An Evolutionary Approach for Achieving Scalability with General Regression Neural Networks

Kenan Casey, Aaron Garrett, Joseph Gay, Lacey Montgomery and Gerry Dozier

***Abstract*— In this paper, we present an approach to overcome the scalability issues associated with instance-based learners. Our system uses evolutionary computational techniques to determine the minimal set of training instances needed to achieve good classification accuracy with an instance-based learner. In this way, instance-based learners need not store all the training data available but instead store only those instances that are required for the desired accuracy. Additionally, we explore the utility of evolving the optimal feature set used by the learner for a given problem. In this way, we attempt to deal with the so-called "curse of dimensionality" associated with computational learning systems. To these ends, we introduce the Evolutionary General Regression Neural Network. This design uses an estimation of distribution algorithm to generate both the optimal training set as well as the optimal feature set for a general regression neural network. We compare its performance against a standard general regression neural network and an optimized support vector machine across four different benchmark classification problems.**

***Index Terms*— general regression neural network, evolutionary computation, support vector machine, estimation of distribution algorithm.**

I. Introduction

INSTANCE-BASED learners (also called memory-based, case-based, or lazy learners) have been used for classification tasks for many years [1]. These approaches typically operate by storing each training instance in some form and then comparing new instances to those stored patterns. In some systems (e.g., radial basis function networks [2], [3]), the training data is pre-processed by an unsupervised learning system to achieve some level of compression before the instances are stored. However, an unsupervised system creates clusters by grouping similar training instances together, but this measure of similarity does not take into account the supervised classification of each instance. In other systems, supervised clustering (such as LVQ-II [3]) can be used to achieve compression while maintaining consideration for the classification task. However, the particular clusters that are created may not be optimal in terms of their use for classification. We believe that an important, and currently missing, enhancement to instance-based learners is a compression method that can achieve an optimal compression ratio as well as maximize classification accuracy.

In addition to the scalability issues for instance-based classifiers described above, all computational approaches are plagued by the so-called "curse of dimensionality" [4]–[7]. This refers to the relationship better higher dimensions and classification complexity. As the dimensions for a classification task increase, the difficulty of the task also increases. The main problem with increasing dimensionality is that functions defined in a higher-dimensional space are typically much more complex than those in a lower-dimensional space. For these complex functions, denser samples of data points must be used to accurately model the underlying function. However, such dense samples are much more difficult to find as dimensionality is increased [5]. Fortunately, higher-dimensional problems often have unnecessary features that can be ignored without reducing classification accuracy. Therefore, a good learning system should be capable of determining the salient features needed for a particular learning task.

To address these issues, we present a hybrid approach which combines an evolutionary computation (EC) and an instance-based learner. The EC is used for both compression and feature extraction in order to optimize the classification accuracy of the learner. We discuss a particular instance of this approach using an estimation of distribution algorithm (EDA) [8] as the EC and a general regression neural network (GRNN) as the classifier. Our approach is compared with two similar systems, the standard GRNN and the support vector machine (SVM), in terms of classification accuracy, compression ratio, and number of salient features selected. Our results show that the evolutionary approach to training set minimization and feature set reduction is effective, especially for larger problems. The evolutionary learner returned classification performance equivalent to the standard learners while utilizing training and feature sets that were significantly smaller.

II. Related Work

Over the last two years there have been a number of studies published comparing GRNNs with SVMs for applications ranging from protein fold recognition, image segmentation and retrieval, and digital mammography [9]–[11]. In each of these studies, the SVMs outperformed the GRNNs; however, the authors of these works failed to investigate the impact of reducing the number of training vectors used for the development of the GRNN. Given a set of training vectors, SVMs will typically reduce the original set of training vectors to one that is composed of the *most essential* vectors known as *support vectors*. Extracting the most essential training vectors used for the development of a GRNN will make for a more even, less biased comparison between GRNNs and SVMs.

Gerry Dozier is the director of the **Applied Computational Intelligence (ACI) Laboratory**, Auburn University, Auburn, AL.

K. Casey, A. Garrett, J. Gay, and L. Montgomery are Ph.D. students in the Department of Computer Science and Software Engineering at Auburn University, as well as members of the **ACI Laboratory**.

III. General Regression Neural Networks

The GRNN is an instance-based method that provides estimates of continuous variables related in a linear or non-linear way [12]. The GRNN makes a single pass through a set of training instances and maps each instance $\vec{t_i}$ to a neuron in the network. Each neuron has a weight equal to the desired output, d_i, of the instance. The predicted output for a given vector $\vec{x}$ is a distance-weighted average of the desired outputs of all neurons [12]. This is represented mathematically as

$$f(\vec{x}) = \frac{\sum_{i=1}^{N} H(\vec{x}, \vec{t_i}) d_i}{\sum_{i=1}^{N} H(\vec{x}, \vec{t_i})}$$

where N is the number of training instances and $H(\cdot)$ is the hidden function. The hidden function for a GRNN compares a given input vector to a particular training instance and returns a value in the range [0, 1] indicating the degree of similarity between the two vectors. A common hidden function used for GRNNs is the Gaussian (or radial basis function), which is defined as follows:

$$H(\vec{x}, \vec{y}) = e^{\frac{-||\vec{x}-\vec{y}||^2}{2\sigma^2}}$$

Here $||\cdot||$ represents vector magnitude and σ represents the standard deviation. The GRNN may either use the same σ value for all neurons or a different σ for each neuron. The σ (or each σ_i, if individual values are used) is provided as a parameter to the algorithm. In our work, we used an EDA to evolve the single σ parameter in all GRNNs.

