

FUZZY LOGIC COURSE PROJECT

FUZZY ROC CURVES FOR THE ONE-CLASS SVM: APPLICATION TO INTRUSION DETECTION

Paul F. Evangelista¹

1 - Rensselaer Polytechnic Institute - Department of Decision Sciences and Engineering Systems
Troy, New York 12180 - United States of America

Abstract. A novel method for receiver operating characteristic (ROC) curve analysis and anomaly detection is proposed. The ROC curve provides a measure of effectiveness for binary classification problems, and this paper specifically addresses unbalanced, unsupervised, binary classification problems. Furthermore, this work explores techniques in fusing decision values from classifiers and using ROC curves to illustrate the effectiveness of the fusion techniques. In describing an unbalanced classification problem, I am addressing a problem that has a low occurrence of the positive class (generally less than 10%). Since the problem is unsupervised, the one-class SVM is utilized. I discuss the curse of dimensionality experienced with the one-class SVM, and to overcome this problem I create subspaces of our variables. For each subspace created, the one class SVM produces a decision value. The aggregation of the decision values occurs through the use of fuzzy logic, creating the fuzzy ROC curve. The primary source of data for this research is a host based computer intrusion detection dataset. Experimental results supported with theoretical discussion of ROC curves and fuzzy logic indicates that synergistic ROC curves emerge when subspaces are orthogonal and T-conorms are utilized for fusion.

1 Introduction to the Problem

The purpose of this paper is to illustrate synergistic combinations of multiple classifiers for the unbalanced, unsupervised binary classification problem. The data explored in this paper is commonly referred to as the Schonlau et. al. or SEA dataset, originally discussed in [9, 10, 11, 24, 23]. Although this is a host based computer intrusion detection dataset, the applications of this work extend beyond computer intrusion detection.

Combinations of multiple classifiers (CMC) is an active area of research today. Given the numerous classification techniques and vast problem area domain, research within this field seems only bounded by creativity and computational power. A particularly interesting problem that involves CMC is the unbalanced, unsupervised binary classification problem. Consider the following example that I will refer to as the “airport security problem”. An airport security system exists in layers, all the way up to and including the aircraft while it is airborne. Each layer in this security system can be considered a classifier; the objective of each layer is to determine whether or not a bad guy, or intruder let us say, is attempting to breach security. How should these classifiers be arranged? How can the results of classifiers be combined to achieve synergistic results?

An unsupervised classifier is handicapped by the fact that it cannot learn from true positive (intruders) examples. The only examples available to learn from are true negatives (non-intruders). Furthermore, the problem I am interested in is unbalanced. This means that the frequency of intruders is very small; in an airport, this number of true positives is a fraction of a percent. Some airports work for years without experiencing an intruder. Yet the cost of not identifying a true positive can be catastrophic, and the cost of falsely identifying non-intruders, or having too many false positives, can create is also formidable.

Given a classification problem of high dimension (perhaps >10 variables), initial experimental results indicate that the creation of subspaces and aggregation of the subspace classification decision values results in improved classification over a model that utilizes all variables at once (the

unsupervised classification model that will be utilized is the one-class SVM [22]). This is often known as the “curse of dimensionality”. Creating subspaces for outlier detection, which is essentially what I am describing, is not a new concept. However, considering this problem as a function of one-class SVM outputs to create a “fuzzy ROC curve” has not been addressed in the literature. Furthermore, it is through careful analysis of receiver operating characteristic (ROC) curves that I will measure performance. As I combine multiple classifiers, I will seek to gain synergistic improvement in our ROC curves. Fuzzy logic is the basis for the aggregation. Each classifier will create a decision value that ranges from -1 to +1, where +1 should indicate the negative (non-intruder) class, and -1 indicates the positive (intruder) class. These decision values represent a degree of membership in an intruder or a non-intruder class. The decision values must be combined to make a final decision, and fundamentals of fuzzy logic can be used to aggregate these decision values.

As mentioned earlier, unsupervised learning does not perform well in higher dimensions. A valid question would be to ask why not try a dimension reduction technique, such as principal components. Principal components work well with balanced classification problems, however caution is necessary with unbalanced problems. Often the difference between an intruder and non-intruder is subtle, and principle components can dilute the information content of the variables. This is why I am considering a different technique.

1.1 Performance Measures

The overall performance measure is the area under the ROC curve, or AUC, with a secondary performance measure of good performance at low false positive rates. Overall, the ROC curve created from the subsets should exceed the curve created by processing one lump sum of variables. This must occur to claim any element of success. The results section shows several instances where this occurred. Secondly, in the range of low false positives, where intrusion detection and security systems typically operate, performance must be good which means that in the range from 0 to 50% true positive, the false positive rate diverges slowly from 0. Examples of this are shown in the results section. Typically good low false positive range performance and increased AUC will occur simultaneously, but not always.

2 Recent Work with Masquerading Dataset

This paper explores many facets of recent research, to include intrusion detection models, outlier detection, and fusion of classifiers using fuzzy ROC curves. Recent work with classifier fusion and fuzzy ROC curves will be discussed during the presentation of the methods for that material.

Schonlau et. al. [9, 10, 11, 24, 23] conducted the original work with this data, hence the reference to the data as the Schonlau et. al. data, or SEA data. Their contributions included a thorough analysis of several statistical techniques for identifying masqueraders. Schonlau et. al. explored approaches that include: Bayes one-step Markov model, hybrid multistep Markov model, text compression, Incremental Probabilistic Action Modeling (IPAM), sequence matching, and a uniqueness algorithm[9]. Schonlau stressed the importance of minimizing false positives, setting a goal of 1% or less for all of his classification techniques. Schonlau’s uniqueness algorithm, explained in [24], achieved a 40% true positive rating before crossing the 1% false positive boundary. Wang [27] used one-class training based on data representative of only one user and demonstrated that it worked as well as multi-class training. Coull [5] applied bioinformatics matching algorithm for a semi-global alignment to this problem. Lee [19] built a data mining framework for constructing features and model for intrusion detection. Yong and Szymanski applied a recursive data mining algorithm for frequent patterns to detect intruders [26]. Evangelista et. al. [13] applied supervised learning through Kernel Partial Least Squares to the SEA dataset.

Roy Maxion contributed insightful work with this data that challenged both the design of the data set and previous techniques used on this data [21, 20]. Maxion uses a 1v49 approach in [21], where he trains a Naive Bayes Classifier one user at a time using the training data from one user

as true negative examples versus data from the forty-nine other users (hence 1v49) as true positive (masquerader) examples. Maxion claimed the best performance to date in [21], achieving a true positive rating of 60% while maintaining a false positive rating of 1% or less. Maxion also examines masquerade detection with a similar data set that contain command arguments in [20].

