Feature Selection and Classification in Noisy Epistatic Problems using a Hybrid Evolutionary Approach

Drew DeHaas
Dept. of Computer Science
University of Vermont
Burlington, VT 05405
Drew.DeHaas@uvm.edu

Jesse Craig
Dept. of Computer Science
University of Vermont
Burlington, VT 05405
JesseCraig@alumni.uvm.edu

Colin Rickert
Dept. of Computer Science
University of Vermont
Burlington, VT 05405
Colin.Rickert@uvm.edu

Paul Haake
Dept. of Computer Science
University of Vermont
Burlington, VT 05405
Paul.Haake@uvm.edu

Margaret J. Eppstein
Dept. of Computer Science
University of Vermont
Burlington, VT 05405
(802) 656-1918
Maggie.Eppstein@uvm.edu

ABSTRACT

A hybrid evolutionary approach is proposed for the combined problem of feature selection (using a genetic algorithm with Intersection/Union recombination and a fitness function based on a counter-propagation artificial neural network) and subsequent classifier construction (using strongly-typed genetic programming), for use in nonlinear association studies with relatively large potential feature sets and noisy class data. We had the dual goals of (a) accurately classifying new individuals based on feature values, and (b) estimating the nonlinear functional relationship of a subset of predictive features and class. The method was tested using synthetically generated mixed-type datasets with various degrees of injected noise, based on a proposed mental health database. Optimal (or nearly optimal) feature sets were accurately identified most of the time, even in the presence of significant amounts of noise in the class data. When presented the selected features, genetic programming consistently outperformed a counter-propagation neural network as a classifier, and reconstructed the correct underlying nonlinear expression associating features with class, even with noisy class data. The results show that the proposed hybrid algorithm has good potential for feature selection, classification, and estimation of the form of even relatively weak nonlinear relationships between features and class.

Categories and Subject Descriptors

I.2.8 Artificial Intelligence [**Problem Solving, Control Methods and Search**]: *Heuristic Methods*

General Terms

Algorithms, Performance, Design, Experimentation

Keywords

Classification, feature selection, data mining, hybrid evolutionary algorithms, genetic algorithms, genetic programming.

GECCO '07 July 7-11, 2007 London, England.

1. INTRODUCTION

The Vermont Department of Public Psychiatry is currently starting the process of developing a comprehensive database of mental health and substance abuse patients, comprising numerous disparate features ranging from general patient data (age, income, living environment, education, occupation, religion, etc.), to self and social worker assessments of life skills (money management, health management, transportation, personal hygiene, etc.), subjective quality of life (money, fun, family, friends, etc.), unacceptable community behaviors (drug or alcohol abuse, theft, assaults, etc.), and specific mental health symptoms (delusions, anxiety, depression, mania, suicide attempts, etc.), treatments (buprenorphine, confinement, counseling, etc.), and patient outcomes. Consequently, there is great interest in developing techniques for mining this future database for useful predictive information that may provide guidance for doctors, courts, and governmental policies, such as identifying promising treatment regimes or risk of physical violence to self or others. In many cases, this reduces to association studies for patient classification, although an understanding of the functional form of the underlying nonlinear associations is also desirable, in order to direct future studies and provide insights into potential causality of the relationship. Such classification problems can be non-trivial due to a variety of factors, including non-linear interactions between features, high dimensionality feature space relative to the sample size, mixed attribute types, and noise in the classification data (e.g., due to factors such as reporting error, insufficient data, and weak associations with the attributes).

Many techniques have been used in data classification problems. For example, artificial neural networks (ANN's) have been shown to be fast and fairly accurate classifiers, and are included in several commercial and academic data mining packages (e.g. Weka [26], XLMiner [27], and Clementine [5]). However, the black box nature of ANN's make them poorly suited to extracting human readable information about the form of the model underlying the classifier, although some information can be extracted from the trained weights [1],[18]. Other standard approaches to classification include support vector machines [6], decision trees (e.g. C4.5 [22]), and Bayesian approaches [4],[10].

Evolutionary algorithms such as learning classifier systems [15], genetic programming (GP) [2],[3],[7],[19], as well as hybrid evolutionary approaches [25], have also emerged as attractive options for use in classification problems. Here, we focus on GP, since it has the advantage of estimating the underlying functional relationship between features and class, which is one of the goals we seek to achieve.

The number of features that can be statistically associated with class data is ultimately limited by the sample size of the dataset and the dimensionality of the feature space [11]. Consequently, there has been significant work done in the field of feature selection, in order to find an optimal subset of features that can enable construction of a predictive classifier and doesn't simply over-fit the data. Various statistical approaches have been applied to feature selection, including those based on entropy and *t*-statistics [17]. However, since the optimal size of a set of features depends on the problem, rather than the classifier [17], it can be difficult to determine how many features to select. As with classification problems, evolutionary algorithms have been shown to have promise for feature selection, including genetic algorithms (GA's) [9],[16],[20],[21],[23],[28] and GP [12]. In general, most evolutionary approaches have the advantage that they do not require pre-specification of the size of the optimal feature set.

