Automatic Classification of Unexploded Ordnances Based on Electromagnetic Induction Data

Grayson Zulauf

1 Introduction

1.1 Background

Unexploded ordnances, in former war zones or military testing sites, pose a massive environmental and humanitarian problem worldwide, rendering huge swaths of land unsafe and unusable for the public. Bombs dropped during warfare or military testing contain a significant number of duds, and these unexploded bombs lay in dangerous wait for years on end. In the United States, a country without a major conflict in over a century, an estimated 11,000,000 acres of land contain a potential unexploded ordnance (UXO) hazard [1].

The successful cleanup of these zones would allow the safe development of acreage across the world, as well as the elimination of many deaths associated with setting these bombs off. Unfortunately, cleanup of these areas is currently very expensive, relying on simple metal detection to find the bombs. This method results in excavating harmless metal clutter in addition to the ordnances, amplifying the cost of site cleanup by at least an order of magnitude.

1.2 Problem Statement

Professor Fridon Shubitidze, an Assistant Professor at Dartmouth's Thayer School of Engineering, has developed an innovative way to discriminate UXOs from harmless metal clutter. The method measures the time decay of the electromagnetic energy emitted by the buried bombs. The time decay curves (3 curves for each target of interest) differ between bombs and clutter, allowing for differentiation and classification. This project won the U.S. Department of Defense's 2011 Project of the Year[2], a testament to the ingenuity of the technology.

Currently, classification is performed manually, with a human combing three times through thousands of objects, sorting clutter and unexploded ordnances. While Professor Shubitidze's group has used this method to successfully identify all UXOs across the DoD's test sites, a robust algorithm would significantly expedite the classification.

1.3 Challenges

This particular problem poses a number of challenges beyond the simple development of a machine learning algorithm. Firstly, the algorithm will receive a very small number of ground truths - at the limit, only one for each type of ordnance. Secondly, success must be judged by the number of false negatives generated - a successful algorithm should generate

zero, while minimizing the number of false positives. The overall error rate, the typical measure for an algorithm, will be considered, but secondarily to the false negative rate.

Lastly, there are a number of different target configurations. For a given 'target' in the field, this target may also be associated with one other target, two other targets, or on its own. Thus, for any 3 target set, we must consider that each of the 7 possibilities may reveal the target to be a UXO. Figure 1 shows all of these possibilities, as well as the 7 graphs associated with each possibility.

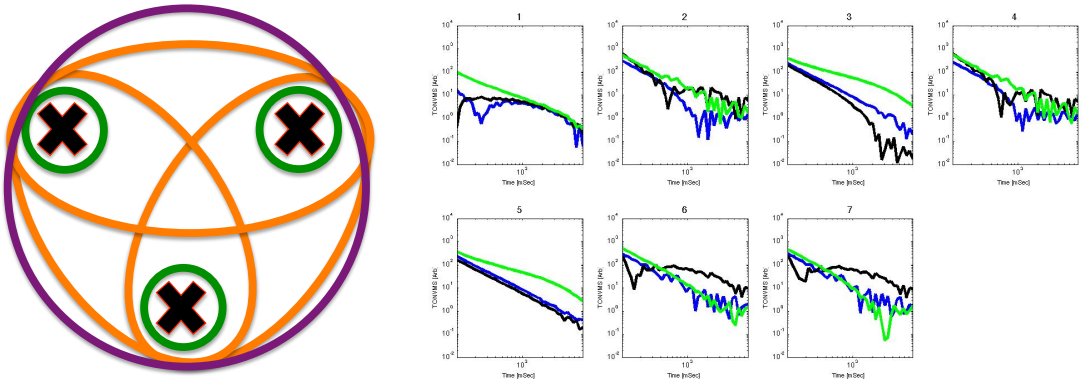

Figure 1: Three Potential UXOs, shown as black 'X's. Green circles show the 3 possibilities if they are considered alone, orange shown the 3 possibilities if they are considered in sets of 2, and purple circle shows the possibility that the three are actually one target. Graphs on right show these 7 distinct possibilities.