The one-class SVM is an outlier detection technique originally proposed in [22]. Stolfo and Wang [25] successfully apply the one-class SVM to this dataset and compare it with several of the techniques mentioned above. Chen uses the one-class SVM for image retrieval[4]. The simplest way to express the one-class SVM is to envision a sphere or ball, and the object is to squeeze all of the training data into the tightest ball feasible. This is analogous to the idea of variance reduction for distribution estimation; given a set of data, I want to estimate a distribution that tightly defines this data, and any other data that does not look the same will not fit this distribution. In other words, once the distribution is estimated, data that does not fit the distribution will be considered an outlier or not a member of the negative class. Consider the following formulation of the one-class SVM originally from [22] and also clearly explained in [4]:

If we consider $X_1, X_2, \dots, X_l \in \chi$ instances of training observations, and Φ is a mapping into the feature space, F , from χ .

$$\begin{aligned} \min_{R \in \mathbb{R}, \zeta \in \mathbb{R}^l, c \in F} \quad & R^2 + \frac{1}{vl} \sum_i \zeta_i \\ \text{subject to} \quad & \|\Phi(X_i) - c\|^2 \leq R^2 + \zeta_i, \quad \zeta_i \geq 0 \text{ for } i \in [l] \end{aligned} \quad (1)$$

This minimization function attempts to squeeze R , which can be thought of as the radius of a ball, as small as possible in order to fit all of the training samples. If a training sample will not fit, ζ_i is a slack variable to allow for this. A free parameter, v , enables the modeler to adjust the impact of the slack variables. The output, or decision value for a one-class SVM, takes on a values from -1 to +1, where values close to +1 indicate datapoints that fit into the ball and values of -1 indicate datapoints lying outside of the ball.

2.1 Curse of Dimensionality

It is commonly understood that high dimensional data suffers from a curse of dimensionality. This curse of dimensionality involves the inability to distinguish distances between points because as dimensionality increases, every point tends to become equidistant as volume grows exponentially. This same curse of dimensionality occurs in the one-class SVM. (The SVM tool used for this research is LIBSVM [3], and for the purpose of consistency I only use the linear kernel. I have experimented with the non-linear kernel options in LIBSVM, and the variance created by additional parameter tuning would distract from the fundamental message of this paper.)

Throughout these experiments I consistently Mahalanobis scale our data (by subtracting the mean and dividing by the standard deviation). The dataset contains 5000 observations and a host of variables to measure these observations (see [13, 26] for a description of variables), and I utilize 2500 observations for training and 2500 observations for testing. After eliminating all positive cases from the training data, 2391 negative cases remain which are used for training the one-class SVM. In the testing data, there are 122 positive cases out of the 2500 observations.

Figure 1 illustrates an experimental example of the curse of dimensionality where there are originally 27 meaningful variables, however meaningless probe variables (uniform (0,1) random variables) are added to create degradation. The area under the ROC curve, or AUC, will serve as a measure of classifier performance. I have included an appendix that provides a quick introduction to ROC curves, and Tom Fawcett provides an excellent discussion of ROC curves in [14] for the reader who is not familiar with ROC curves.

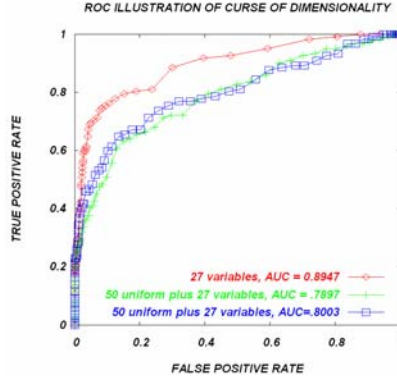


Fig. 1: Curse of Dimensionality induced by the introduction of probe variables.

3 Method to Create Fuzzy ROC Curves

I propose a technique to overcome this curse of dimensionality. The technique involves creating subspaces of the variables and aggregating the outputs of the one-class SVM for each of these subspaces.

3.1 Subspace Modeling

Intelligent subspace modeling is an important first step. Orthogonal subspaces are desired, because we are interested in subspaces that measure different aspects of the data. The idea of creating diverse classifiers is not novel [2, 17, 18, 16], however in the literature the measures of classifier diversity involve functions of the classifier output. This is feasible with supervised learning, however in unsupervised learning this is more difficult because there are no true positive examples to measure diversity against. I propose measuring diversity through the actual data. Our method involves an analysis of the correlation between principal components of each subspace. This is by no means the only measure for subspace diversity, however I have experienced good results with this model.

Given a Mahalanobis scaled data matrix \mathbf{X} , containing m variables that measure n observations, create l mutually exclusive subspaces from the m variables. Assume there are k variables in every subspace if m is divisible by l . Our experience with the one-class SVM indicates that for $k > 7$, increased dimensionality begins to degrade performance, however this is simply a heuristic and may vary depending upon the unsupervised classifier selected. For each subspace, principal components can be calculated. I will refer to the matrix that contains the principal component loading vectors (eigenvectors) as \mathbf{L} . To determine correlation between principal components, we calculate the principal component scores for each subspace, where $\mathbf{S}=\mathbf{XL}$. Let π_i represent subspace i , and consider \mathbf{S}_i as the score matrix for the π_i . Calculate the pairwise comparison for every column vector in \mathbf{S}_i against every column vector in \mathbf{S}_j , $i \neq j$. This would be the equivalent of concatenating \mathbf{S}_i for all i and calculating the correlation matrix, Σ .

We are interested in values approaching zero for every pairwise correlation across subspaces (principal components within subspaces are orthogonal and therefore their correlation is zero, as seen in Figure 3). However, there are a combinatoric number of subspace combinations to explore.

$$\text{Number of subspace combinations} = \binom{m}{k} \binom{m-k}{k} \dots \binom{2k}{k} \quad (2)$$

Equation 2 assumes that m is divisible by l , and even if this is not true the equation is almost identical and on the same order of magnitude. Our approach to search this subspace involved the implementation of a simple genetic algorithm, utilizing a chromosome with m distinct integer

elements representing each variable. There are many possible objective functions that could pursue minimizing principal component correlation between subspaces, and I utilized the following letting $q \in (1, 2, \dots, l)$:

$$\min \max_{\forall \pi_q} |\rho_{ij}| \quad \forall (i \neq j) \quad (3)$$

The fitness of each member is simply the maximum $|\rho_{ij}|$ value from the correlation matrix such that ρ_{ij} measures two principal components that are not in the same subspace.

3.2 Output Processing

After selecting the subspaces, prediction modeling begins. As mentioned previously, our choice for a prediction model is the one-class SVM with a linear kernel. However, the problem of classifier fusion still persists. Classifier fusion techniques have been discussed in [2, 17, 18, 16]. Classifier fusion is a relatively new field and it is often criticized for lack of theoretical framework and too many heuristics [17]. I do not claim to provide a solution to this criticism. Our method of classifier fusion is a blend of techniques from fuzzy logic and classifier fusion, and although it may be considered another heuristic, it is operational and should generalize to other security problems.

3.2.1 Mapping into Comparable Decision Spaces

For each observation within each subspace selected, the classifier will produce a decision value, d_{ij} , where d_{ij} represents the decision value from the j^{th} classifier for the i^{th} observation. Since the distribution of the output from almost any classification technique is questionable, we first consider a nonparametric measure for the decision value, a simple ranking. o_{ij} represents the ordinal position of d_{ij} (for the same classifier, meaning j remains constant). For example, if d_{71} is the smallest value for the 1st classifier, $o_{71} = 1$. This nonparametric measure allows comparison of classifiers without considering the distribution. However, we do not rule out the distribution altogether. We also create p_{ij} , which is the Mahalanobis scaled (normalized) value for d_{ij} . In order to incorporate fuzzy logic, o_{ij} and p_{ij} must be mapped into a new space of real numbers, let us call Λ , where $\Lambda \in (0, 1)$. This mapping will be $p_{ij} \rightarrow \delta_{ij}$ and $o_{ij} \rightarrow \theta_{ij}$ such that $\delta_{ij}, \theta_{ij} \in \Lambda$. For $o_{ij} \rightarrow \theta_{ij}$ this is a simple scaling procedure where all o_{ij} are divided by the number of observations, m , such that $\theta_{ij} = o_{ij}/m$. For $p_{ij} \rightarrow \delta_{ij}$, all p_{ij} values < -1 become -1, all p_{ij} values > 1 become 1, and from this point $\delta_{ij} = (p_{ij} + 1)/2$.

