Pareto Cooperative-Competitive Genetic Programming: A classification benchmarking study*

Andrew R. McIntyre and Malcolm I. Heywood†

December 24, 2008

Abstract

A model for problem decomposition in Genetic Programming based classification is proposed consisting of four basic components: competitive coevolution, local Gaussian wrapper operators, evolutionary multi-objective (EMO) fitness evaluation, and an explicitly cooperative objective. The framework specifically emphasizes the relations between different components of the model. Thus, both the local wrapper operator and cooperative objective components work together to establish exemplar subsets against which performance is evaluated and the decomposition of the problem domain is achieved. Moreover, the cost of estimating fitness over multiple objectives is mitigated by the ability to associate specific subsets of exemplars with each classifier. The competitive coevolutionary model, in combination with a balanced sampling heuristic guides the identification of relevant exemplars and retention of the best classifiers; whereas the EMO component drives the application of selection and variation operators to maintain the diversity of the classifier population. Empirical evaluation is conducted over twelve data sets representative of different application domains, class distributions and feature counts. A benchmarking methodology is developed in order to provide comparison between the proposed model and the deterministic classifier paradigm in general, and the SVM method in particular. Moreover, by adopting performance metrics for classification, model complexity and training scalability, we are able to provide a more informative assessment of the relative merits of each model.

*Extended version of paper published in Genetic Programming Theory and Practice - VI: 2008, R. Riolo *et al.* (eds) Genetic and Evolutionary Computation Series – Copyright 2008 Springer-Verlag

†Faculty of Computer Science, Dalhousie University. Halifax. NS. Canada.

1 Introduction

Classification represents one of the most widely studied domains of machine learning. The current state-of-the-art is generally considered to take the form of the Support Vector Machine (SVM), where such models are both very accurate and effective across large data sets or those having unbalanced class representations [11]. Conversely, canonical Genetic Programming (GP) does not scale well as the number of training exemplars or classes increases, and has no implicit resilience to unbalanced representation of exemplar classes. Naturally, the class imbalance issue may be addressed by assuming problem dependent cost functions, whereas various heuristics have been proposed to address multi-class classification and large data sets (including hardware specific solutions). However, the only implicit advantage that canonical GP is able to maintain relative to the generic SVM model is a bias towards indexing subsets of features, thus providing the potential for more transparent solutions than the 'black box' generally associated with SVM models.

On the other hand, one of the strengths of the GP paradigm is the freedom with which the basic machine learning design questions – cost function, representation, and credit assignment – may be addressed. In this work, we are interested in using this freedom to provide a framework for encouraging GP to actually decompose the problem domain as part of the evolutionary cycle. Thus, solutions might take the form of multiple intra-class mappings per class, each individual associated with the solution responding to a non-overlapping subset of class consistent exemplars. Individuals are therefore explicitly rewarded for correct but also unique behaviors. Moreover, the resulting model should also be scalable. That is to say, tens or hundreds of thousands of training exemplars should not result in recourse to hardware specific solutions. The resulting framework is based on a canonical GP kernel, in this case Grammatical Evolution [20], but makes extensive use of Pareto coevolution and a mechanism for evolving the properties of a 'local' Gaussian wrapper operator. Key characteristics of the resulting Cooperative Multi-objective GE framework – hereafter denoted CMGE – are summarized in Section 2.

In order to evaluate the proposed CMGE framework, we are interested in performing an assessment under multiple criteria: classification, model simplicity and scalability, Section 3. By doing so we demonstrate that the CMGE framework is able to provide competitive classification performance, while also returning parsimonious solutions and scaling gracefully to large data sets, Section 4. Related work on evolutionary methods and problem decomposition are summarized in Section 5. Overall conclusions in which some opportunities for future work are identified in Section 6.

2 A Coevolutionary Multi-objective Framework for Problem Decomposition

In order to develop the proposed approach to GP classification we will introduce the framework in terms of the high level pseudo code listing provided in Figure 1. This establishes the proposed framework in terms of a five step algorithm having two basic components: cooperative coevolution for fitness evaluation and problem decomposition (step 2(c) and 2(e), Figure 1) and a competitive coevolution archiving for training exemplar subset selection and model building (step 2(d), Figure 1).

An interaction between a GP individual and a single exemplar is defined by executing the GP program using the exemplar features as arguments. The result of executing a single program on an exemplar (the interaction) is a real-valued output; mapping the exemplar (from a typically multi-dimensional feature domain) to a one dimensional number line that we refer to as the 'GP Output' space, or simply *gpOut*. Naturally, the mapping process is repeated to map n exemplars to n points on *gpOut*. Such a mapping alone naturally conveys no class information. To do so, canonical GP employs a global membership function (wrapper operator) to impart class labels on the distribution of points in *gpOut*. Such an approach effectively casts all points on the *gpOut* axis that are greater than (less than) zero as in-class (out-class) exemplars, and weights the number of misclassifications by way of a cost function. Such a model implicitly assumes that partitioning the class labels about the origin of the *gpOut* axis is appropriate for the problem domain in question. Under the CMGE model, label assignment is realized by means of a local membership function (LMF) such that class consistent subsets of exemplars should be mapped to the same local neighborhood on the *gpOut* axis.

