GEML: Evolutionary Unsupervised and Semi-Supervised Learning of Multi-class Classification with Grammatical Evolution

Jeannie M. Fitzgerald, R. Muhammad Atif Azad and Conor Ryan
Biocomputing and Developmental Systems Group, University of Limerick, Limerick, Ireland

Keywords: Unsupervised Learning, Semi-supervised Learning, Multi-class Classification, Grammatical Evolution, Evolutionary Computation, Machine Learning.

Abstract: This paper introduces a novel evolutionary approach which can be applied to supervised, semi-supervised and unsupervised learning tasks. The method, *Grammatical Evolution Machine Learning* (GEML) adapts machine learning concepts from decision tree learning and clustering methods, and integrates these into a Grammatical Evolution framework. With minor adaptations to the objective function the system can be trivially modified to work with the conceptually different paradigms of supervised, semi-supervised and unsupervised learning. The framework generates human readable solutions which explain the mechanics behind the classification decisions, offering a significant advantage over existing paradigms for unsupervised and semi-supervised learning. GEML is studied on a range of multi-class classification problems and is shown to be competitive with several state of the art multi-class classification algorithms.

1 INTRODUCTION

Evolutionary algorithms (EAs) are a collection of algorithms which to varying extents emulate aspects of biological evolution. One of these, Genetic Programming (GP) (Koza, 1990) has been successful on a wide range of different problems from several diverse domains (Fogel, 2000), achieving many *human competitive* results (Banzhaf, 2013). However, it is probably fair to say that a significant proportion of previous work has concentrated on supervised learning tasks and, aside from some notable exceptions, studies on unsupervised and semi-supervised learning have been left to the wider machine learning (ML) community.

Two of the most important tasks tackled by ML techniques are regression and classification, and GP has proven itself as an effective learner on each of these: GP has achieved competitive results particularly on symbolic regression and binary classification tasks. Although many multi-class classification (MCC) studies have been undertaken, it remains a problem which is considered challenging for traditional tree based GP (Castelli et al., 2013). This paper takes up a triple challenge: it investigates MCC in a semi-supervised as well as in an unsupervised context.

This study is the first step in a larger investigation for which the motivation is the requirement for an unsupervised algorithm which can be applied to grouping/categorisation tasks involving unlabelled inputs from the medical domain, and where the unsupervised algorithm can supply human interpretable justification for categorisation decisions. The ultimate objective is to go beyond labelling – we want to see which patterns of input data group together, or whether categories so arising can become a teaching aid for training.

Clustering is a natural choice for such tasks, however standard clustering algorithms lack the ability to provide the reasoning behind cluster allocations in a human readable form. In the medical domain, it is usually important that the learner has the capability to provide human understandable explanations of its decisions so that human experts can have confidence in the system. In this respect, decision trees (DTs) have the nice property that the induced DT itself provides an easily comprehensible explanation of all decisions. Unfortunately, traditional DTs rely on ground truth information to make decisions and use of this information is not permissible in an unsupervised context. For these reasons, although each of these methods have attractive properties, we conclude that neither DTs nor clustering approaches are, *in their normal mode of use,* appropriate for unsupervised categorisation tasks which require an explanation from the learner.

Fitzgerald, J., Azad, R. and Ryan, C..
GEML: Evolutionary Unsupervised and Semi-Supervised Learning of Multi-class Classification with Grammatical Evolution.
In *Proceedings of the 7th International Joint Conference on Computational Intelligence (IJCCI 2015) - Volume 1: ECTA*, pages 83-94
ISBN: 978-989-758-157-1

Although there is some important existing work in the area of unsupervised classification in the medical domain, see for example (Mojsilović et al., 1997; Greene et al., 2004; Belhassen and Zaidi, 2010), the topic remains relatively unexplored. This study is the first step of a larger exploration of somewhat uncharted waters.

We hypothesise that it may be possible to combine the desirable qualities of both algorithms by taking the underlying concepts and wrapping them in an evolutionary framework – specifically a grammatical evolution (GE) (O'Neill and Ryan, 1999) framework. This design is attractive due to its symbiotic nature: a GE grammar is used to generate human readable decision tree like solutions and the evolutionary process is applied to the task of optimising the resulting cluster assignments – thus emulating both the decision making behaviour of DTs and the iterative operation of traditional clustering approaches. Not only does GE produce human readable solutions, but it has been shown (Azad and Ryan, 2014) that the paradigm seems to be able to avoid bloated, over-complex ones.

While we can hypothesise that this hybrid approach might be a good idea, objective evaluation of algorithm performance is required before concluding that the resulting models are likely to be of any practical use. One way of accomplishing this is to compare results with other unsupervised algorithms using some common metric of cluster performance. However, it could be argued that without ground truth information any method of comparison is flawed. Another possible approach for evaluating the effectiveness of the proposed method would be to apply it to data about which something is already known, where that knowledge is not used in the learning process – merely to evaluate and compare performance afterwards.

Considering our original objective, we were also interested in learning what sort of performance gaps we could expect between our unsupervised method and, supervised and semi-supervised approaches using the same data. Thus, we choose to construct this study such that it would be possible to compare the performance and behaviour of the hybrid unsupervised learner with another state of the art unsupervised algorithm as well as with supervised and semi-supervised learners on the same data. Rather than using our original medical dataset at this point, we chose to carry out this first study using controllable synthetic data as outlined in section 4.2, with the intention of returning to the medical dataset if these preliminary experiments prove successful.

We set out to investigate the hypothesis that combining ML concepts with GE might facilitate the development of a new hybrid algorithm with two important properties: the ability to learn in an unsupervised way, and the capability of producing human readable results. However, due to the way in which we have designed the experiments – so that meaningful evaluation of the proposed algorithm would be possible, the resulting system delivers much more than initially planned.

