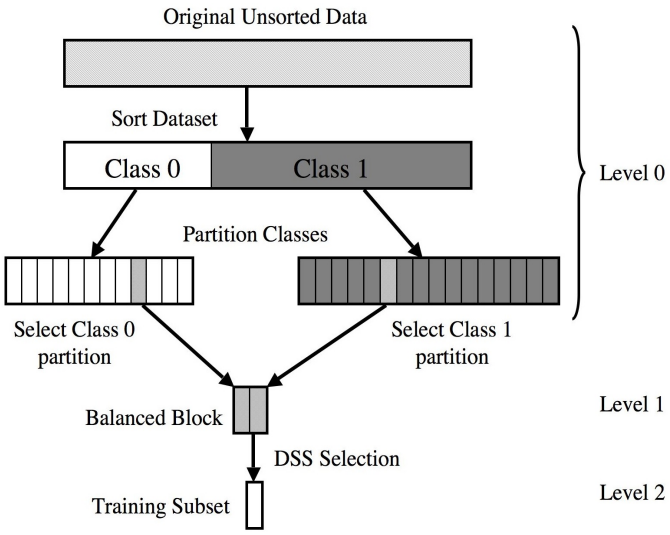

Figure 2: Balanced Block Heirarchical Subset Selection

using the dynamic subset selection (DSS) algorithm of [6]. At Level 1 of Fig. 2 the two selected partitions are combined to form a balanced "block", which is level 1 of the subset selection hierarchy (Fig. 1, Step 4a). Fig. 2, Level 2 of the hierarchy shows DSS again being used, this time to stochastically select exemplars from the Level 1 block biased by exemplar age and difficulty (Fig. 1, Step 4(b)i). The Level 2 "subset" created by these exemplars forms the training dataset for the current tournament of GP individuals. Multiple tournaments are run over a single Level 2 subset selection, and multiple Level 2 subset selections are conducted for each Level 1 block selection. The use of BBA ensures that every level 1 block consists of a balanced set of exemplars, independent of the initial exemplar data distribution [5].

BBA has been modified in this work due to the small number of target class exemplars. This means there can only be a few small target class partitions for forming level one blocks. Moreover, due to the importance of learning from the target class exemplars and finding concise boundaries around the target class the appearance of a sufficient number of target class exemplars is desirable in every Level 2 training subset. For this reason the small Level 1 target class partition is copied directly into the level two subset while only outlier class exemplars are sampled by the DSS algorithm.

In addition to speeding up training time, BBA also has the ability to determine which outlier exemplars are more difficult to classify. In this way and without prior knowledge of what a good outlier exemplar would be, BBA can focus GP training to outlier exemplars that are potentially more relevant to learning the concise boundaries of the target class.

3.4 Class-Consistent Partition Selection

For binary classification datasets GP fitness functions typically utilize a binary switching function as popularized by Koza [8]. The binary switching function maps the one-dimensional 'raw' GP output, or GP_{out}, to one of two classes, as in (1), where y is the label the GP program assigns to the exemplar:

$$y = \begin{cases} 0 & \text{if } GP_{out} \leq 0. \\ 1 & \text{otherwise.} \end{cases} \tag{1}$$

The GP label y is compared to the actual label provided with the training exemplar to determine the error. However, under a one-class model of learning, exemplars belonging to the unseen classes are just as likely to appear on either side of the GP_{out} origin, resulting in high false positive rates. In order to encourage GP to act as a novelty detector the GP_{out} values associated with the single class encountered during training must appear over a small neighborhood of the GP_{out} axis.