2 Algorithms

2.1 AdaBoost

The AdaBoost algorithm, introduced in 1995 by Freund and Schapire, is a supervised learning algorithm that combines a weighted sum of weak classifiers to form a strong classifier [3]. During training, each weak classifier examines a different part of the feature space and performs classification based on this feature space partition. This classification is then assigned a weight (α_t) according to the error it produced. After completing the training of these classifiers, they are summed via their weights to form a strong classifier and a final hypothesis, H(x), where

$$H(x) = sign\left(\sum_{t=1}^{t} \alpha_t h_t(x)\right).$$

AdaBoost requires that each individual classifier, h(t), classify with at least 50% accuracy. A more detailed example of the AdaBoost application is shown below, in Figure 2, for a hypothetical training set containing 4 feature space partitions.

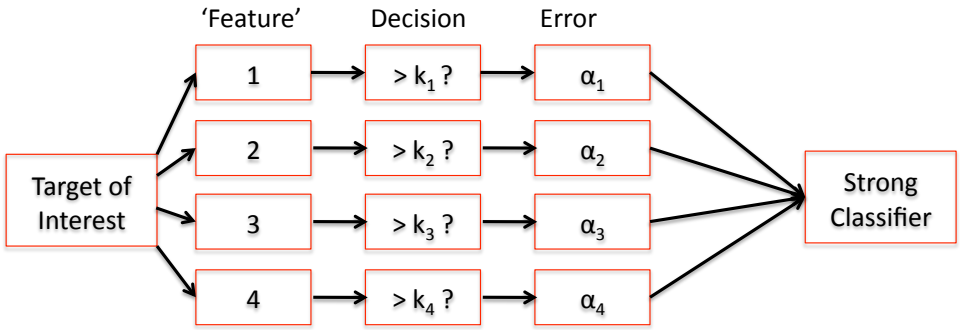

Figure 2: Example of the AdaBoost algorithm, with 4 distinct parts of the feature space examined.

A number of different feature space attributes have been considered to feed into Adaboost. The first weak classifier compared each individual test data point to the same data point in the library and used a Euclidean distance threshold to sort the samples. Each weak classifier, h(t), considered a different point through all feature points. For 42 data points and 3 vector directions, this corresponded to 126 weak classifiers. This comparison method is shown below.

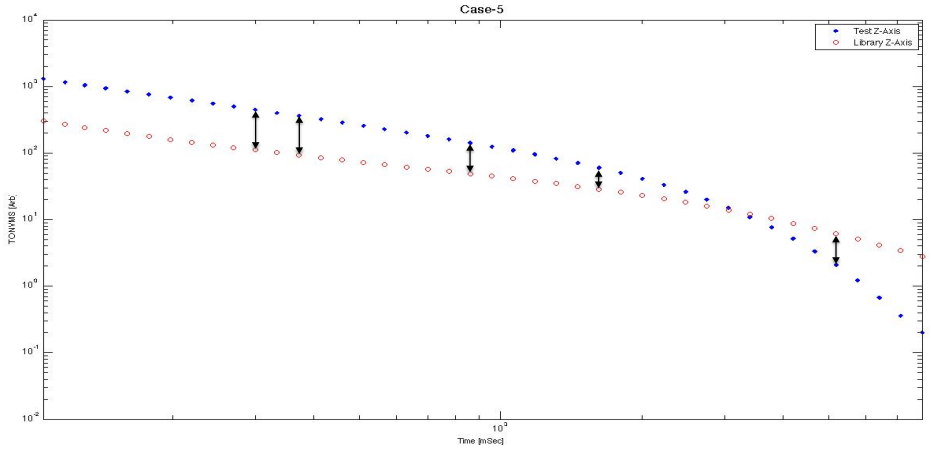

Figure 3: Using each data point comparison as a feature space partition for a weak classifier.

Secondly, the summed Euclidean difference between the test example and the library example across the X, Y, and Z-axis vectors were used to sort the test examples, according to a hyperparameter threshold defined by the user. This resulted in 3 weak classifiers, which were then summed to obtain the final classifier.

The success of these respective methods is discussed later in the results section.