In summary, we propose a novel hybrid GE system which incorporates ideas from two popular ML techniques: decision tree learning which is often applied to supervised tasks, and clustering methods which are commonly used for unsupervised learning tasks. The proposed system which we call GEML is applied to supervised, unsupervised and semi-supervised MCC problems. Its performance is compared with several state of the art algorithms and is shown to outperform its ML counterparts and to be competitive with the best performing ML algorithm, on the datasets studied.

In the remainder of this section we will briefly explain some of the concepts employed.

1.1 Decision Tree Learning

A decision tree is a hierarchical model that can be used for decision-making. The tree is composed of internal decision nodes and terminal leaf nodes. In the case of classification for example, internal decision nodes represent attributes, whereas the leaf nodes represent an assigned class label. Directed edges connect the various nodes forming a hierarchical, tree-like structure. Each outgoing edge from an internal node corresponds to a value or a range of values of the attribute represented by that particular node. Tree construction is a filtering and refining process which aims to gradually separate samples into the various classes with possibly multiple routes through the decision process for a particular class assignment.

1.2 Clustering

Clustering involves the categorisation of a set of samples into groups or subsets called clusters such that samples allocated to the same cluster are similar in some way. There are various types of clustering algorithms capable of generating different types of cluster arrangements, such as flat or hierarchical clustering. One of the best known clustering algorithms is K-means clustering which works in an iterative fashion by creating a number of centroids (aspirationally cluster centres). The algorithm groups samples depending on their proximity to these centres and then measuring the distance between the data and the near-

est centroids – K-means iteratively minimises the sum of squared distances by changing the centroid in each iteration and reassigning samples to possibly different groups. Of the EC work in the existing literature which combines GP or GE with unsupervised methods, K-means is the most popular of those used, as is outlined in Section 2.

1.3 Unsupervised, Semi-supervised and Supervised Learning

In simple terms, supervised, semi-supervised and unsupervised learning methods are differentiated by the amount of ground truth information that is available to the learning system: in supervised learning systems the 'answer' which may, for example, be a target variable or a class label is known to the system; semi-supervised systems may have access to such information for a limited number of samples or may involve revalidation of the automated prediction with expert knowledge; and unsupervised learners do not have any ground truth information with which to guide the learning process.

Although classification and clustering are conceptually similar, in practice the techniques are usually used in fundamentally different ways: clustering methods are generally applied to unsupervised tasks and do not require either training data or ground truth label information, whereas classification is usually a supervised task which requires both. At a basic level the goal of clustering is to group similar things together without reference to the name of the group or what membership of a group represents, other than the fact that the members are similar in some way, whereas the objective of classification is to learn, from examples, relationships in the data which facilitate the mapping of training instances to class labels, such that when presented with a new unseen instance the classification system may assign a class label to that instance based on rules/relationships learned in the training phase

2 PREVIOUS WORK

An in-depth review of the application of EAs to both clustering methods and decision tree induction is beyond the scope of this paper. Here we have chosen to focus on the most recent work and that which we determined to be most relevant to the current study. For a comprehensive survey of EAs applied to clustering, the interested reader is directed to (Hruschka et al., 2009), whereas a detailed review of EAs applied to decision tree induction can be found in (Barros et al., 2012).

Relative to the volume of existing research on supervised learning in the field of Evolutionary Computation (EC), unsupervised and semi-supervised learning have received little attention. Of the existing work, a significant proportion in the area of unsupervised learning recommends the use of clustering methods for feature selection (Kim et al., 2000; Mierswa and Wurst, 2006; Morita et al., 2003), and much of this utilises a traditional K-means approach.

Clustering was used in a novel way in (Omran et al., 2005) which proposed a Differential Evolution (DE) algorithm with built-in clustering functionality. They investigated its effectiveness on an image classification task, and compared their approach with several well known algorithms such as K-means – reporting statistically indistinct results.

Another interesting application of K-means was proposed by (Kattan et al., 2010) who integrated it into GP and used this hybrid approach for problem decomposition – grouping fitness cases into subsets. They applied their strategy to several symbolic regression problems and reported superior results to those achieved using standard GP. They later developed a similar approach (Kattan et al., 2015) for time series prediction.

A different unsupervised GP approach was proposed in (Neshatian and Zhang, 2009) where a novel fitness function was used in feature selection for the purpose of identifying redundant features. The authors reported superior results when performance was compared with several state of the art algorithms. GP was again employed in (Fu et al., 2014) where it was used to develop low level thin edge detectors. In that work the authors demonstrated that edge detectors trained on a single image (without ground truth) could outperform a popular edge detector on the task of detecting thin lines in unseen images.

With regard to multi-class classification problems, there have been several interesting approaches using tree based GP including strategies for decomposing the task into multiple binary ones (Smart and Zhang, 2005), treating MCC problems as regression tasks (Castelli et al., 2013) and experimenting with various thresholding schemes such as (Zhang and Smart, 2004). Other interesting methods have been proposed which utilise GP variants including multi level GP (MLGP) (Wu and Banzhaf, 2011), Parallel linear GP (Downey et al., 2012) and probability based GP (Smart and Zhang, 2004) to name a few.

There have been several other evolutionary approaches to MCC including self-organising swarm (SOSwarm) which was described in (O'Neill and

Brabazon, 2006b). There, particle swarm optimisation (PSO) was used to generate a mapping which had some similarities to a type of artificial neural network known as a self organising map. SOSwarm was studied on several well known classification problems and while the average performance seemed to degrade as the number of classes increased – the best performing solutions were competitive with the state of the art.

Clustering methods have also been applied to MCC problems. A hybrid method which combined a GA with local search and clustering was suggested in (Pan et al., 2003), where it was applied to a multi-class problem on gene expression data. The results of that investigation showed that their method (HGA-CLUS) delivered a competitive performance when compared with K-means and several earlier GA approaches described in (Cowgill et al., 1999) and (Maulik and Bandyopadhyay, 2000). GP was again combined with K-means clustering for MCC in (Al-Madi and Ludwig, 2013) where the researchers used the K-means algorithm to cluster the GP program semantics in order to determine the predicted class labels.