To this end, a fitness function is designed whose goal is to encourage a concise clustering of target exemplars on the GP_{out} axis. In Fig. 1, Step 4(b)iiiA, each GP tournament program is run on the exemplars from the current level 2 training subset identified by the Balanced Block Algorithm (Step 4(b)i). Each program participating in a tournament has now mapped the subset of multi-dimensional training exemplars onto its one-dimensional GP_{out} space. The GP_{out} of each program is then partitioned into *class consistent regions* of sequential target and outlier exemplars (Step 4(b)iiiB). The "*cluster separation distance*" (CSD) [2] is then used to measure the ability of a GP program to distinguish between two neighboring (class consistent) regions and is calculated for each neighboring partition. CSD is estimated from the distance between partition means (μ) normalized by the partition variances (σ^2):

$$CSD_{0/1} = \frac{|\mu_0 - \mu_1|}{\sqrt{\sigma_0^2 + \sigma_1^2}} \tag{2}$$

The 'best' target partition is then identified as the partition that gives the best separability, i.e., maximizes (2), relative to its neighboring partitions representing artificial or outlier data, (Step 4(b)iiiC). For each program participating in a tournament the partition selection heuristic of Fig. 3 selects the best target partition that maximizes the CSD to the nearest pair of outlier partitions:

3.5 Tournament Multi-Objective Fitness and Pareto Ranking

At this point the response of each individual is expressed in terms of a target class consistent partition on the GP_{out} axis. We now need to compare the programs to determine the tournament winners. The quality of the programs is determined for each tournament by way of a multi-objective fitness evaluation (Fig. 1, Step 4(b)iiiE) and the winners are determined by *Pareto ranking* (Fig.

1. Identify the set S of target partitions with size $N > 5\%$ of the target exemplars in the current training subset;
2. **If** $(S = \emptyset)$
   Then $(S = \{\text{target partition with max } N\})$;
3. $\forall$ partitions $s_i \in S$:
   (a) Calculate the *CSD* with respect to the immediate neighboring outlier partitions;
   (b) Choose the min(*CSD*) between the neighboring partitions as the performance of s_i;
4. The partition $s \in S$ with max(*CSD*) is then chosen as the best partition for the current tournament individual;

Figure 3: Partition Selection Heuristic

1, Step 4(b)iv). Four objectives have been identified for optimization in order to encourage GP programs to find 'good' inclass partitions.

1. **Minimize Overlap**: Minimize the number of target patterns that are already being classified by other programs in order to encourage diversity (i.e., discourage GP programs from finding partitions that overlap inclass exemplars)
2. **Maximize Count**: Find partitions that maximize the count of target exemplars (i.e., encourages GP programs to densely map inclass exemplars to regions on the GP_{out} space).
3. **Maximize CSD**: Find partitions that maximize CSD, (2), to the neighboring outlier partitions (i.e., encourages GP programs to form 'robust' boundaries around target partitions).
4. **Minimize Solution Size**: Programs should minimize their solution size (# of instructions) in order to avoid overfitting on the current training subset.

Each individual in a GP tournament now has an associated four dimensional objective vector. An individual A is said to *dominate* another individual B if it performs at least as well as B in all objectives and better than B in at least one objective. Pareto ranking is then used to combine the objectives into a scalar fitness without combining the objectives in any way [9, 19]. Instead, each tournament individual is given a *rank* based on the number of individuals by which it is dominated (Fig. 1, Step 4(b)iv). The two individuals with the

lowest Pareto ranks are chosen as the tournament winners. The winners become parents and have search operators applied to them (Fig. 1, Step 4(b)iv) forming children who replace the tournament losers in the GP population.

The parent programs also update the difficulty ([6, 5, 16]) of target exemplars in their 'best' partition as well as the difficulty of exemplars in the neighboring outlier partitions (Fig. 1, Step 4(b)vi).

$$exemplar_{diff} = \frac{1}{1 + (CSD_{max})} \tag{3}$$

Exemplar ages are also updated ([6, 5, 16]), with exemplars chosen to be in the subset having their ages set to zero, while exemplars not appearing in the subset increment their age.

