Automatic Classification of Unexploded Ordnances Based on Electromagnetic Induction Data
Final Paper

Grayson Zulauf

March 8, 2013

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) are different for bombs and clutter, providing a method for 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 State of the Art

After a very successful demonstration of the technology, Prof. Shubitidze and his team received another grant to develop automated sorting algorithms, resulting in a number

of Ph.D. students working directly on this problem. While human classification identifies 100% of the UXOs and leaves 95% of non-UXO clutter unburied [2], none of the applied algorithms have achieved anywhere near this level of success.

The published results [3], which used Neural Networks, SVMs, and clustering via Ward linkage criteria, were only able to classify 100% of UXOs on a training set at the cost of a large number of false positives. A 3-layer NN returned approximately 500 false positives (out of 1500 targets of interest), while the SVM performed even worse, with nearly 600 false positives on the same training set. The clustering algorithm missed one anomaly. These results are revisited later and compared to the results achieved here.

1.4 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.

Crucially, cross-site validity must be guaranteed with any developed algorithm. Data collected from different test sites have large variations in the EMI decay curves, primarily due to unique soil compositions across the United States (and globally, although no international sites are considered here). Included below, in Figure 1, are the EMI curves of the same bomb type from two different test sites, with vastly different footprints. The variation shown here is indicative of the major challenge of producing a classifier that works well at all test sites.

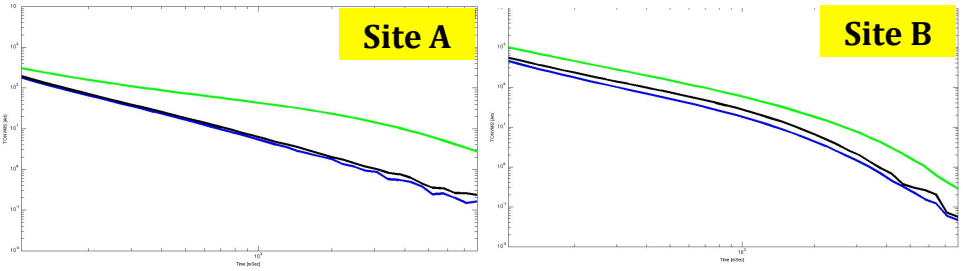

Figure 1: *37-mm targets shown from two test sites, Fort Sill and Spencer Naeva. While these are the same bomb type, the curves (especially the Z-axis, in green, that conveys the most information) are largely different, underscoring the difficulty of developing an algorithm that classifies well across different test sites.*

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 2 shows all of these possibilities, as well as the 7 graphs associated with each possibility.

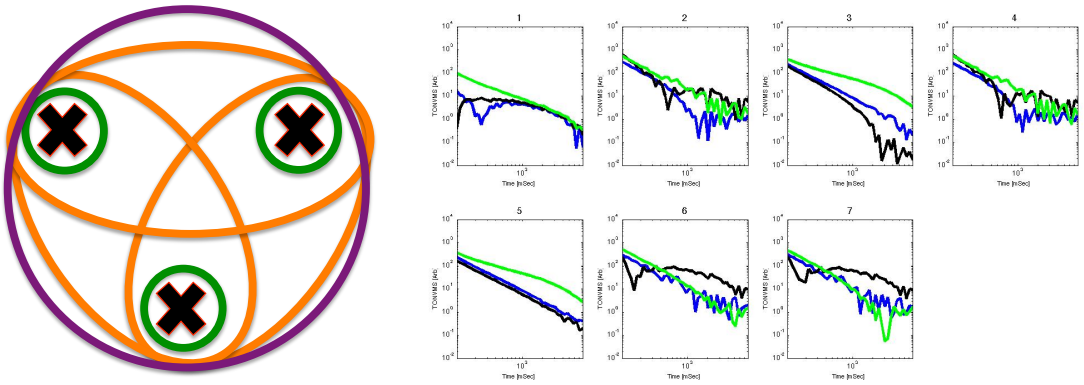

Figure 2: *Three Potential UXOs, shown as black 'X's. Green circles show the 3 possibilities if they are considered alone, orange shown the 3 possibilities if they are considered in sets of 2, and purple circle shows the possibility that the three are actually one target. Graphs on right show these 7 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 [4]. 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 3, for a hypothetical training set containing 4 feature space partitions.