DTs have previously been combined with GE in (Deodhar and Motsinger-Reif, 2010). The algorithm was applied to the binary classification task of detecting gene-gene interactions in genetic association studies. The researchers reported good results when their GE with DT (GEDT) system was compared with the C.45 DT algorithm. Our proposed approach has some similarities to this work. However, that research focused on a supervised binary task where attribute values were restricted to a common set of 3 items.

Competitive results were also reported in (Munoz et al., 2015) in which K-Means clustering was again used with GP for MCC. There, clustering was combined with a multigenic GP approach in which each individual was composed of several solution parse trees having a common root node.

Concerning DTs, (Barros et al., 2012) concluded that good performance of EAs for decision tree induction in terms of predictive accuracy had been empirically established. They recommended that investigation of these algorithms on *synthetic data* should be pursued and also the possibility of using evolutionary computation for the evolution of decision tree induction *algorithms*. In this paper we address the first of these recommendations. The candidate solutions evolved by GE are computer programs which emulate decision trees, and these computer programs are produced using a grammar template capable of generating a multitude of different solutions. Thus, it could be argued that the proposed approach does, at least in some sense, also meet the second objective – the evolution of DT induction algorithms.

The novel contributions of this study are the proposal of a technique for unsupervised learning using an EA where the evolved learning hypotheses are in human readable form, and the extension of this to the development of a hybrid GE framework which can also be used for supervised and semi-supervised learning. The new system which we call GEML is successfully applied to the problem of multi-class classification. This is a proof of concept investigation and given the encouraging results, we anticipate that this approach may be refined to produce even better results on MCC problems but also modified and extended to combine other ML algorithms with GE and also to tackle different tasks such as symbolic regression.

3 PROPOSED METHOD

In ML DTs often use the concept of *information gain* to make branching decisions when constructing a decision tree and the measure of information gain used relies on knowledge of the ground truth labels. Thus, it is not appropriate to use the same mechanism for our semi-supervised and unsupervised methods. Instead, we make use of an *if then else* structure where the if component may be used to test various conditions pertaining to the data, whereby the learning system has access to both the attribute values and also to the variance of each attribute on the training data. For the supervised learner it is acceptable to make use of the ground truth information available in the training data, and so we could, for example, use the within class variance for each class. However, in this initial investigation we chose to use the same grammar for all tasks so that potential differences in behaviour and performance may be more easily understood. Thus, by design our system does not currently implement DTs according to a strict definition of the algorithm, as using label information precludes unsupervised learning.

3.1 Grammatical Evolution

Grammatical Evolution (GE) (O'Neill and Ryan, 1999) is a flexible EC paradigm which has several advantages over other evolutionary methods including standard GP. In common with its traditional GP relative, GE involves the generation of candidate solutions in the form of executable computer programs. The difference is that GE does this using powerful

grammars whereas GP operates with a much more limited tool-set.

A key aspect of the GE approach is *genotype phenotype separation* whereby the genotype is usually (but not necessarily) encoded as a vector of integer *codons*, some or all of which are *mapped* to production rules defined in a user specified grammar (usually in Backus-Naur-Form). This mapping results in a phenotype executable program (candidate solution). GE facilitates focused search through the encoding of domain knowledge into the grammar and the separation of search and solution space such that the search component is independent of the representation and may, in principle, be carried out using any suitable algorithm – a genetic algorithm is often used but other search algorithms such as PSO (O'Neill et al., 2006) and DE (O'Neill and Brabazon, 2006a) have also been used to good effect.

The role of the user-defined grammar is key to guiding the evolutionary search towards desirable solutions. The grammar is essentially a specification of what *can be evolved* and it is up to the evolutionary system to determine which of the many possible solutions which can be generated using the specification *should be evolved* (Ryan and O'Neill, 2002). A simple change in a grammar may induce drastically different behaviour. In this work we have specified a grammar, shown in Figure 1, which facilitates the allocation of data instances to clusters based on the results of applying simple 'if then else' decision rules. Although the rules are simple, the grammar allows for the construction of rich expressions which may be capable of representing both simple and complex relationships between attributes as seen in Figure 3.

```
<expr> ::=<label> if <cond> else <label>
  | <label>if<cond>else<expr>
  | <label>if<cond>and<cond>else <expr>
  | <label>if<cond>or<cond>else<expr>
  | <expr> if <cond> else <expr>
  | <expr> | <label>

<label> ::= 0 | 1 | 2 | 3 | 4

<cond> ::= <attr><relop><const>
  | <subExpr><relop><const>
  | <subExpr><relop><subExpr>
  | <var><relop><var>
  | <attr><relop><attr>

<subExpr> ::= <attr><op><const>
     | <attr><op><var>
     | <attr><op><attr>

<op>  ::= + | - | * | /
<relop> ::= <=
<const> ::= <digit>.<digit>
    | -<digit>.<digit>

<digit> ::= 0 | 1 | 2 | 3 | 4
    | 5 | 6 | 7 | 8 | 9

<attr> ::= x[0] | x[1] | x[2]
<var>  ::= v[0] | v[1] | v[2]
```

Figure 1: Example grammar for five class problem with three attributes ($< attr >$). The $< var >$ entries represent the variance in the training data across each attribute. The 'then if else' format is designed to simplify the syntax required in a python environment – as the system evolves python expressions. The division operation is protected in the implementation.

3.2 Objective Functions

For each learning task: supervised, unsupervised and semi-supervised, we employ a different objective function to drive evolutionary progress. In the supervised case we use classification accuracy which is simply the proportion of instances correctly classified by the system. Since this study uses balanced data sets, we simply use the number of correct predictions to measure system performance. Accuracy values range between 0 and 1 where 1 represents perfect classification. The system is configured with an objective function designed to maximise fitness .

For the unsupervised task we have chosen to use a metric of clustering performance known as a *silhouette co-efficient* or *silhouette score* (SC) as the objective function. The SC is a metric which does not require knowledge of the ground truth which makes it suitable for use in an unsupervised context. For each data point two measures are calculated: a. the average distance (according to some distance metric) between it and every other point in the same cluster and b. the mean distance between it an all of the points in the nearest cluster that is not it's own cluster.

