N-SLOPE: A One-Class Classification Ensemble For Nuclear Forensics

Justin Kehl* Lubomir Stanchev†
Department of Computer Science and Software Engineering
California Polytechnic State University
San Luis Obispo, CA, USA
*jkehl@calpoly.edu †lstanche@calpoly.edu

***Abstract*—One-class classification is a specialized form of classification from the field of machine learning. A traditional classifier always assigns a new element to one of the known classes, but it cannot handle elements that do not belong to any of the existing classes. One-class classification seeks to identify these outliers, while still correctly assigning the rest of the elements to classes appropriately. One-class classification is applied here to the field of nuclear forensics, which is the study and analysis of nuclear material for the purpose of nuclear incident investigations. Nuclear forensics data poses an interesting challenge because false positive identification can prove costly and the data is often small, high-dimensional, and sparse, which is problematic for most machine learning approaches. A web application that incorporates N-SLOPE (a machine learning ensemble) is built using the R programming language and the shiny framework. N-SLOPE combines five existing one-class classifiers with a novel one-class classifier called SCD (Soft Centroid Distance) algorithm and uses ensemble learning techniques to combine output. N-SLOPE is validated on an enhanced version of Galaxy Serpent 3, which is a recent international nuclear forensics exercise. N-SLOPE achieves high classification accuracy of 85% on this difficult data set, while minimizing false positive detection rate to zero by correctly detecting all 16 outliers that are present in the data set.**

I. Introduction

Classification is a well-known problem in the field of supervised learning that has been studied extensively. Traditional classification algorithms learn from a training set and attempt to place a new element from the testing set into one of the known classes. An interesting problem arises when a new element that does not belong to any of the known classes is encountered – see Figure 1. This problem is addressed by *one-class classification*.

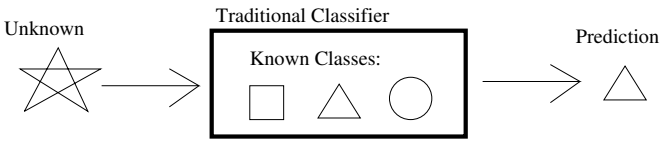

Fig. 1. Incorrect behavior of a traditional classifier when presented with a new element that does not belong to any of the existing classes

One-class classification is interesting because it solves one of the fundamental problems of traditional classification. Traditional classifiers act as discriminators exhibiting only within-class and between-class generalization [20], meaning they are only able to differentiate between elements of the same class and between elements of different classes. True one-class classifiers act as detectors and exhibit within-class, between-class, and out-of-class generalization [20]. In this way, a one-class classifier is able to determine whether a new element belongs to any of the known classes (and if so, to which known class) or if it belongs to a separate class altogether – see Figure 2. This behavior is different from the behavior of traditional classifiers, which will always misclassify outliers as belonging to one of the known classes. Note that throughout this paper we use the term *outlier* to refer to an element that does not belong to any of the classes in the training set.

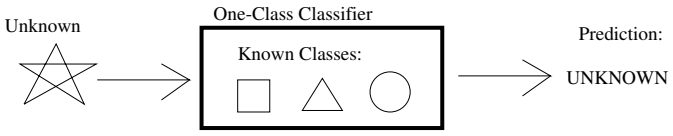

Fig. 2. Correct behavior of a one-class classifier when presented with an outlier

One-class classification is difficult because it must correctly handle a wider range of testing data than traditional classification, while being provided the same training data. This problem is made more challenging by focusing on the domain of nuclear forensics, which introduces several key constraints. First, data is often high-dimensional, small, sparse, and restricted. Second, it is unlikely that any training set will contain all possible classes, meaning that outliers are likely to occur. Third, misclassification of nuclear material, especially false-positive identification, can have serious consequences. Nuclear forensics data poses a unique challenge for any machine learning solution and the domain makes accuracy a prime concern.

There are many existing algorithms that attempt to solve the problem of one-class classification [11] and each algorithm has strengths and weaknesses that cause it to perform better or worse under certain conditions. Most machine learning algorithms struggle under the difficult data domain imposed by nuclear forensics and these one-class classifiers are no different. This specific problem is addressed in the thesis of Justin Kehl [10], which introduces a novel algorithm for one-class classification and employs ensemble learning to combine six algorithms for strong generalized performance.

Each of the six algorithms has an associated training accuracy that is used when calculating ensemble contributions. We calculate the weight of each algorithm in the ensemble using two different approaches: (1) as the overall percentage of correctly classified samples during ten-fold cross-validation, and (2) we calculate each algorithm's training accuracy on each known class and use the appropriate class training accuracy when contributing a decision to the ensemble. In the second approach, when an algorithm identifies a testing sample as an outlier, the overall training accuracy is used for ensemble contribution. The second approach provides more granular control, allowing algorithms to contribute stronger decisions to classes on which they perform better and weaker decisions to classes on which they perform poorly. This modification balances algorithm contributions and provides a slight boost to the overall classification accuracy.