3.6 Block Multi-Objective Fitness and Pareto Ranking

Once training has been completed on a Level 1 block (i.e., the Level 2 termination criteria has been met) the age, difficulty and error rate of the target and outlier partitions making up the Level 1 block need to be updated [5]. A partition's age and difficulty are updated to influence future partition selections in the creation of Level 1 blocks (Fig. 1, Step 4g) while partition error rates are needed to update the Level 2 termination criteria (Fig. 1, Step 4b). To determine these values the entire population of programs is evaluated over the Level 1 block (Fig. 1, Step 4c). In addition to the population of programs, an archive of previously stored programs is evaluated on the same Level 1 block. Archives are maintained for each target partition in an effort to facilitate problem decomposition under a boosting model of classification. That is to say, classifiers with similar levels of pareto performance are identified under different partitions of the training data.

As during a tournament, the GP_{out} for each program is stored (Fig. 1, Step 4(c)i) and sorted into class-consistent partitions. The partition selection heuristic of Fig. 3 is then applied (Fig. 1, Step 4(c)ii). Now that a partition has been selected the error rate is determined by a *local membership function* based on a partition's mean μ and variance σ^2 (see Fig. 4 [12]).

For each exemplar i the $GP_{out}(i)$ value returned by the program is entered into the local membership function, (4), and the value returned provides a confidence measure on whether the associated exemplar is labeled as a target or an outlier by the current program.

$$confidence_i = \exp\left(-\frac{(GP_{out}(i) - \mu)^2}{2 \cdot \sigma^2}\right) \tag{4}$$

A *Cutoff* is used to classify the exemplars, where all confidence values greater than the *Cutoff* are labeled target and all confidence values below the *Cutoff* are labeled outlier (dashed line in Fig. 4), as in Equation 5.

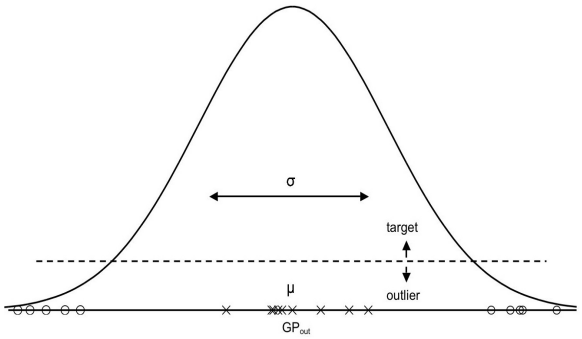

Figure 4: GP local membership function

$$\begin{array}{ll} \text{IF } (\mathit{confidence}_i > \mathit{Cutoff}) & \\ \quad \text{THEN} & GP_{label}(i) = \mathit{target}; \\ \quad \text{ELSE} & GP_{label}(i) = \mathit{outlier}; \end{array} \tag{5}$$

In order to optimize the classifier, different *Cutoff* values are evaluated over the training data. Increasing the *Cutoff* value from zero to one describes an ROC curve, where the knee in the curve is used to define the optimal operating condition (Fig. 1, Step 4(c)iii).

Now that a local membership function and a *Cutoff* value have been found for each program, the programs can be compared in order to decide which will be allowed to participate in the classification of the block. Again the quality of the programs is determined through the use of *Pareto objectives* (Fig. 1, Step 4(c)iv), however the 'maximize CSD' objective from the tournament based evaluation has been replaced by the block objective of minimizing the false positive rate (FPR). This is due to the use of the *Cutoff*, (5), in the local membership function which has the potential to allow outlier exemplars to fall within the GP program's identified target partition. The four objectives used to determine the best programs over the block are now:

1. **Minimize Overlap**.
2. **Maximize Count**.
3. **Minimize Solution Size**.
4. **Minimize FPR**: Find partitions that minimize the count of outlier exemplars labeled incorrectly.

Each program now has an associated four dimensional multi-objective vector and the programs are ranked according to dominance (Fig. 1, Step 4d). A program is said to be *non-dominated* if it is not dominated by any other program

in the population and has a rank of zero. The set of all non-dominated programs is referred to as the *Pareto front* (Fig. 1, Step 4e) and only these programs participate in the classification of the block [19, 9].