2.2 Hierarchical Divisive Clustering

Due to the failures of the AdaBoost algorithm in solving this problem, a new unsupervised learning algorithm, Hierarchical Divisive Clustering, was implemented to better classify the data. A cluster can be defined as a "set of similar points that are highly dissimilar with other points in the dataset"[4]. Clustering algorithms have 3 key stages that must be selected, as shown in Figure 4, below.

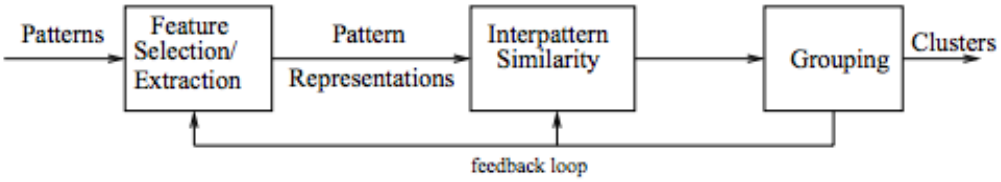

Figure 4: Clustering methodology and parameters selected by the user[5].

For our problem, we have selected the *features* as the X, Y, and Z-axes of the closest library case and the current target of interest. A threshold for maximum Euclidean distance will be used as the *interpattern similarity*, and there are only three eligible *groups*: unclassified targets, UXOs, and Clutter. Because we only care about clustering the data into two classes, we can largely ignore the implications of cluster merging and splitting [6]. In this application, only one cluster split is produced. Future work on this algorithm, however, may include separating types of UXOs, in which a discussion of the optimal splitting strategy must be revisited. The dendrogram for this particular problem is shown below, along with the decision pseudo-code.

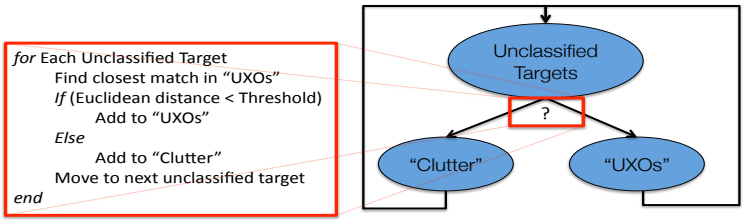

Figure 5: The pseudo-code for the decision point, as well as the dendrogram applied to this problem.

3 Results

3.1 AdaBoost

The AdaBoost algorithm was implemented, and coded with two different feature space partitions. Firstly, I used individual data points across all axes, as discussed previously. During each of these iterations, the error threshold was chosen to maximize the change in entropy, and this preferred threshold was used to define the classification for the data. For the 42 features in each axis, with 3 axes, this yielded 126 weak classifiers. Despite satisfying the necessary threshold of a sub-50% error rate, these classifiers were unable to form a strong classifier to successfully classify more than 75% of the ordnances correctly. Additionally, all classifiers exhibited a very high number of false positives. Figure 6 shows the results for the 126 classifiers.

