A Generic Optimal Feature Extraction Method using Multiobjective Genetic Programming

Y Zhang and P I Rockett

Technical Report No. VIE 2006/ 001
Department of Electronic and Electrical Engineering
University of Sheffield
A Generic Optimal Feature Extraction Method using Multiobjective Genetic Programming: Methodology and Applications

Yang Zhang & Peter Rockett

Abstract—In this paper, we present a generic, optimal feature extraction method using multiobjective genetic programming. We reexamine the feature extraction problem and argue that effective feature extraction can significantly enhance the performance of pattern recognition systems with simple classifiers. A framework is presented to evolve optimized feature extractors that transform an input pattern space into a decision space in which maximal class separability is obtained. We have applied this method to real world datasets from the UCI Machine Learning and StatLog databases to verify our approach and compare our proposed method with other reported results. We conclude that our algorithm is able to produce classifiers of superior (or equivalent) performance to the conventional classifiers examined, suggesting removal of the need to exhaustively evaluate a large family of conventional classifiers on any new problem.

Index Terms—Feature Extraction, Multiobjective Optimization, Genetic Programming, Pattern Recognition

I. INTRODUCTION

Despite its prominence in the field of pattern recognition up to the 1970s, the area of feature extraction - also termed feature construction - together with the related area of feature selection, has been largely overtaken by work on classifier design, principally neural networks. Indeed many elegant theoretical results have been obtained in the classification domain in the intervening years. Nonetheless, feature extraction retains a key position in the field since the performance of a pattern classifier is well-known to be enhanced by proper preprocessing of the raw measurement data – this topic is the main focus of the

Manuscript received 23 March 2006. Yang Zhang is with the Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, S1 3JD, UK (phone: 44-114-222-5436; e-mail: yang.zhang@shef.ac.uk). Peter I Rockett is with the Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield, S1 3JD, UK (e-mail: p.rockett@shef.ac.uk).
Fig. 1 shows a prototypical pattern recognition system in which a vector of raw measurements is mapped into a decision space. Often the feature selection and/or extraction stages are either omitted or are implicit in the recognition paradigm – a multi-layer perceptron is a good example of a classification paradigm where a distinct feature extraction stage is not readily identifiable. Addison et al. [1] and Park et al. [2] have reviewed existing feature extraction and selection techniques while Guyon & Elisseeff [3] have discussed feature extraction in terms of filter and wrapper methods. In this paper we focus principally on feature extraction (although there is some overlap with the related topic of feature selection to which we return in later sections).

The principal difficulty with designing the feature extraction stage of a classifier is that it usually requires deep domain-specific knowledge. (Indeed much of the work in image processing on detecting image cues such as edges and corners is actually feature extraction.) Even for feature extractors designed by domain experts, the issue of optimality is rarely addressed. Ideally, we would require some measure of class separability in the transformed decision space to be maximized but with hand-crafted methods this is hard to guarantee.

In general terms, finding the optimal (possibly non-linear) transformation, \( x \rightarrow y \) from input vector, \( x \) to the decision space vector, \( y \) where \( y = f(x) \) is a challenging task. In the sense that the feature extraction preprocessing stage is a transformation or mapping from input space to decision space, for a given classification problem we seek the mapping which maximizes the separability of the classes in decision space. Thus feature extraction
can be regarded as finding an optimal sequence of operations subject to some criterion. Genetic programming (GP) is an evolutionary problem-solving method which has been extensively used to evolve programs or sequences of operations [4]. Typically, a prospective solution in GP is represented as a parse tree which can be straightforwardly interpreted as a sequence of operations and thus evaluated. Indeed, GP has been used before to optimize feature extraction and selection [5, 6].

Ebner [7, 8] has evolved image processing operators using GP. Bot [9] has used GP to evolve decision space features, adding these one-at-a-time to a k-NN classifier if the newly evolved feature improved the classification performance by more than a certain amount. Bot’s approach is a greedy algorithm and therefore almost certainly sub-optimal. In addition, Koza [4] has produced character detectors using genetic programming while Tackett [10] evolved a symbolic expression for image classification based on image features.

Harvey et al. [11] evolved pipelined image processing operations to transform multi-spectral input synthetic aperture radar (SAR) image planes into a new set of image planes and a conventional supervised classifier was used to label the transformed features. Training data were used to derive a Fisher linear discriminant and GP was applied to find a threshold to reduce the output from the discriminant-finding phase to a binary image. However, the discriminability is constrained in the discriminant-finding phase and the GP only used as a one-dimensional search tool to find a threshold.

Sherrah et al. [12] proposed an Evolutionary Pre-Processor (EPrep) system which used GP to evolve a good feature mapping by minimizing misclassification error. Three typical classifiers: generalized linear machine (GLIM), k-nearest neighbor (k-NN) and maximum likelihood classifiers were selected randomly and trained in conjunction with the search for the optimal feature extractors. The misclassification errors on the validation set from those classifiers were used as a fitness value for the individuals in the evolutionary population. The same procedure was used in the co-evolution of feature extraction/classifiers in [13].
This approach, however, makes the feature extraction procedure dependent on the classifier in an opaque way such that there is a potential risk that the evolved preprocessing can be excellent but the classifier can be poor giving a poor overall performance, or vice versa.

