Classification of Induction Signals for the EXO-200 Double Beta Decay Experiment

Jason Chaves, *Physics, Stanford University*
Kevin Shin, *Computer Science, Stanford University*

I. Introduction

THE EXO-200 experiment searches for double beta decay in Xenon-136, in both the 2-neutrino and 0-neutrino modes. The EXO-200 detector has a set of wires that collect all charge released within the detector volume. Aside from the double beta decay events that occur within the detector, there are other types of background events that can release charge, such as compton scattering. Some of these types of events are single-site, whereas others are multi-site. Being able to classify single-site from multi-site events is important for accurately fitting backgrounds which obscure the double beta decay signal. Multi-site events are simply classified as events where multiple charge clusters are collected by the wires (with certain timing constraints). An issue is, however, that there are some instances where a wire collects charge from a true single-site event, but the closely packed adjacent wires may also show an induction signal, which can be mistaken as an additional charge collection in Event Reconstruction, and can lead to mistakenly classifying the event as multi-site. Removing these induction signals during the Event Reconstruction would improve the single-site/multi-site fraction by $\sim 10\%$ and the total reconstruction efficiency by $\sim 30\%$, two critical improvements for increasing our sensitivity to detecting 0-neutrino double beta decay for the experiment. The ultimate goal is to find a model that can be trained in one sitting and then can be incorporated into Event Reconstruction to classify all signals in the experimental data.

II. Data Sets

Running Event Reconstruction on low-background (LB) physics data from the experiment provides us with 14 real-valued attributes for wire waveforms *(Cluster Raw Energy, UWire Raw Energy, Pulse Width at Half-Max, Unshaped pulse integral, Max Pulse Height, Channel Number, Chi-2 Fit to Template Dep Signal, Restricted Chi-2 Fit to Template Dep Signal, Chi-2 Fit to Template Ind Signal, Restricted Chi-2 Fit to Template Ind Signal, Fit Pulse Amplitude, Time between max and min Pulse, Incoming Pulse Width, Total Pulse Width)*.

A set of 500,000 LB real events has been collected, each sample with the 14 attributes calculated. Moreover, to provide labelled data, there is also a Monte Carlo (MC) simulation package designed to model events in the detector and the electronics which process the wire signals. From running the MC simulations, we have obtained 120,000 MC events, each with the 14 attributes calculated, but also with a labeling of whether a signal is induction or collection (aka deposition). An issue that we kept in mind with the MC data is that the simulation isn't particularly accurate with producing induction signals, because it only uses one template induction signal, whereas in the LB data there can be varying induction signals, and also noise signals which we would also like to discriminate as induction, or better phrased "not-collection" if we can. Luckily, the MC data does accurately model deposition signals, and so there's reasonably good agreement between MC deposition and LB deposition signals.

For the sake of testing different classification models, we have taken advantage of the fact that induction and collection signals are usually distinguishable to a human, and so we have been able to manually parse through the LB data and hand-label several hundred signals that we are confident we can identify. In order to ensure our confidence in our labeling, we also performed a cross-validation of our "human model" on our labeled MC data and attempted to classify some of the labeled MC data and received an accuracy score of 98.75% on the training set. We would like to mention that the above two Monte Carlo and low-background datasets were collected by Dave Moore, a Post-Doc on the experiment. We then converted and formatted the data such that it was no longer dependent on ROOT or any of the EXO libraries located on SLAC computing in order to locally train and test our models.

III. Prior Attempts at Classification

Work on removing induction signals during Event Reconstruction started with comparing labeled MC events for each of the 14 metrics. Figures, such as the one below, were made for each metric, showing the distribution of deposition and induction signals in that metric and where the cut line would be. Based on simple cuts on metrics, a discriminator was proposed consisting of 3 cuts (based on 4 attributes) which a signal must fail all of them to be called induction. The standard for the cut line of the data was chosen to be where 99% of the deposition signals would pass through the discriminator. These figures showed that single metrics could provide very good removal of induction data, but this was all done on MC Data and not on LB data. The distribution of events from real LB runs from the detector for these single metrics, however, was not as nicely bimodal, and so there's the difficulty of proposing a model that we are fairly certain passes 99% of LB deposition while also being able to remove some, if not most, of the induction signals. We suspect that this model is too simplistic in terms of utilizing all of the available information to attain the best possible induction removal and thus plan to build upon it using machine learning techniques.