In what follows, in Section II we present an overview of the algorithms used in the N-SLOPE ensemble. Our main contributions are in Sections III, IV, and V, where we present the design of the N-SLOPE, its implementation, and the modification that was described in the previous paragraph, respectively. In Section VI, we validate N-SLOPE by showing that it is able to achieve high classification accuracy on a realistically simulated nuclear forensics data set. Lastly, Section VII summarizes the paper and outlines areas for future research.

II. Related Research

N-SLOPE is a one-class classification ensemble that employs five existing classifiers including *Partial Least Squares with Discriminant Analysis* (PLS-DA), *Soft Independent Modeling of Class Analogies* (SIMCA), *One-Class Support Vector Machine* (OC-SVM), *Local Outlier Factor* (LOF), and *Extreme Learning Machine* (ELM). Each of these algorithms is described here with justification for being included in N-SLOPE.

PLS-DA is a specific case of Partial Least Squares (PLS) in which the response variable is categorical. Madden and Howley describe PLS as "a two-step multivariate regression method, which first reduces the data using PCA [Principal Component Analysis] (using concentration information to extract the PC [Principal Component] scores) and then performs linear regression on the PC [Principal Component] scores" [16]. PLS-DA differs by additionally creating class boundaries associated with a confidence threshold after performing the linear regression. PLS-DA is commonly used for *chemometric analysis* (classification based on chemical structure or properties) [16], [28] because it excels with high-dimensional, low record count data, particularly when there exist linear relationships between attributes. Support Vector Machine and Neural Networks have been shown to outperform PLS-DA, particularly on data sets with many records and nonlinear relationships between attributes [28], [27]. Despite this, PLS-DA is a proven chemometric standard and requires minimal tuning compared to its alternatives, which justifies its inclusion in N-SLOPE.

SIMCA is a statistical technique for multivariate classification. SIMCA operates by first performing Principal Component Analysis (PCA) on each class in the training set independently, then constructing confidence threshold boundaries for each class based on the residual of standard deviation and distance, and then finally mapping each element into the component space for classification. SIMCA operates as a "soft" classifier in that it can map the elements in the testing set to zero, one, or more distinct classes [4]. Because of this, SIMCA is often outperformed by PLS-DA classification [4], which itself can be outperformed by more traditional machine learning techniques [28], [27]. However, SIMCA excels with small, high-dimensional data sets [23] and acts as a particularly strong within-class classifier by allowing elements in the testing set to be assigned to multiple known classes [4], [23]. SIMCA is included in N-SLOPE for its unique within-class classification abilities, aptitude for dealing with the difficult data domain present in chemometrics, and because it is one of the most common techniques used for chemical spectral data analysis [23].

OC-SVM, also known as Support Vector Data Description or Support Vector Domain Description, is a natural modification of the traditional *Support Vector Machine* (SVM). Instead of creating a decision boundary between known classes as in traditional SVM, OC-SVM creates either a closed hypersphere of minimal volume around the entire training set [25] or a hyperplane between the origin and the training set [22]. Testing set elements are mapped to this space and considered an outlier if they fall outside the decision boundary. OC-SVM is flexible with many optimizations, such as training with outliers (if examples exist) that must fall outside of the decision boundary, kernel functions to alter the behavior, shape, and dimensionality of the decision boundary to better describe different data characteristics, and ignoring statistical outliers in the training set when building the model [25]. Like most machine learning techniques, OC-SVM struggles with statistical outliers and smaller, high-dimensional data sets, although optimizations can be made to improve results under these conditions [1], [25]. Overall, OC-SVM has been shown to have excellent performance [1], [9], [25], [24], [18] and it is often seen as the default one-class classifier in a category of its own [11]. These qualifications make OC-SVM an exceptional candidate for N-SLOPE.

LOF is an algorithmic technique for quantifying how strongly a given element in the testing set resembles a statistical outlier. LOF resembles *k-Nearest Neighbors* (kNN) initially, but differs in that the distance between an element in the testing set and each of its k-nearest neighbors is compared to the average distance between each neighbor's k-nearest neighbors and this ratio is averaged to produce an outlier factor for each element [3]. Unlike other outlier detection methods that attempt to determine whether or not an element in the testing set is an outlier in a binary fashion, LOF assigns a floating-point value, starting at one, to each element of the testing set. Higher values indicate a greater degree of outlying characteristics [3]. This non-binary approach can be useful for

determining the difference between a weak, possible outlier and a strong, definite outlier, although a final classification is not performed. Because LOF is density-based, it struggles with small, high-dimensional, loosely-clustered data sets [9], but its local-density approach excels with between-class generalization and is able to identify outliers often missed by other approaches [3]. Interestingly, LOF has also been shown to perform well compared to OC-SVM [1] and in some cases a relationship may exist where LOF performs well on data sets where OC-SVM struggles and vice versa [9]. LOF is included in N-SLOPE for its between-class generalization and complementary potential with OC-SVM, but challenges due to the nature of the nuclear forensics data domain are expected.