The silhouette score over all points is calculated according to the formula shown in equation 1. Our system tries to maximise this value during the evolutionary process. Note that this approach, which aims at optimising the SC rather than cluster centroids, is quite different from the other EC methods outlined in section 2 where the K-means algorithm was generally used for clustering tasks.

$$(b-a)/max(a,b) \tag{1}$$

The resulting value ranges between -1 *and* 1 where a negative value suggests that samples are in the wrong

clusters, a value near 0 indicates that there are overlapping clusters and a score close to 1 means that clusters are cohesive and well separated.

In order to calculate the silhouette scores it is first necessary to choose an appropriate distance metric from the many and varied options available in the literature. In this work we have used *cosine distance* also known as *cosine similarity* as it is suitable for determining the similarity between vectors of features and obviates the need for data normalisation. Also, we experimented with several metrics including euclidean and mahalanobis distance before choosing *cosine distance* – as its use resulted in the best results over a range of synthetic classification problems. The results for cosine distance were better in terms of classification accuracy when cluster assignments were converted to class labels using a standard approach.

Semi-supervised learning is suitable for classification situations where some but not all ground truth labels are available. It may be the case, for example, that scarce or expensive human expertise is required to determine the labels. In these cases, it is usually possible to improve unsupervised performance by adding a small number of labelled examples to the system. Although, our synthetic data is fully labelled, we simulate partial labelling by only considering a random subset of training data (20% of the full data set) to be labelled; the rest of the data set is treated as unlabelled. We compute prediction accuracy on the labelled data and the silhouette score on the unlabelled set. We then add the two measures to get a final score and strive to maximise this score during evolution.

To summarise, candidate solutions are generated using a specification described in a grammar such as the one shown in Figure 1, where the same grammar is used for all of the GEML setups. Next, using the decision rules defined in the grammar problem instances are assigned to clusters as illustrated in Figure 2 and then depending on whether the task is supervised, unsupervised, or semi-supervised the system tries to optimise the classification accuracy, the silhouette score or a combination of the two. The key point here, is that the same grammar is used for each type of learning – only the objective function is different.

Figure 2 illustrates the expression tree of an example solution for a five-class task. Similar to a DT, the internal nodes of the tree represent branching decision points and the terminal nodes represent cluster assignments.

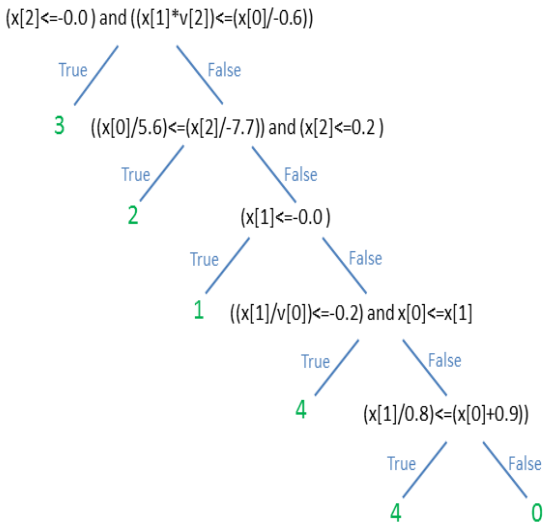

Figure 2: Example Expression Tree.

4 EXPERIMENTS

In this section we outline the construction of our experiments including the parameters, datasets and benchmarks used. We also detail the results of these experiments together with the results achieved on the same problems with our chosen benchmark algorithms. Details of the naming convention for the various experimental configurations are shown in Table 1.

4.1 Benchmark Algorithms

As we incorporate ideas from DT learning and clustering methods into our hybrid GEML approach, it is appropriate that we benchmark the proposed approach against Decision Trees (Breiman et al., 1984) as a supervised method and against the K-means clustering algorithm (Steinhaus, 1957) as an unsupervised paradigm. For semi-supervised learning we compare with a label propagation (LP) (Raghavan et al., 2007) algorithm. The idea behind LP is similar to k-Nearest Neighbours (k-NN) (Altman, 1992) and was originally proposed for detecting community structures in networks.

We also compare with support vector machines (SVMs) (Boser et al., 1992) for supervised learning as the method may provide a useful benchmark as it is known to achieve good results with balanced datasets, which is the case here, and while SVMs are inherently binary classifiers they can perform multi-class classification in various ways, most commonly using a “one versus all” strategy.

For comparison purposes we choose simple classi-

```
2 if x[0]*v[1]<=-0.2 and v[2]<=v[1] else 1 if (x[1]*v[0])<=(x[0]*-0.8)
else 0 if x[2]/v[2]<= x[0]*-2.0 or x[2]<=x[1] else 1 if x[1]<=x[0]
else 1 if x[0]*x[2]<=x[1]/x[0] and x[1]/v[2]<=-0.3) else 2
if x[2]/v[2]<=x[0]*2.9 or x[2]<=x[1] else 1 if x[0]<=-0.1 else 0
```

Figure 3: Example expression generated for a three class, three attribute classification task.

fication accuracy as a performance metric. It has been empirically established in the GP literature that simple classification accuracy is not a reliable measure of classification on *unbalanced* datasets (Bhowan et al., 2012), and that other measures such as average accuracy or Matthews Correlation Co-efficient might be more appropriate especially if combined with a sampling approach (Fitzgerald and Ryan, 2013). However, in this preliminary investigation, the classes *are balanced* which allows us to consider simple classification accuracy as a reasonable measure, particularly as we want to be able to observe differences in performance across the various levels of learning.

Table 1: Experimental Configurations.