If individual features have strong main effects, one can employ a constructive or sequential approach to feature selection. However, if the features exhibit strong nonlinear interactions, then there may be little or no fitness advantage until all the correct features are present. One option is to search exhaustively for all possible ways any number of features might interact to affect an outcome, however for large feature sets this is not practical. In [8], an evolutionary approach to feature selection in highly epistatic problems is proposed, in which the size of the potential feature set is stochastically halved in order to systematically reduce the feature set. However, there is no way for the algorithm to recover, should any of the correct features be inadvertently lost.

In this contribution, we propose a new hybrid approach to combined feature selection and classification in problems with high-dimensionality feature sets containing a small subset of non-linearly interacting features, in the presence of varying degrees of classification noise. Specifically, we propose a GA that uses an intersection/union recombination operator and a noisy but relatively fast fitness function based on a counter-propagation ANN, in order to reduce the feature set to the most predictive subset. A classifier is then constructed using the reduced feature set as the terminal variables in a strongly typed GP and, for comparison, a newly trained counter-propagation ANN. The final classifiers are validated on a second dataset. Preliminary testing of the proposed method is performed using synthetic data modeled after the proposed mental health database, with randomly generated noisy nonlinear associations between the class and a few of many potential mixed-type features.

2. METHODS

2.1 Datasets

As previously mentioned, our work was inspired by a need to build classification models for a specific mental health database. Since the actual data is not yet available, synthetic data were generated to conform to the preliminary structure of the proposed database. In order to test our algorithm against a variety of problems we have created patient datasets as follows. The process consists of three steps: 1) generate the features of simulated patients, 2) generate the class of each simulated patient, and 3) inject noise to the simulated patient class data.

In step 1, we simulated datasets with n = 1000 patients and m = 110 features, 56 of which were integer-valued and 54 of which were boolean-valued. Most of the integer-valued features comprised those with small ranges representing a graded response to a question (typically with ranges less than or equal to 0 to 8) although a few represented larger quantities (such as age or salary). Patient feature values were randomly generated within allowable ranges for each feature, and subject to certain constraints between features. Although space precludes the exact specification of the meaning and range of each feature in this manuscript, the data generator is available upon request.

In step 2, we generate a non-linear relationship between feature values and patient class by randomly generating a boolean-valued expression tree according to certain parameters and subject to certain constraints, as described below. Specifically, we specified the number of true features (m', where m' is much less than m) desired in the expression tree, the desired noise level (specified as the proportion of classes that do not agree with those generated by the expression tree), and a specified *tolerance* for the degree of main effects (defined below). The terminal (T) and functional (F) sets used to generate the tree were:

$$T = \{1, 2, 3\} \cup \{\alpha_1, \alpha_2, \ldots, \alpha_m\} \quad (2)$$

$$F = \{\wedge, \vee, \neg, <, >, =, +\} \quad (3)$$

where

$$\{\wedge,\vee\} : \text{boolean} \times \text{boolean} \rightarrow \text{boolean}$$

$$\neg : \text{boolean} \rightarrow \text{boolean}$$

$$\{<,>,=\} : \text{integer} \times \text{integer} \rightarrow \text{boolean}$$

$$+ : \text{integer} \times \text{integer} \rightarrow \text{integer}$$

and where each α_i is one of the m' unique features randomly selected from the m total features. Note that small integers were included in the terminal set for comparison to integer-valued features. The root of the tree was guaranteed to return a boolean value. Each patient was then classified by evaluating the expression tree on their features. Three final constraints were also enforced on randomly generated "true" classification trees:

a) All m' true features must be included in the randomly generated expression tree representing the true classification.

b) The ratio of the sum of all cases (class = true) to controls (class = false) in the data had to be between 1:3 and 3:1, so that there was sufficient representation from both classes in the database to make classification feasible.

c) Main effects of individual attributes were limited to within the specified tolerance. Specifically, the proportion of agreement between any given individual attribute and the class data (or its complement) must be less than 0.5 + tolerance (where a 0.5 proportion of agreement is purely random). In this study, we used a low tolerance of 0.1 to guarantee highly epistatic problems.

The last constraint makes the generated problems more difficult by ensuring a certain degree of nonlinear interactions among all true features. Random expression tree generation continued until an instance was found that met the above constraints. In the current study, all trees generated that satisfied the constraints were small, with only 3 or 4 terminals.

Finally, in step 3, each patient class value was bit flipped with probability equal to the noise level. In this study, we tested noise levels ranging from 0 (no classification noise) to 0.5 (i.e., equivalent to purely random classification).

2.2 Algorithm Description

Our classification problem was broken into two separate problems. Step 1: feature selection, wherein we attempted to determine an optimal subset of attributes that, together, were predictive of patient class, and step 2: classifier construction, wherein we built a model with the selected attributes to classify patients. The goal was to find a classifier that minimized the classification error, defined as the proportion of patients whose class was incorrectly predicted by the classifier. After the classifier had been built using the first dataset, we validated its accuracy by assessing classification error on a second dataset (step 3), independently generated according to the same underlying true model.

