International Journal on Artificial Intelligence Tools
Vol. 12, No. 1 (2003) 17–35

World Scientific
www.worldscientific.com

EVOLUTIONARY OPTIMIZATION OF GAUSSIAN WINDOWING FUNCTIONS FOR DATA PREPROCESSING

DALE E. COURTE
University of Dayton, Computer Science
300 College Park, Dayton OH 45469

MATEEN M. RIZKI
Wright State University, Computer Science and Engineering
Dayton, OH 45435

LOUIS A. TAMBURINO
AFRL/SNAT, Wright-Patterson, AFB, 45433-7321

RICARDO GUTIERREZ-OSUNA
Texas A&M University, Computer Science
College Station, TX 77843

Received 16 April 2002
Accepted 6 June 2002

The average classification accuracy of an odor classification system is improved using a genetic algorithm to determine optimal parameters for feature extraction. Gaussian windowing functions, called "kernels" are evolved to extract information from the transient response of an array of gas sensors, resulting in a reduced set of extracted features for a linear discriminant pattern classification system. Results show significant improvements are achieved when compared to results obtained using a predetermined and fixed set of four bell-shaped kernels for every sensor. Examination of the evolved kernels reveals the areas of the sensor responses where discriminating information was identified. A novel data migration approach during training helps prevent overtraining, and the fitness measure chosen incorporates adjustments for both population diversity and solution complexity. A variety of adjustable parameters, including the definition of a time-varying dynamic weighting factor, encourage experimentation in order to appropriately tune the sampling methods and fitness measure.

Keywords: Evolution; genetic algorithms; electronic nose; classification; pattern recognition; feature extraction.

1. Introduction

When an array of gas chemical sensors is presented with an odor, the result is a set of time-based responses. Most responses consist of a rapidly changing transitional phase followed by a steady state, as shown in Figure 1, where the response is sampled at 1 Hz

for one minute. When such data is used as input to a pattern analysis engine, a system known as an electronic nose results [1]. A challenge in most pattern classification problems is preprocessing the sensor responses to reduce the amount of data to be considered by the classifier. Gutierrez-Osuna and Nagle [2] describe one such system and discuss findings on several preprocessing techniques.

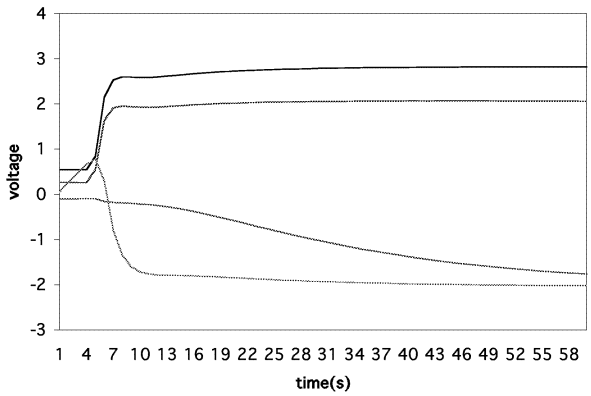

Fig. 1. Response of Four Sensors to a "sniff". Each sensor is sampled once per second for one minute.

The work discussed here obtains performance improvements by focusing on just one of the three aspects of data preprocessing discussed in [2], that being compression via windowed time slicing. Gutierrez-Osuna and Nagle use four identical bell-shaped curves, or "kernels", as windowing functions for each sensor. Each of these kernels is multiplied by the sensor response and integrated with respect to time, resulting in four features extracted for each sensor response. The result is a reduced set of four features per sensor response that captures information in the transient of the response as well as the eventual steady state. Classification is then accomplished by applying a standard linear discriminant analysis/K nearest neighbor (LDA/KNN) approach [3]. LDA is one of many popular statistical methods of dimensionality reduction [4], and nearest neighbor classifiers have long been recognized as an effective approach to nonlinear classification problems [5]. Note that all sensors are treated identically using this method, so individual characteristics evidenced by a given sensor's response to a particular classification problem are not considered. It is reasonable to consider that the information content of different sensors may be centered at different points in the signal.

Rather than consider each sensor identically, using a fixed set of bell-shaped curves as time slicing functions (Figure 2), a different set of kernels is evolved for each sensor, where each kernel is a Gaussian exponential function with a given mean and standard deviation. Similar Gaussian functions have been used in morphological recognition systems [7]. A different set of kernels is evolved for each sensor and the number of kernels per sensor is not fixed, but allowed to vary from zero to a user-defined maximum. The amount of available training data helps guide the choice of this limit, as it needs to be large enough for meaningful features to be discovered while small enough as to not result

in over-training. In this case, the danger of over-training is the detection of false separations resulting from too many features. The goal is therefore two-fold:

- Increase the classification accuracy of the electronic nose through the extraction of a more effective feature set.
- Reduce the overall number of extracted features used for the LDA/KNN classifier.

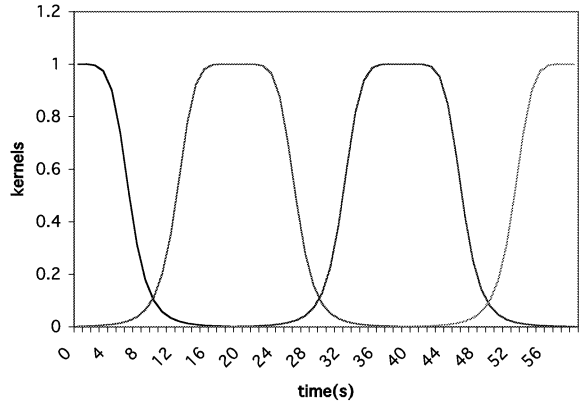

Fig. 2. Bell-shaped Kernels Used in [2] for Windowed Time Slicing.

Evolutionary techniques have been applied to numerical optimization problems for decades, beginning with Bremermann [6]. Evolution is often applied to the problem of feature selection or extraction. The *genetic programming* (GP) approach of Koza [8] is chosen in [9] and [10] for its ability to evolve complex function definitions from a set of primitive operations. Others have used *genetic algorithms* (GA) [11] [12] for the selection and weighting of features [13] [14] [15], or for the evolution of polynomial extractors [16]. And the unique strengths of different evolutionary methods are often combined into effective hybrid feature extractors [17] [18] [19] [20]. Here, a real-valued GA with a unique genotype for the problem at hand has been chosen.