The basic features of the Competitive Multi-objective multi-objective Grammatical Evolution (CMGE) classifier are now summarized as follows relative to the pseudo code listing provided in Figure 1:

Feature 1: A class-wise random balanced sampling heuristic is enforced such that each class has equal representation in the point population, step 2(a). This policy is assumed as individuals (in the point population) represent exemplar indexes. Such a representation means that there is no structure on which to build search operators, whereas there is evidence that a balanced sampling heuristic provides a bias towards optimizing the AUC performance metric [24];

Feature 2: The local membership function is derived from the distribution of points on the *gpOut* axis, step 2(c).i to iv. As such, no attempt is made to incorporate the concept of class labels when deriving the characterization of the individual's local membership function. Instead, we assume that the local membership function is expressed by a Gaussian, the parameters of which are derived by first applying a clustering routine to the individual's point distribution on the *gpOut* axis. Having identified the subset of points associated with the most dense cluster, the mean and variance of the Gaussian local membership function for that particular individual are established. Only the subset of

```
1. Initialize Learner Population ($LP$);
2. WHILE ! (Stop criteria)
    (a) Point Population ($PP$) := random balance sample of training partition;
    (b) Training Subset ($TS$) := $PP$ concatenated with Point Archive contents ($PA$);
    (c) FOR $j$ := 1 to sizeof($C$)
         i. Establish phenotype of individual $I[j]$;
        ii. Map $TS$ to 1-$d$ number line '$gpOut$' of $I[j]$;
       iii. Cluster $gpOut$ of $I[j]$;
        iv. Parameterize Gaussian Local Membership Function (LMF) of $I[j]$;
         v. Evaluate $I[j]$ with respect to:
            SSE, Overlap wrt. Learner Archive ($LA$), Parsimony.
        vi. Rank $I[j]$ with respect to $LP$ and assign fitness;
       vii. Replacement (insert $I[j]$ into $LP$);
    (d) Archive $PP$, $LP$ members based on outcomes (according to IPCA)
         i. Points in $PP$ enter $PA$ if they provide a distinction;
        ii. Learners in $LP$ enter $LA$ if they ar non-dominated wrt. $LA$;
    (e) Evaluate Stop Criteria (method of consecutive Rank Histogram assessment);
```

Figure 1: CMGE Algorithm. *TS*, training subset; *LP*, learner pop.; *PP*, point pop.; *LA*, learner, archive; *PA* point archive; $I[j]$, classifier 'j'.

points within the cluster as evaluated on a specific individual's *gpOut* axis are associated with the local membership function. This is the first property by which we establish problem decomposition;

Feature 3: At this point we have a set of individuals with their corresponding local membership functions and therefore possibly unique subsets of exemplars established. The class label associated with the individual is established by assuming that the individual takes the label of the exemplar with maximum membership of the Gaussian. With the introduction of class labels we may now characterize fitness over multiple objectives, albeit for the subset of exemplars actually mapped to the Gaussian alone, step 2(c).v. Moreover, the multi-objective view provides the opportunity to reward non-overlapping behaviors as well as error minimization; the second property by which problem decomposition is encouraged. However, there is a 'trick' to this process: the estimation of overlap is performed relative to the contents of the learner archive, as established by competitive coevolution, Feature 4. Naturally, having established the relative fitness of individuals, selection and reproduction takes place under a Pareto multi-objective model [12] which encourages diversity without recourse to distance based metrics, step 2(c).vi to vii. This completes the explicitly cooperative aspect of the framework.

Feature 4: Competitive coevolution is now used to identify the best points and classifiers to retain outside of the point and learner populations, step 2(d). To do so, an outcome vector is constructed over the current contents of the point archive and population, relative to the current contents of the learner archive and population. As such this process follows the IPCA algorithm of de Jong [6], however, any form of this class of competitive coevolution would be appropriate. Special attention is necessary to the derivation of an appropriate mechanism for establishing the outcome vector. In particular this is a real-valued pairwise matrix of the behavior of each individual relative to points. Only individuals that are non-dominated in the Pareto sense (with respect to outcome vectors) represent candidates for archiving. Similarly, only the points making the distinction between dominated and non-dominated learners represent candidates for the point archive. It is the contents of the learner archive that represents the set of individuals comprising the solution.

Feature 5: Stop criteria is established in a problem independent manner by making use of the concept of Pareto rank histograms, step 2(e), as established in the Pareto multi-objective technique adopted in Feature 2 above. Unlike the original GA context in which this concept was derived [12], we also deploy it in a class wise manner. This enables us to declare classes converged class-by-class, thus providing the ability to redeploy the individuals associated with that class, such that they are reassigned to classes as yet not converged.

Further algorithmic details are available in [18] and [17].