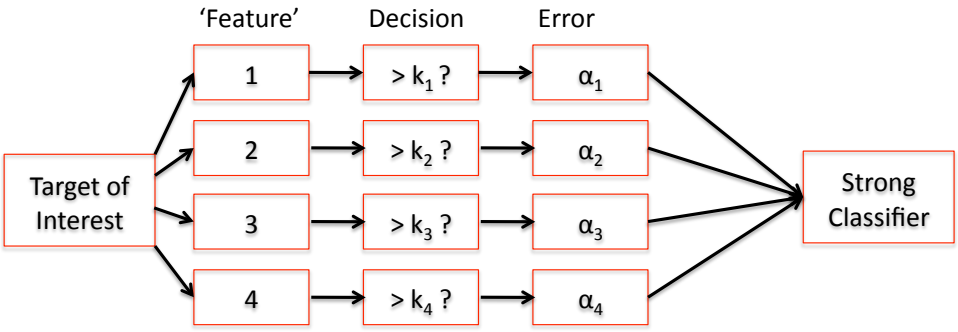

Figure 3: *Example of the AdaBoost algorithm, with 4 distinct parts of the feature space examined.*

The critical design decision is the choice of weak classifier, and especially the feature space partition each weak classifier will consider. Here, a number of different feature space attributes were considered to feed into Adaboost, and the final results used three separate weak classifier partitions. 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, producing a weak classifier h(t). The threshold for sorting was chosen as the one that produced the purest groupings. For 42 data points and 3 vector directions, this corresponded to 126 weak classifiers. This comparison method is shown below.

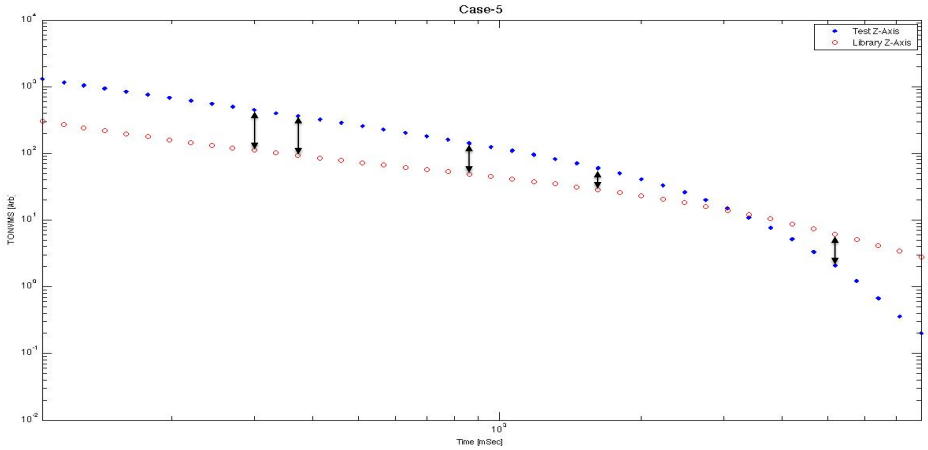

Figure 4: *Using each data point comparison as a feature space partition for a weak classifier.*

Because the data set was hugely skewed toward clutter (by an approximately 20:1 ratio), this comparison method was then altered to weight the misclassification of UXOs 20 times

more than the misclassification of clutter objects in the training set.

Lastly, 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"[5]. Clustering algorithms have 3 key stages that must be selected, as shown in Figure 5, below.