2. The Problem

The domain problem used in this work is the classification of odors with an electronic nose. In particular, we employ a database of fruit juice odors that was reported in [2] and [21]. The dataset consists of equal numbers of samples representing seven types of fruit juice , for a total of 280 data samples. Each sample includes sensor responses from 15 different sensors, each response consisting of 60 readings (1 Hz sampling over one minute). Therefore, each sample consists of 900 values, making feature reduction an essential preprocessing step.

Due to sensor characteristics, the data is noisy, and exhibits apparent shifts in DC offset and amplitude (caused by drift) between different samples, especially between samples taken on different days. The data was compiled over four days, with an equal

number of "sniffs" from each class sampled each day. These facts, along with the small size of the data set, provide a challenging preprocessing problem. Features must be extracted that preserve the real information content of the signals while minimizing the effects of noise and drift, and effective data partitioning and sampling methods must minimize the errors often resulting from small data sets used in training. Where appropriate, these methods are tuned through the use of adjustable parameters, making the environment quite flexible.

3. Genetic Algorithms

First defined by Holland [11] [12], Genetic Algorithms (GA's) have since been applied to a number of difficult optimization problems. Though there are many variants of Holland's original concept, all GA's iteratively alter a population of candidate solutions, each represented by an encoding called a *chromosome*. Each such iteration, or *generation*, introduces new chromosomes produced based on the information in one or more existing chromosomes through operations known as *mutation* and *crossover*. These new chromosomes compete with existing ones for survival based on a *fitness function* that estimates their relative success.

While Holland represented chromosomes as bit strings, developers of GA's today choose a representation for chromosomes that is most appropriate for the problem at hand. The representation chosen here is a collection of chromosomes that specify a number of groups of Gaussian kernels, one for each gas sensor. Figure 3 is a graphical representation of one such group. Both a mutation and crossover operation are included in the GA, the details of which are discussed later.

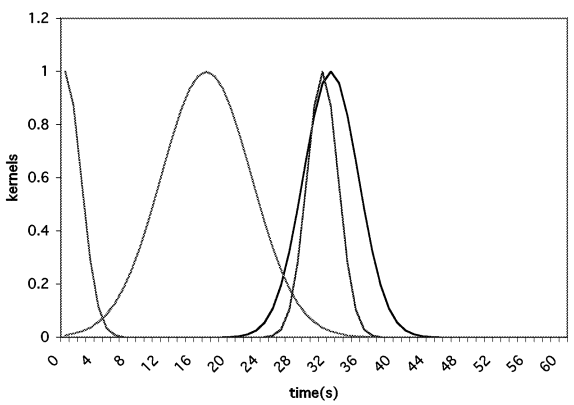

Fig. 3. Phenotype for One Sensor. Four unique Gaussian kernels are shown.

Gutierrez-Osuna [21] applied a basic GA to this problem in order to eliminate certain sensors from consideration, but the four fixed kernels of Figure 2 were still used for all sensors not eliminated. The objective of this was not only improving classification performance but also customizing arrays with few sensors for specific applications. In contrast, the present work allows the GA to select the location, width and number of kernels independently for each sensor. Note that our approach could also be used to

customize small sensor arrays by adding a term to the fitness function that penalized selection of kernels from many different sensors.

4. System Description

Further details of the GA are discussed here. Key decisions and basic algorithms are discussed in five areas of interest.

4.1 Data sampling

The particular characteristics of this data set, along with its relatively small size, led to many problem-specific decisions about the sampling of data for training and testing. Recall that this classification problem is complicated by the fact that the data samples are noisy and subject to "drift", a phenomenon that is most pronounced between samples collected on different days, and that samples were collected over four days. While little can be done in this system to reduce noise, the effect of drift can be reduced by ensuring that the data sets used contain an equal (or at least near equal) number of samples collected each day for each class.

There are 280 classified data samples available, representing seven different odors. For each odor, there are forty samples, ten from each day. Initially, the data is divided into 5 segments, each containing 56 samples - 2 examples per class and day of data collection. Other than this restriction, the exact members chosen for each segment is not fixed from run to run. After the segments are determined, one is chosen as the holdout test data, and the remaining four are used for training during the evolutionary process.

Due to the small amount of data available, it is important to guard against over-training the system to a particular choice of the 4 data segments that comprise the training data set. Therefore, during each generation g, there is a probability that samples will be permitted to migrate between the 4 segments. It is assumed the samples in each segment are sorted first by class, then by day. Given the makeup of the segments, this can be ensured when the original partitioning is done. Because the equal distribution of classes and days in each segment must be maintained, a migration consists of a simple "swap" of two samples from the same index in two randomly chosen segments. The probability a migration will occur is a user-defined parameter known as the *migration rate*. It is desirable that the migration rate be relatively small to ensure that only minor changes in the population characteristics occur between successive generations. This ensures that changes occurring during one generation will still be valuable in the next generation.

4.2 Representation

A population P, of fixed size n_P, is maintained throughout execution. The genotype for one member of the population is a ragged array where the rows represent n_S chromosomes, where n_S is the number of gas sensors. Each row in this ragged array contains between zero and n_{Kmax} pairs of values, one for each kernel representing the mean μ and standard deviation σ of the Gaussian kernel. n_{Kmax} is a predetermined maximum number of kernels allowed per sensor. The genotype of one possible member of the population is shown in Table 1. The phenotype for just one of that member's chromosomes is what is depicted in Figure 3.

count	μ	σ	μ	σ	μ	σ	μ	σ
3.00	26.00	3.77	34.00	6.32	48.00	3.35		
4.00	32.00	3.32	0.00	1.92	17.00	5.43	31.00	1.88
0.00								
1.00	35.00	9.31						
3.00	3.00	8.87	50.00	4.23	8.00	4.62		
2.00	13.00	4.52	30.00	3.12				
1.00	6.00	5.07						
1.00	46.00	3.98						
2.00	33.00	7.94	28.00	5.18				
3.00	3.00	4.42	59.00	3.22	36.00	8.67		
2.00	44.00	6.52	34.00	7.88				
1.00	3.00	6.53						
1.00	45.00	7.50						
3.00	17.00	9.58	33.00	8.54	9.00	7.25		
2.00	44.00	3.97	44.00	3.97				

Table 1. Example Genotype. Count indicates the number of kernels per sensor and a μ-σ pair identifies each kernel.