Kotani et al. [6] used GP to determine the polynomial combination of raw features to be fed into a $k$-NN classifier and reported an improvement in classification performance. Krawiec [14] constructed a fixed-length decision vector using GP proposing an extended method to protect ‘useful’ blocks during the evolution. This protection method, however, contributes to the overfitting which is evident from his experiments. Indeed, Krawiec’s results show that for some datasets, the application of his feature extraction method actually produces worse classification performance than using the raw input data alone. Recently, Guo et al. [15] have evolved features in a condition monitoring task although it is not clear whether the elements in the vector of decision variables were evolved at the same time or hand selected after evolution. Smith and Bull [16] have used GP together with a GA to perform feature construction and feature selection.

Broadly, the previous work on GP feature extraction can be categorized as evolving either: A discrete feature extraction stage which then feeds into a traditional classifier, or evolving a combined feature extraction/classification method which directly outputs a class label. Of the two possible routes, we argue that there is little merit in investing computational effort in evolving classifiers since this area is well understood and has solid theoretical underpinnings. We argue that the available computational effort should be expended on producing good feature extraction; in addition, we question the speed of convergence when exploring a search space which contains not only the set of feature extractors but also the set of all classifiers. Consequently, we adopt the approach here of evolving optimal feature extraction algorithms and performing the classification task using a standard, simple and fast-to-train classifier since the classifier has to be included inside
the evolutionary loop to evaluate an individual’s fitness in terms of a separability measure in the decision space. We draw a clear distinction in the present work between evolving a feature detection stage and evolving a classifier since the outcome of our evolutionary optimization is a mapping into a real-valued (1D) decision space, not a mapping into the space of object labels which is what would result from evolving a classifier. Clearly, our overall ‘system’ does constitute a classification system and our feature extraction stages are conditioned on our (somewhat arbitrary) choice of classifier - in the present case, a single threshold. As a future extension to the present framework, we envisage mapping the input patterns into a multi-dimensional decision space (see [12], for example) in which case the choice of classifier is explicitly much more open.

It is noteworthy that all the previous work on evolving feature extractors/classifiers by GP have used a single objective, typically minimizing the classification error over a training set which is disadvantageous from a number of standpoints. In particular, unless specific measures are taken to prevent it, the trees in a GP optimization tend to grow in size, a phenomenon which is termed tree bloat. This is analogous to overfitting in neural networks and can lead to poor generalization of the trained classifier as well as excessive computational demands. Various heuristic and indirect techniques have been used to suppress bloat but Ekárt & Németh [17] have shown that using a multiobjective fitness function [18] within GP, where one of the objectives is to minimize tree size, prevents bloat by exerting selective pressure in favor of smaller trees; also see [19]. We have thus used a multiobjective framework with Pareto optimality [18] in the present work - as far as the authors are aware, the present paper is the first to report the application of multiobjective genetic programming to the process of designing optimal feature extraction.

Rather than a single solution, the converged output of multiobjective optimization is a set of equivalent solutions whose members are superior to all the other feasible solutions; the members of this so-called Pareto set are said to dominate the other possible solutions
Within this set, none can be considered ‘better’ than any other from the point of view of the simultaneous optimization of multiple objectives and it is left to a Decision Maker (DM) to select one of the optima according to some utility function which expresses their preferences. See [18, 20] for a detailed review of multiobjective evolutionary methods.

Our overall objective in the present work has been to identify the (near-)optimal series of mathematical transformations to pattern data that produces the best class separation in the transformed decision space. Further, our aim has been to produce a generic, domain-independent method for generating optimal feature transformations such that the transformed patterns (or extracted features) can then be accurately classified with a simple and fast classifier. We make no assumptions about the statistical distributions of the original feature data.

For convenience and without loss of generality, we focus here on two-class problems. In common with other approaches to multi-class problems, extension to three or more classes is somewhat more involved and will be the subject of a future publication.

The rest of this paper is organized as follows: We present our generic framework to evolve optimal feature extractors in Section II and demonstrate its utility in Section III by applying it to eight datasets from the UCI Machine Learning [21] and StatLog [22] datasets. We make comparison with nine popular classifiers as well as previous evolutionary results reported by other researchers. We offer conclusions and suggestions for future work in Section IV.

II. METHODOLOGY

Over the years much effort has been expended in the pattern recognition community on finding a ‘best’ classifier (e.g. [22, 23]), the conclusion of which is that there is no single classifier which is best for every problem. In the present work, we focus on the feature preprocessing stage in classification systems. We propose a generic framework to evolve an
optimal feature extraction stage for a given problem, independent of the dimensionality of the input pattern space and with optimized discriminability. If the optimization is effective, patterns in the transformed space should not only be much easier to separate than the original pattern data but should also be ‘optimal’ - within the constraints imposed by the dimensionality of the decision space and the final classifier. By “optimality”, we mean here that our algorithm should, in principle, yield a classification performance which is comparable to the best of the class of all classifiers, thus doing away with the often lengthy process of determining which is the best classifier for a specific classification task. Furthermore, the framework under discussion is not problem-specific and can be easily reused in other domains without or with simple modifications to the GP settings.