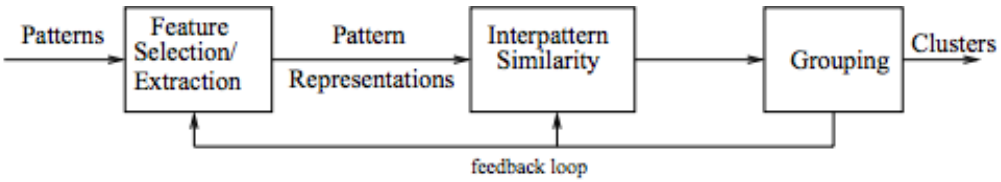

Figure 5: *Clustering methodology and parameters selected by the user [6].*

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. The *interpattern similarity* decision was made using three separate criteria (discussed below), 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 [7]. 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 in Figure 6.

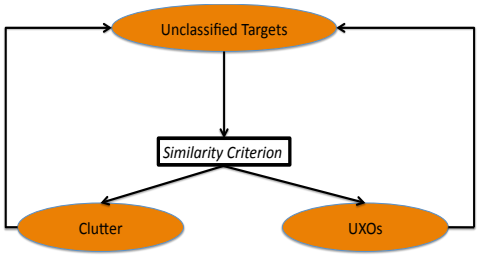

Figure 6: *The dendrogram applied to this problem, with an arbitrary sorting criterion.*

After the milestone, a number of different sorting decisions were considered in an attempt to improve the extensibility across different testing sites, as discussed in *Challenges*.

2.2.1 Closest Centroid

This sorting decision is essentially a modified version of the k-means algorithm, implemented as a benchmark for the program. Here, however, the 'clutter' centroid is well-defined from the beginning of the program by automatically assigning all targets to it, and it thus moves very little from this starting point. This algorithm was implemented as such due to the uneven distribution of the data and the *apriori* assumption that any given target will likely be clutter.

2.2.2 Closest Target Threshold

A maximum Euclidean distance decision point sorts a given target of interest based on its distance from the closest target in the 'UXOs' group. Targets below a set threshold (optimized on the training set) are added to the 'UXO' group, while those above this threshold are assigned to the 'Clutter' cluster.

2.2.3 Centroid Distance Threshold

Here, a threshold for maximum Euclidean distance from the UXO cluster centroid is used as the sorting criterion. This algorithm was implemented after the milestone, with the hypothesis that a centroid comparison would increase extensibility across test sites by dulling the comparison's sensitivity to outliers.

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 7 shows the results for the 126 classifiers.