4.3 Dynamic weighting factor

The GA consists of a primary loop that completes once for each generation. In order to promote diversity and regulate complexity early in the evolution, and yet allow the population to become stable and consistent late in the evolution, a dynamic weighting factor is used. The weighting factor is a user-chosen, time-varying function, and is used to promote "big steps" in the evolutionary search process during early generations, while promoting "fine tuning" of results later in the process. To accomplish this, a weighting function that decreases monotonically from 1.0 to a minimum no less than 0.0 is desirable. This factor will affect data sampling and reproduction, as well as elements of the fitness measure. Equation 1 defines the weighting factor used to produce the results discussed later in this paper.

$$w(g, g_{\max}) = 1 - \frac{\log_e(g)}{\log_e(g_{\max})} \tag{1}$$

Here g is the current generation number, and g_{max} is the predetermined maximum number of generations the system will run. Use of the natural log allows the factor to decrease rapidly during early stages of the evolution and more slowly as the system converges on a solution. We experimented with several different functions for the weighting factor and chose this one because it consistently produced favorable results. Figure 4 shows the weighting factor for g_{max}=300.

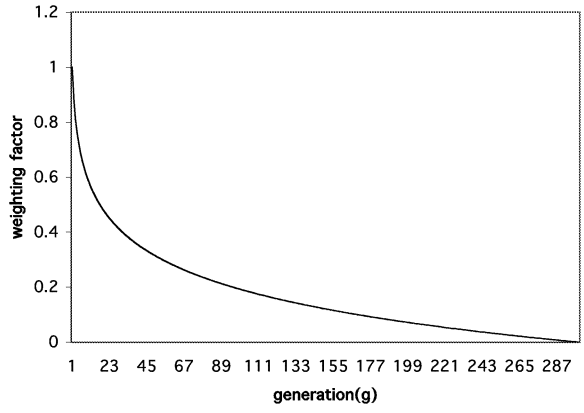

Fig. 4. Dynamic Weighting Factor $w(g, g_{\max})$.

4.4 Evolutionary process

Prior to application of LDA/KNN for a member of the population, the kernels comprising that member must be used to extract features from the raw sensor data for each sample in the training data. Given a member of the population, having n kernels defined for sensor j (represented by $[\mu_1, \mu_{2,} \ldots, \mu_n]$ and $[\sigma_1, \sigma_{2,} \ldots, \sigma_n]$), then a sensor response vector $\mathbf{S^j}$ of length m can be reduced to a feature vector $\mathbf{R^j}$ of length n as shown in equation 2.

$$\begin{aligned} &G(x,\mu,\sigma) = e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \\ &\mathbf{W^i} = \left[G(1,\mu_i,\sigma_i), G(2,\mu_i,\sigma_i), \ldots, G(m,\mu_i,\sigma_i)\right], i = 1..n \\ &\mathbf{R^j} = \left[\mathbf{S^j} \bullet \mathbf{W^1}, \mathbf{S^j} \bullet \mathbf{W^2}, \ldots, \mathbf{S^j} \bullet \mathbf{W^n}\right] \end{aligned} \tag{2}$$

Here, $G(x,\mu,\sigma)$ is the Gaussian exponential function that defines the windowing function for a given μ and σ, and $\mathbf{W^i}$ is a vector representing the values of the i^{th} windowing function for x=[1..m]. Each resulting feature in a vector $\mathbf{R^j}$ can be considered a summary of the j^{th} sensor's activity focused at a given moment in time as indicated by the mean of the corresponding kernel. How closely the measure is focused on that moment is determined by the width of the kernel defined by its standard deviation.

Training this system is computationally intense. The LDA/KNN algorithm is used in a four-fold cross-validation. For each generation, the LDA/KNN algorithm is carried out four times for every member of the population. In turn, each of the four segments of training data is used as a validation set, while the remaining three are used to compute the LDA eigenvectors. The KNN classifier is then used to classify the samples in the validation set, and compute an accuracy measure (percent correct) for each fold. In each generation, the minimum classification accuracy A_i of the four folds is used as an accuracy measure for each member i of the population. This serves as a basis for our fitness measure. Two adjustments are made, each of which decreases over time with the dynamic weighting factor $w(g,g_{max})$.

Given the goal of reducing the overall number of extracted features, a user-adjustable *complexity threshold* value t is set for the total number of kernels in a member of the population. Beyond that limit, the complexity adjustment ca penalizes the i^{th} member of the population during generation g (equation 3).

$$ca(i,g,g_{\max}) = -\max\left(\left(\sum_{j=1}^{15} C_j^i\right) - t, 0\right) \times w(g,g_{\max}) \tag{3}$$

Here $\boldsymbol{C}^i$ is a vector containing the kernel counts representing member i of the population, one count for each of the 15 sensors. Note the dynamic weighting factor is included in the calculation, causing this penalty to ease over time. This focuses the system on evolving good, small kernel sets early, and encouraging incremental improvements in later generations.

It is also desirable to ensure during early stages of evolution that a large portion of the problem space be explored so that early convergence to a local maximum is less likely. Therefore, members of the population that differ from the individual with the highest fitness (after applying the above adjustments) are rewarded. This improvised measure was chosen mainly for its computational simplicity. For each member i of the population the vector representing the number of kernels for each sensor are compared for equality to the vector representing the same components for the most accurate individual. The cells that differ are counted to compute a difference value. The Diversity adjustment da for population member i during generation g is defined in equation 4.

$$da(i,a,g,g_{\max}) = \left(\sum_{j=1}^{15} (C_j^i \neq C_j^a)\right) \times w(g,g_{\max}) \tag{4}$$

Here $\boldsymbol{C}^i$ is as defined above, and population member a is the member achieving highest training accuracy. Assume the operator "$\neq$" returns one when its operands are not equal, zero otherwise. Again, the adjustment is relaxed over time by the dynamic weighting factor. The diversity adjustment for the most fit individual is set to the value of the maximum adjustment awarded to others in the population to ensure that individual maintains the top ranking. More sophisticated diversity measures involving similarity measures between kernel sets are a topic for future research.