3.2.2 Fuzzy Logic and Decisions with Contention

There are now twice as many decision values for every observation as there were numbers of classifiers. Utilizing fuzzy logic theory, T-conorms and T-norms can be considered for fusion. The choice between T-norms and T-conorms depends upon the type of decision. The medical community is cautious of false negative tests, meaning that they would rather have error on the side of falsely telling someone that they have cancer as opposed to letting it go undetected. The intrusion detection community is concerned about minimizing false positives, because too many false positives render an intrusion detection system useless as analysts slog through countless false alarms. In the realm of one-class SVMs, the original decision values will take on values ranging generally from -1 to +1, where values closer to +1 indicate observations that fit inside the ball or estimated distribution (indicating non-intruders), and values closer to -1 indicate outliers (potential intruders). Consider the max and min, simple examples of a respective T-conorm and T-norm. Systems that need to be cautious against false negatives will operate in the realm of the T-norms, creating more false alarms but missing fewer true positives. Systems that need to be cautious against false positives will operate in the realm of the T-conorms, perhaps missing a few true positives but generating fewer false positives. Figure 2 illustrates the domain of aggregation operators.

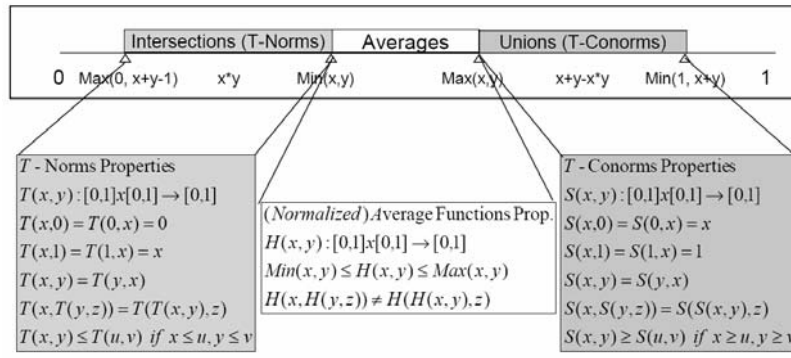


Fig. 2: Aggregation Operators

One problem with T-norms and T-conorms is that contention within aggregation is not captured. By contention I am referring to a vast difference of decision values between classifiers. However, contention can be captured and considered appropriately. Typically, if contention exists, a system needs to reflect caution. In other words, if we are minimizing false positives and contention exists in a decision, we may simply choose negative or choose a different aggregator for contentious decisions. If contention exists in a medical decision, it is likely that the initial diagnosis will report positive (cancer detected) and then further tests will be pursued. There are numerous ways to measure contention, and one of the simplest is to consider the difference between the max and min decision values. If this difference exceeds a threshold, contention exists and it may be best to choose a different aggregator or make a cautious decision.

4 Results with Masquerading Data

Experimental results involved the SEA dataset. There are $m=26$ variables and $n=2500$ observations in the training dat. For our subspace selection, there are $l=3$ subspaces creating subspaces containing 9, 9, and 8 variables respectively. For each subspace we consider three principal components. Our genetic algorithm used the fitness function shown in Equation 3, roulette wheel selection, a crossover rate of .6 and mutation rate of .01. Our number of generations = population size = 50. Figure 3 is the correlation matrix of the principal components from our selected subspaces, where $\max_{\forall \pi_q} |\rho_{ij}| = .4 \forall (i \neq j)$.

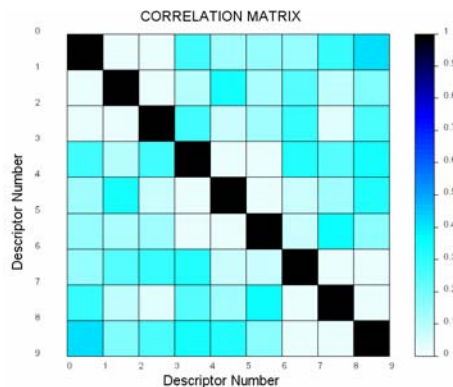


Fig. 3: Correlation matrix of subspace principal components.

Given this subspace configuration, I utilized LIBSVM to calculate the one-class SVM decision variables. Given the decision variables d_{ij} , map $d_{ij} \rightarrow o_{ij} \rightarrow \theta_{ij}$ and $d_{ij} \rightarrow p_{ij} \rightarrow \delta_{ij}$ as described in section 3.2.1. Our decision rule was to take the maximum value unless there was contention $> .5$, and in this case take the median of all decision values. The ROC curves shown in figure 4 and tabled values in Table 2 illustrate the results. The red (superior) ROC curve represents the result with some type of aggregation, the green (inferior) ROC curve represents the result without any aggregation and utilizing 26 variables for the one-class SVM input.

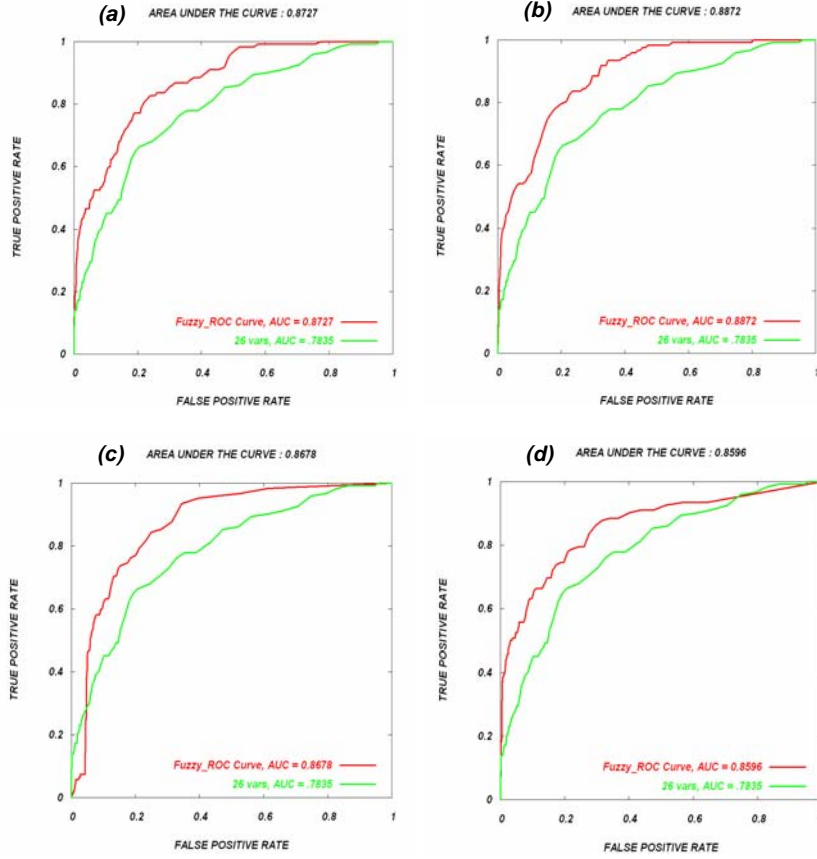


Fig. 4: ROC plots illustrating effect of different decision rules

Table 1: Decision rules for ROC plots in Figure 4

ROC Plot	Decision Rule for Each Observation (i); ($t = \text{threshold for contention}$)
(a)	algebraic sum of $(\theta_{ij})\forall j$
(b)	algebraic product of $(\theta_{ij})\forall j$
(c)	if $t < .5$, algebraic sum of $(\delta_{ij}, \theta_{ij})\forall j$; if $t \geq .5$, median $(\delta_{ij}, \theta_{ij})\forall j$
(d)	algebraic sum of $(\delta_{ij}, \theta_{ij})\forall j$

As Figure 4 illustrates, different decision rules create various outcomes. The best plot, (d), integrates all decision variables as the algebraic sum. Table 1 describes the decision rules.