A. Multiple Objectives

In terms of implementation, we map the input space to a 1-dimensional decision variable since this is a natural and straightforward operation for GP. Within the multiobjective framework, we have used a three-dimensional fitness vector of objectives comprising: Tree complexity, misclassification error and Bayes error, as follows:

1) Tree complexity measurement

As pointed-out above, there is a danger that trees evolved by GP will become very large due to tree bloat. We have observed in early experiments without a complexity objective that huge trees could produce an extremely small error over the training set but a very poor error estimated over an independent validation set. Broadly, for a given training error, the simpler individual is preferred - an observation which is in accord with Ockham’s Razor. Thus we have used node count in the tree as a straightforward measure of tree complexity to be one of our fitness vector elements driving the evolution. We thereby impose a selective pressure that favors small trees, all other things being equal.
2) Misclassification error

The second element we use in the fitness vector is the conventional one of the fraction of misclassified patterns counted over the training set, the so-called 0/1 loss. Since we are projecting the input pattern into a one-dimensional decision space we use a straightforward, single threshold classifier where the threshold is adapted as part of the fitness value determination to give the minimum error. Taken over the whole training set, we find the optimal threshold for that particular mapping by performing a golden section search for the threshold value which minimizes the misclassification rate, bracketed initially by the two extremal responses. The golden section search is terminated when there is no further improvement in the misclassification rate. Thus the training of the classifier is very fast and makes a negligible contribution to the time of a single iteration. The misclassification objective means we are trying to evolve a feature extractor which maps the original pattern space into a new feature space where thresholding is able to yield the smallest possible misclassification.

3) Bayes Error

The use of appropriate fitness functions is critical to the search performance of all evolutionary algorithms - an inappropriate fitness function can seriously mislead the evolution and this has motivated our use of the misclassification error above. In addition to the misclassification error, we have also used an estimate of the Bayes error as the third and final fitness objective. The set of \( n \)-dimensional input patterns from each class project into the 1D decision space forming two class-conditioned probability density functions (PDFs) in the decision space which we histogram. We straightforwardly estimate the Bayes error from the overlap of the two histograms of the class-conditioned PDFs in the projected decision space - see, for example, Fukanaga [24] for further details. The Bayes error is a fundamental lower bound on classification performance in the decision space, dependent solely on the class-conditioned densities and independent of the classifier.
The use of two measures of classification error – the Bayes error estimate and the fraction of misclassified training set patterns requires explanation. During our early experiments we found that using only the fraction of misclassified patterns (and the tree complexity measure) resulted in slow convergence and some cases, a failure to converge altogether. In the initial phases of the evolution when all the randomly generated members of the population were typically poor performers, the misclassification error was close to its maximum. Consequently this objective lacked the sensitivity to identify individuals with slightly more promise than others, hence the poor convergence. In view of this we experimented with the alternative Bayes error measure.

The use of the Bayes error estimate alone (with the tree complexity measure) allowed the evolutions to converge rapidly but the subsequently estimated validation error was disappointingly high. On closer inspection, it became clear that although the Bayes error objective was minimized over the training set, the GP was often opportunistically achieving this goal by producing two transformed class-conditioned densities in the decision space with non-coincident ‘comb’-like structures rather than the desired end of two compact densities with widely separated means [25]. Consequently, although the degree of overlap of the likelihoods from the training set was small, the misclassification error calculated over the validation set was large. This led us to using two error measures: the Bayes error allows the evolutionary search to make rapid progress in the initial stages of the optimization while the fraction of misclassified patterns eventually comes to the fore when the evolution advances to a certain stage of maturity and leads to two well-separated distributions.

Overall, we have found experimentally that the combination of the three objectives is necessary for the algorithm to rapidly generate a Pareto set of parsimonious solutions which generalize well. Without the complexity measure the trees bloat and the optimization tends to stagnate. Without the Bayes error objective, convergence is very
slow or non-existent. Each of the three objectives thus has a key role to play during the evolutionary process although since we are ultimately considering the classification domain, after we have generated a set of non-dominated solutions whose properties are ‘shaped’ by the multiple objectives, we select the solution which has the lowest (mean) validation error. This is a critical distinction between the current area and most other uses of multiobjective optimization: The multiple objectives are vital constraints during the evolutionary process but after convergence, the nature of the problem does not actually form a conventional multiobjective trade-off.