The fitness measurement is complete after the complexity and diversity adjustments are made. The final fitness $f(i,g)$ of member i of the population during generation g is then defined by equation 5.:

$$\begin{aligned} f(i,a,g,g_{\max}) = w_A A_i + w_{ca}\, ca(i,g,g_{\max}) \\ + w_{da}\, da(i,a,g,g_{\max}) \end{aligned} \tag{5}$$

where w_A, w_{ca}, and w_{da} are weighting factors. (Current experiments use equal weighting factors.)

Feature reduction methods are often categorized as either *wrapper* or *filter* approaches. In pattern recognition systems, the difference is distinguished by whether or not the back-end classification scheme is used to analyze the value of a given feature set. Wrapper approaches use the classifier and training data to estimate the accuracy of each potential feature set tested, while filter methods evaluate subsets on less costly measures that do not involve the classification process [22]. In this work, the fitness function involves both performance against training data as well as information that is independent of that performance. Some have called this a *filter+wrapper* approach [23].

Each generation, allowing only selected individuals to survive and replacing the remainder with offspring created through mutation or crossover alters the population. A modified elitist selection strategy [24] is used. The population is first ranked by descending fitness measure. Then the population is divided into three distinct groups:

- *choice* members – These are the most fit members of the population, and they are guaranteed to survive into the next generation.
- *desirable* members – These are reasonably fit members of the population, and they each have a probability of survival S_d.
- *undesirable* members – These are the least fit members of the population, and have a probability of survival S_u.

The size of each of these groups, as well as the probabilities S_d and S_u are user-defined parameters. The goal is to achieve a high probability of maintaining good genetic material in the evolution. The end result is the division of the population P into two subsets, P^s and P^n, of survivors and non-survivors respectively.

4.5 Crossover and mutation

For each offspring created, the probability of it being the result of crossover is a user-defined *member crossover rate,* R_x. The probability that it is the result of mutation is then simply 1- R_x. In either case, the first step is to select a random individual from the survivors' subset P^s.

For crossover, a second individual is chosen randomly from the entire population $P=P^s \cup P^n$. This is a "2nd chance" of sorts for non-survivors, allowing them to contribute some genetic material to the next generation. It is arguable this more closely resembles evolution in nature, where longevity is not a requirement for participation in reproduction.

Recall that an individual consists of several chromosomes, one for each gas sensor. Once two individuals have been chosen as parents, crossover occurs only between pairs of chromosomes defined for the same gas sensor from the two parents. For each such pair of chromosomes having length n and m, with $n \le m$, a uniform crossover [25] is executed:

- Pairs of $<\mu,\sigma>$ values are chosen for inclusion in positions $[1..n]$ of the offspring with equal probability from the same positions in each of the parents.

- Pairs of <μ,σ> values from positions [n+1..m] from the longer parent are chosen for inclusion in the offspring with a 0.5 probability.
- This results in offspring having an average size of (n+m)/2.

If mutation is chosen, there are a total of six different mutations that may occur, one being the replacement of the entire set of chromosomes with a new set generated at random. The probability of this occurring is equal to the current value of the dynamic weighting factor $w(g,g_{max})$. This promotes diversity in the early stages of the evolution when the weighting factor value is high, and stability later when the weighting factor is low.

If the entire chromosome set is not replaced, individual chromosomes are selected with a user-defined probability known as the *chromosome mutation rate* for one of the five remaining kernel mutations. Once a chromosome is selected, the following mutations are possible:

- *kernel replacement* - One of the kernels in the chromosome is replaced with a new kernel.
- *kernel insertion* - A new kernel is added to the chromosome.
- *kernel deletion* - A randomly selected kernel is deleted from the chromosome.
- *μ adjustment* - The μ value of a randomly selected kernel is adjusted by a random integer value in the range [–6..+6].
- *σ adjustment* - The σ value of a randomly selected kernel is adjusted by an random real value in the range [–0.1..+0.1].

The μ and σ values for all new kernels are randomly generated within predefined bounds. The extent of insertion, deletion and adjustments are all also regulated by such bounds. Once selected for kernel mutation, only one of the five is applied. The probability of each of these possibilities is provided by the user in the form of a *mutation selection vector* $\mathbf{M}^{s}$, holding one value in the range [0..1] for each of the above possible mutations. Of course, the sum of this vector should be 1.

4.6 Termination and output

Termination of the GA occurs either when an individual is produced that achieves 100% accuracy during the four-fold training, or when a predetermined number of generations have elapsed. Using the fruit juice database described earlier, accuracy never reached 100% during training.

Once the evolution is complete, the highest ranked individual is used in the LDA algorithm, this time using data from all four of the training segments in the LDA computation of eigenvectors. The holdout test data is then classified using the KNN classifier, and the resulting test accuracy is compared to the performance of the four fixed benchmark kernels from Figure 2.

5. Results

A series of twenty replicates of the experiment were completed. Below are the user-defined parameters that were used for these tests. These settings were determined experimentally via adjustments made between independent sets of runs.

- Each test was run for 300 generations.
- Population size was 20.
  - Number of choice members chosen was 5.
  - Number of desirable members chosen was 7, with P_d=0.9.
  - Number of undesirable members chosen was 8, with P_u=0.1.
- The migration rate was 0.02.
- The dynamic weighting factor is $w(g, g_{\max}) = 1 - \dfrac{\log_e(g)}{\log_e(g_{\max})}$
- The number of kernels per chromosome was restricted to the range [0..5].
- The complexity threshold was 30.
- The member crossover rate was 0.5.
- The chromosome mutation rate was 0.2.
  - The mutation selection vector $\mathbf{M}^s$ was [0.15, 0.15, 0.15, 0.275, 0.275].
- The σ values for kernels were restricted to the range [1.5..10.0].
- The μ values for kernels were restricted to the range [0..60].

During training, significant improvement occurred in the average accuracy during training of the top 50% of the population. Figure 5 shows this improvement over 300 generations of one of the 20 runs. The fluctuations from generation to generation are due in part to the data migration that occurs on each pass, causing successive generations to be evaluated on differing training and validation segments. Experimentation showed that more generations did not often result in significant additional improvements and that often caused the system to become over-trained to the training set.