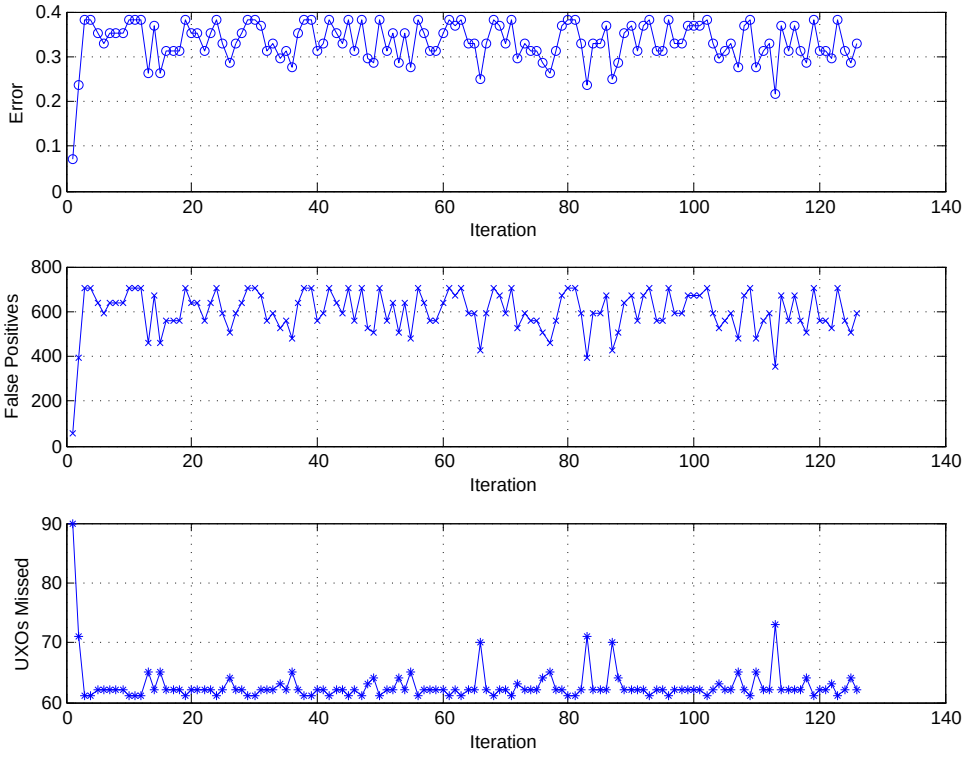

Figure 6: Adaboost error, false positives, and UXOs missed for each iteration.

As expected, Adaboost combines these weak classifiers to form a stronger classifier, with an error rate of **7.24%**, with **54** false positives and **90** UXOs missed out of 92 in the set. Adaboost appears to discount classifying the UXOs. There are nearly 20 times more clutter samples than UXOs in the training set, and the algorithm appears to find it

acceptable to use the error rate associated with essentially ignoring the classification of the UXOs.

In order to remedy this problem, I attempted to weight the classification by assigning them a 20 times greater weight in the error calculation. For the final classifier, this would give greater weight to the classifiers that classified the UXOs correctly. The results from this change are included below, in Figure 7.