IV. Support Vector Machines

Support vector machines (SVMs) [13], [14] represent a specific instance of an entire class of algorithms known as *kernel methods* which map input vectors through a kernel function (e.g., a Gaussian kernel) in order to create more accurate classifications. The concepts needed for support vector machines were first introduced by Vapnik in 1979 [13], [15], but their utility has only truly been realized in recent years [16]. The central idea at the core of an SVM is to translate all input vectors into a higher-dimensional kernel space where the instances of each class are more easily separated by a hyperplane. The SVM can then find the optimal placement of the hyperplane in order to minimize classification error by maximizing the distance between the hyperplane and the instances of each class [17].

The process of training an SVM can be characterized as the following constrained optimization problem [5], [17]:

Minimize $W(\vec{\alpha})$, defined as

$$W(\vec{\alpha}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i Q_{ij}\alpha_j - \sum_{i=1}^{N}\alpha_i$$

subject to the constraints

$$0 \leq \alpha_i \leq C$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0$$

Here, C is a constant that specifies the penalty for misclassification, and $Q_{ij} = y_i K(\vec{x_i}, \vec{x_j}) y_j$, where $\vec{x_i}$ is the i^{th} training instance with output label y_i and $K(\cdot)$ is the kernel function.

The positive α values remaining after training correspond to, and become the weights for, the *support vectors*, which are the training instances that are actually needed to accurately classify the training data. The support vectors can be thought of as those instances in the training set that are the most difficult to correctly classify [5]. They lie very near the boundary between the two classes and can be imagined as "supporting" the boundary.

Only functions that meet Mercer's condition can be used as kernels. Mercer's condition, states that that the kernel matrix must be symmetric and at least semi-positive definite [16]. A common example of such a kernel is the Gaussian,

$$K(\vec{x}, \vec{y}) = e^{\frac{-||\vec{x}-\vec{y}||^2}{2\sigma^2}}$$

where $||\cdot||$ represents vector magnitude and σ represents the standard deviation [17] just like the GRNN in Section III.

The major difficulty faced by a support vector machine is scalability. Training an SVM requires manipulation of the Q matrix, which has a size that is quadratic in the number of training instances. This presents an implementation problem if the number of training instances is large, as is typical for real-world problems. Solutions have been devised that operate only on certain parts of the Q matrix at any particular time so that the entire matrix need not exist in memory. For more details on these approaches, see [17]. However, dealing with the Q matrix even with such techniques is still computationally expensive.

Once the SVM is trained, binary classification is accomplished according to the following equation [5]:

$$f(\vec{x}) = sgn\left(\sum_{i=1}^{n_{sv}} K(\vec{x}, \vec{v_i})\alpha_i y_i + b\right)$$

where $sgn(\cdot)$ is the sign function, n_{sv} is the number of support vectors found through training, and $\vec{v_i}$ is the i^{th} support vector. Here, b is defined to be

$$b = \frac{1}{y_*} - \sum_{i=1}^{n_{sv}} K(\vec{v_*}, \vec{v_i})\alpha_i y_i$$

where $\vec{v_*}$ represents any support vector we choose (any particular support vector will suffice) and y_* is the output label associated with this vector.

V. The Evolutionary General Regression Neural Network

To overcome the scalability issues associated with instance-based learners, we devised an evolutionary approach to reduce both the number of training instances and the number of features used in such systems. The basic idea for this technique is to create two binary masks, one for the training set and one for the feature set. Each element of a mask represents inclusion or exclusion. Then, an evolutionary computation algorithm is used to search for the optimal mask components that simultaneously minimize training error, number of training instances, and number of features. Clearly, these three optimization criteria are related and, thus, require trade-offs. For instance, decreasing the number of training instances may lead to an increase in the training error. In fact, the criteria represent a multi-objective optimization problem for the EC to solve.

As an illustration of this technique, we created a modification of the standard General Regression Neural Network, which we call the Evolutionary General Regression Neural Network (EGRNN). The specific components of our EGRNN are described in the following sections.

A. Estimation of Distribution Algorithms

An Estimation of Distribution Algorithm (EDA) was used as the EC to perform the binary mask optimization. EDAs attempt to leverage the statistical properties of the fitness landscape in order to create children [8]. In EDAs, there are neither crossover nor mutation operators. During each generation, the new population is created by sampling the probability distribution of the current population.

For binary-coded chromosomes, the EDA makes use of the probability distribution function of selected individuals. Real-coded chromosomes use a probability density function instead [8]. In both cases, a set of parents is selected from the population. The probability distribution/density function is calculated for the set and used to create a new population of offspring.