ELM, as applied to one-class classification, is a very simple feed-forward *neural network* (NN) without back-propagation that contains one hidden layer and a single node in the output layer [15]. ELM attempts to minimize both the error and the norm of output weights during training to optimize performance [15]. ELM requires minimal tuning, outperforms the former NN-based one-class classifier known as autoencoder [17] in both computational speed and classification accuracy, and has been shown to outperform OC-SVM in some cases [15]. The ability of ELM to utilize nonlinear kernels also gives it a distinct advantage over linear approaches, such as PLS-DA and SIMCA, when working with nonlinear data [27]. ELM is included in N-SLOPE as a complementary NN approach in the ensemble and because of its high speed and performance potential. However, the minimal tuning, simplicity of the network, and black-box nature of NN in general suggest caution and verification of results.

III. N-SLOPE Visual Design

N-SLOPE is the core classification tool of a web application for comprehensive data analysis of nuclear forensics data that is written in the R programming language [21] and runs using the shiny framework [5]. Both N-SLOPE and the web application have been developed for and remain the property of Lawrence Livermore National Laboratory and the United States Department of Energy.

A. Layout

N-SLOPE is divided into seven separate tabs including one overview tab and one tab for each algorithm. The overview tab provides input controls (described in the following subsection) to modify general algorithm behavior and displays the final results of running N-SLOPE. Each algorithm tab runs the respective algorithm and displays meaningful characteristics and classification results. Any action that involves non-trivial computation is tracked with progress bars.

B. Inputs

N-SLOPE contains several input controls, including principal component retention methods, confidence interval classification range, 10-fold cross validation repititions, and selectors for which algorithms to include in the final results. Each input can be summarized as follows.

1) Principal Components: The number of principal components to retain for the novel SCD, LOF, and PLS-DA algorithms can be determined using *eigenvalue analysis* or *explained variance*. The *eigenvalue analysis* method keeps components with eigenvalues greater than one and discards the remaining components that exhibit only a small amount of variance in the data set. The *explained variance* method keeps as many components as necessary to reach a cumulative explained variance threshold set by the user, with higher thresholds usually retaining more components than the *eigenvalue analysis* method. Both methods use the `FactoMineR package` [14] to determine how many components to retain.

2) Confidence Interval: The novel SCD algorithm and LOF use confidence intervals to produce final classification results. Computed values are compared against statistics from the training set and final classifications are made using a number of standard deviations from training set means. The user is able to select how many standard deviations to include with lower numbers being more sensitive to outlier detection.

3) Training Repetitions: Every algorithm in N-SLOPE is validated on the training set using 10-fold cross-validation repeated a settable number of times with training accuracy listed on each algorithm's tab. Reported training accuracy reflects only the algorithm's ability to correctly classify the training set with classification error existing as misclassifications within the training set or incorrect outlier detection. In this way, a training accuracy of 100% may indicate some degree of overfitting, as no points in the training set were considered outliers.

4) Algorithms: The user is able to manually select which N-SLOPE algorithms to include in the final result calculations, but the selected algorithms must be run before they can be included.

All of these controls allow the user to modify the behavior of N-SLOPE for improved performance on specific data sets based on external factors or intrinsic knowledge of the data. For example, the user may choose to retain fewer principal components to get results more quickly or opt to repeat cross validation multiple times in an attempt to improve results at the cost of training time. With some prior knowledge of the data, the user may select more standard deviations to cope with noisy data or if outliers are unlikely to occur. Similarly, the user may choose to exclude linear algorithms, such as PLS-DA, if the data is expected to be nonlinear.

C. Models

Each N-SLOPE algorithm consists of one or more models that are used to display output in the application. Internally, each model is stored as a `reactiveValue` that updates when a user requests output that depends on the underlying model (e.g., clicking one of the algorithm tabs). Each algorithm is tracked by its own model, except for LOF, which requires a kNN model, calculation model, and summary model, and OC-SVM, which requires a parameter model in addition to its base model. The overall design of N-SLOPE and

its integration with shiny [5] mimics the classic Model-View-Controller (MVC) design pattern where interactions with the controller (inputs on overview tab) alter the model (underlying algorithm models) that the view (output on algorithm tabs) observes to update appropriately. This modular design makes it easy to add additional views to display more information or model characteristics without major changes to the algorithms or their models.