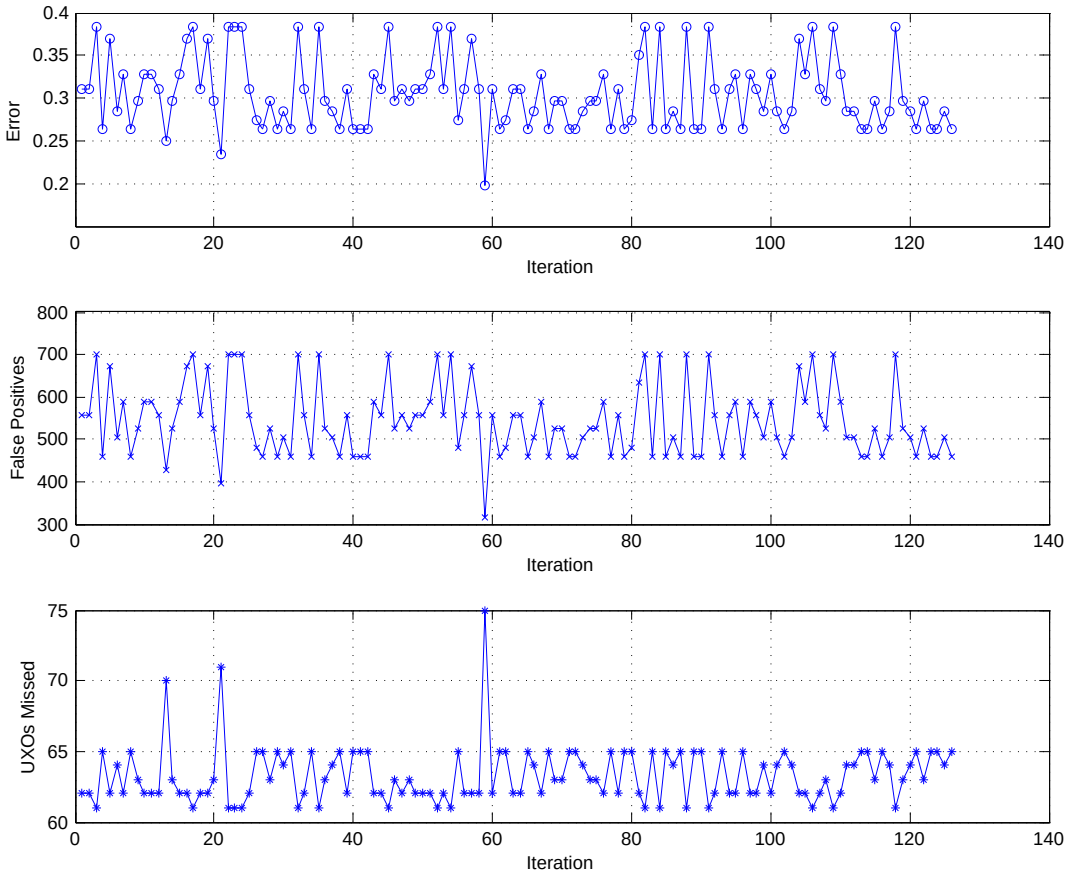

Figure 7: Adaboost error, false positives, and UXOs missed for each iteration, with weighting to classifying UXOs.

Indeed, the weighting curtailed the number of UXOs missed, from 90 down to **62**, but at the expense of **556** false positives and a **31.09%** error rate.

After the failure of this weak classifier, a new one was implemented: classification based on the Euclidean distance between the target of interest and the closest library sample across all points on a particular axis. This reduced the number of classifiers to only three, but we expected to get better classification from each, to ultimately form a strong

final classifier. Instead, none of the classifiers were able to reach the requisite sub-50% error rate on the training data, and the final classifier missed **21** UXOs and classified **1206** false positives, for an overall error rate of **66.75%**. This method would have performed better if the class estimates had merely been reversed.

3.2 Hierarchical Clustering

3.2.1 Hyperparameter Selection

A number of hyperparameters must be set by the user for this particular application. In order to select these values, an end-to-end test data simulation was run across all of the plausible hyperparameter values, while keeping the other ones fixed. Upon completion of the simulation, we selected the value that maximized the number of UXOs found while minimizing the overall error (i.e. minimized the number of false positives).

The error threshold is among the most important hyperparameters to define. As mentioned above, we must run the classifier for the 1-target, 2-target, and 3-target cases, and must define an error threshold for each category. The plots below show the results for the 1-target error threshold, which clearly show that a threshold of 10 gives the optimal results, with a high rate of UXO categorization and the ideal place on the ROC curve.