In order to accomplish this, we use a GA (with an ANN-based fitness function) to perform feature selection, followed by classifier construction using GP (and, for comparison, an ANN) using only the selected features. Figure 1 shows an overview of the combined feature selection and classifier construction approach used. Details of the algorithm are provided in the following subsections.

2.3 Feature Selection with GA+ANN

To perform feature selection, we use a GA with fitness based on an ANN (Figure 1, step 1).

2.3.1 ANN Fitness function for feature selection

The fitness of individuals in the GA population is defined by the the classification error of an embedded counter propagation ANN, implemented in Matlab as described in [13]. For use in this fitness function during feature selection, the training set for the ANN comprised a random 80% of cases plus a random 80% of controls from the original dataset, and the testing set comprised the remaining 20% of cases and controls. Thus, for this study there were approximately 800 patients in the training set and 200 in the testing set of the ANN fitness function used for feature selection. Values of each feature were normalized to the range 0 to 1, prior to input into the ANN. The ANN takes time linearly proportional to the number of features in the set being evaluated. In order to limit computation time for fitness evaluation, we limited the ANN to 30 iterations of training starting from only one set of random initial weight, although this contributed to a small amount of additional noise in the fitness function. The ANN classification error is a noisy evaluation of fitness, especially for large numbers of features, due to over-fitting. For example, in Figure 2 we show the ANN classification error on candidate feature sets of various sizes, averaged over 3 repetitions on 10 random problems, each with 3 true features and no noise. The features in the candidate

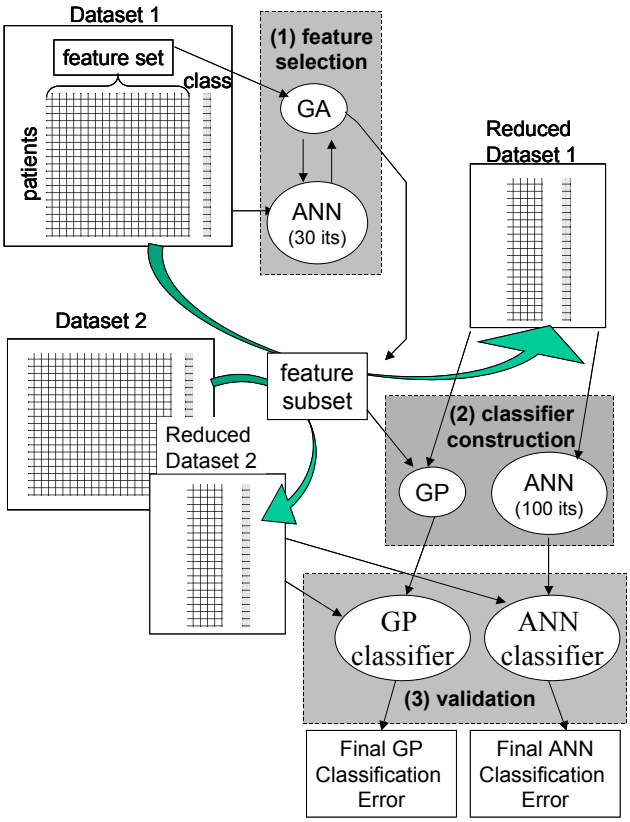

Figure 1: Flow of hybrid GA+ANN→GP and GA+ANN→ANN classification algorithms. Step 1: Feature Selection, step 2: classifier construction using the selected feature set, and step 3: validation with the selected features from a second dataset.

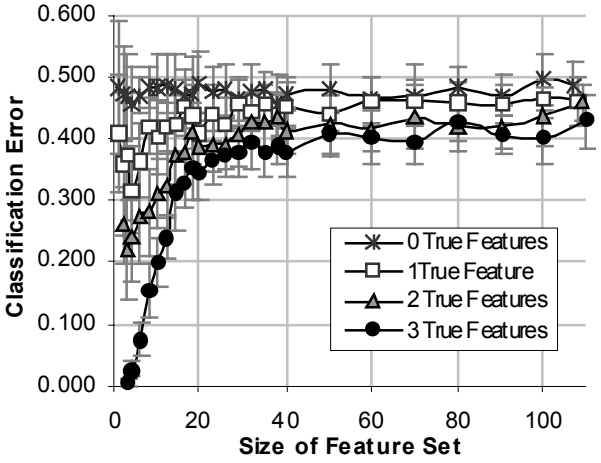

Figure 2. Fitness landscape of feature selection using the ANN on 10 representative problems with 3 true features. Data points indicate means and error bars represent standard deviations for feature sets of various sizes that contain 0 through all 3 of the true features.

sets (both true features, and excess features) are also chosen at random for each repetition. Note that, except for very small feature sets, it is difficult to distinguish sets that contain all 3 of the true features from those that contain fewer or even none of the true features, as indicated by the high degree of overlap in the standard deviations of fitnesses (Figure 2). This is due to the ANN over-fitting the data to the large feature sets. In addition, note the large jump in classification error between the set that contains only the 3 true features and sets that contain exactly two of the true features; this is an indication of the epistatic nature of the problem.