B. MOGP Implementation

A number of multiobjective evolutionary algorithms have been proposed in the past: e.g. SPEA-2 [19], MOGA [26], NSGA-II [27] and MOGLS [20]. How to make quantitative comparisons between multiobjective stochastic optimizers is still very much an open research issue [28] with no clear cut outcomes. Nonetheless, strength Pareto methods have shown good performance when set alongside other multiobjective evolutionary algorithms and we have used SPEA-2 [29] in the present work to approximate the Pareto-optimal set for our multiobjective optimization problem, thereby searching for the optimal sequence of transformations which map input patterns to decision space. (Other, more recent work [25], however, suggests that GP, as opposed to GA, methods based on the steady-state Pareto converging genetic algorithm (PCGA) [30] may confer some benefits for the present application in that PCGP appears able to produce smaller trees for a given misclassification error. The issue of comparisons between multiobjective GP algorithms is an area for future work.)

The SPEA-2 genetic programming implementation used here is a generational strategy in which two sets of individuals are maintained during evolution. One set represents the current population and the second contains the current approximation to the Pareto front. The ranking is done by calculation of the strength, or fitness of each individual in both
sets. When calculating the strength of an individual, we use the method proposed in SPEA-2 [19]. Here we make some modifications and reuse some strategies from SPEA/SPEA-2 to operate with genetic programming although we still store the non-dominated individuals in an external set, and cluster, if necessary.

Using modified binary tournament selection we randomly select two parent trees for breeding from the union of the population and the non-dominated set. If both trees have been drawn from the same set we compare the normalized fitnesses to determine a winner; if not, we use the raw fitnesses to decide which should be chosen.

We use non-destructive, depth-dependent crossover [31] and mutation operators in order to avoid the breaking of building blocks. We choose a sub-tree biased in its complexity (i.e. the number of nodes) using the depth-fair operator [31]; one of the sub-trees at the chosen depth is then picked by roulette wheel selection, biased in its complexity. We initiate both genetic operations by randomly choosing a depth, \( d \in [0, K \max] \) in a tree. At the given depth, \( d \) there are \( N_d \) sub-trees, each comprising \( M_1, M_2, \ldots, M_N \) nodes, respectively. The probability of selecting the \( i \)-th sub-tree is given by:

\[
Pr(i \mid d) = M_i \sum_{j=1}^{N_d} M_j
\]

and we select the target sub-tree using the standard roulette wheel approach. In the crossover operation we exchange the selected sub-trees between the parents. In mutation, the selected sub-tree is replaced by a new, randomly created sub-tree attached at the selected mutation point. See [31] for further details.

We retain only those offspring which dominate either of their parents and in this way, we were able to maintain diversity in the population while avoiding being trapped in local minima in the early stages. The parameters used in the GP implementation are listed in Table I and the function set used in the trees is detailed in Table II.
TABLE I - GP SETTINGS

<table>
<thead>
<tr>
<th>Terminal set</th>
<th>Input pattern vector elements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 floating point numbers $\in {-0.0, \ldots, 1.0}$</td>
</tr>
<tr>
<td>Function set</td>
<td>See Table II</td>
</tr>
<tr>
<td>Standardized fitness</td>
<td>Strength-based fitness</td>
</tr>
<tr>
<td>Population size</td>
<td>500</td>
</tr>
<tr>
<td>Original population</td>
<td>50% full trees, 50% random trees</td>
</tr>
<tr>
<td>Original tree depth</td>
<td>5</td>
</tr>
<tr>
<td>Max. number of generations</td>
<td>500</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>30%</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>70%</td>
</tr>
</tbody>
</table>

The evolution was terminated when one of the following criteria was met: The misclassification error was zero, meaning all patterns in training set are correctly labeled, OR the maximum number of generations was exceeded, OR the Bayes error of the best individual does not improve for some fraction, $f$ of the maximum number of generations; we have used $f = 0.04$ here on the basis of experience.

TABLE II - TREE NODE TYPES

<table>
<thead>
<tr>
<th>Function Nodes</th>
<th>Type</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>sqrt</td>
<td>UNARY</td>
<td>Calculates the square root</td>
</tr>
<tr>
<td>log</td>
<td>UNARY</td>
<td>Calculates the natural logarithm</td>
</tr>
<tr>
<td>pow2</td>
<td>UNARY</td>
<td>Calculates value raised to power of 2</td>
</tr>
<tr>
<td>-</td>
<td>UNARY</td>
<td>Calculates negative value</td>
</tr>
<tr>
<td>sin</td>
<td>UNARY</td>
<td>Calculates sine of the value</td>
</tr>
<tr>
<td>-</td>
<td>BINARY</td>
<td>Subtracts left value from right value</td>
</tr>
<tr>
<td>+</td>
<td>BINARY</td>
<td>Adds left value with right value</td>
</tr>
<tr>
<td>*</td>
<td>BINARY</td>
<td>Multiplies left value by right value</td>
</tr>
<tr>
<td>/</td>
<td>BINARY</td>
<td>Protected division of left value by right value</td>
</tr>
<tr>
<td>max</td>
<td>BINARY</td>
<td>Returns the greater of the two values</td>
</tr>
<tr>
<td>min</td>
<td>BINARY</td>
<td>Returns the smaller of the two values</td>
</tr>
<tr>
<td>if-else</td>
<td>TERNARY</td>
<td>Returns 2nd value if 1st value $\geq 0$, otherwise returns the 3rd value</td>
</tr>
</tbody>
</table>