Configuration	Explanation
GEML-SUP	Supervised GEML
GEML-SEMI	Semi-Supervised
GEML-UN	Unsupervised GEML
DT	Decision Tree Learning
LP	Label Propagation
KM	K-means Clustering
SVM	Support Vector Machine

We adopt a popular mechanism to determine which predicted label represents which a priori class label: the predicted class is mapped to the a priori class which has the majority of instances assigned to it, e.g for a binary task with 1000 training instances, if predicted class 1 has 333 members of ground truth class 1 assigned to it and 667 instances of class 0, then predicted class 1 is determined to represent the a priori class 0. It is important to note that this method is used to calculate the accuracy metric and used only for reporting and comparison purposes across all tasks and methodologies. The measure (classification accuracy) is of course used to drive the evolutionary process in the supervised tasks, and applies to only a percentage of the training instances in the semi-supervised case. In all cases, the same mapping from cluster assignment to class label determined during the training phase also applies when evaluating performance on test data.

For each of the GEML methods the evolved solutions look something like the example shown in Figure 3 which is essentially a python expression that can be evaluated for each training and each test instance. The result of evaluating the expression on a given instance is an integer which is converted into first a cluster assignment and then a class label, using the method already described. Although the objective functions used to determine fitness and drive evolution differ according to the type of learning model as detailed in section 3.2, we calculate the classification accuracy for each unevaluated individual at each generation on training and test data. At no time is the test data used in the learning process.

In each case, the algorithm was run fifty times using the same synthetic datasets and train/test splits as the GEML experiments. A different random seed was used for each run of the same algorithm and these same random seeds were used for the corresponding run of each algorithm. The popular scikit-learn (Pedregosa et al., 2011) python library for machine learning was used for all of these ML experiments.

4.2 Datasets

The various algorithms were tested on several synthetic multi-class datasets which were produced using the scikit-learn library (Pedregosa et al., 2011) which provides functionality for the generation of datasets with the aid of various configurable parameters. For this initial study we investigate balanced multi-class problems of two, three, four and five classes each. The library facilitates user control of the number, type and nature of features selected for experiments. For example, features can be informative, duplicate or redundant. We have chosen to use informative features only for the current work.

Given a problem configuration (number of classes), for each run of each algorithm a dataset of 1000 instances was generated and then split into training and test sets of 700 and 300 instances respectively. Identical random seeds were used for the corresponding run for each configuration, such that the same dataset was generated for each setup for a particular run number.

As this is very much a proof of concept investigation we have chosen to use synthetic datasets: we generated a reasonable number of instances, without added noise, and with few features, each of which is

Table 2: Evolutionary Parameters.

Parameter	Value
Population Size	500
Replacement Strategy	Generational
Number of Generations	100
Crossover Probability	0.9
Mutation Probability	0.01

informative. Employing synthetic datasets allows us to configure the data to have *informative* features such that it is, as far as possible, amenable to being clustered or classified. We have made these choices in an effort to ensure that the data is not biased to favour any particular algorithm or learning paradigm. For example, DTs are known to over-fit and not generalise well where there are a large number of features and few instances.

We note also that the decision to use synthetic datasets also delivers on the recommendation of (Barros et al., 2012) to use synthetic data for decision tree induction, as described in section 2.

4.3 EC Parameters

Important parameters used in these experiments are outlined in Table 2. Evolutionary search operators in GE are applied at the genotypic level, and in this work each individual's genotype is a linear genome represented by a vector of integers. The mutation operator operates by replacing a single integer with a new one randomly generated within a predefined range. One point crossover is used, whereby a single crossover point is randomly and independently selected from each of the two parents (that is, the two points are likely to correspond to different locations) and two new offspring are created by splicing parental segments together. In both cases, these operations take place in the effective portion of the individual, i.e the segment of the integer vector that was used in the genotype to phenotype mapping process – sometimes a complete phenotype is generated before requiring the full integer vector.

4.4 Results

Results for average and best training and test accuracy can be seen in Table 3, where for convenience the best result in each category is in bold text. For comparison purposes we are interested in comparing the supervised methods with each other and the unsupervised approaches with the other unsupervised methods etc. Thus we compare GEML-SUP with both DT and SVM, GEML-UNS with KM and GEML-SEMI with LP. However, we are also interested in observing the relative performances of the three different levels of learning.

Looking first at the supervised approaches, we can see that the SVM approach performs well across all of the problems studied with regard to average classification accuracy on both training and test data. Encouragingly, the GEML-SUP configuration is very competitive with SVM on the first three problems and outperforms DT on each of those tasks.

On the semi-supervised experiments GEML-SEMI outperforms LP on all problems for both training and test data in terms of average classification accuracy.

Finally, with regard to the unsupervised setups GEML-UN outperforms KM for average accuracy on training and test data on all problems.

In all cases, the performance of the various configurations degrades as the number of classes increases which is not surprising as adding more classes increases the difficulty of the problem to be solved. Overall, the SVM algorithm suffered less from this issue, which is again not surprising given that the implementation used here (Pedregosa et al., 2011) solves MCC problems using a binary decomposition strategy.

Looking again at the results in Table 3, we note that values for *best overall* training and test accuracy on the binary and three class tasks for each of the GEML methods are not competitive with the other approaches. For example, on the three class task, the average test accuracy for K-means is 0.74 whereas the best result is 0.99 compared with GEML-UN which has an average test accuracy of 0.75 and a best result of 0.86 and the GEML-SUP which has an average test accuracy of 0.92 and a best result of 0.95. The results for each of the GEML setups show that the reported standard deviation is lower than for the other algorithms.

It is unclear to us at this stage whether this phenomenon is associated associated with the GE paradigm or related to the MCC problem or something else. However, it could be argued that the behaviour is not necessarily a bad thing, as having a larger standard deviation with a higher extreme value can also mean that the algorithm is unreliable. After all, a good test set performance is only useful if it is consistently achieved, not as an exceptional case.

Due to the stochastic nature of GE one might hypothesise that there is a higher probability of many individuals achieving good results across many runs on the easier one and two class problems than on the more difficult problems where individuals have to learn to incorporate a larger number of class labels:

due to the added complexity there are likely to be fewer fit solutions early in the evolutionary process and thus fewer opportunities to improve through the application of genetic operators. One can easily imagine that there could be significant variability across runs depending on the quality of the initial population, and the existence of fewer highly fit solutions reduces the probability of truly excellent ones emerging.