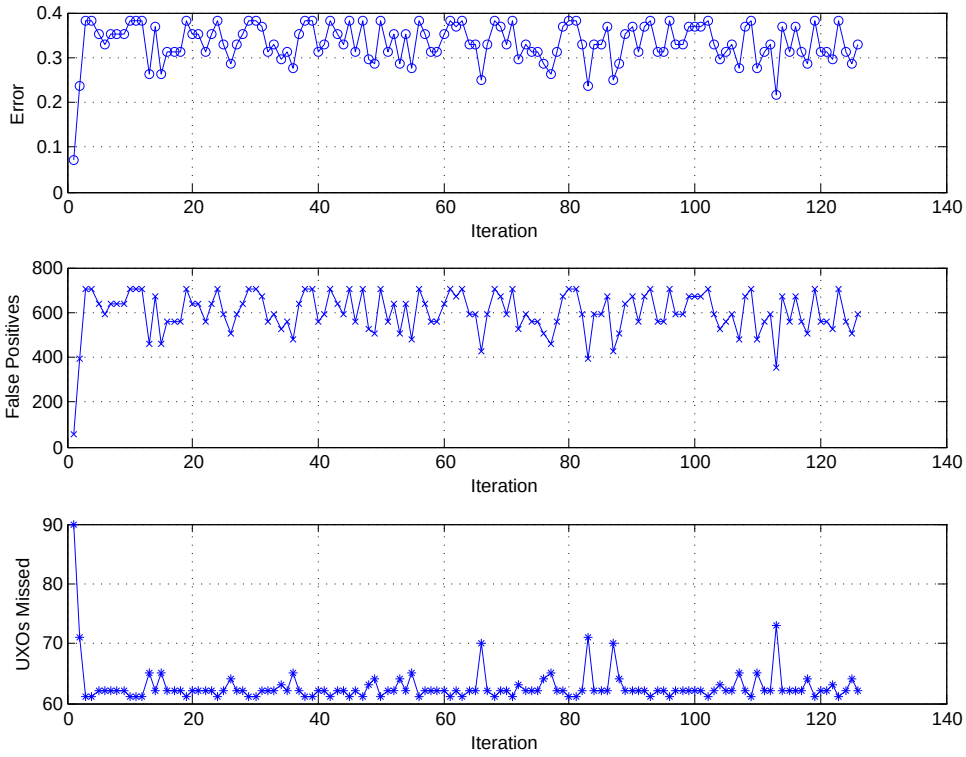

Figure 7: *Adaboost error, false positives, and UXOs missed for each iteration.*

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 8.

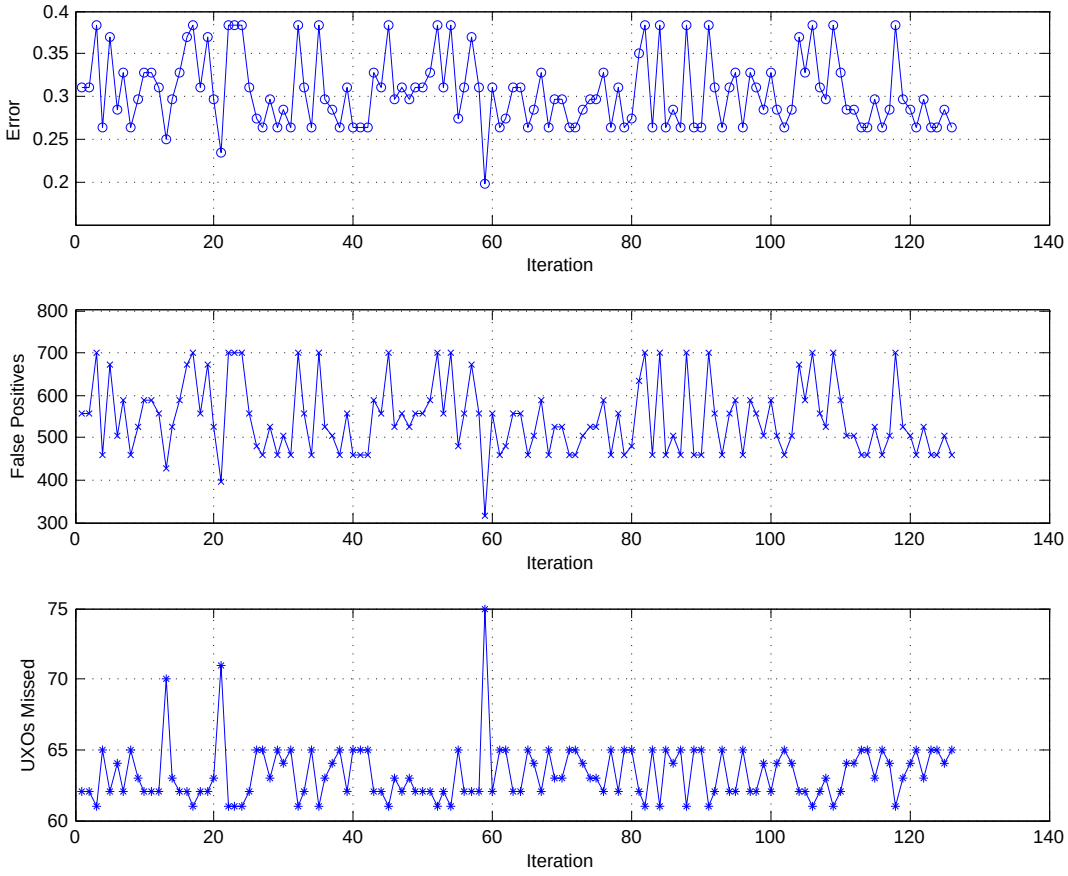

Figure 8: *Adaboost error, false positives, and UXOs missed for each iteration, with weighting to classifying UXOs.*

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.