3 Evaluation

Empirical evaluation is generally considered the basis for establishing the significance of different machine learning algorithms. As such the typical approach takes the form of a 'bake-off' between the proposed model and a 'straw man' alternative, as evaluated on a handful of *a priori* selected data sets. When one algorithm is able to establish significance of a one sided statistical test on more of the data sets than the other, a winner is declared. Such an approach is increasingly being questioned [7]. Under such a context, we consider factors central to establishing a better basis for evaluation of GP methods relative to classical deterministic machine learning methods as follows:

Performance measure: Depending on the characteristics of the underlying data set, assuming a single performance metric may lead to very biased results. The poor performance of 'error' or 'accuracy' based metrics has long been recognized [22], yet such metrics continue to be employed as the sole basis for comparison. Moreover, this has also lead to ignoring other performance factors such as model complexity, in favor of metrics purely associated with classification performance. Unfortunately this can lead to 'black box' solutions in which the complexity of the resulting model is ignored, leading to a low acceptance rate of machine learning based solutions;

Inappropriate use of statistical tests: Aside from the relevance of null hypothesis based statistical testing as a whole [7], a machine learning practitioner is frequently left with an implicit experimental design problem. The non-deterministic methodology of GP requires that evaluation be conducted over multiple runs per data partition. Conversely, deterministic machine learning algorithms only require one evaluation per partition (for a given selection of learning parameters). The evaluation design problem of interest here therefore comes down to establishing how the single performance point from the deterministic algorithm may be compared to multiple points from the evolutionary method;

Data sets: Benchmarking against well known data sets may provide several advantages, including the development of a body of knowledge regarding the attainable level of performance associated with each data set, the implicit biases inherent in the data, and how well particular learning algorithms perform. Conversely, focusing on the same set of data sets also tends to result in learning algorithms 'over fitting' the characteristics specific to the subset of popular data sets.

In this work we are interested in assessing how the proposed CMGE framework compares to an SVM classifier. As such this could result in the classical 'bake off.' In order to avoid this we attempt to address the above generic problems with the bake off approach, at least in part, by adopting the following measures:

Multifaceted performance evaluation: Performance will be considered from three perspectives: Classification performance, model complexity and CPU training time. In the case of specific *classification performance* metrics, we are again faced with the inherent problem of choosing a metric that provides an effective summarization for both binary and multi-class domains and unbalanced class distributions [10]. One approach might be to utilize Receiver Operating Characteristic curves. However, this also implies adopting a suitable heuristic for defining performance thresholds over a set of models (c.f. CMGE intra-class solutions), as well as being limited to pairwise comparisons (thus, not scaling gracefully under multi-class domains). In this work we will therefore build a 'score' metric that measures the class-wise detection rate under an equal class significance weighting assumption:

$$score = \frac{\sum_{i \in C} DR_i}{\mid C \mid} \tag{1}$$

where DR_i is the detection rate of class 'i'; and $\mid C \mid$ is the number of classes.

Model complexity will be assessed through a count of the number of 'elements' used to construct the model; where the elements are naturally a function of the representation assumed by each machine learning model. In the case of CMGE 'elements' are considered synonymous with a normalized instruction count over all individuals participating in a solution, whereas in the case of the SVM paradigm we count the number of support vectors. The CMGE normalization reflects the implicit bias of support vectors to index all features, whereas GP instructions may only index a maximum of two features. This implies that, given a two argument instruction, there are $F-1$ instructions required to index all features, where 'F' is the feature count. The CMGE element count therefore takes the form:

$$CMGEcomplexity = \frac{NumIndividuals \times NumInstructions}{F-1} \tag{2}$$

Naturally, equivalence is not implied, as it is in the hands of the practitioner to decide how transparent the ensuing solution might be.

CPU training time will be assessed in terms of a common computational platform (dual core iMac with 2GB RAM, Tiger OS) for four different training data sets, corresponding to thousands, tens of thousands, 150 thousand and 200 thousand exemplars respectively. As such this is designed to provide feedback in terms of the trade off between fixed memory footprint of CMGE – at the possible expense of accuracy – versus the increasing memory footprint of the SVM implementation.

Normalize relative to deterministic performance point: Models of machine learning such as an SVM, C4.5, or Naive Bayes apply a deterministic procedure for building a model; whereas evolutionary methods may make different design choices on account of their stochastic policy of credit assignment. This results in an inherent disjunction in the number of performance points available for comparison. For each training partition we have one performance point care of the deterministic model, versus fifty or so points under an evolutionary method; performing cross validation does not change this. To this end

Table 1: Data Set Characterization. $\mid T \mid$ is the total number of exemplars in the data set; Train (Test) indicate the exemplar count for training (test) sets, or whether a stratified '10-fold' is deployed; 'F' denotes the number of features; and $\mid C \mid$ the number of classes; '†' denotes a class with a representation at less than 0.7% of the training partition