EDAs operate along one dimension at a time when creating children. Essentially, for binary-coded chromosomes along a particular dimension, a random parent is selected and its gene value is used to create the child gene. For the real-coded chromosomes along a particular dimension, the mean and standard deviation of the set of the parent genes are calculated. Each child gene (along that dimension) is created according to the following equation:

$$child_{id} = \mu_d + \sigma_d \mathrm{N}(0, 1)$$

Here i is the child number, d represents the dimension, μ_d and σ_d represent the mean and standard deviation, respectively, along dimension d, and $\mathrm{N}(\mu, \sigma)$ is a standardized Gaussian random variate that is different for each child and each dimension. For a thorough introduction to EDAs, please see [8].

B. Evaluation Procedure

For each candidate solution, a GRNN was created using the training vectors and features specified by the masks. This GRNN was then trained on those elements of the training set (which simply means that it was initialized with those vectors as kernels) and then tested on the evaluation set. Its performance was measured in terms of the prediction accuracy and mean squared error (MSE) over the evaluation set, as well as the number of training instances that were specified by the mask. These values were used in the calculation of the fitness of the candidate solution.

We developed the GRNN using the standard training method of utilizing a validation set in addition to the evaluation set [1], [18]. Each time a new best evaluation performer was found, it was tested on the validation set. If the validation performance exceeded the previous best validation performer, it replaced the old best validation individual. Otherwise, the previous best validation performer was stored. At the end of the run, we tested the best validation set individual on the test set. The motivation for this training strategy was to avoid over-fitting.

C. Multi-objective Fitness Function

As mentioned earlier, the multi-objective nature of the problem had to be addressed in the fitness function to be minimized. For our work, we used a criterion-based approach for the fitness assignment [19]. Two candidate solutions x and y were compared first according to prediction accuracy on the test set, taking the larger of the two as the "winner". If the two candidate solutions had equal prediction accuracy, they were then compared according to the number of instance vectors used, taking the smaller of the two as the "winner". Finally, if both prediction accuracy and number of instance vectors were equal for x and y, the solution with the smaller MSE on the test set was judged to be the "winner". Note that no preference is given to individuals with fewer features.

It may at first appear, in light of the preceding statement, that the feature mask would evolve to include all of the features, since there is no explicit penalty for doing so. However, as we will show later, the most successful individuals found by the EDA had feature masks that represented only subsets of the total features. The explanation for this seemingly paradoxical result is that, while there is no explicit penalty for keeping all the features, there may be an *implicit* penalty if some of the features are misleading. For this reason, the EGRNN has the ability to selectively ignore misleading features.

VI. Test Suite Description

In order to test the performance of the EGRNN, we used four different benchmark problems. The first three problems were selected from the Machine Learning Repository at the University of California, Irvine. This repository can be accessed through the Web at `http://www.ics.uci.edu/~mlearn/MLRepository.html` as of May 2006. The final problem was taken from the Enhanced Learning for Evolutive Neural Architecture (ELENA) project at the Université catholique de Louvain (UCL). This dataset can be accessed through the Web at `http://page.mi.fu-berlin.de/~prechelt/NIPS_bench.html` as of May 2006.

Each of these problems represents a binary classification task with numeric data for features. (A binary classification task was necessary in order to use the support vector machine as a comparison with our approach.) Additionally, these particular datasets were chosen because they represented a varying range of sizes, both in number of instances and number of features. The details of each dataset are given in the following sections and are summarized in Table I.

A. Breast Cancer Dataset

The first dataset was the University of Wisconsin breast cancer dataset, obtained from the University of Wisconsin

TABLE I
SUMMARY OF TEST SUITE

	Dataset			
	Breast Cancer	**Internet Ads**	**SPAM**	**Cloud**
Training Set	478	1652	3221	3500
Evaluation Set	68	236	460	500
Validation Set	68	236	460	500
Test Set	69	236	461	500
Features	9	1558	57	2
Positive Instances Per Set	34% – 40%	13% – 18%	$\approx 40\%$	$\approx 50\%$

Hospitals, Madison, from Dr. William H. Wolberg between 1989 and 1991 [20]. This set consisted of 699 classification instance vectors. However, there were 16 instances that were incomplete and had to be removed, so our final data set consisted of 683 points. Of these 683 instances, 478 points (70%) were used for training, 68 (10%) for evaluation, 68 (10%) for validation, and 69 (10%) for testing. Each instance consisted of 9 features (clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses) and one classification (benign or malignant). Each feature was represented by an integer in the interval [1, 10]. The percentage of malignant cases in each set ranged from 34%–40%.

B. Internet Advertisements Dataset

The second dataset represented possible advertisements on Internet pages collected by Kushmerick in 1998 [21]. Each instance in the dataset consisted of 1558 features (mostly binary), including such things as the geometry of the image and whether or not certain phrases appeared in the URL or anchor text. The classification task was to predict whether the image represented an advertisement. This set consisted of 3279 classification instance vectors, but 920 of these vectors contained missing features. Therefore, we used the remaining 2359 vectors for our experiments, allocating 1652 instances (70%) for training and 236 (10%) each for evaluation, validation, and testing. The percentage of actual advertisements in each set ranged from 13%–18%.