Table 1 shows a summary of these results, below. None performed well enough on the training set (Fort Sill) to move to the test sets.

Table 1: *The error rates for the three different weak classifiers implemented with the Adaboost algorithm, on a site with 1988 targets and 92 UXOs. None performed satisfactorily on the training set, and thus were not tested.*

Adaboost Algorithm Results			
Weak Classifier Method	**Overall Error**	**UXOs Found**	**False Positives**
Points Comparison	7.25%	2 out of 92	54
Weighted Points Comparison	31.09%	30 out of 92	556
Axes Comparison	66.75%	71 out of 92	1206

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.

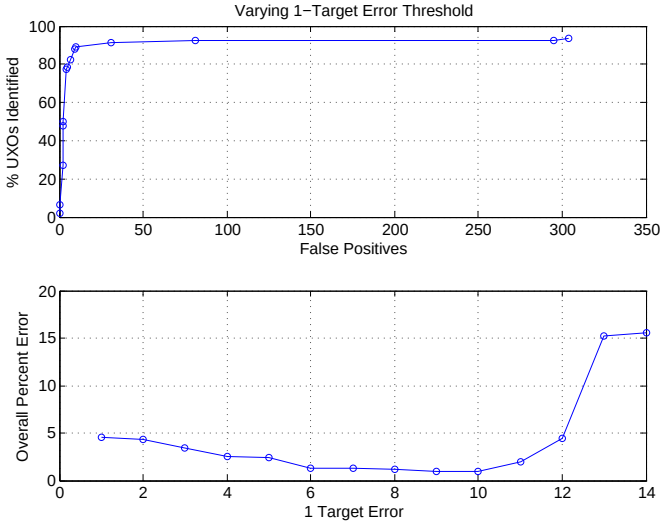

Figure 9: *Curves showing train results while varying the 1-target error threshold. Both plots point to an ideal threshold of 10.*

Next, the same procedure was performed for the 2- and 3-target error threshold, while holding the other hyperparameters fixed. 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. For the 2-target case, we have selected the first plateau as the optimum error, with a threshold set at 9.