Dataset	$\mid T \mid$	Train	Test	F	$\mid C \mid$	% Distribution
Boston	506	10-fold	10-fold	13	3	≈ balanced
Breast	675	10-fold	10-fold	10	2	(65):(45)
Census	295 173	196 294	98 879	41	2	(94):(6)
Contra	1 425	10-fold	10-fold	9	3	(43):(22):(35)
Image	2 310	210	2 086	19	7	≈ balanced
Iris	147	10-fold	10-fold	4	3	≈ balanced
KDD99	222 871	145584	77 287	41	5	(60):†:(37):(1.5):†
Liver	341	10-fold	10-fold	6	2	(42):(58)
Pima	768	10-fold	10-fold	8	2	65:35
Shuttle	58 000	43 500	14 500	9	7	(78):†:†:(15.5): (5.65):†:†
Thyroid	7 129	3709	3420	21	3	(2):(5):(93)
Wine	178	10-fold	10-fold	13	3	(33):(40):(27)

we will use the performance point from the deterministic model to normalize the distribution of points provided by the evolutionary model. Plotting the result as a box plot establishes the likelihood of the evolutionary method performing any better than the deterministic. Thus, by retaining the distribution of points we are able to directly resolve the significance or cost of assuming a stochastic model relative to the *a priori* post-training performance metric. Naturally, the deterministic model will be evaluated under multiple parameterizations of learning parameters and the best case selected with respect to the same performance function under the training partition.

Seek out a 'diverse' cross section of data sets: Multiple sources for benchmarking data sets are now available e.g., UCI and KDD. Moreover, the major sources are receiving additional data sets on a continuous basis. As such this provides the opportunity to sample data sets with properties such as: different application domains, different exemplar distributions, multiple classes, or feature counts. In addition, various benchmark studies have identified data sets that are in some way 'difficult' to classify [15]. In this case a total of twelve data sets are utilized, Table 1, with the goal of retaining some 'old favorites' with varying degrees of difficulty (e.g., Breast, Iris, Liver, Pima, Wine) as well as introducing large multi-class / multi-feature / very unbalanced data sets (e.g., Census-Income, Contraceptive, ImgSeg, KDD Cup 99, Shuttle, Thyroid).

Table 2: CMGE Parameters. 'Archive' sizes are quoted per class.

Clustering	value	Archive or Pop.	value
α	150	Learner Pop. Size	50
β	155	Learner Archive Size	30
γ_{lower}	0.5	Point Pop. Size	30
γ_{upper}	0.75	Point Archive Size	30
GE	value	Early Stopping	value
max. prog. length	4,096	Min. Difference	0.1
max. Codon count	256	Min. Pop.	20
max. Epochs	500	Faction Evolved	$\frac{1}{5}$
Crossover and Mutation rates: canonical, (structural)			
P(crossover)	0.5, (0.9)	P(mutate)	0.01, (0.9)

3.1 Parameterization

3.1.1 CMGE

The CMGE model requires parameterization of the basic Grammatical Evolution (GE) model, archive and population sizes, cluster algorithm parameters, and definitions for early stopping, Table 2. GE parameters set limits on the maximum length of an individual before and after codon transcription [20]. Variation operators take two basic forms: canonical and structure preserving. Canonical crossover and mutation tend to be global in their behavior, on account of the action of the grammar, and are therefore applied with a comparatively low likelihood, Table 2. Structure preserving variation operators make use of context information returned by genes associated with building terminals during application of the grammar [9]. As such this provides the basis for local variants of crossover and mutation, designed such that the impact is more likely to be restricted to single 'subtrees,' and are therefore applied at a higher frequency. The context free grammar itself is designed to support the four arithmetic operators, sine and cosine, exponential and natural log, square root and index the data set features.

Point and learner archives enforce a fixed upper bound on the memory mechanism used to retain points supporting the Pareto front of Learners identified during competitive coevolution. Separate point and learner archives are retained for each class, whereas a single population is maintained for establishing search diversity, as per the original IPCA algorithm [6]. A constant archive and population limit is maintained for all cases other than the learner population, which is larger to encourage greater learner diversity. The early stopping parameters are modeled on a simplified version of the Pareto histogram method [12] and effectively declare the degree of similarity between sequential generations of the EMO inner loop of CMGE. Learners and points from other classes continue to evolve, and search resources associated with the converged class are reassigned to the remaining classes. The final set of parameters from Table 2 define the radii (α and β) and stop conditions (γ) for the Potential function clustering algorithm [5] used in CMGE to independently configure the local membership function of each learner.

All of the above remain fixed across all runs, irrespective of data set. Needless to say, no claims are made regarding the optimality of the parameter selection or grammar. Relative to the training partitions of Table 1, fifty runs are performed in each case, thus data sets defined in terms of 10-fold cross validation imply a total of 500 runs under CMGE.

3.1.2 SVM

The method of Support Vector Machines (SVM) is based on a quadratic optimization problem designed about a 'kernel function' that maps the original input space to an orthogonal space, the support vectors. Much of the research for providing improved SVM approaches is directed towards establishing more efficient methods for performing the process of optimization. One of the first breakthroughs in this respect was to consider the process as a series of decompositions in which the Sequential Minimal Optimization method provided a very efficient paradigm [21]. More recent refinements have included the use of second order information to speed the rate of convergence, while also addressing the additional computational cost of doing so [8]. The LIBSVM implementation (release 2.85) supports the use of the aforementioned second order model [4] (see also Chapter 1 in [1]) and is therefore adopted in this benchmarking study.