2.3.2 GA for feature selection

A GA was wrapped around the ANN for selecting the optimal feature set. The GA was implemented by modifying the genetic algorithm provided in the Matlab Genetic Algorithms and Pattern Search Toolbox. Potential solutions are represented by chromosomes of binary strings of length m, where a 1 in column i means that the i^{th} feature is in the feature set represented by the potential solution. We assume to have some knowledge of the maximum size (f) of the optimal feature subset. This assumption is reasonable, since even in the absence of *a priori* knowledge regarding the size of the "true" feature set, the sample size of the dataset imposes statistical limitations on the maximum number of features that can be included without significant over-fitting [11]. In order to handle problems that may be highly epistatic, our algorithm generates an initial population designed to contain q individuals comprising supersets of the optimal feature subset. We then select, recombine and mutate individuals with the goal of eliminating the less predictive features and finding a smaller superset (or, ideally, the exact optimal subset) of the correct features. As feature sets are reduced in size, fitness improves due to less over-fitting to extraneous features (Figure 2). Assuming that chromosomes are initialized such that ones are generated with probability P_1, the minimum population size μ_{GA} required to generate approximately q individuals containing all f of the correct features can be approximated as follows:

$$\mu_{GA} \approx q / (P_1^f) \quad (5)$$

The higher the proportion P_1 (i.e., the more 1's in the initial chromosomes), the smaller the required population size μ_{GA}, but the noisier the ANN fitness function (due to over-fitting) and the longer individual ANN fitness evaluations take. In the preliminary results reported here, we arbitrarily used uniform initialization (P_i = 0.5) and q = 15, and future work will include optimization of these parameters. Since we limited this study to 3-feature problems (f = 3), the GA populations used here for feature selection thus comprised μ_{GA} = 120 individuals.

Individuals in the population were evaluated for fitness using the ANN described in Section 2.3.1, and parent selection was performed using tournament selection (with replacement). The maximum generations was set at 20, which was found to be sufficient for the problems tested here.

We propose and implement an Intersection/Union recombination operator that creates children based on the commonality of the two selected parents. With some (high) probability P_A we create a child using the bit-wise AND of the parent's bit-strings (candidate feature set intersection), otherwise we create a child using the bit-wise OR (candidate feature set union). The emphasis is on set intersection of high fitness solutions, so that the size of the feature set is rapidly reduced (which also speeds up ANN fitness evaluations and reduces the noise in the fitness function, as indicated by Figure 2). However, allowing a non-zero probability of set union allows potential repair of candidate solutions that do not contain all the correct solutions. We tested $P_A \in \{1.0, 0.95, 0.8, 0.7, 0.5\}$ on 5 random 3-feature problems with no noise, using tournaments of size 8, and selected the best of 4 repetitions on each problem. Averaged results are shown in Table 1. Although the Intersection/Union operator proved surprisingly insensitive to P_A, performance at P_A =0.95 resulted in optimal solutions for these problems (Table 1, bold).

Table 1. Performance of GA+ANN feature selector for different values of P_A, using tournaments of size 8 and Intersection/Union recombination.

P_A	*# of 3 true features found*	*# excess features found*
0.5	3	0.6
0.7	3	0.2
0.8	3	0.2
0.95	**3**	**0**
1.0	3	0.2

We also compared the performance of the feature selector using the Intersection/Union recombination operator (with P_A = 0.95) *vs.* standard single-point crossover, for a variety of tournament sizes. For example, Table 2 summarizes results with tournament sizes of 2 and 8, averaged over the best of 4 repetitions on each of 5 random 3-feature problems with no injected noise. With single-point crossover, feature sets were only reduced slightly from the initial average of 55 features, regardless of tournament size. With the Intersection/Union recombination and tournaments of size 8, however, the feature selector was able find optimal solutions in all of these test problems (Table 2, bold). Performance on both criteria was slightly reduced for both recombination operators, when tournaments were reduced to size 2.

Table 2. Performance of GA+ANN feature selector, for two different recombination operators and two different tournament sizes.

Type of Recombination	*Tourney size*	*# of 3 true features found*	*# excess features found*
Single-point	2	3	44.6
Single-point	8	3	42
Intersection/ Union (P_A=0.95)	2	2.8	1
Intersection/ Union (P_A=0.95)	**8**	**3**	**0**

Based on these experiments, all further studies were performed using tournaments of size 8 and the Intersection/Union recombination with P_A = 0.95. The final parameter settings for the GA feature selector are summarized in Table 3.

Table 3. Parameter settings for GA feature selection.