The Pareto front is then stored in the current target partition's archive (Fig. 1, Step 4f). If the Pareto front is smaller than the size of the archive the remainder of the archive is filled with next least Pareto-ranked programs. If the Pareto front is larger than the archive size then for each program the distance to the nearest neighboring program is found where distance is measured in terms of similarity of Pareto objective vectors. The programs are then ranked by this nearest neighbor distance and those with the greatest distance, or least similarity, are chosen for the archive.

The archive programs are then evaluated on the exemplars from the block, recording each correctly classified target exemplar and incorrectly classified outlier exemplar (only one count per exemplar). Partition difficulties are updated as the error rate found on each partition as in Equation 6 (Fig. 1, Step 4g).

$$\begin{aligned} \text{target}_{error} &= 1 - \sum \frac{\text{correct target exemplars}}{\text{target partition size}} \\ \text{outlier}_{error} &= \sum \frac{\text{incorrect outlier exemplars}}{\text{outlier partition size}} \end{aligned} \tag{6}$$

The age of the partitions making up the Level 1 block are set to 0 while all other partitions have their age incremented. The error rates of each partition are also stored in order to determine the number of subset iterations to perform the next time either partition is chosen to be in the Level 1 block (Fig. 1, Step 4h).

3.7 Post Processing

Once the Level 1 termination criteria has been met (Fig. 1, Step 4) all archive programs are run on the entire artificial training dataset. For each program the best *Cutoff* is found which determines its overall detection rate (DR) and false positive rate (FPR). A final filter is applied to all archive programs in order to remove any programs with a FPR greater than a specified threshold. Furthermore, any duplicated programs found across the archives are removed.

Finally, the remaining archive programs are run again on the entire artifical training dataset and the actual test dataset to determine classification accuracy. If any program determines an exemplar has a confidence value greater than it's *Cutoff* than that exemplar receives a vote to be labeled as a target exemplar. The number of votes necessary for an exemplar to be labeled as a target is varied, from requiring only 1 vote up to the maximum number of votes possible before the false positive rate becomes 0. The exemplars labeled as target by the GP programs are compared to the actual labels provided with the datasets to determine the final detection rates, false positive rates and classification accuracy.

Table 1: Binary classification datasets

	Breast		Liver		C-heart	
Features	9		6		13	
Class	Art.	Test	Art.	Test	Art.	Test
0	10,000	114	10,000	50	100,000	41
1	181	60	109	36	105	34
Total	10,181	174	10,109	86	100,105	75

4 Experiments

4.1 Experimental Setup

The one-class multiobjective genetic programming algorithm was tested on the Breast, Liver and C-heart datasets from the UCI machine learning repository [13]. Each dataset was first divided into a 75% training dataset and a 25% test dataset while maintaining the class distribution of the original dataset. The class 0 exemplars are removed from the training dataset to form the one-class target dataset. Then, the artificial outlier exemplars are generated around the target training data. Table 1 lists the number of features in each dataset, the number of target exemplars provided in the training dataset, the total number of artificial outlier exemplars generated and the number of target and outlier exemplars in the test dataset. For the Breast and Liver datasets $10,000$ artificial exemplars are generated. Experiments on the C-heart dataset with only $10,000$ exemplars resulted in frequent degenerate solutions in which all exemplars were labeled as outliers. The cause was attributed to the increase in the dimensionality of the feature space for C-heart, from 6 and 9 features for Liver and Breast to 13 features for C-heart. Therefore $100,000$ artificial exemplars were generated to deal with this wider range of possible values.

Training was performed on a dual G4 1.33 GHz Mac Server with 2 GB RAM. All experiments are based on 50 GP runs where runs differ only in their choice of random seeds for initializing the population while all other parameters remain unchanged. Table 2 lists the common parameter settings for all runs.