These results provide interesting insight. It is my overall assumption that T-conorms provide the best rules for the type of problem addressed in this paper. However, for the ordinal decision variables (θ_{ij}) , I observed good results with T-norms. The ordinal variables are interesting in them-

Table 2: Overall Results with Best Subsets.

	$\forall \delta_{ij}, \theta_{ij}$		$\forall \theta_{ij}$		$\forall \delta_{ij}$	
	AUC	comment	AUC	comment	AUC	comment
T-norms						
minimum	.695		.8632	poor low FP	.7	
algebraic product	.5204		.887	low FP not the best	.676	
minimum with contention	.6907		.8458		.6929	
algebraic product with contention	.4999		.8363	poor high FP	.677	
T-conorms						
maximum	.8206	poor high FP	.7967		.825	
algebraic sum	.8596	best low FP	.8727		.858	
maximum with contention	.849		.8105		.86	
algebraic sum with contention	.8678	poor low FP	.8548		.887	low FP not the best

selves since there is no distance considered when comparing variables; the only aspect considered is order. This is a non-parametric method for comparison, and this is done since the distribution of the decision variables is not easy to estimate and perhaps should not be assumed. The distance based decision variables (δ_{ij}) do not perform well with T-norms. Overall, I would select ROC plot d as the best since there is exception performance for low false positive rates and the overall AUC is among the best. The idea of contention deserves much more study, and this parameter is likely to need tuning. It obviously does impact the overall results and warrants consideration, however it is not entirely understood. Lastly I illustrate an example of poor results, which should reinforce some of the claims previously made.

Table 3: Overall Results with Worst Subsets.

	AUC $\forall \delta_{ij}, \theta_{ij}$	AUC $\forall \theta_{ij}$	AUC $\forall \delta_{ij}$
T-norms			
min	.6167	.7767	.617
algebraic product	.5204	.775	.62
min with contention	.726	.787	.648
algebraic product with contention	.557	.790	.653
T-conorms			
max	.7457	.769	.7466
algebraic sum	.814	.775	.818
max with contention	.6874	.769	.6929
algebraic sum with contention	.7563	.725	.762

Table 3 illustrates the results obtained for the worst subset encountered (the configuration that scored highest for our minimization fitness function). This is interesting to observe since it is difficult to show that some type of optimality has been reached by minimizing the correlation

between principal components. Empirically, this shows that there are much worse configurations and it is worth exploring for improved subspace configurations.

5 Conclusions

This paper discusses a framework for a difficult domain of decision making: the unsupervised, unbalanced, binary classification problem with high dimensionality. It is common to encounter this domain in both the medical community and the security community. However, different risk aversion creates different policies for decisions. The framework in this paper capitalizes on theory from multivariate statistics, optimization, and information theory to present an approach for decision making and creation of such policies. Future work includes finding alternate approaches for finding optimal orthogonal subspaces and further research on decisions with contention. The goal of the research discussed in this paper is to improve our ability to find synergistic combinations of classifiers and subspaces, and more importantly that this research will grow into applications for the improvement of security policies and other policies that address unbalanced, unsupervised, binary classification problems.

References

- [1] Jinbo Bi and Kristin P. Bennett. Regression error characteristic curves. Washington, D.C., 2003. Proceedings of the Twentieth International Conference on Machine Learning.
- [2] Piero Bonissone, Kai Goebel, and Weizhong Yan. Classifier fusion using triangular norms. Cagliari, Italy, June 2004. Proceedings of Multiple Classifier Systems (MCS) 2004.
- [3] Chih Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. <http://www.scie.ntu.edu.tw/~cjlin/libsvm>, Accessed 5 September, 2004.
- [4] Yunqiang Chen, Xiang Zhou, and Thomas S. Huang. One-class svm for learning in image retrieval. Thessaloniki, Greece, 2001. Proceedings of IEEE International Conference on Image Processing.
- [5] Scott Coull, Joel Branch, and Boleslaw K. Szymanski. Intrusion detection: A bioinformatics approach. Las Vegas, Nevada, December 2001. Proceedings of the 19th Annual Computer Security Applications Conference.
- [6] James M. DeLeo. Measuring classifier intelligence. pages 694–699. Proceedings of the 2002 PerMIS Workshop, 2001.
- [7] James M. DeLeo and Gregory Campbell. Fundamentals of Fuzzy Receiver Operating Characteristic (ROC) Functions. pages 543–548. Proceedings of the 21st Symposium Interface, 1989.
- [8] James M. DeLeo and Gregory Campbell. The fuzzy receiver operating characteristic function and medical decisions with uncertainty. pages 694–699. Proceedings of the First International Symposium on Uncertainty Modeling and Analysis, 1991.
- [9] William DuMouchel, Wen Hua Ju, Alan F. Karr, Matthias Schonlau, Martin Theus, and Yehuda Vardi. Computer intrusion: Detecting masquerades. *Statistical Science*, 16(1):1–17, 2001.
- [10] William DuMouchel and Matthias Schonlau. A fast computer intrusion detection algorithm based on hypothesis testing of command transition probabilities. pages 189–193. The Fourth International Conference of Knowledge Discovery and Data Mining, August 1998.
- [11] William DuMouchel and Matthias Schonlau. A comparison of test statistics for computer intrusion detection based on principal components regression of transition probabilities. pages 404–413. Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics, 1999.
- [12] Mark J. Embrechts. *AnalyzeTM*, Version 6.86, Rensselaer Polytechnic Institute. <http://www.drugmining.com>, Accessed 4 June, 2004.
- [13] Paul F. Evangelista, Mark J. Embrechts, and Boleslaw K. Szymanski. Computer intrusion detection through predictive models. St. Louis, Missouri, November 2004. Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining and Complex Systems.
- [14] Tom Fawcett. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. Palo Alto, CA, 2003. Technical Report HPL-2003-4, Hewlett Packard.
- [15] Tom Fawcett and Foster Provost. Robust classification for imprecise environments. *Machine Learning Journal*, 42(3):203–231, 2001.
- [16] Ludmila I. Kuncheva. ‘Fuzzy’ vs. ‘Non-fuzzy’ in Combining Classifiers Designed by Boosting. *IEEE Transactions on Fuzzy Systems*, 11(3):729–741, 2003.