GA Parameter	*Value*
Elitism	1
Survivor Selection	(μ, λ)
Selection Operator	Tournament w/Replacement (size 8)
Crossover Probability	0.80
Crossover Operator	Intersection/Union ($P_A = 0.95$)
Mutation Probability	0.05
Mutation Operator	Uniform
Maximum Generations	20
Population Size μ_{GA}	120

2.4 Classifier Construction with GP

Once k features had been selected with the GA+ANN, we constructed a binary classifier using the selected features as the terminal variables in a strongly typed GP (Figure 1, step 2), also written in Matlab. The objective function of the GP evaluates a boolean-valued expression tree and returns the proportion of patients that are classified incorrectly. In this preliminary study, the function set F we used was the same as used in the problem generator, as shown in equation (3), although we recognize that it would make the problem more challenging to use a different function set here.

The terminal set T' for the GP classifier is

$$T' = \{1, 2, 3\} \cup \{\alpha_1, \alpha_2, \ldots, \alpha_k\} \quad (6)$$

where in this case each α_i is one of the k distinct integer- or boolean-valued features previously selected by the GA+ANN.

Strong typing was enforced during crossover in the GP as follows. When performing crossover, a random subtree was selected from one parent and its return type (integer or boolean) was noted. A random subtree location was then chosen in the second parent. If the return type of this second subtree was integer, while the return type of the first parent's subtree was boolean, then the subtree location in the second parent was moved up the tree until a boolean return value was found. Since the root of the tree was forced to be a boolean-valued function (so that the expression tree could act as a binary classifier), this condition is guaranteed to be ultimately satisfied. However, if the return type of the first subtree was integer and that of the second was boolean, then the subtree location in the second parent was moved down the tree until an integer return value was found. Since the leaves of the trees were permitted to be either boolean or integer terminals, in the event that a boolean-valued terminal was encountered, it was simply treated as an integer with value 0 or 1. Once the return type of the subtree from the second parent matched that of the first, the second parent's subtree was swapped with the first parent's subtree. Strong typing was also guaranteed for mutation, by simply choosing a random location for mutation and then generating random new subtrees until one with the compatible return type was generated.

We tested six population sizes for the GP: $\mu_{GP} \in \{50, 100, 200, 400, 800, 1600\}$ (Figure 3). For each population size, we performed two runs on each of 20 randomly generated problems (a total of 40 runs per population size) with a noise level of 0.1.

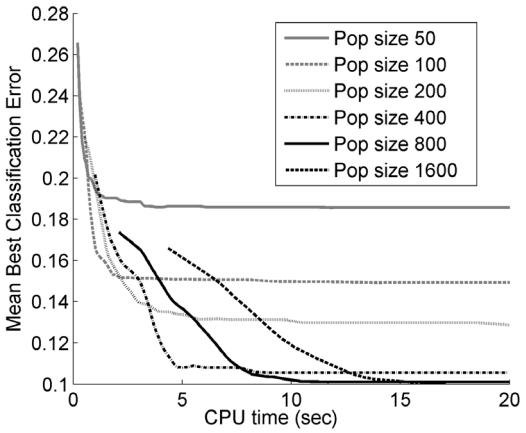

Figure 3: Minimum GP fitness as a function of CPU time, averaged over 40 runs for each population size (a total of 240 runs)

Each generated problem consisted of 1,000 patients with 3 true features, with $k = 8$ total features (including the 3 true features and 5 additional extraneous features). Since the noise level was set to 0.1, a fitness of 0.1 was considered optimal. The curves do not begin at 0 because of the CPU time (roughly proportional to population size) required to initialize and evaluate the first generation. Based on these results we decided to use a population size of $\mu_{GP} = 800$, an acceptable compromise between CPU time required for convergence and minimum fitness. Our GA+ANN algorithm selected fewer than 8 features for all 3-feature problems except at very high noise levels, so this population size was deemed adequate for the problems tackled. Table 4 summarizes the settings for the GP used in our algorithm.

Table 4. Parameter settings for GP classifier.

GP Parameter	*Value*
Population Size μ_{GP}	800
Initialization method	Ramped half-and-half
Selection	Tournament (size 2)
Recombination method	Strongly typed subtree swap
Probability of selecting internal node as crossover point	0.9
Probability of crossover	0.7
Probability of mutation	0.3
Replacement	Replace if better or equal
Termination condition	45 generations or 15 generations w/o improvement or classification error = 0
Maximum tree height	17

2.5 Validation

For final testing and validation of each classifier on each problem, we generated a second dataset of 1000 new synthetic patients, according to the same true expression tree and with the same level of noise as was used to construct the original dataset. Classification error of the best final expression tree (the GP

classifier) resulting from the GA+ANN→GP classifier that had been constructed using the original dataset was then assessed on this new dataset (Figure 1, step 3). This cross-validation approach was used in order to detect over-fitting to the original dataset on which the classifier was trained.

For comparison to the GP results, we also retrained the counter propagation ANN (this time, allowing up to 100 iterations, which was always sufficient for convergence) using data from only the *k* selected features of the original dataset (Figure 1, step 2) and validated the resulting ANN classifier with the second dataset (Figure 1, step 3). With the reduced feature set and 100 iterations, the initial values of the random ANN weights was found to have little effect of the final ANN classification error, so again, only one set of initial ANN weights was employed in this step.