C. SPAM Dataset

The final dataset was made up of features collected from email messages, some of which were SPAM (i.e., unsolicited commercial email). This data was collected by Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt at Hewlett-Packard Labs in 1999. The dataset was complete and contained 4601 instance vectors. Each instance was made up of 57 continuous-valued features and one binary classification (i.e., SPAM or not SPAM). The features represented whether particular words or characters frequently occurred in the email along with the length of sequences of consecutive capital letters. We used 3221 instances (70%) for training, 460 (10%) for evaluation, 460 (10%) for validation, and 461 (10%) for testing. The percentage of SPAM messages in each set was approximately 40%.

D. Cloud Dataset

The cloud dataset was an artificial construction of a difficult classification problem created by the Enhanced Learning for Evolutive Neural Architecture (ELENA) group at UCL. It was generated in order to obtain some defined characteristics and to serve as a dataset in which the theoretical Bayes error could be computed. The requirements when creating this dataset included having a heavy intersection of the class distribution, a high degree of nonlinearity of the class boundaries, already published results on these databases, and various dimensions of the vectors. The dataset was complete and contained 5000 instance vectors. Each instance was made up of 2 continuous-valued features and one binary classification (i.e., cloud class 0 or cloud class 1). Class 0 is the sum of three different normal distributions while the the class 1 is another normal, overlapping the class 0. Figure 1 shows a plot of the two cloud classes. Class 0 is plotted with black squares and class 1 with the gray crosses. We used a total of 5000 instances with 2500 being class 0 and 2500 being of class 1 clouds. We used 3500 instances (70%) for training, 500 (10%) for evaluation, 500 (10%) for validation, and 500 (10%) for testing. The percentage of class 0 clouds in each set was approximately 50%.

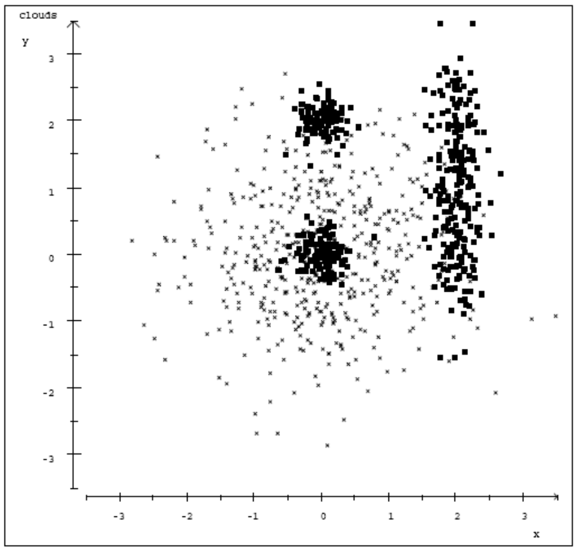

Fig. 1. A plot of the clouds dataset in x and y. The black squares correspond to class 0 and the gray crosses represent class 1

VII. Methodology

For each dataset, we compared the performance of the support vector machine, the standard GRNN, and two versions of the EGRNN. Each of these classifiers, as well as the EDA, required certain parameters to be specified. In each instance of the EDA, a population of size 40 was used along with elitism [8], which ensured that the best individual in the current population would survive to the next generation. The number of function evaluations allowed for each EDA was 5000. Unless otherwise noted, a "function evaluation" in this work refers to an instantiation of a classifier that is subsequently trained and tested on a dataset. We performed 30 independent runs of the GRNN and EGRNN for each dataset in order to gain an understanding of the underlying statistical distribution. Due to its deterministic nature, the SVM only required one run per dataset.

A. SVM Details

The SVM implementation used in our work was SVM$^{\text{light}}$, a free (for scientific use) implementation of an SVM written in C [22]. This software (source code and binaries) can be downloaded from `http://svmlight.joachims.org` as of May 2006. We downloaded and installed the SVM$^{\text{light}}$ Windows XP binaries for use in our work. In all experiments, the default settings for the SVM were used. Since the SVM did not require an evaluation or validation set during training, we included those instances in the SVM's training set.

B. GRNN Details

For the GRNN, we created an implementation in C++ corresponding to the description of the GRNN given in Section III. The implementation required the specification of the σ parameter(s). In our experiments, we used a real-coded EDA to evolve the optimal value for σ on a given dataset. In this case, a candidate solution was simply a possible σ value. To determine the fitness of each candidate solution, a GRNN was constructed with that value for σ, trained on the training set, and used to classify the instances in the evaluation set. The criterion-based multi-objective fitness function described in Section V-C was used to calculate the fitness.