Unlike the GP scenario, sources of variation in the model are now limited to the SVM learning parameters, of which the principal parameters are the kernel function and regularization parameter C. Two of the most widely utilized kernel functions take the form of the Gaussian (or radial basis) and Sigmoidal function, hence the following evaluation will consider both cases. Three values for the C parameter will be employed $\{1, 10, 100\}$ under each data set, with the SVM model selected corresponding to the best post training performance as evaluated by the 'score' metric of Equation (1). In addition the input features will be normalized to the unit interval relative to the default application values (no such normalization is applied under the evolutionary model); the SVM output will take real valued responses, where this represents a significant improvement over the default of binary values for multi-class domains.

One significant difference between SVM and CMGE methodologies with regards to computational resource is the requirement for extensive cache memory support with the SVM paradigm. Under the data sets previously benchmarked by [8], two cache memory limits were considered: 40 MB and 100KB, when the largest data set considered consisted of around 15,000 exemplars. In this study the Census data set consists of around 200,000 training exemplars. The resulting SVM memory utility went up to 2 GB; whereas the CMGE model has a constant memory utility, as defined by the union of point population and class archives. The impact of this from a computational perspective will be considered in the following performance evaluation.

4 Performance Evaluation

Following the above discussion, for each kernel function, three SVM models are trained per partition (reflecting different values for C) and then the score metric of Equation (1) employed to select the representative solution. Performance of that model is then evaluated on the test partition, again in terms of the score metric. This establishes either a single performance point, or ten performance points, as in the case of the cross-validation data sets i.e., those without an *a priori* defined training partition, Table 1. In the latter case we therefore establish a single SVM performance point by assuming the median of the ten test results. The single SVM performance point is then used to normalize the distribution of CMGE solutions. A CMGE distribution is established by taking the fifty best solutions as defined by the score metric over the training partition, post training, and evaluating the model under test. Naturally, data sets with a single training partition will utilize all fifty initializations, whereas the runs from the seven 10-fold cross validation scenarios will assume a sample from the 500 CMGE solutions.

4.1 Classification Performance

Figures 2 and 3 detail the distribution of SVM normalized CMGE test partition performance on each of the twelve data sets with respect to Gaussian and Sigmoid kernels respectively. Naturally, results above unity imply that the SVM performance point was bettered, whereas values below unity imply an SVM score bettering the CMGE distribution. Moreover, CMGE distributions are summarized in terms of a box plot, thus we are able to establish the contribution of any outlier behavior and explicitly identify the quartile performance points of each data set.

It is clear that of the two SVM models, the Gaussian kernel results in stronger scores than the Sigmoid on all but the Shuttle data set. Generally, the CMGE model is as capable as the best of the two SVM models for Breast, Iris, Shuttle, Thyroid and Wine data sets. In the case of the Liver and Puma data sets a minimum of twenty five percent of the CMGE results might be expected to perform better. However, the SVM is clearly much stronger under the Boston, Image Segmentation and KDD-99 data sets; and depending on the kernel Census and Contraceptive.

Table 3 details the numerical score for each of the twelve data sets; as such we can discern the specific data sets on which the SVM and CMGE models are most likely missing entire classes, as score values tend towards a multiple of $\mid C \mid$. In the case of both models, a tendency to ignore all but the major class exists on Census, and the two major classes on KDD99 (plus half of the third larger class). On Shuttle, the SVM tends to classify the four larger classes and half of a fifth; whereas the CMGE model focuses on two of the three in Boston and five of seven under Image Segmentation. In short, both CMGE and SVM models have specific strengths and weaknesses, or no free lunch.