2.6 Experiments

For each of six different noise levels {0, 0.025, 0.1, 0.2, 0.35, 0.5}, and each of the two different classification strategies (GA+ANN→GP and GA+ANN→ANN), we ran 4 repetitions on each of 10 randomly generated 3-feature problems. For each different problem, we then selected the best of the 4 repetitions, for each classifier type. Thus, in some cases, different repetitions of the same problem were selected for the two different classifiers.

3. RESULTS

The performance of the two types of classifiers on the validation dataset is shown in Table 5, as a function of injected noise in the class data. With little or no noise (≤ 0.025 noise) the feature selector was able to successfully identify the 3 correct features 100% of the time in the sample problems tested, and almost never included any extraneous features in the final feature set. In fact the feature selector continued to perform quite well even at 0.1 and 0.2 noise levels. However, at noise levels or 0.35 or higher, many extraneous features were included and most final feature sets did not include all of the 3 true features.

Table 5. Final results for GA+ANN feature selector and for the GP (bold) and ANN classifiers.

noise	*classifier*	*from GA+ANN feature selector*		*classification error on validation dataset*	
		# 3 true features	*# excess features*	*mean*	*std*
0.000	**GP**	**3.0**	**0.0**	**0.000**	**0.000**
	ANN	3.0	0.1	0.003	0.006
0.025	**GP**	**3.0**	**0.0**	**0.023**	**0.004**
	ANN	3.0	0.1	0.032	0.016
0.100	**GP**	**2.9**	**0.6**	**0.107**	**0.018**
	ANN	2.9	0.8	0.126	0.044
0.200	**GP**	**2.7**	**2.9**	**0.212**	**0.024**
	ANN	2.4	2.3	0.261	0.039
0.350	**GP**	**1.7**	**34.9**	**0.358**	**0.020**
	ANN	0.9	26.0	0.463	0.026
0.500	**GP**	**1.2**	**35.2**	**0.364**	**0.011**
	ANN	1.1	30.6	0.456	0.019

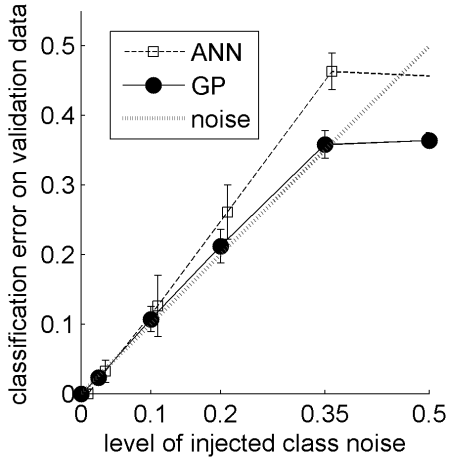

Figure 4. Comparison of classification errors of the final GP and ANN classifiers on the second dataset, as a function of the level of injected noise. Data points represent means, and error bars represent one standard deviation. Note that the ANN data points have been shifted to the right by 0.01 in order to make the error bars more visible.

The final classification error of the two classifiers increases linearly with the amount of injected noise, up to a noise level of 0.35 (Table 5, Figure 4). Note that the proportion of noise injected into the class data (Figure 4, gray dotted line) represents the expected optimal potential classification error of the classifiers up to about noise = 0.35. At noise levels of 0.5, where the class data has been totally randomized and thus has no statistical association with the features, the expected classification error is still expected to be only about 0.35, when the ratio of cases to controls is 1:3 or 3:1. As shown in Figure 4, the GP classifier consistently achieves essentially optimal results at all noise levels. In all cases, the means and standard deviations of classification error resulting from the GP classifier were lower than that of the ANN classifier (although not significantly so at low noise levels), and the difference between them increases linearly with noise (Table 5, Figure 4). Part of the reason for the better performance in the GP, relative to the ANN, is that the GP did additional feature selection in the process of evolving the optimal expression tree, whereas the ANN necessarily used all features provided to it and therefore had a greater tendency to over-fit the data. In fact, the correct "true" expression tree (or a functional equivalent) was recovered for all problems during GP classifier construction at noise levels of 0.0 and 0.025, for all 9 of 10 problems in which the feature selector identified all 3 of the true features at noise 0.10, and for 6 or the 7 problems in which the feature selector found all 3 of the true features at noise level 0.20.

At noise level 0.35, the feature selector found all 3 of the correct features in only one of the 10 problems, so consequently the GP classification trees tended to include many extraneous variables. However, even at the high noise level of 0.35, there was only one problem in which the GP discarded one of the true features that had been provided to it by the feature selector. At 0.5 noise, there is no statistical association between features and classes, so both classifiers performed poorly, as expected.

It is worth noting that any variance in quality of the solutions between the 4 repetitions on each problem was primarily due to

variance in the quality of the results coming from the feature selector, rather than to variance in the performance of the classifier construction itself. When the selected feature set was a small superset of the 3 true features, the GP invariably found the correct underlying expression tree, and so the results of the 4 repetitions were identical.