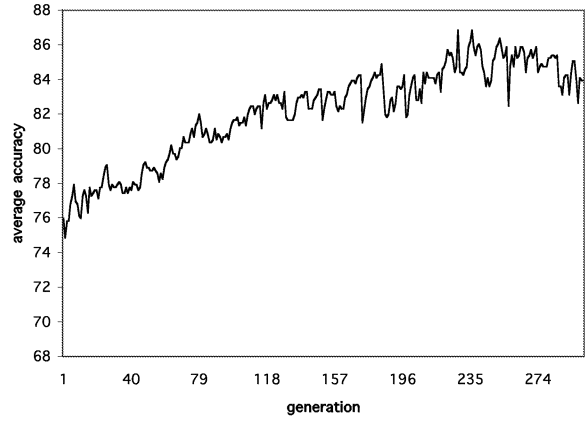

Fig. 5. Mean Accuracy v. Generation.

5.1 Overall performance

The results of these tests show that evolved kernels can outperform the four fixed benchmark kernels when applied to this data. Table 2 summarizes performance against the benchmark kernels in the 20 tests, while Figure 6 graphically depicts the performance of each test relative to the benchmark. There is an average improvement in classification accuracy of 7.07%. The best result showed an accuracy improvement of 18.8%, while the worst degraded performance by only 4.26%. Analyzed in terms of elimination of errors, this equates to an average error reduction or 26.3%. In addition the total number of features extracted during data preprocessing was reduced by an average of 50%. (Recall the benchmark kernels always result in 60 features). In order to ensure these results are not dependent on a specific choice of training and test data, each trial shown represents a new sampling of the data. This accounts for the variation of results using the baseline kernels. A simple one tail, pair-wise t-test applied to the accuracy vectors for the benchmark and evolved kernels reveals a 99% confidence of average improvement greater than 2.9%.

Trial	Benchmark Kernels (% correct)	Evolved Kernels (% correct)	Improvement	Complexity* (total kernels)
1	76.7857	80.3571	4.65%	26
2	78.5714	89.2857	13.64%	30
3	80.3571	83.9286	4.44%	32
4	75.0000	76.7857	2.38%	33
5	76.7857	83.9286	9.30%	30
6	76.7857	83.9286	9.30%	32
7	78.5714	82.1429	4.55%	27
8	80.3571	85.7143	6.67%	28
9	76.7857	80.3571	4.65%	29
10	76.7857	83.9286	9.30%	31
11	76.7857	82.1429	6.98%	32
12	76.7857	87.5000	13.95%	29
13	76.7857	80.3571	4.65%	27
14	78.5714	87.5000	11.36%	34
15	83.9286	80.3571	-4.26%	30
16	85.7143	83.9286	-2.08%	29
17	75.0000	85.7143	14.29%	29
18	78.5714	92.8571	18.18%	26
19	82.1429	82.1429	0.00%	28
20	75.0000	82.1429	9.52%	30
σ	2.9127	3.6596		
Ave	78.30	83.75	7.07%	29.60
Min	75.00	76.79	-4.26%	26.00
Max	85.71	92.86	18.18%	34.00

* Complexity using benchmark kernels is 60.

Table 2. Summary of results.

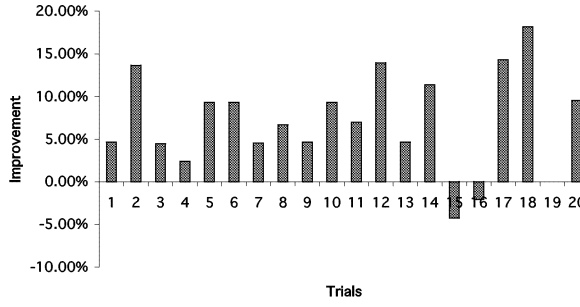

Fig. 6. Mean. Accuracy Relative to Benchmark (20 tests).

5.2 Observed details

The overall performance summarized in the previous section indicates there is an advantage to evolving custom kernels for data preprocessing in this problem. What follows are observations about the nature of the evolved kernels themselves. With just 20 test runs applied to a single problem, it is doubtful that any definitive conclusions can be drawn from these observations, but many are interesting enough to warrant further study and suggest continued research.

While it is clear from Table 2 that the average complexity (measured as number of evolved kernels) is reduced from four kernels per sensor to around two per sensor, more interesting information can be obtained from closer examination of kernel counts evolved for each sensor over the 20 tests. Figure 7 provides insight into how much useful information may be provided by each of the 15 sensors. The figure is a compressed view of 15 graphs, and therefore requires some explanation. Included is a histogram for each sensor containing six bars representing the six possibilities for kernel counts for that sensor. (Recall that anywhere from 0 to 5 kernels could be evolved for each sensor.) Sensors 0-15 are shown in order, arranged from left to right and top to bottom. The height of each bar represents the number of test runs (out of 20) that resulted in that count. The bars range in height from 1 to 12. This graphically shows that the kernel sets for some sensors tended to evolve more kernels than others, indicating those sensors may contain useful information at more locations within their responses. Also of note is that for a few sensors evolution resulted on several occasions in no kernels at all (i.e. the 9^{th}, 11^{th} and 12^{th} sensors), and that for some sensors there was a tendency toward a particular kernel count, as in the 2^{nd}, 4^{th}, and 12^{th} sensors, where at least half of the 20 tests resulted in the same kernel count.

Examining only the number of kernels generated for each sensor, while easy to summarize over the 20 test runs, provides little information about the nature of the evolved kernels themselves. Figure 8 shows one example of complete sets of evolved kernels for all 15 sensors. For the purposes of illustration, the sensors' responses for one specific "sniff" have been normalized and superimposed on the graphs. The graph clearly illustrates that the evolved kernel sets are very different from the fixed kernels of Figure 2, presumably focusing more precisely on the discriminating areas of each sensor's response.