III. UCI MACHINE LEARNING AND STATLOG PROJECT DATASETS

In this section we address our guiding issue of producing a generic methodology by
examining performance across a wide range of two-class classification problems from the UCI Machine Learning [21] and Statlog [22] databases. Since GP is able to synthesize a feature extraction stage which is (near-)optimal with respect to the learning task at hand, we conjecture that the classification performance of our method should, at worst, be identical to the very best conventional classifier on any given problem. Koza et al. [32] have discussed the potential of GP to invent new solutions to established problems. As part of our conjecture, we suggest that our methodology is ‘inventing’ a near-optimal classifier for every dataset to which it is applied; in some instances these evolved classifiers may be similar to existing classifiers and in other cases, quite unlike any known classifier paradigm. The key issue is that the generation of the feature extraction stage is being driven by the notion of optimality.

We consider an extensive set of comparisons across eight 2-class learning problems. For each dataset we make a statistical comparison of the classification performance between our MOGP algorithm and a range of established classifiers. If our conjecture about the generic power of our method is supported, then MOGP should perform at least as well as any other classifier (and in a number of cases, better). In addition, we compare – where possible – with previously reported evolutionary feature extraction techniques.

A. The UCI & StatLog Datasets

The datasets used in the current work are:

1. Glass - 163 instances with nine attributes - This dataset has been converted to a two-class problem by seeking to distinguish between float glass and non-float glass.

2. BUPA Liver Disorders – Prediction of whether a patient has a liver disorder.
   There are two classes, six numerical attributes and 345 records.

3. Wisconsin Diagnostic Breast Cancer - This dataset has been discussed before by
Mangasarian et al. [33]. 569 examples with thirty numerical attributes.

4. Pima Indians Diabetes (PID) - Records with missing attributes were removed. This dataset comprises 532 complete examples with seven attributes.

5. Wisconsin Breast Cancer (WBC) - Sixteen of the instances with missing values were removed; 683 out of original 699 instances have been used here. Each record comprises ten attributes. This dataset has been used previously in [34].

6. Australian credit approval (AUS) - Credit card applications; comprises 690 instances in 14 attributes. 55.5% instances from positive decisions. This dataset has previously been investigated by Quinlan using decision trees [35].

7. Heart disease (HEA) - Contains 13 attributes, 270 samples. 120 samples present heart disease.

8. German credit (GER) - Classifies people described by 24 attributes as good or bad credit risks. 1000 instances, 700 of which are in good credit condition.

For convenience, the details of the datasets used in the current work are summarized in Table III.

<table>
<thead>
<tr>
<th>Name</th>
<th>Features</th>
<th>Size and Distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>9</td>
<td>163 = 87 (float) + 76 (non-float)</td>
</tr>
<tr>
<td>BUPA</td>
<td>6</td>
<td>345 = 200 (Benign) + 145 (Malignant)</td>
</tr>
<tr>
<td>WDBC</td>
<td>30</td>
<td>569 = 357 (Benign) + 212 (Malignant)</td>
</tr>
<tr>
<td>PID</td>
<td>7</td>
<td>532 = 355 + 177 (Diabetic)</td>
</tr>
<tr>
<td>WBC</td>
<td>10</td>
<td>699 = 458 (Benign) + 241 (Malignant)</td>
</tr>
<tr>
<td>AUS</td>
<td>14</td>
<td>690 = 383 (Positive) + 307</td>
</tr>
<tr>
<td>HEA</td>
<td>13</td>
<td>270 = 150 (Healthy) + 120 (Diseased)</td>
</tr>
<tr>
<td>GER</td>
<td>24</td>
<td>1000 = 700 (Good) + 300 (Bad)</td>
</tr>
</tbody>
</table>

B. Classification Algorithms

As a basis for comparison with MOGP, we have used nine existing classification
algorithms. The implementations used were all taken from the Weka machine learning system\textsuperscript{1} [36] and we used the default parameter settings except where noted below. The classifiers used were:

1. Radial Basis Functions (RBF) – A normalized Gaussian radial basis function network using the $k$-means clustering algorithm. We estimated the number of clusters ($k$) for a given dataset by considering a random split of the dataset, training the classifier on the first half and calculating a validation error on the second half. We adopted the value of $k$ which gave the lowest validation error for each dataset by this method.

2. Logistic – Modified multinomial logistic regression model with a ridge estimator.

3. NNge – Nearest-neighbor-like algorithm using non-nested generalized exemplars.


5. IB1 – Instance-based learning algorithm. Uses a simple distance measure to find the training instance closest to the given test instance and predict the same class as this training instance.