D. Validation

Every algorithm in N-SLOPE is validated on the training set using 10-fold cross-validation. SIMCA, PLS-DA, ELM, and the novel SCD algorithm undergo manual cross-validation purely to calculate a training accuracy that represents the percentage of correct classifications performed on the training set. Except for ELM, these algorithms rely solely on hyperparameters and do not need to train additional parameters as part of a learning process. ELM modifies weights between nodes in the network as it trains, but this process is entirely automated with no external tuning necessary. Manual cross-validation is run all at once and independently of the algorithms themselves, meaning that changing N-SLOPE inputs will automatically revalidate these four algorithms, but each algorithm will still have to be run to update the underlying model(s). Manual cross-validation is performed on all four algorithms at once rather than idependently for simplicity and to reduce code repetition.

LOF and OC-SVM perform indepentent cross-validation because both algorithms train parameters as part of the learning process. LOF trains a kNN model to determine an optimal k neighbors to consider when calculating local density and OC-SVM trains *gamma* and *nu*, which represent the bias/variance tradeoff and minimal percentage of support vectors to include in the model, respectively. These two algorithms are validated and run sequentially due to their specialized training procedures, unlike the other four algorithms in N-SLOPE.

E. Outputs

N-SLOPE outputs are designed similarly to models with all outputs being tracked by `reactiveValues` that update when an algorithm is validated or run. These values separately track both the classification results (including probabilities if applicable) and the training accuracy of each algorithm. Outputs are combined using the sum rule [12], which involves summing each algorithm's prediction output and choosing the maximal combined sum to produce unified classification results for the entire N-SLOPE ensemble.

IV. N-SLOPE Implementation

Each algorithm in N-SLOPE has specific strengths and weaknesses. Combining these algorithms with ensemble learning attempts to minimize their weaknesses and generate more stable and accurate results across data sets. Some properties of each algorithm are displayed in Table I.

TABLE I
SUMMARY AND CHARACTERISTICS OF N-SLOPE ALGORITHMS

Algorithm	Package	Linearity	Prone to Overfitting	High Dimensional	Small Data	Relative Runtime
SCD		Linear		✓		Fast
SIMCA	`rrcovHD` [26]	Linear		✓	✓	Fast
LOF	`Rlof` [7]	Linear		✓		Slow
OC-SVM	`e1071` [19]	Nonlinear	✓	✓		Slow
PLS-DA	`caret` [13]	Linear		✓	✓	Medium
ELM	`elmNN` [6]	Nonlinear	✓	✓		Medium

A. Ensemble Learning

N-SLOPE consists of several different algorithms whose results must be combined to generate a singular, cohesive output. This is done using the sum rule because it has been shown to outperform other ensemble learning techniques [12]. The basic principle is that each algorithm makes predictions which are summed, and the prediction with the greatest value is selected as the ensemble's decision. This requires that each algorithm in the ensemble produce output that can be summed in a meaningful way.

$$\begin{aligned} \text{N-SLOPE}_{pred} &= \sum_{\text{Alg}\in\text{N-SLOPE}} \text{Alg}_{cont} \\ \text{Alg}_{cont} &= \text{Alg}_{acc} \times \text{Alg}_{pred} \end{aligned} \tag{1}$$

N-SLOPE implements the sum rule as shown in Equation 1. First, each algorithm in N-SLOPE undergoes 10-fold cross-validation to calculate a training accuracy, which represents the percentage of correct classifications made on the training set. Each algorithm is then run on the testing set and predictions are made. Each algorithm produces slightly different output that must be coerced to a common state before being combined. Once the predictions have been standardized, they are multiplied by each algorithm's respective training accuracy before being summed. Equation 1 shows the general form of these calculations. The final results are tabulated and displayed on the N-SLOPE overview tab and the greatest decision sum for each testing sample is selected as the final N-SLOPE prediction.

B. SIMCA

SIMCA operates by independently performing PCA on each class in the training set and then projecting all elements in the testing set to the same space and comparing against a threshold to determine class membership. SIMCA in N-SLOPE is provided by the `rrcovHD package` [26], which implements a robust SIMCA approach, or RSIMCA, and returns *Orthogonal Distance Scores* (ODSC) and *Score Distance Scores* (SDSC) for each element in the testing set. ODSC represent the Euclidean distance between a point and the center of each PCA subspace, whereas SDSC represent the Mahalanobis distance, which is the distance along each principal component between a point and the PCA subspace [2]. These scores are used to compute the classification rules `R1` and `R2` proposed by

Branden and Hubert [2]. The minimum of the two values is kept for each class with membership being associated with any value less than one. In this way, RSIMCA can assign an element in the testing set to zero, one, or more classes with lower values indicating a stronger association. An element with both `R1` and `R2` greater than one is not assigned to any known class and thus labeled as an outlier.

C. LOF

LOF operates by comparing the local density of each element in the testing set to the local density of neighboring classes with values of approximately one indicating membership. The LOF implementation in N-SLOPE is provided by the `Rlof package` [7] and requires a number of neigbors to examine when computing densities. This number is determined by training a kNN classification model courtesy of the `caret package` [13] and using the same optimal *k*. Since LOF produces a score rather than a fixed classification, statistics including the mean and standard deviation are calculated for each sample in the training set by class. Each element's LOF is compared against the LOF statistics of its nearest neighboring class and class membership is determined by whether or not the LOF falls within a user-settable threshold of standard deviations from that class's mean.