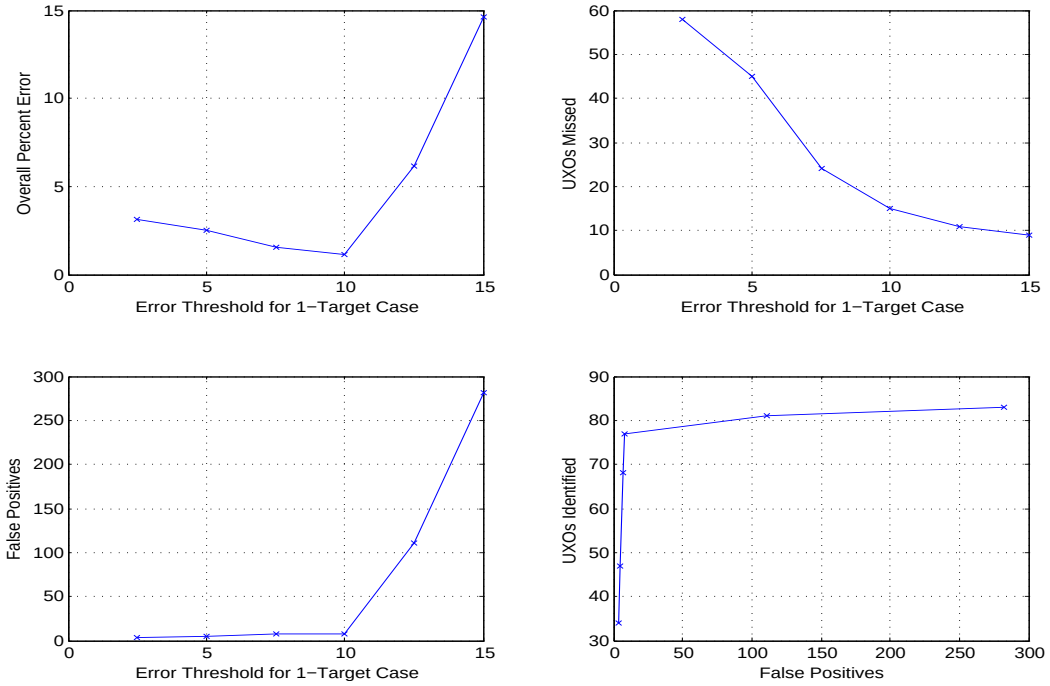

Figure 8: Curves showing train results while varying the 1-target error threshold. All four plots point to an ideal threshold of 10.

Next, the same procedure was performed for the 2- and 3-target error threshold, while holding the other hyperparameters fixed (for simplicity, these were set to be the same - this constraint will be lifted in future work). Here, the threshold is not so easily defined, as the error rate has no minimum and the ROC appears to have two plateaus, where one may trade off 4 identified UXOs for 500 false positives. Currently, we have selected the first plateau as the optimum error, with a threshold set at 9.

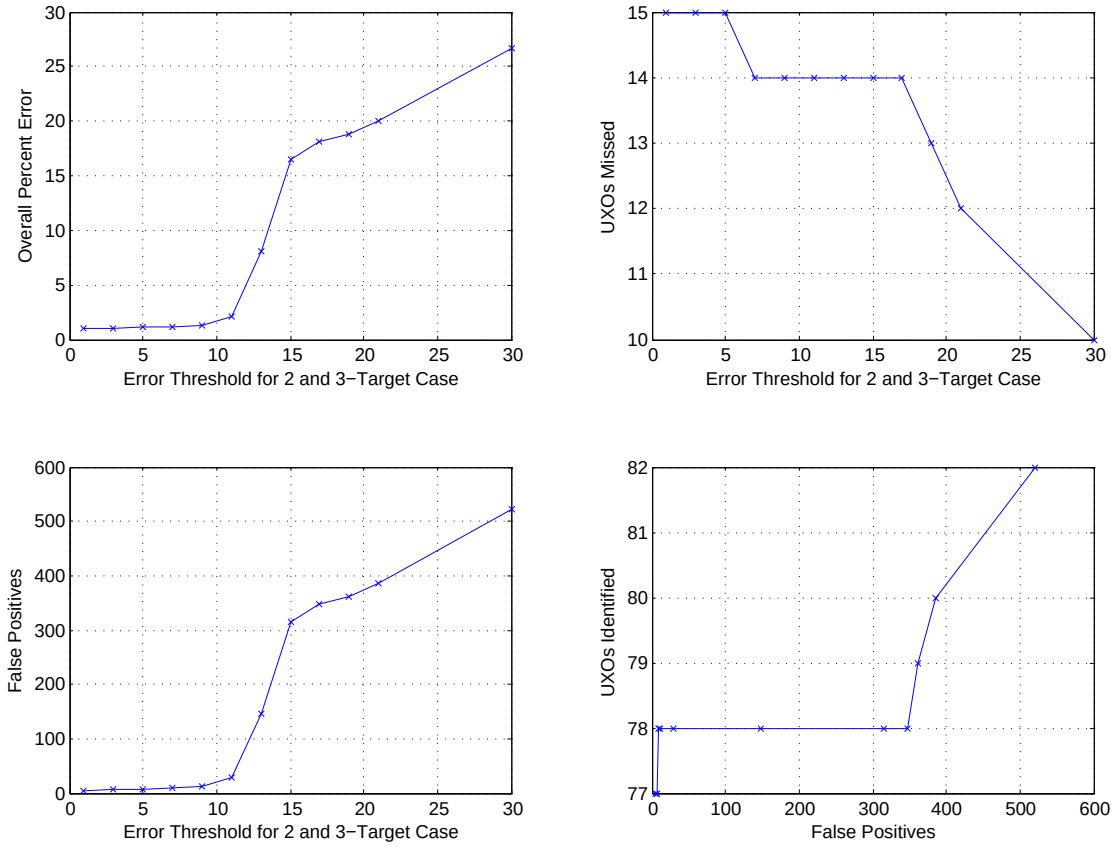

Figure 9: Curves showing train results while varying the 2 and 3-target error threshold. An ideal threshold of 9 was selected, based on the ROC curve's first plateau.