The performance of the final GP programs are reported in terms of first, second (median) and third quartiles for training time, the number of participants in a GP solution and test accuracy. The votes for each final GP program on the test exemplars were also recorded so that the detection rate (DR) and false positive rate (FPR) could be determined based on different voting schemes. Detection rate and false positive rate are estimated as in Equation 7.

$$\begin{aligned} \text{DR} &= \left(\frac{\text{\# of True Positives}}{\text{Total \# of Positives}} \right) \\ \text{FPR} &= \left(\frac{\text{\# of False Positives}}{\text{Total \# of Negatives}} \right) \end{aligned} \tag{7}$$

Table 2: Parameter Settings

Page Based Linear GP	
Parameter	Setting
Population size	125
Maximum # of pages	32
Page size	8 instructions
Maximum page size	8 instructions
Prob. Crossover, Mutation, Swap	0.9, 0.5, 0.9
Tournament size	4
Number of registers	8
Instr. prob. type 1, 2 or 3	0/5, 4/5, 1/5
Function set	$\{+, -, \times, \div\}$
Terminal set	{# of exemplar features}
Balanced Block Algorithm Parameters	
Target Partition Size	$\approx$ NumPatterns / NumArchives
Outlier Partition Size	500
Max block selections	2000
Max subset iterations	5
Tournaments per subset	6
Level 2 subset size	100
Archive Parameters	
Number of Archives	Liver = 2, C-heart = 3, Breast = 4
Archive Size	10

Table 3: ν min, max and step size for ν-SVM

Dataset	ν_{min}	ν_{max}	ν_{step}
Breast	0	0.035556	0.00250
Liver	0	0.021565	0.00250
C-Heart	0	0.002098	0.00025

4.2 Support Vector Machines

Support vector machines (SVMs) were used to compare to the one-class GP results. In particular the ν-SVM algorithm [15] and the one-class ν-SVM [14] variant as implemented in LibSVM [3] were used. The binary ν-SVM algorithm uses the entire artificial dataset for training while the one-class ν-SVM algorithm uses the target class data only. For both algorithms the radial basis kernel was used, Equation (8), with kernel parameter set to the LibSVM default $\gamma = \frac{1}{k}$ where k is the number of features in a training exemplar.

$$K(x, y) = \exp^{-\gamma \|x-y\|^2} \tag{8}$$

The ν parameter for the standard ν-SVM was varied in the interval $[\nu_{min}, \nu_{max}]$ where $\nu_{min} = 0$ and ν_{max} is determined by Equation 9:

$$\nu_{max} = 2 \ \frac{min(m_+, m_-)}{m} \tag{9}$$

where m is the total number of exemplars in the training dataset and m_+ and m_- are the total number of positive and negative exemplars respectively [4]. Table 3 lists the ν values for ν-SVM on each dataset where ν_{step} is the step size used to vary ν between ν_{min} and ν_{max}.

The ν parameter for the one-class ν-SVM was varied through the interval $[0.1, 0.9]$ with $\nu_{step} = 0.1$.

4.3 Results

Table 4 lists the GP results in terms of 1st, median and 3rd quartiles over all runs in terms of run time in minutes, the number of programs participating in the GP solution (# of Participants) and the percent accuracy on the test dataset. The median run time for each GP run over all datasets was less than 5 minutes. In terms of classifier participants, on the Liver dataset there can be up to 20 participants in 2 archives, the C-heart can have up to 30 participants in 3 archives and the Breast dataset can have up to 40 participants in 4 archives (Table 2). Table 4 shows that on the Liver dataset the GP algorithm used a median of 100% participation (20 participants out of a possible 20) in the final solution, while on the Breast and C-heart datasets the GP algorithm used a median of only 43.8% (17.5 out of 40) and 70.0% (21 out of 30). Having a higher rate of participants on the Liver dataset, which consists of $10,000$ artificial exemplars, than on the Breast dataset ($10,000$) or on the C-heart