D. OC-SVM

OC-SVM operates by creating a hypersphere decision boundary around the training data and assigning class membership to the elements in the testing set based on whether or not they fall within this boundary. OC-SVM in N-SLOPE is provided by the `e1071 package` [19] and has two tunable parameters: *gamma* and *nu*. *Gamma* controls the bias/variance tradeoff with small *gamma* leading to high variance and overfitting and large *gamma* leading to high bias and underfitting. *Nu* controls the minimal percentage of support vectors used when building the model, as well as the maximal percentage of training data allowed to be misclassified. *Gamma* and *nu* are tuned during training and vary between data sets. This OC-SVM implementation supports linear, polynomial, radial basis, and sigmoid kernel functions. Since OC-SVM is included in N-SLOPE to add support for complex, nonlinear relationships, radial basis or sigmoid are both attractive options. Since N-SLOPE includes two nonlinear solutions, OC-SVM uses the radial basis kernel function and ELM, the other nonlinear solution, uses sigmoid. Since OC-SVM only produces a true/false value for each element, indicating whether or not the element belongs to any of the known classes, a traditional SVM is created using the same *gamma* and *nu* values to complete classification for elements that belong to one of the known classes. The final output of OC-SVM is simply a classification: one of the known classes if it is predicted to belong, otherwise it is marked as an outlier.

E. PLS-DA

PLS-DA operates by performing PCA on the entire training set to reduce dimensionality and then running linear regression and creating class decision boundaries based on confidence intervals. PLS-DA in N-SLOPE is provided by the `caret package` [13] and supports two different classification decision algorithms: Softmax and Bayes. Since PLS-DA does not act as a true one-class classifier and always attempts to place an element into one of the known classes, two models are trained with the same number of principal components using different decision algorithms. The final output for each element is the probability of it belonging to each of the possible known classes and a classification as either one of the known classes (if both models agree) or the element is marked as an outlier if they do not agree. In this way, this implementation of PLS-DA marks elements as outliers more aggressively than other algorithms in N-SLOPE because true outliers and confusion between known classes are treated the same. This flaw is factored into the PLS-DA contribution to the ensemble by including class probabilities whether or not the element is marked as an outlier with the final contribution to the ensemble from PLS-DA being calculated by multiplying the training accuracy with each element's associated class probability. If PLS-DA predicts that an element is an outlier, then the full training accuracy is applied, but the remaining probabilities are still included.

F. ELM

ELM operates by building a NN with one input node for each dimension, a single hidden layer, and a single output node that predicts whether or not an element belongs to any of the known classes. ELM in N-SLOPE is provided by the `elmNN package` [6] and starts with randomly initialized weights between nodes that are updated during a single training phase. This implementation of ELM allows for several different kernel functions and setting the number of nodes in the hidden layer. The sigmoid function is chosen to provide a nonlinear approach and to complement the radial basis kernel function used by OC-SVM. Selecting an optimal number of hidden layer nodes for generalized performance is a difficult problem, but there is some suggestion that single-layer, feed-forward networks like ELM are more reliant on updating connection weights than the number of nodes for general performance [8]. Through trial and error, the hidden layer has been chosen to contain 5000 nodes. Since ELM only indicates whether or not an element is an outlier, its contribution to the ensemble is fairly straightforward. If the element is determined to belong to one of the known classes, the ELM training accuracy is evenly divided among all known classes. If the element is determined to be an outlier, then the entire training accuracy supports this classification.

G. Novel SCD Algorithm

The novel SCD algorithm that is included in N-SLOPE is inspired by PCA, LOF, Nearest Centroid, and SIMCA. PCA is performed for dimensionality reduction and summary statistics are computed for each class, similar to the decision method used by the LOF implementation in N-SLOPE. Ideas from Nearest Centroid and SIMCA are incorporated through scaling

the final decision scores into a soft classifier based on distances between the elements in the testing set and class centroids. The algorithm is shown in Algorithm 1.