- [17] Ludmila I. Kuncheva. That Elusive Diversity in Classifier Ensembles. Mallorca, Spain, 2003. Proceedings of 1st Iberian Conference on Pattern Recognition and Image Analysis.
- [18] Ludmila I. Kuncheva and C.J. Whitaker. Measures of Diversity in Classifier Ensembles. *Machine Learning*, 51:181–207, 2003.
- [19] Wenke Lee and Salvatore J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):227–261, 2000.
- [20] Roy A. Maxion. Masquerade detection using enriched command lines. San Francisco, CA, June 2003. International Conference on Dependable Systems and Networks.
- [21] Roy A. Maxion and Tahlia N. Townsend. Masquerade detection using truncated command lines. Washington, D.C., June 2002. International Conference on Dependable Systems and Networks.
- [22] Bernhard Scholkopf, John C. Platt, John Shawe Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.
- [23] Matthias Schonlau and Martin Theus. Intrusion detection based on structural zeroes. *Statistical Computing and Graphics Newsletter*, 9(1):12–17, 1998.
- [24] Matthias Schonlau and Martin Theus. Detecting masquerades in intrusion detection based on unpopular commands. *Information Processing Letters*, 76(1-2):33–38, 2000.
- [25] Salvatore Stolfo and Ke Wang. One class training for masquerade detection. Florida, 19 November 2003. 3rd IEEE Conference Data Mining Workshop on Data Mining for Computer Security.
- [26] Boleslaw K. Szymanski and Yongqiang Zhang. Recursive data mining for masquerade detection and author identification. West Point, NY, 9-11 June 2004. 3rd Annual IEEE Information Assurance Workshop.
- [27] Geoffrey I. Webb and Zijan Zheng. Multi-strategy ensemble learning: Reducing error by combining ensemble learning techniques. *IEEE Transactions on Knowledge and Data Engineering*, 16(8):980–991, 2004.

APPENDIX A - Implementation of the Genetic Algorithm

The main body of the paper does not discuss the implementation of the GA because it would distract from the main points. I did address the fitness function in the main body, therefore I will not address it again here. As seen in the results section, selection of the wrong subspaces creates terrible results. The purpose of this GA is to find orthogonal subspaces through the analysis of principal components and the correlation of these principal components. Principal components from the same subspace will have 0 correlation, however principal components from different subspaces will have correlation. If principal components are correlated, then this is an indicator that the subspaces measure the same behavior. Therefore, uncorrelated principal components are desirable.

DESIGN OF THE CHROMOSOME

The chromosome consisted of 26 elements, and each of these elements represented a variable. The initial population consisted of 50 randomly ordered chromosomes. Figure 5 is an example of the chromosome encoding used in this algorithm (this is actually the best configuration discovered to date). The three subsets are extracted as shown, and then I used the *AnalyzeTM* [12] software package to calculate the principal components and correlation of the principal components across subspaces.

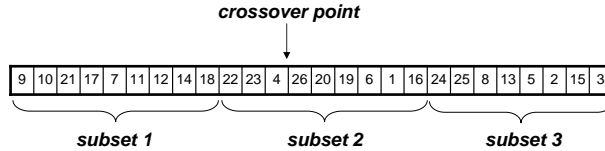


Fig. 5: Illustration of chromosome and subspaces

SELECTION, BREEDING, and MUTATION

The selection of chromosomes followed the common roulette wheel process, where chromosomes with a better fitness receive a higher probability of being selected. In order to retain the fittest chromosome found, I implemented an elitist rule to retain the single best chromosome and pass it to the next generation. I also implemented immigrants, which were two new random chromosomes who took the place of the least fit chromosomes selected for breeding. After selecting a group of chromosomes for breeding, these chromosomes had a 40% chance of directly passing to the next generation and a 60% chance of crossover. Crossover occurred at the point shown in Figure 5. This crossover option was somewhat unorthodox since a clean crossover could not occur due to the fact that every element in the chromosome must be unique. Therefore, after executing a crossover, the new chromosomes required adjustment to ensure unique elements. This adjustment occurred only on the first half of the new chromosome (before the crossover point); ie, if a redundant element occurred in the new chromosome, this would randomly be replaced by one of the variables not yet accounted for in the new chromosome.

The last step involved mutation. Mutating chromosomes also required special consideration due to constraints of chromosome elements. Each element had a 1% chance of encountering mutation. If an element was selected for mutation, the element was simply swapped with its mirror element. If the chromosome contains l elements, and the k^{th} element is selected for mutation, then I swap this with the $(l - k)^{th}$ element, which I refer to as its mirror.

APPENDIX B - Receiver Operating Characteristic (ROC) Curves

A Receiver Operating Characteristics Curve is a very complete, simple, and elegant way to display the performance of a binary classification system. ROC curves date back to World War II during the advent of radar. It was necessary to detect friendly planes from enemy planes, and the military needed reliable systems to perform this task. Continuing with our concept of referring to a true positive as the correct detection of something dangerous or malicious, it was desirable for the military to possess a system which performed at a high true positive (correctly identify an enemy plane) and low false positive (identify a friendly plane as enemy). The reasons for this is obvious. The medical community uses ROC curves extensively for measuring the accuracy of medical diagnosis tests [7, 8, 6].

Tom Fawcett published several papers regarding the application of ROC curves. He considers ROC curves and multiple classification systems in the context of game theory in [15]. In [14], he provides a very thorough theoretical review concerning in regard to everything that an ROC curve can (and sometimes cannot) represent. Kristin Bennett extended ROC theory into regression, publishing a paper that discussed “Regression Error Characteristic Curves” [1].

ROC curves are a critical component of any binary classification problem. Classification systems create a real valued number as a decision value for each observed instance of the testing sample. Given a sample of testing data, each data point will be assigned a decision value that typically ranges from -1 to +1. The data points that are true positive should fall closer to +1, where as the true negative cases closer to -1. The modeler must determine a threshold value for which to segregate and predict classes. The outcome in regard to true positive and false positive for one threshold value would correspond to one point on the ROC curve. If this threshold value is now incremented by some small value *epsilon*, from -1 to +1, a range of operating points will emerge. Plotting these operating points with respect to true positive (*y* axis) and false positive (*x* axis) will generate the ROC curve.

An ROC curve can also be considered a graphic representation of the relationship between the probability of a true positive outcome (sensitivity, $1-\alpha$ error) and the probability of a false positive outcome (Type II error(β) or 1 -specificity). The overall curve reflects the quality of the classification system. The area under the curve (AUC) is typically used as method of comparing alternate ROC curves; the better ROC curve typically has more AUC.

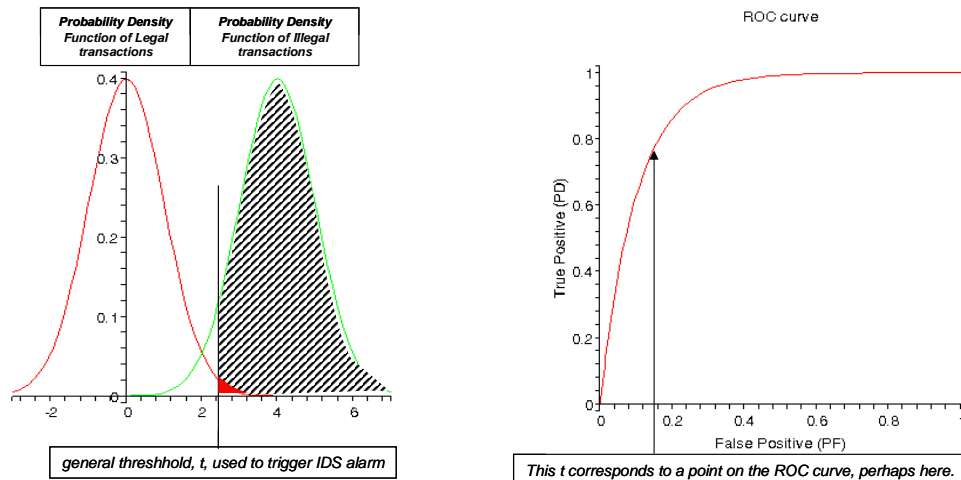


Fig. 6: The plot on the left shows the PDFs of legal and illegal transactions (non intruders and intruders), respectively. The ROC curve on the right shows the possible operating point represented by the tolerance threshold shown on the PDF plots.