C. EGRNN Details

For the EGRNN, we made use of the standard GRNN implementation and wrapped it in a real-coded EDA to evolve the binary masks and sigma value. We evolved two variants of the EGRNN: one with feature selection and one without feature selection. We denote the EGRNN without feature selection as EGRNN-I and the EGRNN with feature selection as EGRNN-II. In both cases, the EDA evolved a reduced training set for the EGRNN. We used a real-coded EDA (as opposed to a binary-coded EDA) for practical reasons. First, we had an existing implementation of a real-coded EDA. Second, we needed to evolve not only the binary masks but also the σ parameter for each EGRNN. In the case of the binary masks, a positive-valued gene represented a 1 and a non-positive gene value represented a 0.

VIII. Results

We trained the SVM, GRNN, and EGRNN on each of the three datasets and recorded the mean squared error and classification accuracy. For the GRNN and EGRNN, both of these values were meaningful. However, since the SVM was not performing regression, the mean squared error was not a particularly useful measure of its success. For completeness, both values for each dataset are presented.

We also attempted to compare the classifiers in terms of the computational complexity of the training process. For the GRNN and EGRNN, we specified the number of function evaluations for the evolution of the binary masks (for the EGRNN) and the σ parameter (for both). We felt that this provided a meaningful measure of the computational time required. For the SVM, however, we were limited by the implementation that was used and, thus, had no choice in the types of measurements that could be made. SVM$^{\text{light}}$ did provide information on both CPU usage time as well as the number of kernel evaluations. Just as with the MSE, meaningful comparisons are difficult to make between these different measures of complexity, but all values are reported. The results across the test suite are summarized in Table II.

A. Breast Cancer Dataset

All of the classifiers performed very well on this dataset, most likely due to its small size, both in training instances and features.

The SVM achieved an MSE of 1.711 and a classification accuracy of 98.6% on the test set, accurately classifying all but one of the test instances. Additionally, the SVM was able to reduce the training set (consisting of 615 instances) to only 51 support vectors. The amount of time required for training was 0.03 CPU-seconds, using 5,924 kernel evaluations.

The GRNN achieved a classification accuracy of 98.0% on the test set. In terms of MSE, the GRNN achieved a mean squared error of 0.069.

The EGRNN-I (no feature selection) and EGRNN-II (feature selection) returned identical prediction performance on the breast cancer dataset. Both were able to correctly predict 96.9% of the test vectors. The mean squared error over the test set was 0.110 for both EGRNN. The EDA reduced the average training set from 478 vectors to 95.8 (EGRNN-I) and 125.6 (EGRNN-II). The number of features used for classification by the EGRNN-II was reduced to 6.2 on average. Figure 2 shows the percentage each feature was present in the 30 individuals found by the EGRNN-II. Notice that 3 of the 9 features were present less than 50% of the time, impling that these features were either misleading or uninformative.

The reason for the superior performance of GRNN over the EGRNNs is primarily due to the fact that the EGRNNs had many more parameters to evolve (instead of just σ). Therefore, it was difficult for the EGRNNs to have a number of individuals in the population that had all of the training vectors turned on.

B. Internet Advertisements Dataset

The most striking characteristic of this dataset was the incredible number of features (1558) that had to be considered for each training instance. Such ultra-high-dimensional data typically reveals the curse of dimensionality that plagues computational learning approaches.

The SVM achieved an MSE of 0.813 and a classification accuracy of 89.8% on the test set, misclassifying 24 of the 236

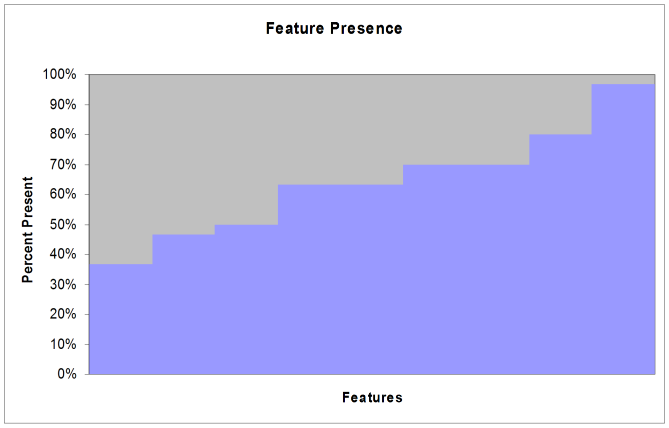

Fig. 2. Feature presence for the breast cancer dataset for 30 runs sorted by percent present

test instances. The SVM was able to reduce the training set, which contained 2124 instances, to 546 support vectors. The amount of time required for training was 51.44 CPU-seconds, using 50,943 kernel evaluations.

The GRNN was able to classify 96.1% of the test set correctly with an MSE of 0.122.

The EGRNN-I classified the test set with an accuracy of 93.9% and an MSE of 0.194. The EGRNN-II returned an average prediction accuracy of 94.5% and an average MSE of 0.186. This difference was statistically significant. EGRNN-I lowered the average number of training vectors to 675.9. EGRNN-II reduced the average training vector set to 738.9 and the average feature set length to 789.8. The presence of the 1558 features in the 30 best solutions found by the EGRNN-II is shown in Figure 3. Only about half of the features were present in more than 50% of the runs. The GRNN outperformed the EGRNNs for the same reasons as stated in the for the breast cancer dataset.