6. ADTree – The alternating decision tree learning algorithm.

7. SMO – Sequential minimal optimization algorithm for training a support vector classifier.

8. C4.5 – The well-known decision tree algorithm. (This is referred to as J48 in Weka.)

In addition, we have used the classical Fisher Linear Discriminant since comparative studies [23] show that this classifier is competitive across a wide range of datasets. The version used here calculates the threshold assuming equal covariance Gaussian classes with a correction for the priors [37]; the priors were estimated from the dataset in question.

\textsuperscript{1} See: http://www.cs.waikato.ac.nz/\~ml/weka/. We have used Version 3.4.5 of Weka in this work.
C. Comparison Methodology

Conventionally, classifiers have been compared in the literature using \( N \)-fold cross-validation followed by a \( t \)-test to gauge the statistical difference between the results. Dietterich [38], however, has pointed-out this is unsound due to the implicit assumptions about independence being violated and has proposed an empirical \( 5 \times 2 \ cv \) test. In turn, Alpaydin [39] has modified Dietterich’s test to remove the unsatisfactory aspect of the result depending on the ordering of the folds: it is Alpaydin’s \( F \)-test which we use here to statistically compare classifier performance.

To compute the Alpaydin \( F \)-statistic we perform five repetitions of splitting the dataset into two folds, treating each fold as a training set and the other as the test set. We then compute the \( F \)-statistic (see [39] for full details) and use this figure to decide whether or not to reject the null hypothesis that the performances of the two classifiers are identical. Throughout this work we have used a 95\% confidence level to infer a statistical difference which is equivalent to an \( F \)-measure \( \geq 4.74 \).

D. Comparisons with Conventional Classifiers

The outcome of the multiobjective optimization is an approximation to the Pareto set, although as we explain in Section II.A, the classification domain does not constitute a trade-off in the conventional multiobjective sense. (Alternatively, the utility function which expresses the Decision Maker’s preferences [20] considers only the validation error to the exclusion of the other two objectives.) Therefore, to effect comparison of the classifier algorithms we have calculated the mean validation errors over the ten folds of the \( 5 \times 2 \ cv \) test for each classifier, both evolutionary and conventional.

The mean test errors for all ten classification algorithms across the eight datasets are summarized in Table IV. Reassuringly, the MOGP returns the lowest mean error for each dataset apart from the LOG/GER pairing. The results in Table V need to be treated with
some caution since the *statistical significance* of these differences is not clear from this table. In order to quantitatively compare statistical significance of the cross-validation experiments we have computed the Alpaydin $F$-statistic as described in the previous sub-section for our MOGP algorithm paired with every other comparator classifier. The comparisons are summarized in Table V where a tick in a cell represents superiority of MOGP over the given classifier on the given dataset whereas a dash denotes no statistical difference between MOGP and its comparator. Taken over the datasets considered here, MOGP is consistently superior to radial basis function (RBF), C4.5 and Fisher’s linear discriminant (FD). At the other extreme, the logistic regression (LOG) algorithm is only bettered by MOGP on two datasets (Glass and BUPA) and not statistically different on the remaining six. Highly significantly, MOGP is not out-performed by any of the comparator classifiers on any of the examined datasets.

At the start of this Section we conjectured that the MOGP optimizations would yield classifiers which were either better than, or at worst, statistically identical to the best performing algorithm among the class of all classifiers. Clearly we are not able to prove such a conjecture as this would involve testing MOGP against the universe of all possible datasets with every possible classifier (including those as yet undiscovered). Nonetheless, we argue that the results presented here constitute strong evidence to support our conjecture.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Classifiers</th>
<th>RBF</th>
<th>LOG</th>
<th>NNge</th>
<th>Bayes Net</th>
<th>IB1</th>
<th>ADTree</th>
<th>SMO</th>
<th>C4.5</th>
<th>FD</th>
<th>MOGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td></td>
<td>0.354</td>
<td>0.364</td>
<td>0.322</td>
<td>0.311</td>
<td>0.300</td>
<td>0.317</td>
<td>0.392</td>
<td>0.338</td>
<td>0.510</td>
<td>0.227</td>
</tr>
<tr>
<td>BUPA</td>
<td></td>
<td>0.442</td>
<td>0.383</td>
<td>0.449</td>
<td>0.485</td>
<td>0.421</td>
<td>0.335</td>
<td>0.452</td>
<td>0.391</td>
<td>0.434</td>
<td>0.264</td>
</tr>
<tr>
<td>PID</td>
<td></td>
<td>0.255</td>
<td>0.233</td>
<td>0.279</td>
<td>0.249</td>
<td>0.299</td>
<td>0.258</td>
<td>0.234</td>
<td>0.316</td>
<td>0.336</td>
<td>0.205</td>
</tr>
<tr>
<td>WBC</td>
<td></td>
<td>0.048</td>
<td>0.045</td>
<td>0.038</td>
<td>0.026</td>
<td>0.042</td>
<td>0.051</td>
<td>0.030</td>
<td>0.057</td>
<td>0.932</td>
<td>0.021</td>
</tr>
<tr>
<td>WDBC</td>
<td></td>
<td>0.061</td>
<td>0.068</td>
<td>0.077</td>
<td>0.054</td>
<td>0.046</td>
<td>0.052</td>
<td>0.030</td>
<td>0.067</td>
<td>0.364</td>
<td>0.026</td>
</tr>
<tr>
<td>AUS</td>
<td></td>
<td>0.171</td>
<td>0.132</td>
<td>0.162</td>
<td>0.151</td>
<td>0.200</td>
<td>0.141</td>
<td>0.145</td>
<td>0.142</td>
<td>0.146</td>
<td>0.126</td>
</tr>
<tr>
<td>HEA</td>
<td></td>
<td>0.170</td>
<td>0.152</td>
<td>0.233</td>
<td>0.170</td>
<td>0.248</td>
<td>0.222</td>
<td>0.163</td>
<td>0.233</td>
<td>0.222</td>
<td>0.144</td>
</tr>
<tr>
<td>GER</td>
<td></td>
<td>0.269</td>
<td>0.231</td>
<td>0.270</td>
<td>0.277</td>
<td>0.322</td>
<td>0.273</td>
<td>0.236</td>
<td>0.261</td>
<td>0.276</td>
<td>0.248</td>
</tr>
</tbody>
</table>
TABLE V - F-Test Comparison Between Algorithms For Each Dataset At 95% Confidence Level. A tick represents superiority of MOGP for that comparator classifier/dataset combination; a dash denotes no statistical difference.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>RBF</th>
<th>LOG</th>
<th>NNge</th>
<th>BayesNet</th>
<th>IB1</th>
<th>ADTree</th>
<th>SMO</th>
<th>C4.5</th>
<th>FD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUPA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WBC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WDBC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HEA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E. Comparisons with Other Evolutionary Feature Detectors