The plot of the probability density functions illustrates the idea of false positives and true positives. The hashed area represents the probability of a true positive, and the small red area represents a false positive. Imagine shifting the threshold, t , to the left. The true positive rating would definitely increase, but so would the false positive. The user must identify what point on the curve is acceptable, and this is usually accomplished through a cost-benefit analysis.

APPENDIX C - Comments on Computing Techniques and Source Code

This project required a host of software and several programs that I wrote in Perl. I also relied heavily on scripts that would simultaneously execute Perl programs, *AnalyzeTM* for statistical analysis, and create plots with Gnuplot. Unsupervised classification would not have been possible without LIBSVM [3] which is written in C and C++. In order to extract the decision values from LIBSVM required a bit of code manipulation, and thankfully the authors of LIBSVM provide the source code.

I have included in this appendix the two most significant Perl programs that I wrote for this project. I have not included the scripts or some of the uninteresting Perl programs for the sake of brevity. The two programs that I am including are the Fuzzy ROC curve program and Genetic Algorithm program.

The Fuzzy ROC curve program accepts decision values created by LIBSVM, maps the values into δ_{ij} and θ_{ij} values, and then performs the desired aggregation operation (T-norms or T-conorms) with or without contention. The resulting decision value is then translated into ROC plotting data which is visualized with Gnuplot.

The Genetic Algorithm program implements the GA as I have described it in this paper. In order to evaluate the fitness of each chromosome, this program calls upon *AnalyzeTM* which receives the variable selections, creates the subspaces and their principal components, and then creates a correlation matrix of the principal components. The GA program opens this correlation matrix and finds the maximum correlation value (of two components not in the same subspace).

Fuzzy ROC Curve Program

```
#####
# DEC 04; This file creates a Fuzzy ROC curve #
# from several arrays of dec values #
# Paul Evangelista #
#####

## READ IN ALL DEC VALUES ##
#
$start=time();
print "Start time: $start\n\n";
print "CALCULATING THE ROC CURVE\n";

open (unique,"scaled_xu.txt");
@unique=<unique>;
@sortxu_unique = sort {$a <=> $b} @unique;

#open (one_dec,"1to4dec_value.txt");
#open (one_dec,"set1_dec.txt");
open (one_dec,"wset1_dec.txt");
@xu_one=<one_dec>;
@sortxu_one = sort {$a <=> $b} @xu_one;
$length=@xu_one;
$scale_one = abs($sortxu_one[$length-1] - $sortxu_one[0]);

#open (two_dec,"5to10dec_value.txt");
#open (two_dec,"set2_dec.txt");
open (two_dec,"wset2_dec.txt");
@xu_two=<two_dec>;
@sortxu_two = sort {$a <=> $b} @xu_two;
$scale_two = abs($sortxu_two[$length-1] - $sortxu_two[0]);

#open (three_dec,"12to27dec_value.txt");
#open (three_dec,"set3_dec.txt");
```

```

open (three_dec,"wset3_dec.txt");
@xu_three=<three_dec>;
@sortxu_three = sort {$a <=> $b} @xu_three;
$scale_three = abs($sortxu_three[$length-1] - $sortxu_three[0]);

```

```
#####
```

```
# T NORM AND T CONORM FUNCTIONS
```

```
#####
```

```

sub maximum {
my @input = @_;
my $length = @input;
my $i;
my $max = 0;
for ($i=0;$i<$length;$i++) {
if ($input[$i] > $max) {
$max = $input[$i];
}
}
return $max;
}

```

```

sub alg_sum {
my @input = @_;
my $length = @input;
my $i;
my $prod = 1;
my $result = $input[0] + $input[1] - $input[0]*$input[1];
for ($i=2;$i<$length;$i++) {
$result = $input[$i] + $result - $input[$i]*$result;
}
return $result;
}

```

```

sub minimum {
my @input = @_;
my $length = @input;
my $i;
my $min = 1;
for ($i=0;$i<$length;$i++) {
if ($input[$i] < $min) {
$min = $input[$i];
}
}
return $min;
}

```

```

sub alg_prod {
my @input = @_;
my $length = @input;
my $i;
my $prod = 1;
my $result = $input[0]*$input[1];
for ($i=2;$i<$length;$i++) {
$result = $input[$i]*$result;
}
return $result;
}

```

```

sub median {
my @input = @_;
my $length = @input;
my $i;
my $med;

```

```

my $lb;
my $ub;
@input = sort {$a <=> $b} @input;
if (($length % 2) == 0) {
    $ub = $length/2;
    $lb = $length/2 - 1;
    $med = ($input[$ub]+$input[$lb])/2;
    print "\n $input[$ub]\n";
    print "$input[$lb]\n\n";
}
if (($length % 2) > 0) {
    $lb = $length/2 -.5;
    $med = $input[$lb];
}
return $med;
}

#####
##THIS OPERATION BELOW SORTS THE LIST AND CAPTURES ORDER STATISTICS;
for ($i=0;$i<$length;$i++)
{
    for ($j=0;$j<$length;$j++)
    {
        if ($xu_one[$i]==$sortxu_one[$j])
        {
            $one[$i]=$j-1250;
        }
        if ($xu_two[$i]==$sortxu_two[$j])
        {
            $two[$i]=$j-1250;
        }
        if ($xu_three[$i]==$sortxu_three[$j])
        {
            $three[$i]=$j-1250;
        }
        if ($unique[$i]==$sortxu_unique[$j])
        {
            $s_unique[$i]=$j-1250;
        }
    }
}

#####

$sum_one=0;
$sum_two=0;
$sum_three=0;
$sum_unique=0;

for ($i=0;$i<($length);$i++)
{
    $sum_one=$xu_one[$i]+$sum_one;
    $sum_two=$xu_two[$i]+$sum_two;
    $sum_three=$xu_three[$i]+$sum_three;
    $sum_unique=$unique[$i]+$sum_unique;
}

$mean_one=$sum_one/$length;
$mean_two=$sum_two/$length;
$mean_three=$sum_three/$length;
$mean_unique=$sum_unique/$length;