Algorithm 1: SCD Algorithm

Input : `trainingData`, `testingData`
Output: Calculated values for each testing sample and known class that provide one-class classification with class assignment on values (0,1)

```
1  pcaData ← PCA(trainingData, testingData)
2  train ← pcaData[trainingData]
3  test ← pcaData[testingData]
4  foreach class in train do
5  |  classCenter[class] ← Mean(train[class])
6  end
7  foreach sample in train do
8  |  distToCenter[sample] ←
   |   Euclidean_Distance(sample,
   |   classCenter[sample.class])
9  end
10 foreach class in train do
11 |  avgDist[class] ← Mean(distToCenter[sample.class
   |   == class])
12 |  stdDevDist[class] ←
   |   Std_Dev(distToCenter[sample.class == class])
13 |  threshold[class] ← avgDist[class] ±
   |   stdDevDist[class]
14 |  foreach unknown in test do
15 |  |  unknownDist[unknown, class] ←
   |  |   Euclidean_Distance(unknown,
   |  |   classCenter[class])
16 |  |  pred[unknown, class] ←
   |  |   (unknownDist[unknown,class]−threshold[class].lower)
   |  |   ──────────────────────────────────────────────
   |  |   (threshold[class].upper−threshold[class].lower)
17 |  end
18 end
19 return pred
```

1) Implementation: The SCD algorithm implementation is described here with parenthetical references to line numbers shown above in Algorithm 1. The algorithm begins by combining the training and testing data into a single data frame, performing PCA on the data courtesy of the `FactoMineR package` [14], and retaining the number of components that are selected through *eigenvalue analysis* or *explained variance* as specified by the user (Line 1). The data is then split back into the original training (Line 2) and testing (Line 3) sets. The center point of each class is calculated (Line 5) by averaging every column in each class. Next, the Euclidian distance between each element and its respective class center is calculated (Line 8). These distances are used to calculate an average distance-to-center (Line 11) as well as a standard deviation of distance-to-center measurements (Line 12) for each class, which is similar to how LOF computes class summary statistics. These values are used to generate loose class boundaries in the form of confidence intervals by calculating upper and lower acceptance thresholds (Line 13) as

$$\begin{aligned} \text{Threshold}_{lower} &= \text{Dist}_{avg} - (CI \times \text{Dist}_{sd}) \\ \text{Threshold}_{upper} &= \text{Dist}_{avg} + (CI \times \text{Dist}_{sd}) \end{aligned} \tag{2}$$

for each class, where *CI* is the user-settable number of standard deviations to include for N-SLOPE decision boundaries. The algorithm then computes the distance between each element and the center of each class (Line 15) and compares this distance with the acceptance thresholds of each class. This comparison is normalized into the range $(0, 1)$ for each element and class by calculating the following (Line 16).

$$\frac{(\text{Dist}_{toCenter} - \text{Threshold}_{lower})}{(\text{Threshold}_{upper} - \text{Threshold}_{lower})} \tag{3}$$

Class membership is assigned to any computed values between zero and one. Negative computed values indicate the element is uncharacteristically close to the mathematical center of the class, while values greater than one indicate the element lies uncharacteristically far from the mathematical center of the class. Since this algorithm acts as a soft classifier like SIMCA, it is possible to assign a single element to multiple classes. Unlike SIMCA, the computed value does not necessarily indicate a strength of belonging, meaning that any value between zero and one is equally indicative of class membership. The algorithm's contribution to the ensemble reflects this fact. If a single decision is required, then the algorithm will draw on ideas from Nearest Centroid and select the minimal computed value between zero and one as the final class prediction because this indicates that the element is statistically closest to that class centroid.

2) Limitations: The SCD algorithm is somewhat naïve and assumes data is roughly spherical, causing it to perform poorly on strongly shaped data. Shaped data is less likely to exist in the high-dimensional data common to this domain, but even moderate shaping will negatively impact this algorithm's classification accuracy. Another limitation of this algorithm is that it is distance-based and will thus struggle with sparse data. Furthermore, since the algorithm uses summary statistics to make classification decisions, small data sets may also pose a problem. These last two considerations are common problems for most machine learning techniques and are one of the interesting challenges posed by the chemometric domain.

3) Strengths: The SCD algorithm runs significantly faster than the other N-SLOPE algorithms, particularly on larger, high-dimensional data sets. It is also able to clearly detect outliers in the case where data is roughly donut-shaped with classified elements in a spherical shape surrounding an empty or hollow core. This is a specialized and uncommon case, but not unheard of for high-dimensional, sparse data sets.

4) Contribution to Ensemble: Once the novel algorithm has made class predictions for each element in the testing set, contributing its results to the ensemble is fairly straightforward. The SCD algorithm's training accuracy is added to each class,

where membership is predicted, or to the outlier class if the element is predicted to not belong to any of the known classes.