Looking at the generalisation performance of each method in terms of the variance component, we adopt a simple measure whereby the variance error is simply the difference in performance of the various learning hypothesis between training and test data. In this respect, of the algorithms studied only the LP approach exhibits high variance. The various GEML methods all produce good generalisation performance. This is quite interesting as its close relation GP is known to exhibit a low bias high variance behaviour (Keijzer and Babovic, 2000). We can hypothesise that possible contributing factor to this contrast in behaviour is due to the grammar used, even if it contains recursive rules it is likely to constrain the size of the evolved programs. (Azad and Ryan, 2014) demonstrated empirically that while program size tends to increase steadily during GP runs, the average size of GE genomes remains roughly static after an initial period of growth or shrinkage. In those experiments, GP genomes were consistently larger than GE genomes after only fifty generations. It may be the case that these effects are preventing the evolved models from becoming over complex. It would be informative to apply the system to some regression problems where the required grammar would be fundamentally different, to see if the lack of over-fitting observed in the current experiments would also be seen there. The recent results shown in (Azad and Ryan, 2014) would suggest that GE does not over-fit on those problems either.

If we analyse the difference in performance between the various supervised, semi-supervised and unsupervised algorithms, it is not surprising that in all cases the supervised algorithms produced the best results and that the semi-supervised algorithms performed better than the unsupervised ones. Of course, the unsupervised and semi-supervised methods are not usually evaluated in the same way as supervised classification approaches: using accuracy as a performance metric. We have chosen to do so here as a convenient and practical way to gain some insight into the likely relative performance of our hybrid technique when it is applied to the three learning approaches.

The results suggest that while the performance of all of the algorithms deteriorates as the number of classes increases, this effect is even more evident for the unsupervised and semi-supervised methods where the performance of GEML-UN drops from 90% on the binary task to 66% for the five class problem, although this is still better than the corresponding LP algorithm. Again, we can hypothesis that while adopting a binary decomposition approach may seem attractive, this would be very challenging in an unsupervised context. However, there may be some scope for the strategy in the semi-supervised paradigm.

We carried out tests for statistical significance on the test results using the non-parametric Mann-Whitney U-Test. This revealed that statistical significance of results sometimes varied depending on the problem. Comparing SUP against SVM any differences were not significant for the two and three class problems but for the four and five 5 ones SVM is significantly

Table 3: Average and Best Classification Accuracy on Training and Test Data.

Task	Method	Avg. Training Accuracy	StdDev	Best Training Accuracy	Avg, Test Accuracy	StdDev	Best Test Accuracy
C2	GEML-SUP	**0.96**	0.01	0.98	**0.96**	0.01	0.97
	DT	0.93	0.04	0.99	0.93	0.04	0.99
	SVM	0.95	0.03	0.99	0.95	0.03	0.99
	GEML-SEMI	0.90	0.01	0.93	0.91	0.01	0.92
	LP	0.90	0.06	0.99	0.88	0.07	0.99
	GEML-UN	0.90	0.01	0.92	0.91	0.01	0.93
	KM	0.84	0.08	0.99	0.84	0.08	0.99
C3	GEML-SUP	**0.93**	0.01	0.94	**0.92**	0.02	0.95
	DT	0.87	0.04	0.97	0.88	0.05	0.99
	SVM	0.92	0.03	0.98	**0.92**	0.04	0.99
	GEML-SEMI	0.88	0.04	0.94	0.87	0.04	0.92
	LP	0.83	0.05	0.96	0.79	0.08	0.93
	GEML-UN	0.76	0.04	0.87	0.75	0.04	0.86
	KM	0.75	0.07	0.91	0.74	0.08	0.99
C4	GEML-SUP	0.86	0.01	0.88	0.86	0.02	0.89
	DT	0.82	0.04	0.94	0.83	0.04	0.93
	SVM	**0.88**	0.03	0.94	**0.88**	0.03	0.96
	GEML-SEMI	0.78	0.05	0.84	0.78	0.05	0.85
	LP	0.77	0.05	0.88	0.71	0.07	0.85
	GEML-UN	0.71	0.04	0.79	0.71	0.04	0.81
	KM	0.65	0.06	0.83	0.67	0.07	0.84
C5	GEML-SUP	0.77	0.06	0.85	0.75	0.04	0.83
	DT	0.77	0.04	0.88	0.79	0.05	0.89
	SVM	**0.85**	0.03	0.93	**0.86**	0.03	0.94
	GEML-SEMI	0.72	0.04	0.76	0.75	0.06	0.82
	LP	0.71	0.05	0.84	0.65	0.07	0.83
	GEML-UN	0.66	0.06	0.77	0.69	0.07	0 .82
	KM	0.63	0.05	0.78	0.63	0.06	0.79

better with 99% confidence. Comparing SUP against DT, the differences are significant at the 95%, 99% and 99% confidence levels for the two, three and four class problems respectively (SUP is better), but not significant for the five class ones. For the semi-supervised tasks, any differences are not significant for the binary task but the GEML-SEMI results are significantly better, at the 99% confidence level for the other three problems. Finally, the analysis comparing GEML-UNS with K-Means appears similar, where GEML-UNS is significantly better on the two, four and five class tasks having confidence levels of 99%, 95% and 95% respectively, and with a p-value of 0.58 there was no significant difference of the three class task.

4.5 Discussion

This is a simple study into the potential of the GEML system to perform effectively in unsupervised and semi-supervised tasks. Although the results are quite encouraging we feel that there is potential for improvement in the existing system. The obvious place to look for improvement is the all-important grammar. Our next steps will be to examine this to see how we can make it more effective. As a first move in that direction we will analyse the best individuals from our existing runs to determine which rules are contributing most and which are not performing. We will then modify the system applying this new information and use it to tackle a large, potentially noisy real-world medical dataset.