```

```

serr_one=0;
serr_two=0;
serr_three=0;
serr_unique=0;

print "unique mean is: $mean_unique\n";

for (i=0;i<($length);i++)
{
serr_one=(($xu_one[i]-$mean_one)*($xu_one[i]-$mean_one))+serr_one;
serr_two=(($xu_two[i]-$mean_two)*($xu_two[i]-$mean_two))+serr_two;
serr_three=(($xu_three[i]-$mean_three)*($xu_three[i]-$mean_three))+serr_three;
serr_unique=($unique[i]-$mean_unique)*($unique[i]-$mean_unique))+serr_unique;
}

$stdev_one = sqrt($serr_one/($length-1));
$stdev_two = sqrt($serr_two/($length-1));
$stdev_three = sqrt($serr_three/($length-1));
$stdev_unique = sqrt($serr_unique/($length-1));

$cont_count = 0;

for (i=0;i<$length;i++)
{
$xu_one[i]=($xu_one[i]-$mean_one)/$stdev_one;
$x[0]=$xu_one[i];
$xu_two[i]=($xu_two[i]-$mean_two)/$stdev_two;
$x[1]=$xu_two[i];
$xu_three[i]=($xu_three[i]-$mean_three)/$stdev_three;
$x[2]=$xu_three[i];
$unique[i]=($unique[i]-$mean_unique)/$stdev_unique;
$xu_sum[i]=($xu_one[i]+$xu_two[i]+$xu_three[i])/3;
#$x[3]=$unique[i]*(1);
$x[0]=($x[0]+1)/2;
$x[1]=($x[1]+1)/2;
$x[2]=($x[2]+1)/2;
#$x[3]=($x[3]+1)/2;
for (j=0;j<3;j++) {
if ($x[j] < 0) {
$x[j] = .00000001;
}
if ($x[j] >1) {
$x[j] = 1;
}
}
#print "@x\n";
@x = sort {$a <=> $b} @x;
$xu_min[i]=$x[0];
$xu_mid[i]=$x[1];
$xu_max[i]=$x[2];
$xu_minmid[i]=($xu_min[i]+$xu_mid[i])/2;
$xu_min_sq[i]=$xu_min[i]*$xu_min[i]*($xu_min[i]/(abs($xu_min[i])));
$xu_mid_sq[i]=$x[1]*$xu[1];
$xu_max_sq[i]=$x[2]*$xu[2];

$order_sum[i]=$one[i]+$two[i]+$three[i];
$y[0]=$one[i];
$y[1]=$two[i];
$y[2]=$three[i];
#$y[3]=$s_unique[i];
@y = sort {$a <=> $b} @y;
$y_min[i]=$y[0];
$y_mid[i]=$y[1];

```

```

$y_max[$i]=$y[2];
$s_y[0]=($y[0]+2500)/5000; #s prefix means "scaled"
$s_y[1]=($y[1]+2500)/5000;
$s_y[2]=($y[2]+2500)/5000;
#print "@s_y\n";
@all = @s_y;
push @all,@x;
@all = sort {$a <=> $b} @all;
$min_all[$i] = $all[0];
$max_all[$i] = $all[5];
$med_all[$i] = ($all[2]+$all[3])/2;

#####
@array = @s_y; # @x is cardinal, @s_y is ordinal, @all is all 6

$maxi = maximum(@array);
$mini = minimum(@array);
$a_sum = alg_sum(@array);
$a_prod = alg_prod(@array);
$cont = $max_all[$i]-$med_all[$i];
$med = median(@array);

#####

$t_cont = 1; # EXP. CONTROL PARAMETERS
$decision = $a_prod;
#####

if ($cont > $t_cont) {
# $c_all[$i] = 1;
$c_all[$i] = $med;
$cont_count = $cont_count+1;
}
if ($cont < $t_cont) {
$c_all[$i] = $decision;
}

}
print "Contention count: $cont_count\n";

open (true,"label.txt");
@true=<true>;
$AUC=0;
#####
# CHOOSE THE AGGREGATION ARRAY BELOW
#####

@xu=@c_all;

#####
@sortxu = sort {$a <=> $b} @xu;
$length=@xu;
$scale = abs($sortxu[$length-1] - $sortxu[0]);
print "scale is $scale\n";

#####
# CREATE THE DATA FOR THE ROC CURVE
#####

open (ROC,">ROC.txt");
$j=0;

```

```

print ROC "0 0\n";
for ($j=0;$j<200;$j++)
{
  #if ($j==101)
  #{
  # @xu=@s_unique;
  # @sortxu = sort {$a <=> $b} @xu;
  # $length=@xu;
  # $scale = abs($sortxu[$length-1] - $sortxu[0]);
  #}
  $tolerance=$scale*($j/199)+$sortxu[0]+(1/199);
  for($i=0;$i<$length;$i++)
  {
    if ($xu[$i]<$tolerance)
    {
      @testresult[$i]=1;
    }
    if ($xu[$i]>=$tolerance)
    {
      @testresult[$i]=-1;
    }
  }

  $truepositive=0;
  $falsepositive=0;
  $falsenegative=0;
  $truenegative=0;
  for ($i=0;$i<$length;$i++)
  {
    if (@testresult[$i]==@true[$i])
    {
      @finalresult[$i]=1;
    }
    if (@testresult[$i]!=@true[$i])
    {
      @finalresult[$i]=0;
    }
    if (@testresult[$i]==1 && @true[$i]==1)
    {
      $truepositive=$truepositive+1;
    }
    if (@testresult[$i]==-1 && @true[$i]==-1)
    {
      $truenegative=$truenegative+1;
    }
    if (@testresult[$i]==1 && @true[$i]==-1)
    {
      $falsepositive=$falsepositive+1;
    }
    if (@testresult[$i]==-1 && @true[$i]==1)
    {
      $falsenegative=$falsenegative+1;
    }
  }
  $truepositive[$j]=($truepositive/($truepositive+$falsenegative));
  $falsepositive[$j]=($falsepositive/($truenegative+$falsepositive));
  if ($j>0)
  {
    $AUC=$AUC+.5*(abs($falsepositive[$j]-$falsepositive[$j-1]))*($truepositive[$j]+$truepositive[$j-1]);
  }
  print ROC "@falsepositive[$j] @truepositive[$j]\n";
}
print ROC "1 1";

open (AUC,">AUC.txt");

```

```
print AUC "$AUC";

print "Program complete.\n";
$time=(time()-$start)/60;
$time=sprintf("%.2f",$time);
print "Total elapsed time: $time min\n";
print "\n";
print "AREA UNDER THE ROC CURVE IS $AUC\n";

print "press enter to continue\n";
$zz=<stdin>;
```