4. DISCUSSION

A hybrid evolutionary strategy was proposed for the combined problem of feature selection (using a GA with an ANN-based fitness function) and subsequent classifier construction (using GP), for use in nonlinear association studies with relatively large potential feature sets and noisy class data. We had the dual goals of (a) accurately classifying new individuals based on feature values, and (b) estimating the nonlinear functional relationship of a subset of predictive features and class.

In problems like this, the total feature set must be reduced prior to classifier construction, in order to avoid simply over-fitting to training data. In [8], a method was proposed to systematically reduce the size of the potential feature set through stochastic halving; however, in that approach once any of the true features are lost they cannot be reintroduced. Here, we also attempt to systematically reduce the feature set by using set intersection of potential feature subsets as the primary recombination operator in a population of potential solutions. By starting with an initial population that is expected to have numerous individuals that represent supersets of the optimal feature subset, intersection between supersets enables rapid elimination of extraneous features and convergence on the exact optimal subset of features. However, selection of which individuals will recombine is performed using a fixed-iteration counter-propagation ANN, which is quite noisy when evaluating large feature subsets, due to over-fitting. Thus, especially in early generations when individuals comprise fairly large subsets of potential features, parents selected due to high fitness may not necessarily contain the entire optimal subset of features. Thus, with a lower probability, we also employ set union as a recombination operator, thereby enabling the size of potential feature subsets to increase and providing an opportunity to introduce missing critical features into individuals. This hybrid GA+ANN approach to feature selection worked well, and optimal (or nearly optimal) feature sets were accurately identified most of the time, even when class data was bit-flipped with probability 0.2. This represents a fairly high degree of noise, especially considering that the ratio of cases to controls (or *vice versa*) could have been as little as 1:3 before the noise was added.

One could, alternatively, use a GP as the fitness function in a feature selection GA. However, this would necessitate evolving entire GP populations for every fitness evaluation of individuals in the GA population. Since the size of the terminal set varies between individuals in the GA population, it would be very difficult to accurately size such GP populations, runtimes of the GP-based fitness functions could be quite long and variable, and results would still be noisy (especially in early generations when evaluating large terminal sets) due to the stochastic nature of GP evolution.

In contrast, the fixed-iteration counter-propagation ANN runs in known time that is linearly proportional to the size of the feature subset being evaluated. Preliminary experimentation had shown that limiting the ANN to 30 iterations yielded results almost as accurate as allowing the ANN to run to full convergence. Other bounded-time fitness functions are also good candidates for use in the feature selection GA. For example, in [8] a bounded-time fitness function was based on the ReliefF data mining algorithm [24], and their results indicate that this may be less noisy than the ANN-based fitness function proposed here. A direct comparison of these two fitness functions is planned for the future.

Both the speed and accuracy of the ANN- (or ReliefF-) based fitness function could be further improved by starting the GA with individuals that contained smaller feature subsets (lower P_I). However, this would necessitate a larger initial population in order to expect a given number of individuals that contained the optimal feature subset. This trade-off is worth exploring, and in future studies we will seek to optimize P_I.

Once a subset of potential features has been selected, one must still determine the optimal way in which to use them to predict the class of an individual. For this step, we used GP to construct a predictive expression tree. For comparison, we also trained a counter-propagation ANN with the selected subset of features, this time allowing it to run for 100 iterations (which was sufficient for convergence). Not only did the GP consistently outperform the ANN as a classifier, it also was able to do additional feature selection and reconstruct the correct underlying nonlinear expression associating features with class, even with noisy class data. Although all "true" (synthetically generated) underlying associations were designed to be highly nonlinear, the true expressions trees were still relatively small. Furthermore, the GP used for classifier construction used the same functional set as was used in the true expression generation process. Thus, it is not surprising that, when provided with the optimal feature subset and noiseless class data, the GP was able to recover the correct underlying tree in all cases. However, we were surprised by how well the GP continued to perform even when the strength of the relationship between features and class was deliberately lessened, by injecting noise into the class data. Even when class data was randomly bit-flipped 20% of the time, the GP was almost always able to recover the correct underlying expressions when provided with a superset of the true features.

While the experiments performed here were by no means exhaustive, they demonstrate that the proposed hybrid algorithm has good potential for feature selection, classification, and estimation of the form of even relatively weak nonlinear relationships between features and class. Although this project was initially motivated by data mining needs in a proposed mental health database, the method proposed is general and is applicable to other similar problems, such as detecting genetic interactions underlying complex disease traits [14] or other nonlinear association studies.

5. ACKNOWLEDGEMENTS

This manuscript grew out of a class project in CS 395 Evolutionary Computation, F'06; we thank the other members of this class for helpful discussions, and in particular Kirsten Stor, Ian Kavanagh and J. Brooks Zurn. We thank Joshua L. Payne, who provided a preliminary un-typed GP code and gave helpful feedback on this manuscript, Donna M. Rizzo who provided the ANN code, and Thomas A. Simpatico and Emily Levy for discussions regarding the proposed mental health database.