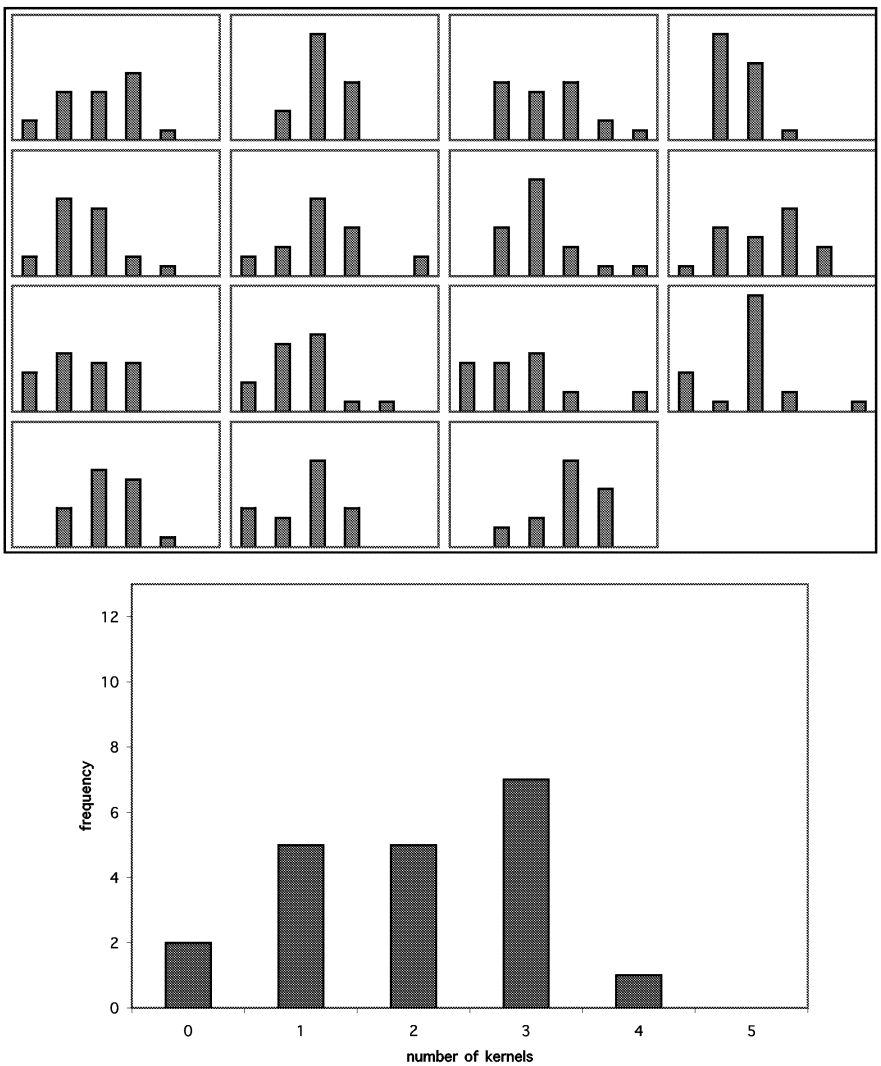

Fig. 7. Kernel Count Frequencies for Sensors over 20 Runs. Large figure is labeled expanded view of sensor 1.

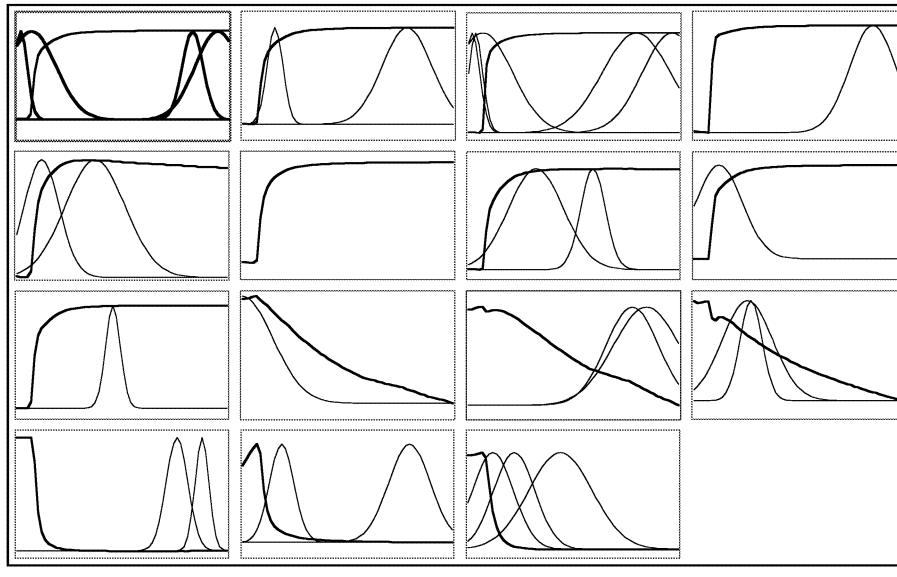

Fig. 8. Evolved Kernels for 15 Sensors with one "Sniff". For illustrative purposes, sensor responses have been scaled and shifted to align with the kernels.

Another interesting view involves looking at the kernels evolved for a single sensor over the twenty test runs. It is reasonable to expect that similar kernels might appear in different test runs, since the goal here is to discover near-optimal kernel sets. However, given no prior knowledge as to how many optimal sets may exist, there is no guarantee that such similarities would be apparent. Figure 9 shows the kernel sets for the 3rd sensor over the twenty test runs. Graphs including similar kernels are highlighted. Here, "similar" means the kernels focus attention on regions of the sensor response that overlap considerably.

Larger test suites and tests against different data sets may be necessary to determine the significance of these observations, but the observations are nonetheless notable. It is conceivable that extensive testing could determine what subset of sensors is most useful for a particular problem, or whether or not a given sensor's signal transient is more or less important than its eventual steady state value. There are several kernels that appear to recur (or at least nearly recur) in the sets, indicating these kernels may provide substantial discriminating power.