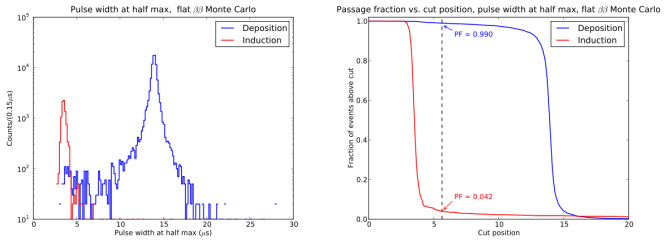

Fig. 1.

IV. Plan of Attack

Our approach to classify deposition and induction signals involves taking full advantage of the 14 calculated attributes and working with a multitude of classifier algorithms. We also strive to come up with some methods for validating and comparing performance of different classification models, so we can gain as much knowledge as possible about the LB induction signals without having LB labelings. We quickly saw that supervised learning on the MC data gave the best balanced results on both the cross validation and the hand-labeled testing data and we tried to use a variety of classifiers, each addressing certain expected patterns in the data. While we make a naive assumption that the single induction template from the simulation will suffice to generalize for real-world testing data, we note that the results of testing the classifier on real hand labeled data show great gains over the prior attempts at classification. By the end of this work, we evaluate the performance of our classifiers, as well as compare them to the previously proposed discriminator described above.

V. Supervised Learning

A. Logistic Regression

The first method we explored was regularized logistic regression (sklearn.lm.LogisticRegression()). We have a few model parameters we are free to adjust, such as regularization type ($l1$ or $l2$) and regularization strength; we also chose to increase the strictness of convergence testing from the default value, which we found to improve model predictions. For a given model configuration, we set up a 5-fold cross-validation on the MC data to make sure that the model is at least consistent on MC data. For each of the cross-validations, a confusion matrix is reported to check deposition passage and induction blockage. Then the model is trained on the whole MC set and tested on the hand labeled LB data, with a confusion matrix reported. It was found that the default $l2$-regularization with a strength factor of 1.0 actually gave the best performance on the MC data and the hand labeled LB data. When tested on the hand labeled LB set, the model passes 99.69% of deposition signals while correctly classifying 57% of induction signals. This deposition accuracy is comparable to the MC k-fold performances of 99.67%, while the induction accuracy is expectedly worse than the 99.6% on the MC data, since the MC induction signals are all from the same template waveform. Additionally, it was found that feature scaling (centering and scaling) slightly improved accuracy on induction signals, but significantly reduced accuracy on deposition signals, which was unacceptable, so it was omitted from the final logistic regression model.

B. SVM

Linear Kernel: Motivated by the early promising results of logistic regression, we proceeded to apply the SVM classifier (sklearn.svm.LinearSVC() and sklearn.svm.SVC()) to consider the well regarded out-of-the-bag classifier. Tuning the SVM, we note that we are free to adjust the kernel function, the cost parameter, the loss function type (hinge loss or squared hinge loss), as well as the class weights to continuously lower our error rates. In testing the model, we again utilized a 5-fold-cross-validation on the original training data and a testing session on a hand-labelled data set while training on the labelled MC simulation data. To account for the many parameters to tune, we also implemented a custom grid search algorithm over the parameters and repeated running the SVM hundreds of times using varying parameters optimizing on the testing data. We began running the SVM on our regular unscaled features. As with logistic regression, we set a minimum deposition rate threshold to be 99% and the results showed that using the linear kernel provided the most balanced test results as other kernels such as the radial basis kernel tended to over-fit on our training set and result in high deposition pass-through rates but very low induction block rates. Moreover, we found that a combination of a cost parameter around 23, a squared hinge loss in addition to a equally balanced class weight yielded in the best results on the hand-labelled test set of an overall accuracy of 96.3277% with a deposition accuracy of 99.6951% and an induction block of 53.8462%. We investigated further into the theory order to account for the similar passing rates in the linear SVM as well as logistic regression to see if we could improve on the results. We note that the regularized empirical loss minimization appear very similar in both the case of the SVM and logistic regression. As shown by Jaakkola [1] for the SVM we find the weights with respect to minimizing

$$1/m \cdot \sum_{i=1}^{m} (1 - y^{(i)}[w_0 + (x^{(i)})^T \cdot w])^+ + \lambda ||w||^2/2$$

and for logistic regression

$$1/m \cdot \sum_{i=1}^{m} -log(g(w_0 + (x^{(i)})^T \cdot w])) + \lambda ||w||^2/2$$

From the minimization functions, it is clear that the main difference between the two models lie in the use of the kernel trick in the SVM and the varying error functions between the two model. Because we are using the linear kernel for the SVM, the main source of variation between the two models then lies in the definition of the error function which could potentially explain the similar error rates in both of the models. Probing further, if the deposition and induction events of the EXO-200 experiment were best modeled by the logistic regression error function over the SVM error function, it would explain the slight improvement variations we see in the logistic regression model over the SVM. As with logistic regression, scaling the data only served to lower deposition pass rates and therefore was not considered for the linear kernel SVM.