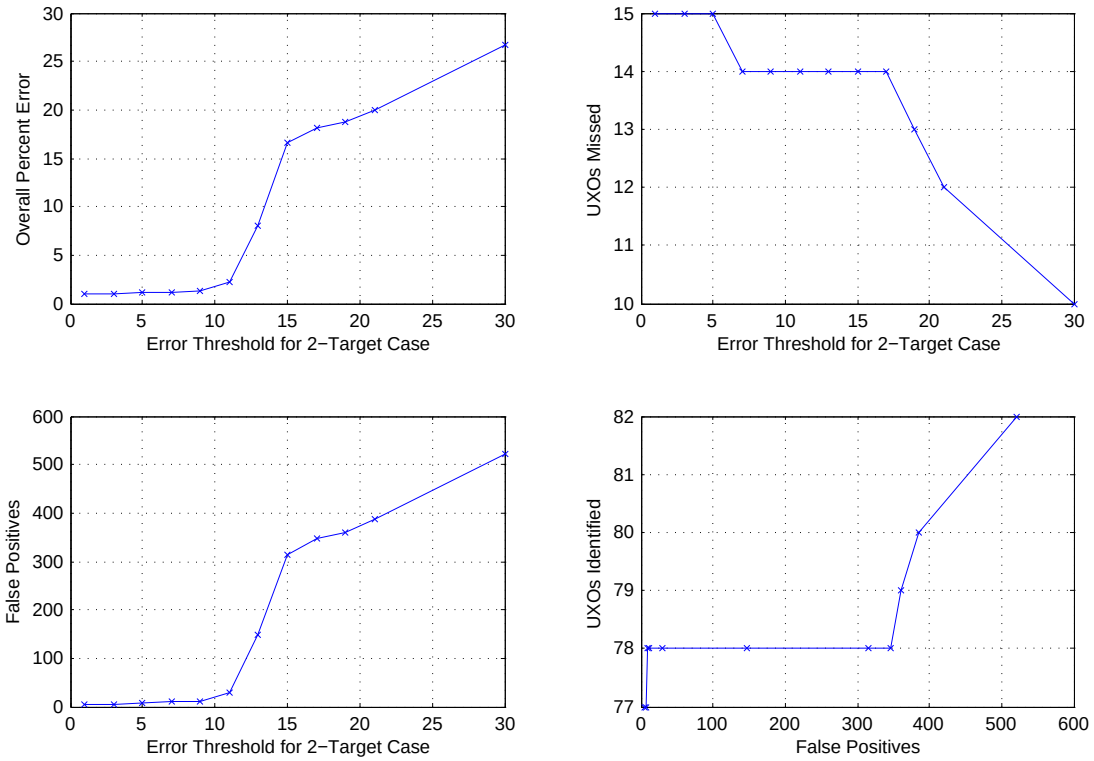

Figure 10: *Curves showing train results while varying the 2 and 3-target error threshold. An ideal threshold of 9 was selected, based on the ROC curve's first plateau.*

The same method was run for the 3-target case, with a threshold of 3 chosen. This was an important iteration from the milestone report, where the 2-target and 3-target thresholds were optimized together. Optimizing these hyperparameters separately significantly reduced the number of false positives returned with no effect on the UXOs categorized.

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.