Table 4: One-class GP Results

	Time (m)	# of Participants	Acc. (%)
	Breast		
Q1	1.63	13.75 (34.4%)	67.39
Med.	1.74	17.50 (43.8%)	70.11
Q3	1.88	20.25 (50.6%)	74.57
	Liver		
Q1	1.63	19.00 (95.0%)	63.08
Med.	1.74	20.00 (100%)	66.28
Q3	1.90	20.00 (100%)	68.31
	C-Heart		
Q1	1.75	18.75 (62.5%)	55.00
Med.	3.02	21.00 (70.0%)	57.33
Q3	4.87	23.00 (76.7%)	63.67

Table 5: Best Case Accuracy

	OC GP	OC ν-SVM	ν-SVM
Breast	90.80	95.98	85.06
Liver	72.09	65.12	56.98
C-heart	77.33	66.67	68.00

dataset $(100,000)$ suggests that the rate of participation is related more to the difficultly of the dataset and less on the size of the training dataset.

Table 5 lists the best case accuracies for the one-class GP (OC GP), the one-class ν-SVM (OC ν-SVM) and the standard ν-SVM. The one-class GP had the highest best case accuracy found on the more difficult Liver and C-heart datasets while achieving a higher accuracy than the standard ν-SVM on the Breast dataset. The one-class ν-SVM had the highest best case accuracy on Breast. Moreover, the median GP accuracy on the Liver dataset (Table 4) was higher than the best case accuracy of both the one-class and the standard ν-SVM.

Fig. 5, Fig. 6 and Fig. 7 plot the detection rate and false positive rates of the one-class GP's and the one-class and standard ν-SVM's on the Breast, Liver and C-heart datasets respectively. The one-class GP results differ depending on the number of votes required for an exemplar to be labeled a target, starting from one vote up to the number of votes that results in no exemplars being labeled target. Increasing the number of votes required to label an exemplar as a target requires a consensus from the GP programs and gives a more confident prediction.

A point in these figures represents a solution and a solution can be said to dominate another solution if it is at least as good in all objectives (DR and FPR) and better in at least one (DR or FPR). In other words, a solution is dominated

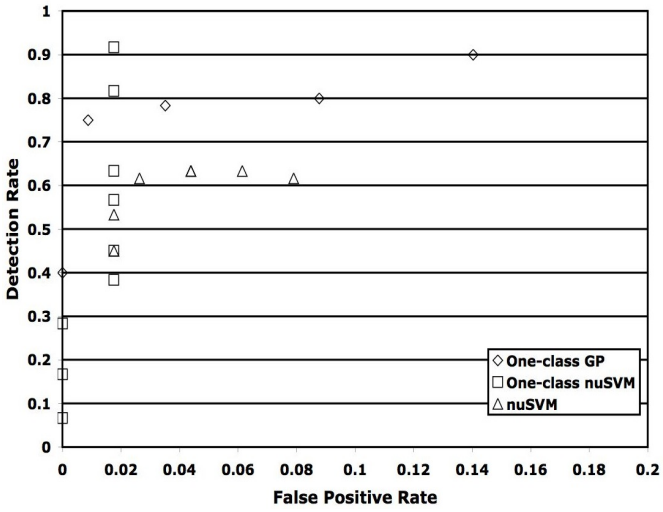

Figure 5: Breast Results

by any solutions that are in the box created by the solution in question and the upper left corner (the optimum with a DR of 100% and an FPR of 0%). Only non-dominated GP solutions are shown in these figures.

Fig. 5 shows that for the case of having a false positive rate of zero the one-class GP was able to find higher detection rates than both SVM's, with the best GP having a DR of 40% and FPR of 0%. Furthermore, the one-class GP solution with a DR of 75% and FPR of $\approx 1\%$ dominates the remaining one-class ν-SVM solutions with the exception of the two one-class ν-SVM solutions with DR $> 80\%$. The one-class GP also managed to find at least one solution that dominates each of the standard ν-SVM solutions.