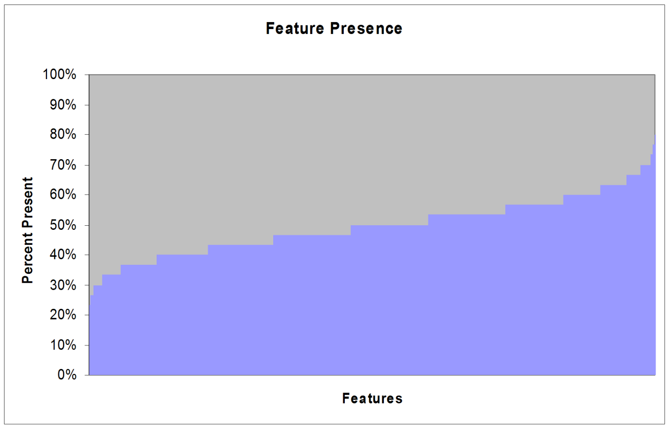

Fig. 3. Feature presence for the Internet ads dataset for 30 runs sorted by percent present

C. SPAM Dataset

The SPAM dataset contained the largest number of training instances in our test suite (4141 for the SVM and 3221 for the neural networks). For many problems of interest, this is a relatively small training set. However, it was large enough to provide a challenge to our classifiers.

The SVM achieved an MSE of 17.156 and a classification accuracy of 74.6% on the test set, misclassifying 117 of the 461 test instances. The SVM was able to reduce the training set from 4141 instances to 2526 support vectors. The amount of time required for training was 1949.96 CPU-seconds (32.5 CPU-minutes). This was significantly more time than was needed for either of the other datasets. The SVM required 2,185,981 kernel evaluations to find the support vectors.

The GRNN achieved a classification accuracy of 77.0% with an MSE of 0.697.

The EGRNN-I performed somewhat poorly in this dataset, returning an average prediction accuracy of 76.2% and an MSE of 0.756. It minimized the average training set to 1604.7 vectors. The EGRNN-II returned an average classification success rate of 89.6% and an MSE of 0.322. The average training set was minimized to 1565.8 vectors and the average feature set considered 34.9 features. Figure 4 summarizes the features which were present in the 30 best solutions evolved by the EGRNN-II. Notice that one feature (capital_run_length_total) was never present in any of the solutions.

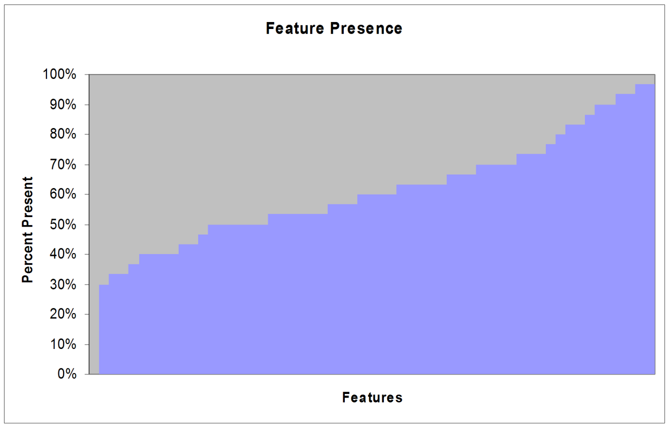

Fig. 4. Feature presence for the SPAM dataset for 30 runs sorted by percent present

D. Clouds Dataset

The EGRNNs and the SVM performed moderately well on this dataset. The clouds dataset had only 2 attributes for the learner to consider. As a result, no learner was able to reduce the number of features needed for classification. This means that the EGRNN-I and EGRNN-II were effectively performing the same function. The small number of features also resulted in a shorter training time compared to the datasets with larger feature sets. Although the number of features was small, the clouds dataset had a large number of instances so training set reduction was very relevant. For the EGRNN we used 3500 for training and then 500 each for evaluation, validation, and for testing. The SVM used 4500 instances for training and 500 for testing.

The SVM achieved a mean square error of 1.334 and the classification accuracy of 75.40% on the test set. The SVM reduced the number of training instances from 4500 to 2539 support vectors.

The GRNN classified 89.5% of the test instances correctly. It returned an average MSE of 0.297.

The EGRNNs were able to accurately classify 89.9% (EGRNN-I) and 90.4% (EGRNN-II) of the test instances giving an average MSE of 0.284 (EGRNN-I) and 0.282 (EGRNN-II). EGRNN-I reduced the number of training vectors from 3500 to an average of 1671.7 vectors while EGRNN-II further reduced it to 1636.3 vectors.