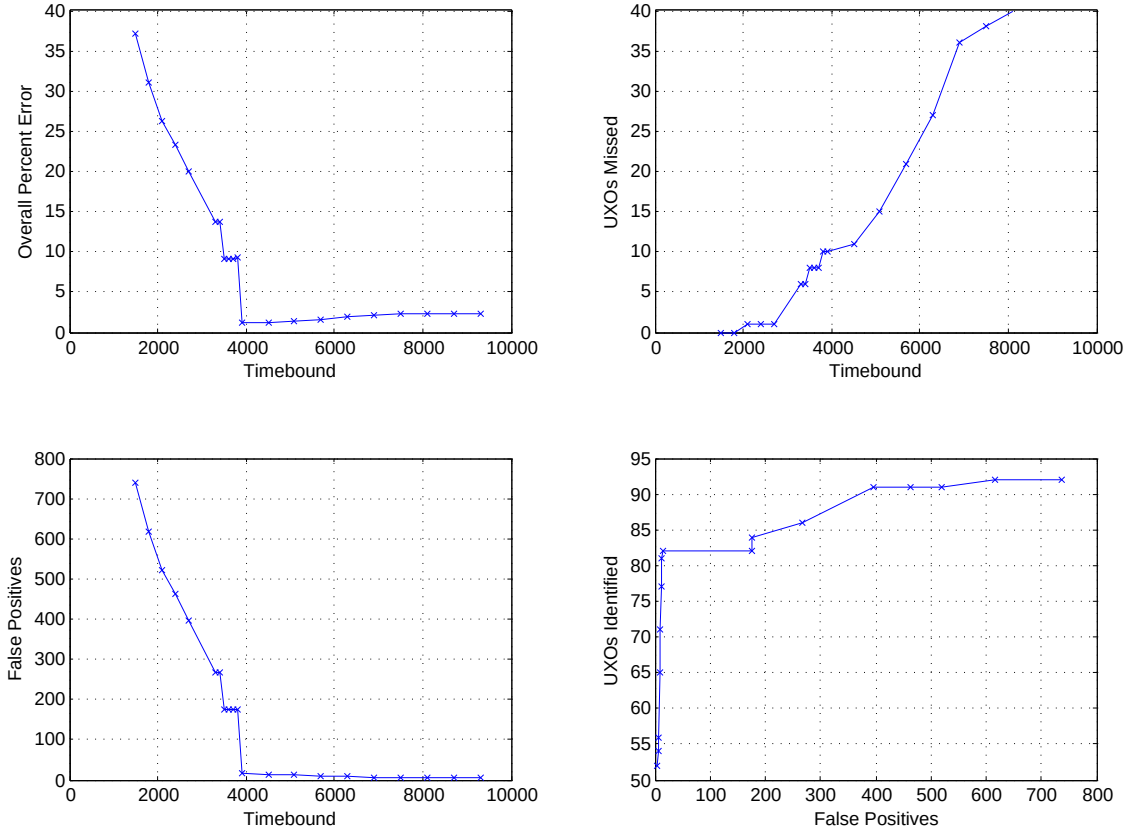

Figure 11: *Curves showing train results while varying the time boundary, keeping the error thresholds fixed. An ideal time bound of 3900 seconds was selected, based on the ROC curve's plateau.*

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. The current default order is 1-target to 2-target to 3-target, and alternating this order in a number of different ways produced worse training set results.

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 **3.17%**, with **55** false positives and **8** UXOs missed out of 92 in the set. This was an improvement over the 10 UXOs misclassified at the time of the milestone report.

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.

This number of false positives is comparable to the state of the art research in the field. Figure 12, below, shows 4 different training set results set against the results from this method.

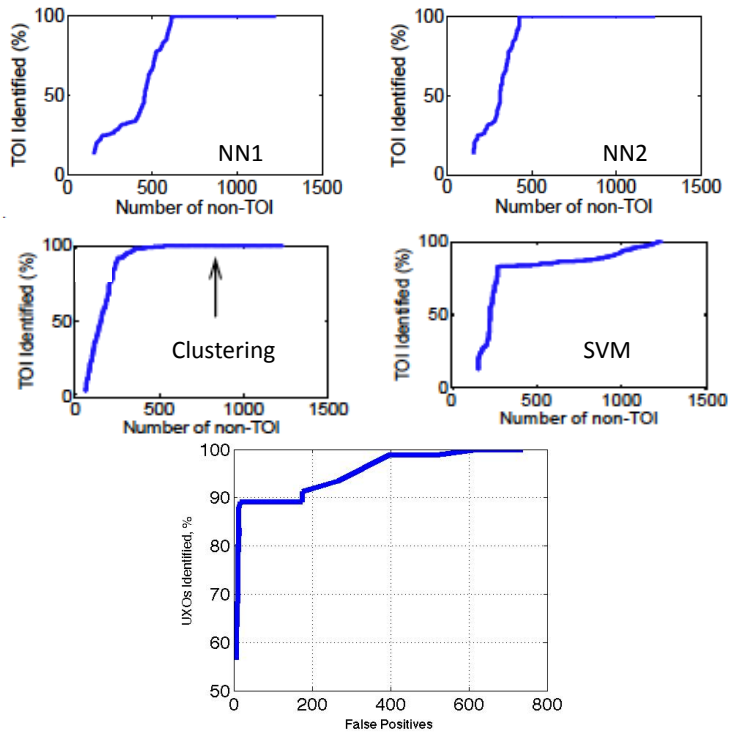

Figure 12: *Four ROC curves from the state of the art research in the field [3], with the text showing the type of classifier used. The bottom curve is the ROC curve (for varying the timebound) for the training set used in this project, with 617 false positives for 100% UXO classification. This project achieved parity with the state of the art research.*

The state of the art training set, however, had only 1464 targets, while our training set had 1988 targets of interest. Thus, this method classifies all UXOs with 31% of the non-UXOs misclassified. This actually improves upon the best published results for this

application. Table 2 compares the results in more depth.