Genetic Algorithm Program

```
#####
# Genetic Algorithm written by
# Paul F. Evangelista
# DEC 04
#
# This program utilizes a genetic algorithm
# to find uncorrelated subspaces by measuring
# their principal component correlation (across
# subspaces
#####

sub shuffle #Randomly shuffle any given array
{
my $array = shift;
my $i;
for ($i=@array;--$i;)
{
my $j = int rand ($i+1);
next if $i == $j;
$array[$i,$j] = $array[$j,$i];
}
}

sub fitness { #Calculates fitness of each member utilizing Analyze
my @rand_chrom = @_;
our $gen_size;
my ($i,$k,@array,@subset_1,@subset_2,@subset_3,$call,@fitness);
for ($k=0;$k<$gen_size;$k++) {
@array = split /\t/, $rand_chrom[$k];
#print "chrom$k: @array\n";

for ($i=0;$i<9;$i++) { #creates subsets and eventually writes them to file
$subset_1[$i] = $array[$i];
$subset_2[$i] = $array[$i+9];
if ($i<8) {
$subset_3[$i]=$array[$i+18];
}
}

@subset_1 = sort{$a<=>$b} @subset_1;
@subset_2 = sort{$a<=>$b} @subset_2;
@subset_3 = sort{$a<=>$b} @subset_3;

open(set_1,">set_1.txt");
open(set_2,">set_2.txt");
open(set_3,">set_3.txt");

for ($i=0;$i<9;$i++) {
print set_1 "$subset_1[$i]\n";
print set_2 "$subset_2[$i]\n";
$subset_2[$i] = $array[$i+9];
if ($i<8) {
print set_3 "$subset_3[$i]\n";
}
}
}
$call = "PCA_embrechts.bat>output.txt";
system $call;
open(fit,"avg_cov.txt");
$fitness[$k] = <fit>;
}
return @fitness;
}
}
```

```

sub sel_breed { #receives array of fitness values
my @input = @_;
my $length = @input;
my ($i,@prob,$rn,$j,@sel_gen,$comp,@sort_input,@new_input);
my $cum_prob = 0;
my $sum = 0;
my @new_input = sort{ $a <=> $b } @input;
$new_input[$length] = $new_input[$length-1] + 1;
for ($i=0;$i<$length;$i++) { #creates sort input which is an index array to original input in sorted order
$comp = 0;
for ($j=0;$j<$length;$j++) {
if ($input[$j] == $new_input[$i]) {
$sort_input[$i] = $j;
}
}
}

    my $maxer = 0;
for ($i=0;$i<$length;$i++) { #find max fitness of current generation
if ($new_input[$i] > $maxer) {
$maxer = $new_input[$i];
}
}
$maxer = $maxer+.1;
for ($i=0;$i<$length;$i++) { #Create probabilities for selection
$sum = $sum + ($maxer - $new_input[$i]);
}

for ($i=0;$i<$length;$i++) { #Create probabilities for selection
$prob[$i] = ($maxer-$new_input[$i])/$sum; #adjusts probability for minimization problem
$cum_prob = $cum_prob + $prob[$i];
$prob[$i] = $cum_prob;
print "$sort_input[$i] $new_input[$i] $prob[$i]\n";
}

for ($i=0;$i<$length-1;$i++) { #Select next generation
$rn = rand;
$prob[$length] = 1.1;
for ($j=0;$j<$length;$j++) {
if ($j==0) {
if ($rn > 0 && $rn <= $prob[$j]) {
$sel_gen[$i] = $sort_input[$j];
last;
}
}

if ($j>0) {
if ($rn > $prob[$j-1] && $rn <= $prob[$j]) {
$sel_gen[$i] = $sort_input[$j];
last;
}
}
}
}
#print "rand#: $rn index: $sel_gen[$i]\n";
}
$sel_gen[$length-1] = $sort_input[0]; #elitist selection
print "Elite member: $sort_input[0] fitness: $input[$sort_input[0]]\n";
return @sel_gen;
}

our @ordered_array;

for ($i=0;$i<26;$i++) {
$ordered_array[$i] = $i+1;
}

```

```

}

for ($i=0;$i<50;$i++) { #create a random number of initial members
@array = @ordered_array;
shuffle(\@array);
$rand_chrom[$i] = join "\t",@array;
}

#####

our $no_gens = 50; #Experimental Control parameters
our $gen_size = 50;
$worst_fitness = 0;

#####

@current_chrom = @rand_chrom;

for ($zz=0;$zz<$no_gens;$zz++) {

@pop_fitness = fitness(@current_chrom);
$best_fitness = 1; #best fitness will range from 0 to 1, where 0 is best
print "fitness of generation:\n";
for ($i=0;$i<$gen_size;$i++) {
print " $pop_fitness[$i]\n";
#print "$current_chrom[$i]\n";
if ($pop_fitness[$i] < $best_fitness) {
$best_fitness = $pop_fitness[$i];
$fittest = $current_chrom[$i];
}
if ($pop_fitness[$i] > $worst_fitness) {
$worst_fitness = $pop_fitness[$i];
$worst = $current_chrom[$i];
}
}
print "CURRENT BEST FITNESS: $best_fitness\nFITTEST CHROMOSOME: $fittest\n";
print "\n";

@breeders = sel_breed(@pop_fitness);

@array = @ordered_array;
shuffle(\@array); #insert two immigrants for worst performers
$curr_chrom[$breeders[$gen_size-3]] = join "\t",@array;
shuffle(\@array);
$curr_chrom[$breeders[$gen_size-2]] = join "\t",@array;

print "the breeders (elitist is last one):\n @breeders\n";
$elitist = $breeders[$gen_size-1];
$elite_chrom = $current_chrom[$elitist];
print "elite at 158: $elite_chrom\n";

pop(@breeders);
shuffle(\@breeders);

$k=0;
$l=0;
for ($i=0;$i<($gen_size-1);$i++) {
$rn = rand;
if ($rn > .4) {
$crossover[$k] = $i;
$k++;
}
if ($rn <= .4) {
$new_chrom[$l] = $current_chrom[$breeder[$i]];
$l++;
}
}

```

```

}
}

shuffle(\@crossover);

$x_over_pu = $l;
$no_xover = $k;

for ($i=0;$i<$no_xover;$i=$i+2) {
#print "creating offspring from parents $crossover[$i] and $crossover[$i+1]\n";
@parent_1 = split /\t/, $current_chrom[$crossover[$i]];
@parent_2 = split /\t/, $current_chrom[$crossover[$i+1]];
$length = @array;
$c_over_point = int(@array/2);
for ($j=0;$j<$c_over_point;$j++) {
$child_1[$j] = $parent_1[$j];
$child_2[$j] = $parent_2[$j];
}
for ($j=$c_over_point;$j<$length;$j++) {
$child_1[$j] = $parent_2[$j];
$child_2[$j] = $parent_1[$j];
}
$m=0;
$n=0;
for ($j=0;$j<$length;$j++) {
$dup_1[$j] = 0;
$dup_1[$j] = 0;
$sused_1 = 0;
$sused_2 = 0;
for ($l=0;$l<$length;$l++) { #checks to see what integers went unused and assigns them to an "avail" array
if ($child_1[$l] == $j+1) {
$sused_1 = 1;
}
if ($child_2[$l] == $j+1) {
$sused_2 = 1;
}
}
if ($sused_1 == 0) {
$avail_1[$m] = $j+1;
$m++;
}
if ($sused_2 == 0) {
$avail_2[$n] = $j+1;
$n++;
}
}
}
$m=0;
$n=0;
for ($j=0;$j<$c_over_point;$j++) {
for ($l=$c_over_point;$l<$length;$l++) {
if ($child_1[$j] == $child_1[$l]) {
$child_1[$j] = $avail_1[$m];
$m++;
}
if ($child_2[$j] == $child_2[$l]) {
$child_2[$j] = $avail_2[$n];
$n++;
}
}
}
}
@check1 = sort{$a<=>$b} @child_1;
@check2 = sort{$a<=>$b} @child_2;
#print "child_1 sorted after check:\n @check1\n";

```

```

#print "child_2 sorted after check:\n @check2\n";

$new_chrom[$x_over_pu] = join "\t", @child_1;
$x_over_pu++;
$new_chrom[$x_over_pu] = join "\t", @child_2;
$x_over_pu++;
}

print "\n\n";
$size = @new_chrom;
print "size should be gen_size: $size\n";

for ($l=0;$l<$gen_size-1;$l++) { #MUTATION
for ($k=0;$k<$length;$k++) {
$rn = rand;
if ($rn < .01) {
#print "chrom before mutation:\n $new_chrom[$l]\n";
@array = split /\t/, $new_chrom[$l];
$mute = $array[$k];
$mute_a = $array[$length-$k];
$array[$k] = $mute_a;
$array[$length-$k] = $mute;
$new_chrom[$l] = join "\t", @array;
#print "chrom after mutation:\n $new_chrom[$l]\n";
}
}
#print "$new_chrom[$l]\n\n";
$current_chrom[$l] = $new_chrom[$l];
}
$current_chrom[$gen_size-1] = $elite_chrom;
print "elitist at end of program: $current_chrom[$gen_size-1]\n";

}
print "Worst fitness is: $worst_fitness\n\n$worst\n\n";

print "Best fitness is : $best_fitness\n\nFittest chromosome:\n$fittest\n";

```