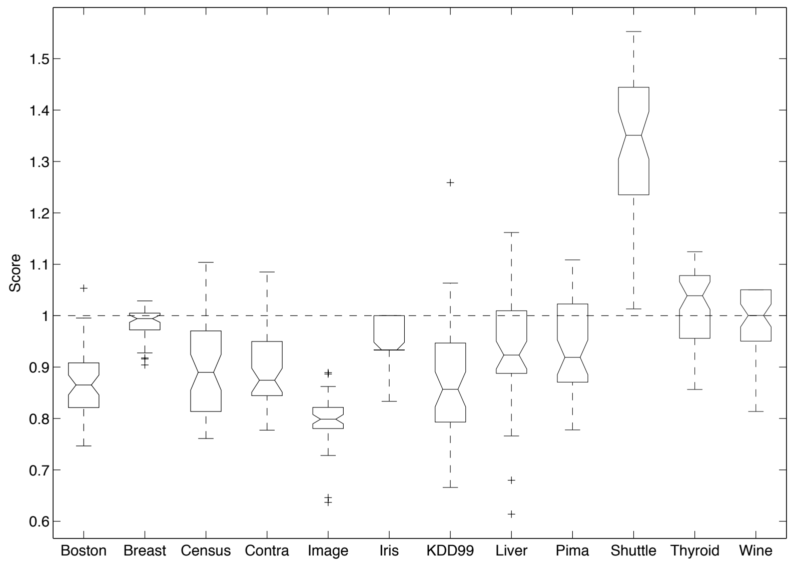

Figure 2: Gaussian kernel SVM performance point normalized CMGE score on test. CMGE distribution is summarized in terms of rank statistics of 1st, 2nd (median), and 3rd quartile, with wiskers indicating the limits of the distribution after which outliers are identified. Notches imply the 95th percentile associated with the variance of the median.

Table 3: Median SVM score per data set. Corresponds to the normalization factors used in Figures 2 and 3

Kernel	Boston	Breast	Census	Contra	Image	Iris
Sigmoid	0.789	0.963	0.592	0.452	0.904	0.904
Gaussian	0.788	0.972	0.657	0.500	0.906	1.00
–	KDD99	Liver	Pima	Shuttle	Thyroid	Wine
Sigmoid	0.554	0.616	0.707	0.637	0.784	0.948
Gaussian	0.589	0.704	0.713	0.607	0.807	0.952

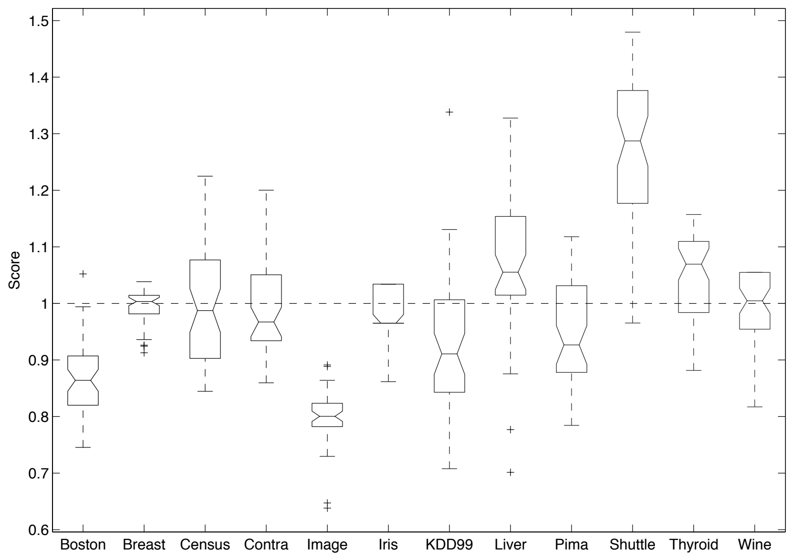

Figure 3: Sigmoid kernel SVM performance point normalized CMGE score on test. For comments on the Box Plot, see previous figure.

4.2 Model Complexity

In the case of model complexity, we again use the SVM performance point as the normalizing factor. As per the discussion in the previous section, an SVM element is considered a support vector, whereas an element under the CMGE model will be considered to be a normalized instruction count taken across all individuals comprising a solution, Equation (2). Figure 4 summarizes the resulting SVM normalized median complexity of CMGE solutions on each data set with respect to Gaussian and Sigmoid kernel functions. A log scale was necessary on account of the resulting wide variation in solution complexities. Values less (more) than unity imply a model with a lower (higher) count of CMGE elements than SVM elements. In eight of the twelve cases, CMGE models had a lower count, whereas in four cases the SVM solution utilized less. However, the factor by which CMGE solutions where deemed simpler was up to an order of magnitude in three cases, and over three orders of magnitude in one case (Census).

The underlying theme in Figure 4 is that the SVM 'simplicity' is generally correlated with the smaller data sets; Breast, Image, Iris, Liver, and Wine corresponding to the 5th, 3rd, 1st, 4th, and 2nd smallest data sets respectively. However, after this it more difficult to identify specific second order relations; with the easier Breast, Iris and Wine data sets having a strong preference for an SVM model from a simplicity perspective, but Breast and Wine benefiting from the more complex CMGE model in terms of classification. At the other extreme, the CMGE models that were generally simpler than the SVM solutions returned both weaker and stronger classification scores than the corresponding SVM models (Census and KDD99 classification performance was generally weaker, whereas Shuttle and Thyroid were generally stronger).