Table 2: *A comparison between the results from our clustering algorithm and the state of the art research in the field [3]. All false positive values taken at the point in the ROC curves where all UXOs are identified.*

A comparison between the state of the art and our clustering method					
	Algorithm Implemented				
Error Measure	**NN 3-node**	**NN 10-node**	**SVM**	**Cluster**	**Our Cluster**
% UXOs Identified	100%	100%	80%	99%	**100%**
% False Positives	33.7%	41.2%	20.6%	55.0%	**31.0%**

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 **64** out of 73 UXOS, with **715** false positives and an overall error rate of **36.99%**. The final test site used for the milestone report was Camp Bealle, with a total of 81 unexploded ordnances to be identified. Here, the algorithm returned **53** out of 81 UXOs with **181** false positives and an overall error rate of **10.51%**.

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. We thus implemented the centroid distance threshold for the decision point, with the hypothesis that it would reduce the sensitivity to outliers in the clustering algorithm.

As expected, the centroid distance comparison maintained its error and UXO classification rates over the training and test sites better than the closest target comparison. Although the centroid distance comparison did not perform as well on the training or test data, it appears to hold promise as a more extensible application. Table 3 shows the final performance of these two unsupervised classifiers over all of the sites.

Table 3: *A comparison between the results from the closest target threshold and the centroid distance threshold. The closest target performs better on the task of classifying UXOs, but the centroid distance threshold appears to be more extensible across sites. False Positive rates (False Positives/Number of Targets) are denoted as % FP, with the percentage of UXOs found denoted as % UXOs.*

A comparison between two different clustering decision criterion.						
	Site					
	Ft. Sill		**Spencer Naeva**		**Camp Bealle**	
Criterion	*% UXOs*	*% FP*	*% UXOs*	*% FP*	*% UXOs*	*% FP*
Closest Target	91.3%	2.7%	87.6%	42.9%	65.5%	10.5%
Centroid	78.2%	5.7%	73.9%	6.8%	55.5%	2.8%

4 Future Work

Firstly, this algorithm produced positive results, indicating a step forward from the current state of the art in a real-life, humanitarian problem to which machine learning can significantly incentivize cleanup of these areas. I hope to be working with Professor Shubitidze towards the publication of these results during spring term, after refining the algorithm and testing its application across more DoD test sites.

In this training set, we attempted to identify two types of unexploded ordnances: 37-mm and small pipes. They have been considered together in 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 8 combined UXOs missed when considered together. The potential separation of the targets, either through further clustering or multiple iterations, is a topic that should be explored further.

Additionally, the centroid comparison algorithm was implemented after the milestone in an effort to improve extensibility across the different test sites. Based on the promise shown here, I will be refining this algorithm and spending more time attempting to improve its classification of UXOs.

Lastly, for the eventual real-life applications of this algorithm, it should have a probabilistic (or confidence) component. Ideally, those targets with a low confidence could be flagged and examined by a human expert. A large, but addressable, shortcoming of this algorithm is the strictly binary classification with no confidence measure.

5 External Code

No external software was used for this project. Professor Shubitidze provided the code for loading in the data correctly and was crucial in decoding the format of the .mat files. After receiving guidance in getting the data loaded correctly, the algorithms were independently written.

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] Alex Bijamov, Fridon Shubitidze, Juan Pablo Fernandez, Irma Shamatava, Benjamin E. Barrowes, Kevin O'Neill. *Comparison of supervised and unsupervised machine learning techniques for UXO classification using EMI data.* Proc. SPIE 8017, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XVI, 801706 (May 23, 2011); doi: 10.1117/12.884076.

[4] 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.

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

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

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