Lastly, we defined the ideal time bound, i.e. the range of points for a given curve that we will accept as data. From mere inspection, we can see that using the entire available feature set introduces large noise variations, and the training results support this assumption. With very stringent time restrictions (ostensibly to eliminate the noise), however, we have not received enough data to correctly differentiate the results and classify numerous false positives. Using the same method as above, we have chosen an ideal time bound of 3900 seconds.

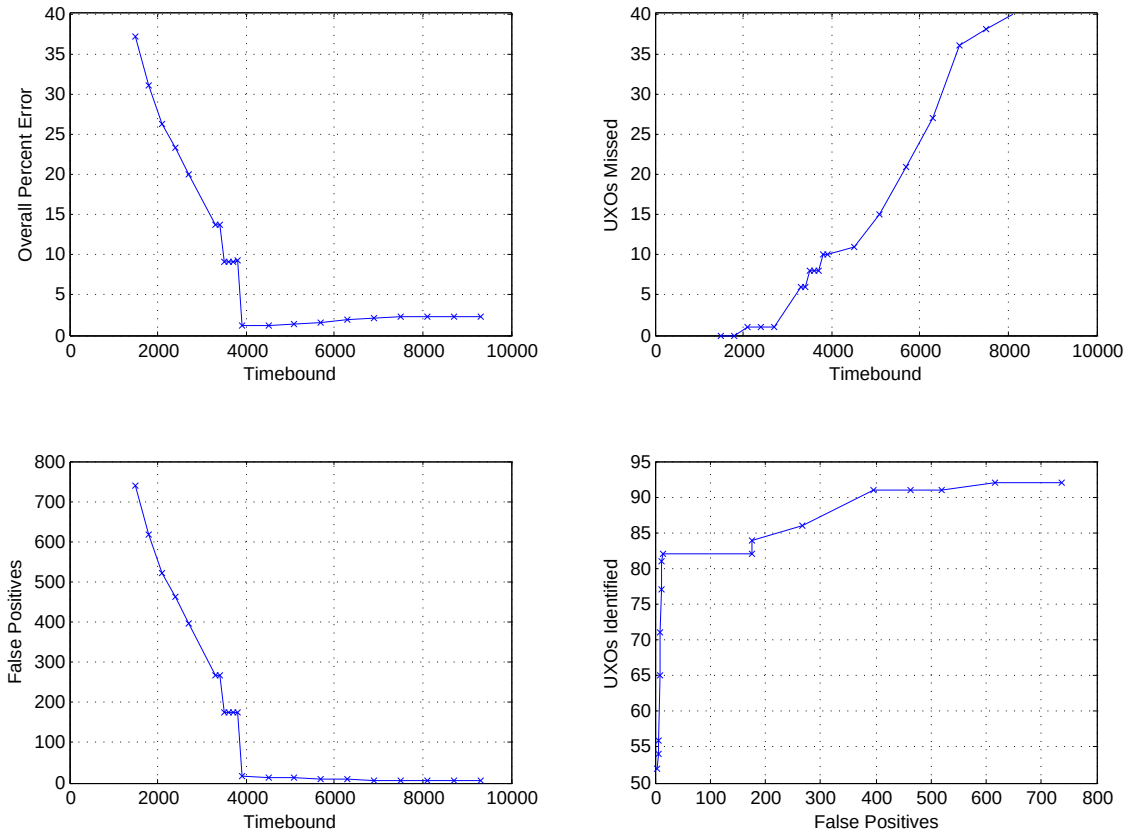

Figure 10: Curves showing train results while varying the time boundary, keeping the error thresholds fixed. An ideal time bound of 3900 seconds was selected, based on the ROC curve's plateau.

Other important hyperparameters exist: the number of iterations to run for each target case (1, 2 and 3), the order through which to approach each target, and the number of iterations to run through the entire program. I have examined the number of iterations for each case, with two iterations providing the best results (highest number of UXOs identified and least number of false positives) in the 1-target case, and 1 iteration each for the 2-target and 3-target case. To this point, the ideal number of iterations for the clustering algorithm has not been identified, and a default of 1 has been used. The current default order is 1-target to 2-target to 3-target, but the selection of these will be explored before the final presentation.