4.3 Computational Requirements

Computational requirements of each model are summarized for the Thyroid, Shuttle, KDD99 and Census data sets on account of the relative increments associated with the training partition; that is approx. {3,700, 44,000, 140,000, 200,000} exemplars respectively. Moreover, we also detail this characteristic in terms of the duration necessary to conduct a 'typical' run and that to build all the models necessary to make the performance evaluation. This implies fifty models in the case of CMGE and six models in the case of the SVM (two kernels by three regularization parameters). Note, this is summarized as a multiple of the typical run time, some of the SVM regularization parameters resulting in instability, thus runs of over ten hours. From Figure 5 the trade off between the SVM and CMGE approaches is again immediately apparent. The constant size of the point population and archives effectly decouple the CMGE model from the specific size of the training data set. When combined with the ability to class-wise introduce early stopping (care of the Pareto front behavior), there is little increase in computation time when training on tens of thousands of exemplars to hundreds of thousands. Conversely, the LIBSVM implementation of the SVM

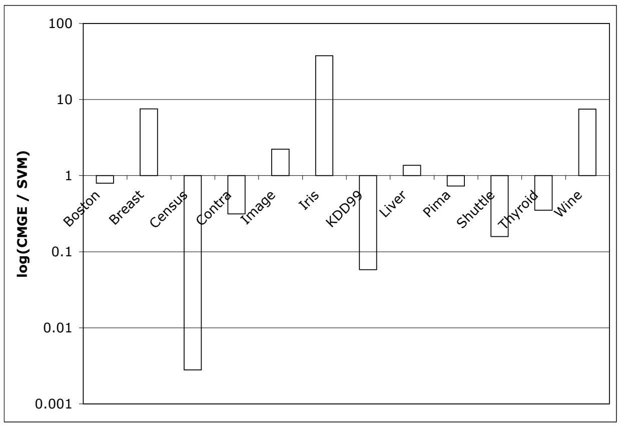

Figure 4: SVM performance point normalized CMGE. 'log' ratio of normalized CMGE instruction count versus SVM support vector count under Gaussian kernel. Values less (greater) than unity imply a simpler CMGE (SVM) solution.

model results in an increasing computational overhead as the training exemplar count increases, where this is also reflected in the strong correlation between SVM model complexity and size of the training data partition.

5 Related Work

Problem decomposition in GP has been a recurring theme, with one of the earlier models taking the form of the 'teaming' metaphor [2]. All teams were limited to have the same *a priori* specified number of individuals per team, thus providing context for variation operators limited to modifying the same individual in both teams. A recent development has been to combine the teaming metaphor with a multi-population island approach in which team members are developed independently in each population [23]. Ensemble methods provide a natural mechanism for establishing multi-individual behaviors. The negative-correlation model of [16] directly addresses the goal of establishing unique uncorrelated behaviors in the cooperating individuals from a single population, and does not assume *a priori* specification of the number of participating individuals. A recent extension to this added an EMO component for trading off error minimization and correlation objectives [3]. However, some drawbacks remain, such as the cost of estimating the correlation coefficient and post training imposition of a voting heuristic which may or may not be appropriate. One final

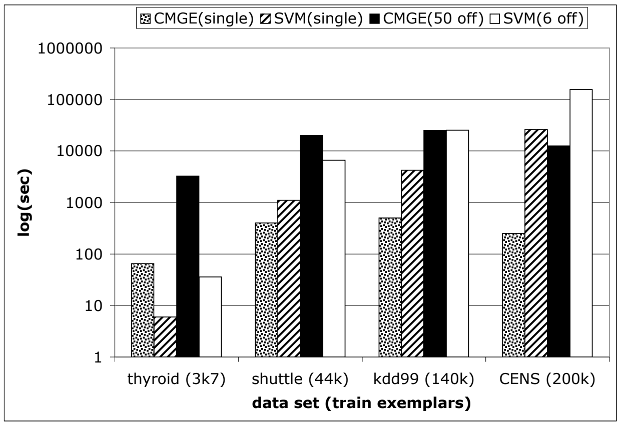

Figure 5: SVM and CMGE CPU training time in log seconds. Performance is expressed as a tuple: CMGE and SVM time for a typical run; CMGE and SVM time for 50 and 6 runs respectively.

model of interest takes the form of a bid based paradigm [13]. At initialization individuals are assigned a scalar representing class label, whereas the goal of the evolved program behavior is to identify a bidding strategy for which they receive most reward under the preassigned class label. Naturally suitable mechanisms are required to reward diversity in the bid behaviors; with a symbiotic model being recently adopted to establish a clear model for credit assignment between teams and programs [14].

6 Conclusions

Advances in areas such as Evolutionary Multi-objective Optimization (EMO) and competitive coevolution are providing new opportunities for addressing credit assignment in GP. Specifically, the proposed CMGE model scales to large data sets using a competitive coevolutionary model in combination with a class-wise balanced uniform sampling heuristic [24]. A local Gaussian wrapper operator is employed to delimit exactly what subset of exemplars and individual responds to. This insight enables us to evolve the region over which the Gaussian is built, thus associating it with the most dense set of exemplars on the *gpOut* axis. This provides the first mechanism for establishing problem decomposition. EMO evaluation is then very efficient, as evaluation is only performed over the exemplars explicitly associated with the region of the Gaussian. One

EMO objective explicitly rewards behavior that minimizes the overlap of exemplars mapped to the Gaussian; the second mechanism for establishing problem decomposition. However, a second insight is to 'bootstrap' this objective, by evaluating it against the individuals contained in the archives built during competitive coevolution. Thus, the class-wise archive produced by the competitive coevolutionary model identifies the team of learners; whereas EMO drives the mechanism by which the diversity of learners is maintained, and class-wise early stopping is established.