Fig. 6 shows that neither SVM algorithm was able to find a solution with FPR of 0% while the one-class GP found several, with the best being a DR of $\approx 30\%$ and FPR of 0%. Moreover, the one-class GP was able to dominate all of the one-class and standard ν-SVM solutions.

Fig. 7 shows that while having a FPR of 0% the one-class ν-SVM and the one-class GP were both able to find the best DR of $\approx 24\%$. For FPR $> 0\%$ the one-class GP was able to find solutions that dominate the rest of the one-class ν-SVM solutions and all of the standard ν-SVM solutions.

5 Conclusions and Future Works

This work proposed a methodology for applying GP to the problem of one-class learning comprising of four components. (1) Artificial outclass (outlier) data was generated in and around the provided inclass (target) data to emphasize the limit of inclass (target) data. (2) A local Gaussian wrapper then allowed

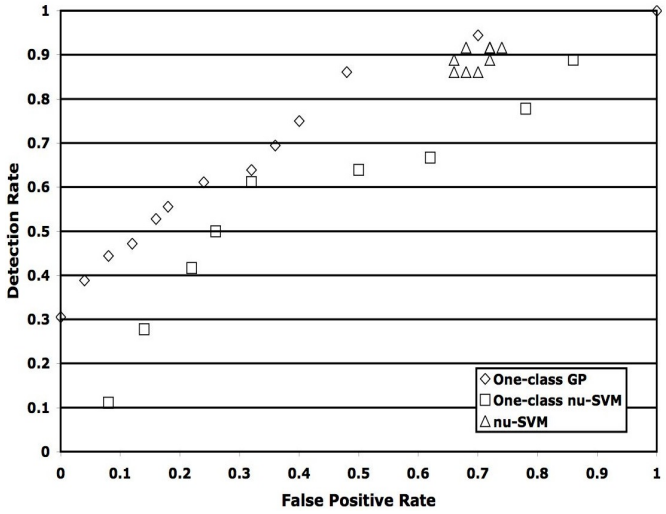

Figure 6: Liver Results

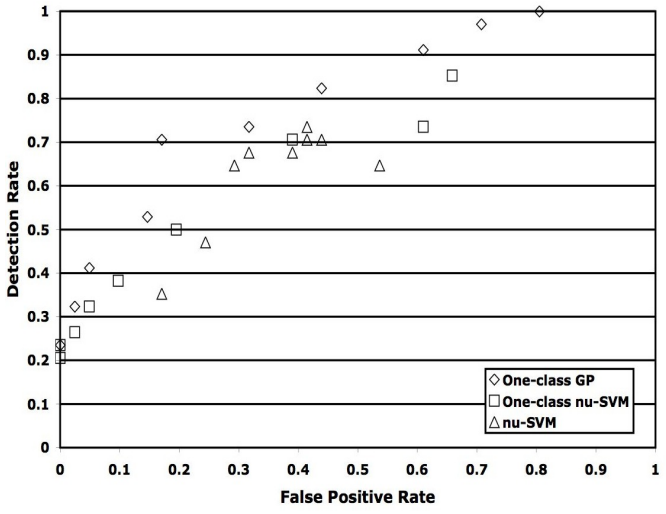

Figure 7: C-heart Results

GP to concentrate on classifying only a subset of the target data, as opposed to using a global wrapper to classify the entire target class. (3) Evolutionary multi-objective optimization then allowed GP to decompose the problem by simultaneously developing multiple classifiers that each identify their own target exemplar subsets while also driving the GP programs to search for improved target subsets. Moreover, the ensuing problem decomposition enables a voting scheme to be employed whereby increasing the number of votes required to label an exemplar as target increases the confidence in the prediction reducing the likelihood of false positives. (4) An active learning algorithm provides an effective means for training GP over the large unbalanced data sets that result from the artificial outclass data.