TABLE II
TEST SUITE RESULTS

Breast Cancer Dataset				
Learner	SVM	GRNN	EGRNN-I	EGRNN-II
Prediction Accuracy	98.6%	98.0%	96.9%	96.9%
Mean Squared Error	1.712	0.069	0.110	0.110
Number of Vectors	51	478	95.8	125.6
Number of Features	9	9	9	6.2
Internet Ads Dataset				
Learner	SVM	GRNN	EGRNN-I	EGRNN-II
Prediction Accuracy	89.8%	96.1%	93.9%	94.5%
Mean Squared Error	0.813	0.122	0.194	0.186
Number of Vectors	546	1653	675.9	738.9
Number of Features	1558	1558	1558	789.8
SPAM Dataset				
Learner	SVM	GRNN	EGRNN-I	EGRNN-II
Prediction Accuracy	74.6%	77.0%	76.2%	89.6%
Mean Squared Error	17.156	0.697	0.756	0.322
Number of Vectors	2526	3222	1604.7	1565.8
Number of Features	57	57	57	34.9
Cloud Dataset				
Learner	SVM	GRNN	EGRNN-I	EGRNN-II
Prediction Accuracy	75.4%	89.5%	89.9%	90.4%
Mean Squared Error	1.334	0.297	0.283	0.282
Number of Vectors	2539	3501	1671.7	1636.3
Number of Features	2	2	2	2

IX. ANALYSIS

Our results suggest that for large datasets (i.e., those with large training and/or large feature sets) the EGRNN algorithm is particularly well-suited for minimizing the training set and classifying instances correctly. It returned strong prediction performance for each dataset and reasonably good compression performance especially for the larger datasets.

The breast cancer data set is the smallest dataset in terms of training set size. All three learners performed very well on this data set with prediction accuracies all greater than 96%. The EGRNN performed only 2% worse than the GRNN and SVM. Although this was statistically significant, it represents a relatively small performance difference. The SVM minimized the training set most efficiently reducing the 478 vectors to 51. The EGRNN-I, by comparison, reduced the set to 95.8. Given its smaller computational complexity, greater prediction accuracy, and more efficient training vector compression, the SVM was the clear winner for the breast cancer dataset.

The other three datasets, however, proved to be more challenging. On the Internet Ads problem, the SVM returned the worst prediction accuracy of the three learners, 89.8% compared to 96.0% (GRNN), 93.9% (EGRNN-I), and 94.5% (EGRNN-II). Each performance was statistically distinct. As in the breast cancer problem, the SVM returned the minimum training set (546 vectors) followed closely by the EGRNN-I and EGRNN-II (675.9 and 738.9 respectively). The EGRNN-II reduced the feature set almost in half, returning an average feature set length of 789.8 (of 1558 total). The overall winner for this dataset is not so apparent. The GRNN classified most accurately but required more than twice as many training vectors as the EGRNN. Given the reasonably good feature set reduction achieved by the EGRNN-II over the GRNN and SVM (50%) and its strong performance in terms of accuracy, we consider it the best performer on this dataset.

The EGRNN-II performed best in terms of both accuracy and compression on the SPAM dataset. It classified the test set 15% more accurately than the SVM (89.6% as compared to 74.6%) and 13% better than the GRNN (77.0%) and 14% better than the EGRNN-I (76.2%). Even more impressive was the training set reduction it achieved. The EGRNN-II was able to minimize the training set to 1565.8 vectors while the SVM required 2526 training instances (61% more). Likewise, the feature set was reduced to from 57 features to 34.9 by the EGRNN-II. For this problem, the EGRNN was the clear-cut winner since it provided better classification accuracy with fewer training vectors (and features).

On the clouds dataset, the GRNN and EGRNN both significantly outperformed the SVM in terms of prediction accuracy. The EGRNN-I and EGRNN-II, not surprisingly, had very similar accuracy performance. The EGRNN-II achieved the best training set minimization, reducing the average number of training vectors from 3500 to 1636.3. This was 64% better than the SVM which returned 2539 support vectors. Due to the small feature set, no learner was able to eliminate any features. As expected, both of the two features were necessary for classification. This property of the clouds dataset allows for a more equivalent comparison of the SVM and EGRNN in that both were essentially performing the same task: training set minimization. The results imply that, for large training sets, the EGRNN is able to achieve greater training set reduction and higher prediction accuracy than the SVM, even without feature set minimization. Thus, the definite winner on the clouds dataset was the EGRNN.

Our results demonstrate that the EGRNN can achieve significant training set and feature set reduction without sacrificing classification accuracy. The EGRNN was better able to minimize the larger training sets than the smaller set. This supports the goal of scalability for even larger datasets where scalability is most relevant. The EGRNN's ability to reduce the feature set further enhances its scalability by reducing the number of features that must be stored by the learner. Most notably, the EGRNN was the clear-cut winner on the SPAM and Clouds datasets and a strong performer on the Internet Ads problem. Although the EGRNN is computationally more expensive than the SVM, the results may be well worth the price for problems with large numbers of training vectors and/or features.

X. Conclusions and Future Work

Our results demonstrate the strong performance of the EGRNN with respect to accuracy and training set minimization. The EGRNN achieves training set reduction similar to that produced by an SVM but also capable of feature extraction and dynamic learning. Further consideration of other ECs is needed to determine whether the EDA is the best EC for this task. We would also like to test other instance-based learners in our framework. The EGRRN represents a very promising approach for scalable instance-based learners.