6. REFERENCES

[1] Andrews R., Diederich J., and Tickle A.B. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge Based Systems*, 8:373-389, 1995.

[2] Bojarczuk, C.C., Lopes, H.H., and Freitas, A.A. Discovering comprehensible classification rules using genetic programming: a case study in a medical domain. *Proc. Genetic and Evolutionary Computation Conference (GECCO), 953-958, 1999.*

[3] Brameier, M., and Banzhaf, W. A Comparison of Linear Genetic Programming and Neural Networks in Medical Data Mining. *IEEE Trans. on Evol. Comp.,* 5:17-26, 2001.

[4] Cheng, J. and Greiner, R. Comparing bayesian network classifiers. *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pp. 101-107, 1999.

[5] Clementine: http://www.spss.com/clementine/

[6] Cortes, C., and Vapnik, V. Support-vector networks. *Machine Learning*, 20(3):273-297, 1995.

[7] De Falco, I., Cioppa, A.D., and Tarantino, E. Discovering Interesting Classification Rules with Genetic Programming. *Applied Soft Computing*, 23:1-13, 2002.

[8] Eppstein, M.J., Payne, J.L., White, B.C., and Moore. J.H., Hill-climbing through "random chemistry" for detecting epistasis, *Late-breaking papers, Genetic and Evolutionary Computation Conference (GECCO) 2006.*

[9] Ferri, F., Pudil, P., Hatef, M., and Kittler, J. Comparative study of techniques for large-scale feature selection. *In: Pattern Recognition in Practice IV, Multiple Paradigms, Comparative Studies and Hybrid Systems,* eds. E.S. Gelsema and L.S. Kanal. Amsterdam: Elsevier, pp. 403-413. 1994.

[10] Friedman, N., Geiger, D., and Goldszmidt, M., Bayesian Network Classifiers, *Mach. Learn.*, 29:131-163, 1997.

[11] Gauderman, W.J. Sample size requirements for association studies of gene-gene interaction. *Am. J. Epidem*., 155:478-484, 2002.

[12] Guo, H., Jack, L.B., and Nandi, A.K. Feature Generation Using Genetic Programming with Application to Fault Classification. *IEEE Transactions on Systems, Man and Cybernetics, Part B.* 35(1):89-99, 2005.

[13] Hecht-Nielsen, R. Counterpropogation networks. *Applied Optics.* 26(23):4979-4984, 1987.

[14] Hirschhorn, J.N, and Daly, M.J. Genome-wide association studies for common diseases and complex traits. *Nature Reviews Genetics* **6**, 95-108, 2005.

[15] Holmes, J.H, Lanzi, P.L, Stolzmann, W., and Wilson, S.W Learning classifier systems: New models, successful applications, *Inf. Proc. Lett*., 82:23-30, 2002.

[16] Kudo, M., and Sklansky, J. Comparison of Algorithms that Select Features for Pattern Classifiers. *Patt. Recog.,* 33:25-41, 2000.

[17] Liu, H., Li, J., and Wong, L. A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Informatics,* 13:51-60, 2002.

[18] Lu, H., Setiono, R., and Liu, H. Effective data mining using neural networks. *IEEE Transactions on Knowledge and Data Engineering,* 8(6):957-961, 1996.

[19] Muni, D.P., Pal, N.R., and Das, J. A Novel Approach to Design Classifiers Using Genetic Programming. *IEEE Trans. Evol. Comp.,* 8:183-196, 2004.

[20] Oh, I.-S., Lee, J.-S., and Moon, B.-R.Hybrid Genetic Algorithms for Feature Selection, *IEEE Trans. Patt. Anal. And Mach. Intel*., 26:1424-1437, 2004.

[21] Pernkopf, F., and O'Leary, P. Feature Selection for Classification Using Genetic Algorithms with a Novel Encoding, *Proceedings of the 9th International Conference on Computer Analysis of Images and Patterns*, pp. 161-168, 2001.

[22] Quinlan, J.R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, 1993.

[23] Raymer, M.L., Punch, W.F., Goodman, E.D., Kuhn, L.A., and Jain, A.K. Dimensionality reduction using genetic algorithms. *IEEE Trans. Evol. Comp.*, 4:164--171, 2000

[24] Robnik-Sikonja, M., and Konenenko, I. Theoretical and empirical analysis of ReliefF and RReliefF. *Mach. Learning*, **53**, 23-69 (2003).

[25] Tan, K.C., Yu, Q., Heng, C.M., and Lee, T.H. Evolutionary Computing for Knowledge Discovery in Medical Diagnosis. *Artificial Intelligence in Medicine.* 27:129-154, 2003.

[26] Witten, I.H., and Frank, E. *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[27] XLMiner: http://www.resample.com/xlminer/

[28] Yang, J. and Honavar, V. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems,* 13:44-49, 1998.