V. Modified Ensemble Contribution

The ensemble contribution of each algorithm in the original implementation of N-SLOPE is calculated as shown in Equation 1. This calculation is somewhat naïve in that it does not account for varying degrees of algorithm accuracy between different known classes and therefore does not take full advantage of each algorithm's individual classification strengths. A modified ensemble contribution calculation is shown in Equation 4, where the predicted *class* is matched with the appropriate *class* accuracy. In the case where the predicted class is UNKNOWN, meaning the testing sample is not assigned to any of the known classes and therefore considered an outlier, the overall classification accuracy is used as in the original N-SLOPE implementation.

$$\begin{aligned} \text{N-SLOPE}_{pred} &= \sum_{\text{Alg} \in \text{N-SLOPE}} \text{Alg}_{cont} \\ \text{Alg}_{cont} &= \text{Alg}_{acc.class} \times \text{Alg}_{pred.class} \end{aligned} \tag{4}$$

This modification requires that each algorithm's classification accuracy be calculated for each class independently during the training and cross-validation stage. This is not computationally expensive as predictions are already made during training and averaged to produce an overall training accuracy. Under this new methodology, training accuracies for each class are recorded before averaging them together to produce an overall training accuracy. This simple change weights ensemble contributions differently depending on how well a given algorithm is able to classify samples belonging to each known class. This improves the final ensemble prediction by capitalizing on each algorithm's unique strengths and produces more accurate results as shown in the following section.

VI. Experimental Validation

N-SLOPE with the modified ensemble contribution presented here was validated on the Galaxy Serpent 3 (GS3) data set and compared against the original implementation of N-SLOPE. Both versions of N-SLOPE were run on the same data with the same inputs. Validation results are displayed in Table II and include Classification Accuracy (CA - overall percentage of elements correctly classified), True Negative Rate (TNR - percentage of outliers correctly identified as not belonging to any of the known classes), False Positive Rate (FPR - percentage of outliers incorrectly classified as belonging to one of the known classes), and False Negative Rate (FNR - percentage of inlying elements incorrectly classified as outliers).

Ideal classification performance should yield a high CA, high TNR, low FPR, and low FNR. FPR and FNR are typically inversely coorelated such that one must be prioritized over the other. Since this application is designed to aid nuclear forensic investigations, minimizing FPR is the priority as falsely assigning an outlier to a known class is more costly than failing to identify an inlying element.

A. GS3 Data

GS3 is a tabletop exercise run by the Nuclear Forensics International Technical Working Group (ITWG) to support the development of a National Nuclear Forensics Library (NNFL), which aims to facilitate nuclear forensic investigations. GS3 uses simulated data that attempts to mimic the properties or characteristics of real uranium ore concentrate. Simulated data is used due to the sensitive and proprietary nature of such information. This data set comes courtesy of Naomi Marks from Lawrence Livermore National Laboratory and accurately simulates chemometric data used in nuclear forensic investigations.

The GS3 data set contains both a training set of labeled knowns and testing set of unknowns. The training set has 821 rows, 45 columns of numerical data, and 4 described classes: IAB, MORB, OIB, and ZCRFB. IAB is linearly separable from the other classes, which are overlapping. The testing set contains 60 elements whose true identities are excluded from the original exercise, but have been revealed here for the purpose of validation. The testing set contains a number of samples from each known class, as well as a number of samples from at least four distinct groups that do not belong to any of the known classes for a total of 44 and 16 samples, respectively.

The training set, by design, is missing about one third of all possible measurements. This further increases the difficulty of the exercise, but accurately reflects the reality where certain samples undergo certain tests and other samples are subject to different scientific measurements. Missing data in the training set is substituted with imputation, while columns containing missing data are dropped from the testing set, as imputation would prove inaccurate. Columns are then synchronized between the training and testing set to ensure each set includes measurements of the same metrics.

B. Results

The first implementation of N-SLOPE misclassified 10 samples in the testing set, while successfully identifying every outlier in the testing set and producing no false positive classifications. The modification to N-SLOPE allowed it to correctly classify one additional sample in the testing set. A classification summary detailing these results is shown in Table II.

TABLE II
N-SLOPE classification results on GS3 test data

	SCD	SIMCA	LOF	OC-SVM	PLS-DA	ELM	Original N-SLOPE	Modified N-SLOPE
CA	80.0%	85.0%	78.33%	73.33%	58.33%	88.33%	83.33%	85.0%
TNR	100%	100%	68.75%	100%	37.5%	62.5%	100%	100%
FPR	0%	0%	8.33%	0%	16.67%	10.0%	0%	0%
FNR	10.0%	6.67%	1.67%	23.33%	25.0%	3.33%	6.67%	5.0%

N-SLOPE performed perfectly when it came to classifying MORB, ZCRFB, and recognizing outliers, but struggled with

classifying IAB and OIB. Internally, certain algorithms appear better suited for identifying specific classes in this data set. SIMCA was the only algorithm that correctly identified every testing sample belonging to IAB, but SIMCA struggled with MORB (unlike the other algorithms) and OIB (like the other algorithms). LOF was the only algorithm that correctly identified every testing sample belonging to OIB, but LOF struggled with IAB (like the other algorithms) and detecting outliers (unlike the other algorithms).

C. Discussion

N-SLOPE performed very well on the GS3 data given the unique challenges posed by this data set. N-SLOPE correctly classified 85% of testing samples with the modified ensemble contribution calculation and correctly identified each of the 16 outliers as not belonging to any of the known classes. This second metric is particularly important, and impressive, as correctly identifying outliers is the main focus of one-class classification and the outliers included in this set were specifically chosen from a variety of excluded classes with the expectation that certain outliers would be difficult to distinguish from the known classes.