References

[1] T. M. Mitchell, *Machine Learning*, 1st ed. McGraw-Hill, 1997.

[2] P. Picton, *Neural Networks*. Palgrave, 2000.

[3] K. Mehrotra, C. K. Mohan, and S. Ranka, *Elements of Artificial Neural Networks*. MIT Press, 1997.

[4] R. E. Bellman, *Adaptive control processes: A guided tour*. Princeton University Press, 1961.

[5] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Prentice Hall, 1999.

[6] M. Köppen, "The curse of dimensionality." [Online]. Available: citeseer.ist.psu.edu/524717.html

[7] C. Böhm, S. Berchtold, and D. A. Keim, "Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases," *ACM Computing Surveys*, vol. 33, no. 3, pp. 322–373, 2001.

[8] P. Larranaga and J. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[9] I. El-Naqa, Y. Yang, N. P. Galatsanos, R. M. Nishikawa, and M. N. Wernick, "A similarity learning approach to content-based image retrieval: application to digital mammography." *IEEE Transactions on Medical Imaging*, vol. 23, no. 10, pp. 1233–1244, October 2004.

[10] J. Fu, S. Lee, S. Wong, A. Yeh, J.Y.and Wang, and H. Wu, "Image segmentation feature selection and pattern classification for mammographic microcalcifications," Computerized Medical Imaging and Graphics*, Vol. 29, pp. 419-429, Elsevier.*, 2005.

[11] O. Okun and H. Priisalu, "Fast nonnegative matrix factorizaton and its application for protein fold recognition," EURASIP Journal on Applied Signal Processing*, Vol. 2006, Article ID 71817, pp. 1-8*, 2006.

[12] D. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, 1991.

[13] V. Vapnik, *The nature of statistical learning theory*. Springer, 1995.

[14] B. Schölkopf, "Statistical learning and kernel methods," Microsoft Research, Tech. Rep. MSR-TR-2000-23, 2000.

[15] V. Vapnik, *Estimation of dependences based on empirical data; translated by Samuel Kotz*. Springer Verlag, 1982.

[16] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 955–974, 1998.

[17] M. A. Hearst, "Support vector machines," *IEEE Intelligent Systems*, pp. 18–21, 1998.

[18] A. P. Engelbrecht, *Computational Intelligence: An Introduction*. John Wiley and Sons, Ltd., 2002.

[19] E. Zitzler, M. Laumanns, and S. Bleuler, "A tutorial on evolutionary multiobjective optimization," in *Metaheuristics for Multiobjective Optimisation*, X. Gandibleux, M. Sevaux, K. Sorensen, and V. T'kindt, Eds. Berlin: Springer, 2004, pp. 3–37.

[20] O. L. Mangasarian and W. H. Wolberg, "Cancer diagnosis via linear programming," *SIAM News*, vol. 23, no. 5, pp. 1–18, 1990.

[21] N. Kushmerick, "Learning to remove Internet advertisements," in *Proceedings of 3rd International Conference on Autonomous Agents*, 1999.

[22] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods – Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. MIT Press, 1999.

PLACE PHOTO HERE

Kenan Casey Kenan Casey is currently a Ph.D. student in computer science at Auburn University. He received a Bachelor's degree in mathematics and computer science at Freed-Hardeman University, Henderson, TN in 2004. In 2005, Kenan was awarded a GAANN Graduate Fellowship in Wireless Technology from Auburn University. His interests include computational intelligence and sensor networks.

PLACE PHOTO HERE

Aaron Garrett Aaron Garrett is currently completing his Ph.D. in computer science at Auburn University. He received a Bachelor's degree in mathematics and a Master's degree in computer science from Jacksonville State University in Jacksonville, AL. In 2001 while pursuing the Master's degree, Aaron became a NASA Summer Graduate Fellow at Ames Research Center, Mountain View, CA. Since 2002, he has served as an instructor in the computer science department at Jacksonville State University.

PLACE PHOTO HERE

Joseph Gay Joseph Gay is currently a Ph.D. student in computer science at Auburn University. He received a Bachelor's degree in software engineering at Auburn University in 2004. Joseph currently serves as a veteran instructor for COMP 1000, Auburn's computer literacy course. His interest include interactive evolutionary robotics, EMG, and EEG.

PLACE PHOTO HERE

Lacey Montgomery Lacey Montgomery is currently pursuing a Ph.D. in Computer Science from Auburn University. She received a Bachelor's degree in Software Engineering from Auburn University in 2004. Lacey currently serves as the assistant course coordinator for COMP 1000, Auburn's computer literacy course. Her research interests include artificial intelligence, interactive evolutionary computation, and usability engineering.

PLACE PHOTO HERE

Gerry Dozier Gerry Dozier is the director of the Applied Computational Intelligence Lab and an Associate Professor in the Computer Science & Software Engineering Department at Auburn University. Gerry is a member of the IEEE Computational Intelligence Society's Technical Committee on Evolutionary Computation as well as an associate editor for the IEEE Transactions on Evolutionary Computation and the Journal for Intelligent Automation and Soft Computing (AutoSoft).