In the results section of this paper we have compared with several multi-class classification algorithms, and the reported results demonstrate that the most successful supervised technique is SVM. However, it is perhaps fair to point out that SVMs are not inherently multi-class, rather the algorithm usually (but not always) implements multi-class problems in either a "one versus one" or a "one versus all" approach, which in fact was how SVMs were implemented in this study. Thus, the performance of GEML and SVMs are, in one sense, not directly comparable. Given that the average performance of GEML on the two and three class problems is very competitive with that of the SVM, it is reasonable to hypothesise that equivalent performance to SVMs which use binary decomposition, might be expected on problems with greater numbers of classes if the GEML method were adapted to also perform multi-classification by way of binary decomposition. It may also be worth re-iterating that unlike SVMs the GEML setups all provide human readable solutions, which is an important consideration in many problem domains.

We have seen in this investigation that the GEML system which incorporates ideas from decision tree and cluster based learning has produced some statistically significant results. As GE is such a flexible paradigm there is no reason why alternative ML algorithms could not be incorporated instead. Once the candidate ML algorithm has some aspect which requires optimisation it should be suitable for an evolutionary approach.

Memetic algorithms (MA) are one of the interesting research topics in evolutionary computation at present, where the term MA is generally used to describe a synergistic algorithm which combines evolutionary approaches with population-based methods that use local search techniques or facilitate separate individual learning for problem search. It has been shown recently in the literature (Fitzgerald and Ryan, 2014) that individualised techniques can be effective when applied to GP generally. It may be illuminating to examine the effects of similar techniques in GE.

5 CONCLUSIONS

In this paper we introduced a prototypical, novel GE system which we call GEML which incorporates ML techniques commonly used for supervised and unsupervised learning into a flexible evolutionary framework which although designed primarily to perform unsupervised learning while providing human readable results, can also be used with minor modification for supervised and semi-supervised learning. We provided a brief description of important existing work in the areas of unsupervised learning and multi-class classification in both GP and GE and we briefly outlined how our novel approach may contribute to the existing literature. Following this, we described our algorithm in some detail together with some necessary background information about GE and other relevant concepts. Next, we described a set of experiments undertaken using GEML together with several other state of the art comparative ML algorithms. We presented and discussed the results of these experiments noting the promising performance of the new system which delivered competitive, statistically significant accuracy and good generalisation on all of the problems studied. Finally, we ended by suggesting some possible improvements to the existing system as well as proposing some potentially interesting new directions for GEML.

REFERENCES

Al-Madi, N. and Ludwig, S. A. (2013). Improving genetic programming classification for binary and multiclass datasets. In Hammer, B., Zhou, Z.-H., Wang, L., and Chawla, N., editors, *IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013*, pages 166–173, Singapore.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Azad, R. M. A. and Ryan, C. (2014). The best things dont always come in small packages: Constant creation in grammatical evolution. In *Genetic Programming*, pages 186–197. Springer.

Banzhaf, W. (2013). Evolutionary computation and genetic programming. In Lakhtakia, A. and Martin-Palma, R. J., editors, *Engineered Biomimicry*, chapter 17, pages 429–447. Elsevier, Boston.

Barros, R. C., Basgalupp, M. P., De Carvalho, A. C., Freitas, A., et al. (2012). A survey of evolutionary algorithms for decision-tree induction. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(3):291–312.

Belhassen, S. and Zaidi, H. (2010). A novel fuzzy c-means algorithm for unsupervised heterogeneous tumor quantification in pet. *Medical physics*, 37(3):1309–1324.

Bhowan, U., Johnston, M., and Zhang, M. (2012). Developing new fitness functions in genetic programming for classification with unbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(2):406–421.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Castelli, M., Silva, S., Vanneschi, L., Cabral, A., Vasconcelos, M. J., Catarino, L., and Carreiras, J. M. B. (2013). Land cover/land use multiclass classification using GP with geometric semantic operators. In Esparcia-Alcazar, A. I., Cioppa, A. D., De Falco, I., Tarantino, E., Cotta, C., Schaefer, R., Diwold, K., Glette, K., Tettamanzi, A., Agapitos, A., Burrelli, P., Merelo, J. J., Cagnoni, S., Zhang, M., Urquhart, N., Sim, K., Ekart, A., Fernandez de Vega, F., Silva, S., Haasdijk, E., Eiben, G., Simoes, A., and Rohlfshagen, P., editors, *Applications of Evolutionary Computing, EvoApplications 2013: EvoCOMNET, EvoCOMPLEX, EvoENERGY, EvoFIN, EvoGAMES, EvoIASP, EvoINDUSTRY, EvoNUM, EvoPAR, EvoRISK, EvoROBOT, EvoSTOC*, volume 7835 of *LNCS*, pages 334–343, Vienna. Springer Verlag.

Cowgill, M. C., Harvey, R. J., and Watson, L. T. (1999). A genetic algorithm approach to cluster analysis. *Computers & Mathematics with Applications*, 37(7):99–108.

Deodhar, S. and Motsinger-Reif, A. A. (2010). Grammatical evolution decision trees for detecting gene-gene interactions. In Pizzuti, C., Ritchie, M. D., and Giacobini, M., editors, *8th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics (EvoBIO 2010)*, volume 6023 of *Lecture Notes in Computer Science*, pages 98–109, Istanbul, Turkey. Springer.

Downey, C., Zhang, M., and Liu, J. (2012). Parallel linear genetic programming for multi-class classification. *Genetic Programming and Evolvable Machines*, 13(3):275–304. Special issue on selected papers from the 2011 European conference on genetic programming.

Fitzgerald, J. and Ryan, C. (2013). A hybrid approach to the problem of class imbalance. In Matousek, R., editor, *19th International Conference on Soft Computing, MENDEL 2013*, pages 129–137, Brno, Czech Republic.

Fitzgerald, J. and Ryan, C. (2014). Balancing exploration and exploitation in genetic programming using inversion with individualized self-adaptation. *International Journal of Hybrid Intelligent Systems*, 11(4):273–285.

Fogel, D. B. (2000). What is evolutionary computation? *Spectrum, IEEE*, 37(2):26–28.