As mentioned in Section I, a number of other workers have explored (single objective) genetic programming to evolve either feature detectors or overall classifier systems. Muni et al. [40] have used GP to produce a c-class classifier (as distinct from a feature extraction stage) using a multi-tree representation. Bot [9] evolved new features which he added one-at-a-time until the improvement in classification performance dropped below a threshold. Bot & Langdon [41] produced linear classification trees using strongly typed GP. Loveard & Ciesielski [42] have explored five strategies for evolving classifiers while Krawiec [14] also constructed features using GP for subsequent classification with the C4.5 decision tree.

The error rates, where known, from these earlier studies are summarized in Table VI. Typically, error rates were the means estimated over ten-fold cross-validation. The results of Bot & Langdon [41] are the mean validation error of the best individual of 30 runs while those for Smith and Bull [16] are the best result from twenty repetitions with a 90%-10% partitioning of the dataset into training and test sets, respectively. In the case of the data of Loveard & Ciesielski [42], we show the most favorable results from the five strategies.
investigated by these authors. Although in every case our MOGP method yields the lowest error rate, we are unable to assess the statistical significance of the differences in error rates on the basis of the published information. Nonetheless, the fact that MOGP records the lowest error rates is very promising and implies that MOGP is at very worst, equally good as the comparator evolutionary methods and quite possibly superior.

F. Interpretation of the Generated Trees

Figs. 9 to 13 show the trees which display the smallest validation error on each dataset from a single run; the generation number at which they were generated is also shown. (We are concerned only with the lowest 0/1 loss; the complexity and Bayes error objectives are important shaping forces during the evolution but do not form part of any ultimate design trade-off.) The terminal nodes displayed in the following tree graphs are labeled \( X_n \) where \( n \in [1..N] \) denotes an element in the input pattern vector.

<table>
<thead>
<tr>
<th>Training algorithm</th>
<th>AUS</th>
<th>Glass</th>
<th>BUPA</th>
<th>PID</th>
<th>WBC</th>
<th>WDBC</th>
<th>GER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muni [40]</td>
<td>-</td>
<td>-</td>
<td>0.3007</td>
<td>-</td>
<td>0.0281</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Bot [9]</td>
<td>0.169</td>
<td>0.4800</td>
<td>0.4160</td>
<td>0.3050</td>
<td>-</td>
<td>-</td>
<td>0.37</td>
</tr>
<tr>
<td>Bot &amp; Langdon [41]</td>
<td>-</td>
<td>0.368</td>
<td>-</td>
<td>0.250</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Krawiec [14]</td>
<td>-</td>
<td>0.3361</td>
<td>-</td>
<td>0.2359</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Loveard [42]</td>
<td>-</td>
<td>-</td>
<td>0.308</td>
<td>0.242</td>
<td>0.032</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Smith &amp; Bull [16]</td>
<td>-</td>
<td>-</td>
<td>0.3403</td>
<td>0.265</td>
<td>0.0437</td>
<td>0.0438</td>
<td>-</td>
</tr>
<tr>
<td>MOGP</td>
<td>0.126</td>
<td>0.2271</td>
<td>0.2644</td>
<td>0.2057</td>
<td>0.02634</td>
<td>0.026</td>
<td>0.248</td>
</tr>
</tbody>
</table>