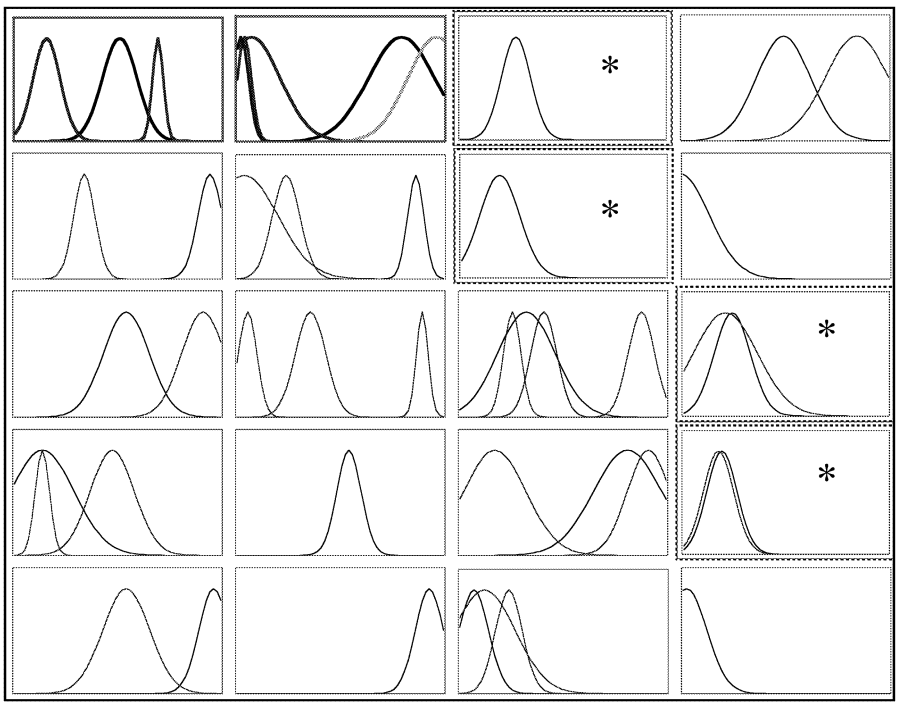

Fig. 9. Kernel Sets for Sensor #3 (20 runs). Highlighted (*) graphs show similar kernels evolved during separate runs.

The widths, positions, and counts of evolved kernels provide useful information, but those kernels are only useful if they provide information useful in discriminating between samples from different classes. It is therefore useful to look at the output of individual kernels applied to input sensor responses. Figure 10 shows the response of one evolved kernel set for the 2nd sensor to each of the 280 samples in the complete data set, arranged by class. The vertical dividers mark the class boundaries. Figure 11 does the same for an evolved kernel set for the 15th sensor. Both of these kernel sets were the result of the same test run. The seemingly periodic nature of the graphs within each class is due to the "drift" discussed earlier in Section 2 and the order in which the samples are presented in the graph. Recall that the drift is most apparent between days. From left to right, samples are presented sequentially from days 1, 2, 3, and 4 in a repeating cycle, resulting in a visual indication of how this drift helps make this a difficult classification problem. It is clear from examining just these two kernel sets that each kernel shown provides unique response characteristics, some having more apparent individual discriminating power than others.

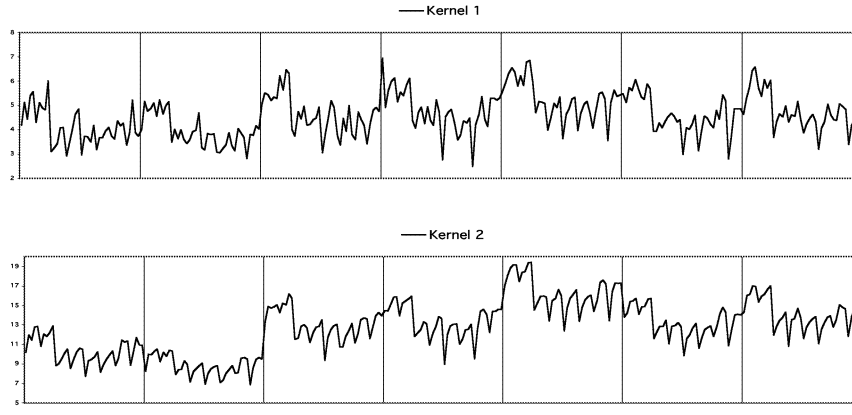

Fig. 10. Sensor #2 (2 kernels) Response by Class. Note that scaling differs from kernel to kernel.

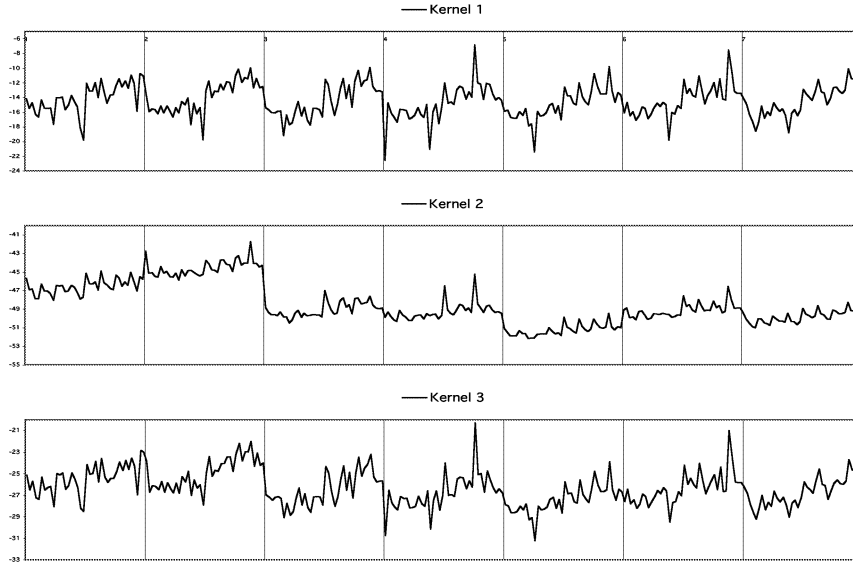

Fig. 11. Sensor #15 (3 kernels) Response by Class. For each sample, the output (extracted feature) from each kernel is shown. Vertical dividers separate samples from different classes.

6. Conclusions

The system demonstrates that improved data preprocessing kernels for an electronic nose can be evolved given a system that provides the user with adjustable parameters to facilitate tuning. Not only can classification accuracy be increased, but also the number of features extracted during preprocessing can be decreased significantly, lowering the size of the feature space used for LDA/KNN classification. Not only was overall classification accuracy observed, but also the output of individual kernels. In addition,

trends apparent in the evolution of kernel sets for specific sensors have been identified, which provide information as to what the performance characteristics of each sensor are for this particular problem. While the size of this test was limited, we believe the results do warrant further study. Future work could apply this approach to different data and applications.