Fu, W., Johnston, M., and Zhang, M. (2014). Unsupervised learning for edge detection using genetic programming. In Coello Coello, C. A., editor, *Proceedings of the 2014 IEEE Congress on Evolutionary Computation*, pages 117–124, Beijing, China.

Greene, D., Tsymbal, A., Bolshakova, N., and Cunningham, P. (2004). Ensemble clustering in medical diagnostics. In *Computer-Based Medical Systems, 2004. CBMS 2004. Proceedings. 17th IEEE Symposium on*, pages 576–581. IEEE.

Hruschka, E. R., Campello, R. J., Freitas, A., De Carvalho, A. C., et al. (2009). A survey of evolutionary algorithms for clustering. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 39(2):133–155.

Kattan, A., Agapitos, A., and Poli, R. (2010). Unsupervised problem decomposition using genetic programming. In Esparcia-Alcazar, A. I., Ekart, A., Silva, S., Dignum, S., and Uyar, A. S., editors, *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010*, volume 6021 of *LNCS*, pages 122–133, Istanbul. Springer.

Kattan, A., Fatima, S., and Arif, M. (2015). Time-series event-based prediction: An unsupervised learning framework based on genetic programming. *Information Sciences*, 301:99–123.

Keijzer, M. and Babovic, V. (2000). Genetic programming, ensemble methods and the bias/variance tradeoff - introductory investigations. In Poli, R., Banzhaf, W., Langdon, W. B., Miller, J. F., Nordin, P., and Fogarty, T. C., editors, *Genetic Programming, Proceedings of EuroGP'2000*, volume 1802 of *LNCS*, pages 76–90, Edinburgh. Springer-Verlag.

Kim, Y., Street, W. N., and Menczer, F. (2000). Feature selection in unsupervised learning via evolutionary

search. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 365–369. ACM.

Koza, J. R. (1990). Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical report.

Maulik, U. and Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. *Pattern recognition*, 33(9):1455–1465.

Mierswa, I. and Wurst, M. (2006). Information preserving multi-objective feature selection for unsupervised learning. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1545–1552. ACM.

Mojsilović, A., Popović, M. V., Nešković, A. N., and Popović, A. D. (1997). Wavelet image extension for analysis and classification of infarcted myocardial tissue. *Biomedical Engineering, IEEE Transactions on*, 44(9):856–866.

Morita, M., Sabourin, R., Bortolozzi, F., and Suen, C. Y. (2003). Unsupervised feature selection using multi-objective genetic algorithms for handwritten word recognition. In *2013 12th International Conference on Document Analysis and Recognition*, volume 2, pages 666–666. IEEE Computer Society.

Munoz, L., Silva, S., and Trujillo, L. (2015). M3GP: multiclass classification with GP. In Machado, P., Heywood, M. I., McDermott, J., Castelli, M., Garcia-Sanchez, P., Burelli, P., Risi, S., and Sim, K., editors, *18th European Conference on Genetic Programming*, volume 9025 of *LNCS*, pages 78–91, Copenhagen. Springer.

Neshatian, K. and Zhang, M. (2009). Unsupervised elimination of redundant features using genetic programming. In Nicholson, A. E. and Li, X., editors, *Proceedings of the 22nd Australasian Joint Conference on Artificial Intelligence (AI'09)*, volume 5866 of *Lecture Notes in Computer Science*, pages 432–442, Melbourne, Australia. Springer.

Omran, M. G., Engelbrecht, A. P., and Salman, A. (2005). Differential evolution methods for unsupervised image classification. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 966–973. IEEE.

O'Neill, M. and Brabazon, A. (2006a). Grammatical differential evolution. In Arabnia, H. R., editor, *Proceedings of the 2006 International Conference on Artificial Intelligence, ICAI 2006*, volume 1, pages 231–236, Las Vegas, Nevada, USA. CSREA Press.

O'Neill, M. and Brabazon, A. (2006b). Self-organizing swarm (soswarm): a particle swarm algorithm for unsupervised learning. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 634–639. IEEE.

O'Neill, M., Leahy, F., and Brabazon, A. (2006). Grammatical swarm: A variable-length particle swarm algorithm. In Nedjah, N. and de Macedo Mourelle, L., editors, *Swarm Intelligent Systems*, volume 26 of *Studies in Computational Intelligence*, chapter 5, pages 59–74. Springer.

O'Neill, M. and Ryan, C. (1999). Automatic generation of programs with grammatical evolution. In Bridge, D., Byrne, R., O'Sullivan, B., Prestwich, S., and Sorensen, H., editors, *Artificial Intelligence and Cognitive Science AICS 1999*, number 10 in , University College Cork, Ireland.

Pan, H., Zhu, J., and Han, D. (2003). Genetic algorithms applied to multi-class clustering for gene expression data. *Genomics, Proteomics, Bioinformatics*, 1(4):279–287.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Raghavan, U. N., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106.

Ryan, C. and O'Neill, M. (2002). How to do anything with grammars. In Barry, A. M., editor, *GECCO 2002: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, pages 116–119, New York. AAAI.

Smart, W. and Zhang, M. (2004). Probability based genetic programming for multiclass object classification. Technical Report CS-TR-04-7, Computer Science, Victoria University of Wellington, New Zealand.

Smart, W. and Zhang, M. (2005). Using genetic programming for multiclass classification by simultaneously solving component binary classification problems. In *Genetic Programming*, pages 227–239. Springer.

Steinhaus, H. (1957). Sur la division des corps matériels en parties. *Bull. Acad. Pol. Sci., Cl. III*, 4:801–804.

Wu, S. X. and Banzhaf, W. (2011). Rethinking multilevel selection in genetic programming. In Krasnogor, N., Lanzi, P. L. et al., editors, *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1403–1410, Dublin, Ireland. ACM. Best paper.

Zhang, M. and Smart, W. (2004). Multiclass object classification using genetic programming. In *Applications of Evolutionary Computing*, pages 369–378. Springer.