VII. Conclusion and Future Research

We showed the design and implementation of N-SLOPE and modified its ensemble contribution calculation. We validated N-SLOPE experimentally by measuring its one-class classification potential on a realistic nuclear forensics data set. The results show that N-SLOPE has high classification accuracy, increased further with the modification presented here, and excellent outlier detection capabilities.

One area for future research is exploring alternate weighting schemes for emphasizing specific algorithms in the ensemble under certain conditions.

VIII. Disclaimer

References

[1] M. Amer, M. Goldstein, and S. Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In *ODD '13 Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pages 8–15, 2013.

[2] K. V. Branden and M. Hubert. Robust classification in high dimensions based on the SIMCA method. *Chemometrics and Intelligent Laboratory Systems*, 79(1-2):10–21, 2005.

[3] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[4] M. Bylesjö, M. Rantalainen, O. Cloarec, J. K. Nicholson, E. Holmes, and J. Trygg. OPLS discriminant analysis: combining the strengths of PLS-DA and SIMCA classification. *Journal of Chemometrics*, 20(8-10):341–351, 2006.

[5] W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2017. R package version 1.0.5.

[6] A. Gosso. *elmNN: Implementation of ELM (Extreme Learning Machine) algorithm for SLFN (Single Hidden Layer Feedforward Neural Networks)*, 2012. R package version 1.0.

[7] Y. Hu, W. Murray, Y. Shan, and Australia. *Rlof: R Parallel Implementation of Local Outlier Factor(LOF)*, 2015. R package version 1.1.1.

[8] G. Huang, L. Chen, and C. K. Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Networks*, 17(4):879–892, 2006.

[9] J. H. Janssens, I. Flesch, and E. O. Postma. Outlier detection with one-class classifiers from ML and KDD. *International Conference on Machine Learning and Applications*, 2009.

[10] J. Kehl. N-SLOPE: A One-Class Classification Ensemble For Nuclear Forensics. Master's thesis, California Polytechnic State University, San Luis Obispo, CA, USA, 2018.

[11] S. S. Khan and M. G. Madden. One-class classification: Taxonomy of study and review of techniques. *CoRR*, abs/1312.0049, 2013.

[12] J. Kittler, M. Hatef, R. P. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[13] M. Kuhn, J. Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, C. Candan, and T. Hunt. *caret: Classification and Regression Training*, 2018. R package version 6.0-79.

[14] S. Lê, J. Josse, and F. Husson. FactoMineR: A package for multivariate analysis. *Journal of Statistical Software*, 25(1):1–18, 2008.

[15] Q. Leng, H. Qi, J. Miao, W. Zhu, and G. Su. One-class classification with extreme learning machine. *Mathematical Problems in Engineering*, 2015, 2015.

[16] M. G. Madden and T. Howley. A machine learning application for classification of chemical spectra. *Applications and Innovations in Intelligent Systems XVI*, pages 77–90, 2009.

[17] L. M. Manevitz and M. Yousef. Document classification on neural networks using only positive examples (poster session). In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 304–306. ACM, 2000.

[18] L. M. Manevitz and M. Yousef. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2001.

[19] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2017. R package version 1.6-8.

[20] M. M. Moya and D. R. Hush. Network constraints and multi-objective optimization for one-class classification. *Neural Networks*, 9(3):463–474, 1996.

[21] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018.

[22] B. Schölkopf, R. C. Williamson, A. Smola, and J. Shawe-Taylor. SV estimation of a distribution's support. In *Advances in Neural Information Processing Systems*, 1999.

[23] B. Stumpe, T. Engel, B. Steinweg, and B. Marschner. Application of PCA and SIMCA statistical analysis of FT-IR spectra for the classification and identification of different slag types with environmental origin. *Environmental Science & Technology*, 46(7):3964–3972, 2012.

[24] D. M. Tax and R. P. Duin. Support vector domain description. *Pattern Recognition Letters*, 20:1191–1199, 1999.

[25] D. M. Tax and R. P. Duin. Support vector data description. *Machine Learning*, 54(1):45?66, 2004.

[26] V. Todorov. *rrcovHD: Robust Multivariate Methods for High Dimensional Data*, 2016. R package version 0.2-5.

[27] H. Yang, P. R. Griffiths, and J. Tate. Comparison of partial least squares regression and multi-layer neural networks for quantification of nonlinear systems and application to gas phase Fourier transform infrared spectra. *Analytica Chimica Acta*, 489(2):125 – 136, 2003.

[28] T. Zou, Y. Dou, H. Mi, J. Zou, and Y. Ren. Support vector regression for determination of component of compound oxytetracycline powder on near-infrared spectroscopy. *Analytical Biochemistry*, 355(1):1 – 7, 2006.