The strength of the proposed CMGE model naturally lies in the ability of the model to decompose problems such that multiple models support the classification of a single class. The decomposition is entirely an artifact of the evolutionary model. This is most apparent in the general simplicity of the solutions, as compared to the SVM model. Moreover, this is also achieved without sacrificing the classification performance; with data set preferences being returned for both CMGE and SVM solutions. Needless to say, alternative SVM models are available that might result in a different relative weighting of performance factors. A case in point being SVM models that are based on an active learning strategy to explicitly address scaling to large data sets [1]. Needless to say, such models introduce a performance trade off in terms of model accuracy; whereas the benchmarking comparison conducted here was designed to contrast the relative merits of a known strong classification implementation (LIBSVM) versus an evolutionary model designed to address weaknesses in canonical GP. An additional evaluation of the problem decomposition property of CMGE in terms of GP competitive coevolution versus the cooperative-competitive teaming model of CMGE is made in [19] and [17].

Future work will consider the utility of the CMGE framework to problem domains with large feature spaces. The intuition being that as the CMGE model is able to identify intra-class as well as inter-class models, it is now explicitly possible to decouple features appropriate for models at the intra-class level, rather than having to tie feature selection to representation for all class consistent classification (as in one-classifier-per-class frameworks). Such a property could be of particular importance to bio-informatic and document or multi-media classification/ categorization domains in which the feature spaces are both very large, but also sparse and multi-labelled. Results on the data sets considered in this work have been shown to support the above hypothesis for strong intra-class feature subset identification by intra-class classifiers [17].

Acknowledgements

The authors gratefully acknowledge the support of PRECARN, 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] L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors. *Large-Scale Kernel Machines*. MIT Press, 2007.

[2] M. Bremeier and W. Banzhaf. Evolving teams of Predictors with linear Genetic Programming. *Genetic Programming and Evolvable Machines*, 2(4):381–407, 2001.

[3] A. Chandra, H. Chen, and X. Yao. Trade-off between diversity and accuracy in ensemble generation. In Y. Jin, editor, *Multi-Objective Machine Learning*, volume 16 of *Studies in Computational Intelligence*, chapter 19, pages 429–464. Spinger-Verlag, 2006.

[4] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2007. http://www.csie.ntu.edu.tw/c̃jlin/libsvm.

[5] S. L. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2:267–278, 1994.

[6] E. D. de Jong. The incremental pareto-coevolution archive. In *Proceedings of the Genetic and Evolutionary Computation Conference*, number 3102 in Lecture Notes in Computer Science, pages 525–536. Springer, 2004.

[7] C. Drummond. Machine learning as an experimental science (revisited). In *AAAI Workshop on Evaluation Methods for Machine Learning*, pages 1–5, 2006. WS-06-06.

[8] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[9] R. Harper and A. Blair. A structure preserving crossover in Grammatical Evolution. In *IEEE Congress on Evolutionary Computation*, volume 3, pages 2537–2544, 2005.

[10] N. Japkowicz. Why question machine learning evaluation methods? In *AAAI Workshop on Evaluation Methods for Machine Learning*, pages 6–11, 2006. WS-06-06.

[11] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–450, 2002.

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

[13] P. Lichodzijewski and M. I. Heywood. Coevolutionary bid-based Genetic Programming for Problem Decomposition in Classification. *Genetic Programming and Evolvable Machines*, 9(4):331–365, 2008.

[14] P. Lichodzijewski and M. I. Heywood. Managing team-based problem solving with Symbiotic Bid-based Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 363–370, 2008.

[15] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40:203–228, 2000.

[16] Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.

[17] A. R. McIntyre. *Novelty Detection + Coevolution = Automatic problem decomposition: A framework for scalable Genetic Programming classifiers.* PhD thesis, Dalhousie University, Faculty of Computer Science, November 2007. http://www
.cs.dal.ca/m̃heywood/Thesis.

[18] A. R. McIntyre and M. I. Heywood. Multi-objective competitive coevolution for efficient GP classifier problem decomposition. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 1930–1937, 2007.

[19] A. R. McIntyre and M. I. Heywood. Cooperative problem decomposition in pareto competitive classifier models of coevolution. In *Proceedings of the European Conference on Genetic Programming*, number 4971 in Lecture Notes in Computer Science, pages 289–300. Springer, 2008.

[20] M. O'Neill and C. Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, 2001.

[21] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods – Support Vector Learning*, B. Scholkopf et al., (eds), pages 185–208, 1998.

[22] R. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *International Conference on Machine Learning*, pages 43–48, 1998.

[23] R. Thomason and T. Soule. Novel ways of improving cooperation and performance in Ensemble Classifiers. In *ACM Genetic and Evolutionary Computation Conference*, pages 1708–1715, 2007.

[24] G. M. Weiss and R. Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, 2003.