Gaussian Kernel: We then attempted various different kernels for the SVM. It was found that the Gaussian kernel gave the best results and as we had before, we explored further into varying the results by scaling and centering the dataset. The SVM was retuned on top of these features to see if we could gain any increases in performance. We found that this instance, scaling the dataset actually increased our performance and our initial results without accounting for varying class weights resulted in a promising 97.5623% deposition pass through rates and a 73.56% induction block rates. With the desire of increasing deposition to meet our 99% threshold, we then varied our class weights to give a higher weight towards deposition (60%) and found that very interestingly, unlike logistic regression, after retuning the SVM with feature scaling and varying the kernel functions, we resulted in better performance than the unscaled unweighted model with a final deposition rate of 100% pass through and an induction block rate of 69.2308% on the testing data. On the training data, we received deposition and induction rates of 99.5276% and 77.8499% respectively. Moreover, we noticed that in our tuning there were results with deposition rates near 98% while induction block rates had increased near 85%. By running the Gaussian SVM on the scaled feature set, we received very promising results, but we also noted that we could not increase induction beyond 70% while keeping our threshold even by altering the class weights. As a result, we gained a new optimistic perspective to selecting for future different classifiers that would inherently bias themselves towards weighting deposition more significantly with the belief that we would meet our deposition pass through rates from the classifier and be able to tune the classifier beyond that to increase the induction block rates. Keeping this in mind, we researched and began experimenting with two such classifiers: the one-class SVM and the random forest classifier.

C. One-Class SVM

One-class SVM as a method for outlier detection was appealing since it captured the spirit of ultimately wanting a classifier robust enough to distinguish between deposition and any not-deposition signals. Being able to tightly confine the region of existence for deposition signals with a one-class SVM is an unlikely accomplishment, but perhaps the decision boundary of the one-class SVM could better exploit the geometry of our data. The OneClassSVM class was used to train the model with a gaussian kernel, which was easily found to be the best kernel for our task. The classifier was optimized over choices of convergence tolerance, kernel degree, and a νparameter, which defines an upper bound on the number of training errors and a lower bound on the fraction of support vectors. The final model was trained on the entire MC set, and this time, utilized feature scaling to get better results. While this classifier consistently achieved over 99% deposition passage on MC data, it only achieved 97.56% deposition passage on the hand-labeled set, which excludes it from being considered for final adoption. However, we can learn about the geometry of our data in parameter space by comparing the performance of this model with the performance of linear classifiers. The one-class SVM achieved lesser deposition passage on the MC data and hand-labeled than the above two classifiers, which tells us that the deposition signals do not lie with a well-restricted region in parameter space, and that a hyperplane defining a halfspace does better than a finite volume in space to capture the amount of deposition that we require of ourselves. It is also of interest to compare induction blockage among these models. The numbers can be found above for the previous models, but for one-class SVM, we had 57% blockage on the MC data and 73% blockage on the hand-labelled data. This much worse performance on the MC induction signals is likely due to the fact that it is the only model that didn't include any induction signals in its training.

D. Decision Tree and Random Forest

Decision trees (sklearn.tree.DecisionTreeClassifier) were a promising classifier model because of two characteristics. First, we hypothesized that given the widespread use of decision tree boundaries among Physics research in general, it could very well be that a rule-based cutting system could perform an accurate distinction between induction and deposition. In fact, the current filtering system as implemented in the EXO-200 experiment involves using three cuts on the attribute values which create rectangular decision boundaries in an attempt to filter out the induction. Decision trees come closest of our classification models to performing such logical boundaries and thus had from the onset had potential to perform better than the current system as it would be a more rigorous cutting process rather than three speculative cuts. Second, decision trees fit our requirement of biasing themselves towards deposition classes as they have a tendency to inherently biasing towards the class with more training examples. We knew that we had more deposition examples than induction examples and as we saw in the SVM, if we could maintain a high deposition pass through rate, we could scale the features to increase our induction block. We decided to take this attribute of the decision tree and convert it to our advantage. Understanding these two characteristics, we began our process of applying the model by working from the decision tree classifier to gain a preliminary insight into the performance of the model then began varying the decision boundary geometrics by including a majority ruling random forests of decision trees. First, we discuss the result of the single decision tree.