3.2.2 Overall Results

The clustering algorithm has performed very well on the training data, with an error rate consistently under 5%, the consistent identification of around 90% of the unexploded ordnances included, and an acceptably low false positive rate. For the ideal hyperparameters outlined above, the Fort Sill data set returns an overall error rate of **1.28%**, with **15** false positives and **10** UXOs missed out of 92 in the set.

For most algorithms, this small error and 10% miss rate would likely be considered a success. As discussed previously, however, we must return zero false negatives, and have not reached that critical point while minimizing the number of false positives. For small timebounds, all of the UXOs are identified, but at the expense of over 600 false positives.

In this training set, we are attempting to identify two types of unexploded ordnances: 37-mm and small pipes. They have been considered together in all previous parts of this paper but may also be considered separately. If considered separately, only 1 of the 37-mm targets is missed (with 37 false positives), and only 2 of the small pipes are missed (with 39 false positives). This is a significant improvement over the 10 combined UXOs missed when considered together. The potential separation of the targets, either through further clustering or multiple iterations, is a topic that will be explored in the coming weeks.

After using the training set to define the hyperparameters, the algorithm was tested on 2 other DoD training sites, Spencer Naeva and Camp Bealle. On the Spencer Naeva test site, the algorithm identified **111** out of 121 UXOS, with **616** false positives and an overall error rate of **31.99%**. When the 37-mm and small pipe categories were considered separately, the algorithm returned **111** UXOs, **650** false positives, and a **33.72%** error rate.

The final test site used for the milestone report was Camp Bealle, with a total of 40 unexploded ordnances to be identified. Here, the algorithm returned **49** out of 81 UXOs with **62** false positives and an overall error rate of **4.73%**. When the UXO types were considered separately, the algorithm returned **50** UXOs, with **107** false positives and an error rate of **6.94%**.

As the test results show, there is significant improvement needed in the algorithm to ensure its extensibility across all test sites without requiring the redefinition of the hyperparameters. This will be the focus of the work conducted in the last few weeks of the term.

4 Future Work

The checkpoints for the milestone have been completed (code completed and initial results returned), and the project is on pace to be completed for the end of the term. I plan to refine the Hierarchical Clustering method code for efficiency gains, ideal selection of hyperparameters, and the expansion of the number of sets upon which the algorithm is tested. The Adaboost algorithm appears ill-suited to this problem, and I will likely cease

to pursue it, with one exception.

Depending on the results of the hyperparameter selection, I may choose to incorporate the AdaBoost algorithm into the clustering code. While the clustering code does not appear to be a *weak* classifier (under 5% error rate), I would like to explore feeding different hyperparameter values into the Adaboost algorithm and allowing it to generate weights for the results. For example, a number of different clusters could be generated based on the error threshold, and Adaboost could weight these results on a training class.

References

[1] Fridon Shubitidze Dartmouth College, Thayer School of Engineering http://engineering.dartmouth.edu/emsg/

[2] Thayer School News November 30, 2011 'Dartmouth Engineering Professor earns DoD Project-of-the-Year Award' http://engineering.dartmouth.edu/news/dartmouth-engineering-professor-earns-dod-project-of-the-year-award/

[3] Yoav Freund and Robert E. Schapire. *A decision-theoretic generalization of online learning and an application to boosting.* Computational Learning Theory: Second European Conference, EuroCOLT 95, pages 2337, Springer-Verlag, 1995.

[4] S. Vadapalli, S. Valluri, and K. Karlapalem. *A simple yet effective data clustering algorithm.* Center for Data Engineering, IIIT, Hyderabad, India, 2006.

[5] A.K. Jain, M.N. Murty, and P.J. Flynn. *Data Clustering: A Review.* ACM Computing Surveys, Vol. 31, No. 3, September 1999.

[6] C. Ding and X. He. *Cluster merging and splitting in hierarchical clustering algorithms.* NERSC Division, Lawrence Berkeley National Laboratory, 2002.