It is noteworthy that these ‘optimal’ feature transformations are all non-linear which is in contrast to traditional feature extraction methods which emphasize linear projections. In most cases it is difficult to identify any underlying rationale the GP is employing for the transformations although it is interesting to remark on the extreme case of the Pareto set
for the WBC dataset which contains one non-dominated solution comprising a single leaf node, the Marginal Adhesion attribute [21]. If we examine the values of this attribute across the dataset, 90.09% of the benign patterns have a Marginal Adhesion value less than 3. In comparison, 78.24% of the malignant patterns have this attribute set greater than or equal to 3. In which case, if we judge malignancy based on Marginal Adhesion attribute alone thresholded at 3, we will obtain an error of 0.1399 assuming equal costs\(^2\). This is to be compared with the error of 0.02634 obtained from the more complex tree shown in Fig. 12 which has a node count of 37. Nonetheless, it appears that the GP is eliciting sensible structure from this dataset – when (effectively) posed with the question of which is the best variable to use if constrained to use just a single leaf, it correctly selects the most discriminatory.

![Fig. 9. MOGP transformation evolved for the PID dataset at generation 52.](image)

Although one of our multiple objectives has been tree size, used to suppress tree bloat, a number of the trees are not of the absolute minimum size and contain a few redundant sub-trees. For example, Fig. 12 contains the sub-trees: \(\max(0.5, 0.8)\) which, of course,

\(^2\) In medical diagnosis, equal costs are, of course, generally unacceptable; we use them here for convenience of statistical comparison.
always returns the value of 0.8 as well as the sub-tree \( \text{if}(0.1 \geq 0) \ then \ X6 \ else \ X6 \) which always return the value of X6. We present the trees in Figs. 9 - 13 unedited since these are what have been generated by the evolutionary algorithm. It is clear, however, that these are not completely optimal in that the identical classification performance could be obtained in some cases with slightly smaller trees. Nonetheless, the work presented here produces near-optimal trees which are reasonable for a stochastic search method such as genetic programming and an advance on previous work on feature extraction. In practice, any redundant sub-trees could be easily removed from the final solution by hand.

![Diagram](image)

Fig. 10. MOGP transformation evolved for the Glass dataset at generation 125.

IV. CONCLUSIONS & FUTURE WORK

In this paper we have demonstrated the use of multi-objective genetic programming (MOGP) to evolve an ‘optimal’ feature extractor which transforms input patterns into a decision space such that class separability is maximized. In the present work we have projected the input pattern to a one-dimensional decision space since this transformation
naturally arises from a genetic programming tree although potentially, superior classification performance could be obtained by projecting into a multi-dimensional decision space [12] - this is currently an area of active research. What arises from our method is a family of equivalent solutions - the Pareto set - which simultaneously present the decision maker with the trade-off surface between training error and the complexity of the feature extractor; in the classification domain we will generally prefer the solution with the smallest 0/1 loss although this is not a necessary constraint. Although we effectively ignore two of the three objectives in selecting a final solution, the complexity and Bayes error objectives have a crucial role in rapidly shaping compact - and by implication, well-generalizing - solutions.

![Fig. 11. MOGP transformation evolved for the BUPA liver disorders dataset at generation 176](image)

A major objective in this work has been to produce a domain-independent method and we have applied our algorithm to five machine learning tasks from the UCI database and three Statlog datasets. In comparison with a number of other representative classifier paradigms, the performance of our MOGP method turns-out to be better (or at worst, not
statistically different). In no case was MOGP bettered by a conventional classifier which supports our conjecture that GP is finding the (near-)optimal feature extraction stage for a given classification problem.

Fig. 12. MOGP transformation evolved for the WBC dataset at generation 98.
The use of multiple objectives, particularly multiple classification error objectives has been shown to be effective in guiding and speeding the optimization. It is interesting that when only the Bayes error objective was used, GP was able to meet its goals of minimizing the overlap of the two likelihoods in a way which was both unintended and unwanted. Clearly the intuitively straightforward concept of discriminability needs to be very carefully framed for use in an evolutionary setting to avoid the generation of opportunistic and unhelpful solutions.

The node number objective employed penalizes an individual according to its complexity. This appears to be essential both in order to prevent tree bloat as well suppressing over-fitting of the training set leading to poor generalization. We believe that combined with the non-destructive, depth-dependent crossover [31] and mutation operators used, the bloating problem has been largely overcome.

![MOGP transformation evolved for the WDBC dataset at generation 85.](image)

Although we have treated only binary classification problems in this paper, extension to multiple classes is a logical development and is currently underway. Extension to large
datasets is also a key issue where stochastic sub-sampling [43] may well prove a fruitful avenue.

ACKNOWLEDGMENTS

One of us (YZ) is grateful for the financial support of a Universities UK Overseas Research Student Award Scheme (ORSAS) scholarship and the Henry Lester Trust.

REFERENCES