Acknowledgements

Funding for this research was provided by the U.S. Air Force AFRL/SNAT grant F33615-99-C-1441, "Evolving Pattern Recognition Systems". Hardware was made available for these computations thanks to Dr. Oscar Garcia (grant NSF-9601670) and Dr. Francis Quek (grant NSF-KDI-9980054, "Cross-Modal Analysis of Signal and Sense: Multimedia Corpora and Computational Tools for Gesture, Speech, and Gaze Research").

References

[1] J.W. Gardner P.N. Bartlett, "A Brief History of Electronic Noses", *in Sens. Actuators B*, pp. 18-9, 211-220, 1994.

[2] Ricardo Gutierrez-Osuna and H. Troy Nagle, "A Method for Evaluating Data-Preprocessing Techniques for Odor Classification with an Array of Gas Sensors", in *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Vol. 29, No. 5, pp. 626-632, Oct. 1999

[3] Robert Schalkoff, *Pattern Recognition, Statistical, Structural and Neural Approaches*, John Wiley and Sons, Inc., 1992.

[4] A. Jain, P. Duin, and J. Mao, "Statistical Pattern Recognition: A Review", *IEEE Transactions on PAMI* 22(1), pp. 4-37, 2000

[5] T.M. Cover, P.E. Hart, "Nearest Neighbor Pattern Classification", *IEEE Transactions on Information Theory*, IT-13, 21-27, January 1967

[6] H.J. Bremermann, Optimization through Evolution and Recombination", *Self-Organizing Systems*, M.C. Yovits *et al*, eds., Spartan, Washington DC, 1962

[7] M.M. Rizki, L.A. Tamburino and M.A. Zmuda, "Evolution of Morphological Recognition Systems", *Evolutionary Programming IV - Proceedings of the 4th Ann. Conf. on Evolutionary Programming*, J.R. McDonnell, R.G. Reynolds and D.B. Fogel, Eds., pp. 95-106, MIT Press, 1995

[8] J.R. Koza, "Hierarchical Genetic Algorithms Operating on Populations of Computer Programs", *Proceedings of the 11th Annual Joint Conference on Genetic Algorithms*, pp. 768-774, Morgan Kaufmann, 1989

[9] J.R. Sherrah, R.E. Bogner, and B. Bouzerdoum, "Automatic Selection of Features for Classification Using Genetic Programming", *Proceedings of the 1996 Australian New Zealand Conference on Intelligent Information Systems*, pp. 284--287, IEEE, 1996

[10] J.R. Sherrah, R.E. Bogner, A. Bouzerdoum, "The Evolutionary Pre-Processor", *Genetic Programming: Proc. 2nd Annual Conference*, pp. 304-312, Morgan Kauffman, 1997

[11] J.H. Holland, "Outline for a Logical Theory of Adaptive Systems", in *J. ACM* 9 297-314, 1962

[12] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975

[13] W. F. Punch, E. D. Goodman, M. Pei, L. ChiaShun, P. Hovland, and R. Enbody, "Further Research on Feature Selection and Classification Using Genetic Algorithms", *International Conference on Genetic Algorithms 93*, pp. 557-564, 1993

[14] M.L. Raymer, W.F. Punch, E.D. Goodman, P.C. Sanschagrin, L.A. Kuhn, "Simultaneous Feature Extraction and Selection Using a Masking Genetic Algorithm", *Proc. 7th International Conference on Genetic Algorithms (ICGA97)*, Morgan Kaufmann, 1997

[15] M.L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn, A.K. Jain, "Dimensionality Reduction Using Genetic Algorithms", *IEEE Transactions on Evolutionary Computation*, Vo. 4, No. 2, July 2000

[16] E.J. Chang, R.P. Lippmann and D.W. Tong, "Using Genetic Algorithms to Select and Create Features for Pattern Classification", *Proceedings of the International Joint Conference on Neural Networks*, Vol. 3, 1990

[17] H. Vafaie and K. De Jong, "Genetic Algorithms as a Tool for Feature Selection in Machine Learning", *Proceeding of the 4th International Conference on Tools with Artificial Intelligence*, pp. 200-204 Arlington, VA, 1992

[18] M.M. Rizki, L.A. Tamburino and M.A. Zmuda, "Evolution of Morphological Recognition Systems", *Evolutionary Programming IV - Proceedings of the 4th Ann. Conf. on Evolutionary Programming*, J.R. McDonnell, R.G. Reynolds and D.B. Fogel, Eds, pp. 95-106, MIT Press, 1995

[19] C. Guerra-Salcedo, S. Chen, D. Whitley, S. Smith, "Fast and Accurate Feature Selection Using Hybrid Genetic Strategies", *Proc. Congress of Evolutionary Computation*, pp. 177-184, 1999

[20] L.A. Tamburino, M.M. Rizki, M.A. Zmuda, "HELPR Evolved Pattern Recognition Systems", *Proc. Of the 5th World Multi-Conference on Systems, Cybernetics and Informatics (SCI2001)*, Orlando FL, July 2001

[21] Ricardo Gutierrez-Osuna, *Signal Processing and Pattern Recognition for an Electronic Nose*, Ph.D. Thesis, North Carolina State University, 1998

[22] A.L. Blum and P. Langley, "Selection of Relevant Features and Examples in Machine Learning", *Artificial Intelligence*, 97(1-2), pp. 245-271, 1997

[23] J. Bala, K. De Jong, J. Huang, H. Vafaie, and H. Wechsler, "Using Learning to Facilitate the Evolution of Features for Recognizing Visual Concepts", *Evolutionary Computation*, 4(3), 1996

[24] K.A. DeJong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Ph.D. Thesis, University of Michigan, 1975

[25] G. Syswerda, "Uniform Crossover in Genetic Algorithms", pp. 2-9 in *Proceedings of the 3rd Intl. Conf. on Genetic Algorithms and their Applications*, J.D. Schaffer, Ed., Morgan Kaufmann, 1989