After testing on three classification datasets the GP solutions were found to be competitive with an one-class SVM trained on inclass exemplars alone and a standard SVM trained on the generated artificial dataset. In particular, against the standard SVM the one-class GP was able to find a better best case solution on all datasets. Moreover, at a false positive rate of 0% the one-class GP had the highest detection rate on all three datasets compared with both the one-class and the standard SVM. The power of the GP voting scheme for classifying target exemplars was demonstrated by the ability of GP to find a variety of solutions over varying levels of false positive rate. This would allow the user to decide on an acceptable level of FPR in choosing a GP solution.

6 Acknowledgments

The authors gratefullly acknowledge the support of a Walter Sumner Fellowship, NSERC Discovery, MITACS, and CFI New Opportunities programs; and industrial funding through the Telecom Applications Research Alliance (Canada), and SwissCom Innovations AG (Switzerland).

References

[1] M. Brameier and W. Banzhaf. *Linear Genetic Programming Genetic and Evolutionary Computation Series*. Springer Verlag, 2007.

[2] K. R. Castleman. *Digital Image Processing*. Prentice Hall, Englewood Cliffs, NJ, USA, 1996.

[3] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*. http://www.csie.ntu.edu.tw/∼cjlin/libsvm, 2001.

[4] C.-C. Chang, C.-J. Lin, and B. Scholkopf. A tutorial on nu-support vector machines. *Applied Stochastic Models in Business and Industry*, 21:111–136, 2005.

[5] R. M. Curry, P. Lichodzijewski, and M. I. Heywood. Scaling genetic programming to large datasets using hierarchical dynamic subset selection.

IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, August 2007.

[6] C. Gathercole and P. Ross. Dynamic training subset selection for supervised learning in genetic programming. In *Parallel Problem Solving from Nature III*, volume 866 of *Lecture Notes in Computer Science*, pages 312–321. Springer Verlag, 1994.

[7] M. I. Heywood and A. N. Zincir-Heywood. Dynamic page-based linear genetic programming. *IEEE Transactions on Systems, Cybernetics and Man - Part B: Cybernetics*, 32(3):380–388, June 2002.

[8] J. R. Koza. Genetic programming: On the programming of computers by means of natural selection. *Statistics and Computing*, 4(2), June 1994.

[9] R. Kumar and P. Rockett. Improved sampling of the pareto-front in multiobjecctive genetic optimizations by steady-state evolution: A pareto converging genetic algorithm. *Evolutionary Computation*, 10(3):283–314, 2002.

[10] M. Markou and S. Singh. Novelty deteciton: a review – part 1: statistical approaches. *Signal Processing*, 83:2481–2497, 2003.

[11] M. Markou and S. Singh. Novelty deteciton: a review – part 2: neural network based approaches. *Signal Processing*, 83:2499–2521, 2003.

[12] A. McIntyre and M. I. Heywood. MOGE: GP classification problem decomposition using multi-objective optimization. In ACM, editor, *Proceedings of Genetic and Evolutionary Computation Conference*, 2006.

[13] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases.

[14] B. Scholkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

[15] B. Scholkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

[16] D. Song, M. I. Heywood, and A. N. Zincir-Heywood. Training genetic programming on half a million patterns: An example from anolmaly detection. *IEEE Transactions on Evolutionary Computation*, 9(3):225–239, 2005.

[17] D. M. J. Tax and R. P. W. Duin. Uniform object generation for optimizing one-class classifiers. *Journal of Machine Learning Research 2*, pages 155–173, December 2001.

[18] H. Zhang, W. Huang, Z. Huang, and B. Zhang. A kernel autoassociator approach to pattern classification. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 35(3):593–606, June 2005.

[19] E. Zitzler and T. Thiele. Multiobjective evolutionary algorithms: A comparitive case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.