Single Decision Tree: In order to assure ourselves that we weren't actually impairing the results of the decision tree by running it on the deposition heavy data set, we ran a sanity check by using a filtered balanced example set and comparing the results with the original unscaled feature set. We saw that the equal distribution tuned decision tree resulted in a deposition pass through rate of 98.4756% and an induction block rate of 53.8462%. The decision tree trained from the entire data saw a slight increase in induction of 57.7923% and a decrease in deposition of 98.1707%. From these preliminary results, we saw that a single tree itself would not give the performance as desired. However, having a rule based cutting system appealed to us as a logical way of understanding the

data further. As a result of running the single tree, we were at least able to see a visible rule-based system (subtree shown in Figure 4) to logically and easily intuit how the system was classifying between the two classes. Inspired by that there was potential in such a cutting system and after reading more about the gains in ensemble methods, we decided to apply random forest as an ensemble method of the decision tree.

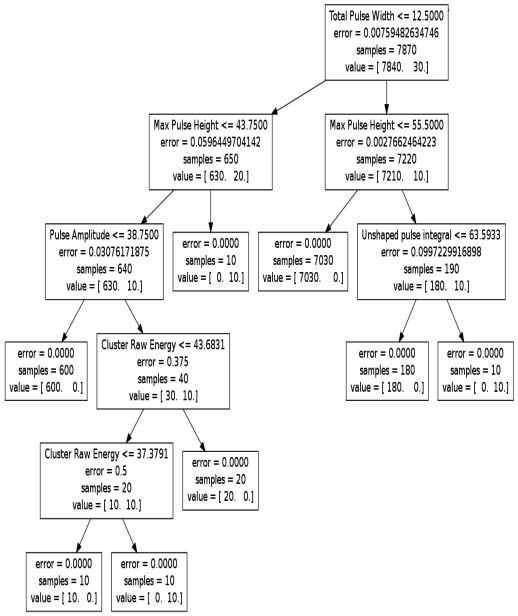

Fig. 2. Subtree of the Decision Tree

Random Forest Ensemble Method: The random forest model gave great gains in performance from the other classifiers. Interestingly, however, we saw that scaling the feature set did not have any different results on the final tuned forest. This may be that the loss function used to compute the boundaries is unaffected by the affine transformation of the data. However, regardless, we saw a great increase in performance as the final results on the hand-labeled testing set resulted in 99.3902% deposition pass rates and an 80.7692% induction block rate. On the training data, both deposition and induction rates were 100%. Even though the results of the random forest took a slight hit in deposition as opposed to the SVM, we note that the random forest gives the most balanced results among all of our classifiers. There is a slight room for error as the hand labelled testing set is not 100% representative of all of the events that occur in the real world, but given the high rates on the training data as well as the high rates on the hand labelled test set we note that the random forest performed the most admirably of the classifiers. The high performance of the classifier could be attributed to the boolean logic of the system; since the data is based on 14 real world attributes, it could very well be that a rule-based classifier best models the difference in the inherent characteristics between deposition and induction.

VI. Unsupervised Learning

Our work with supervised-learning-based classifiers above has indeed been fruitful, but we felt that we wanted to briefly explore some unsupervised learning methods. For our purposes, unsupervised learning has the advantage of directly dealing with the LB data and not the MC data. We expect that unsupervised learning could help us understand that geometry of the LB data that we care about, and we decided that clustering would be the method-of-choice.

A. K-Means Clustering

K-Means (sklearn.clustering.kmeans()) clustering was run to find two clusters, hopefully aligning with the deposition/induction distinction in the data. Training a k = 2 clustering model on the LB data set, and then predicting classes for the LB set, it first found that all but one point is classified in a single group. So we adopted a process of training the model, predicting classes, inspecting the confusion matrices on the MC and hand-labeled sets, and then creating a new set for the model to train against composed of just the data points in the super-dominant class, and repeat until there is a reasonable split of signals between the two classes. After 4 iterations of this process, we arrived at a clustering model that achieved 99.39% deposition passage on the MC set and 97.87% deposition passage on the hand-labeled set. Oddly enough, its induction blockage on the MC set was significantly worse than on the hand-labeled set, 8.5% and 30.7% respectively. This is odd because the supervised-learning models always had a very easy time blocking MC induction signals when k-fold cross-validation was used, and it was presumed to be because the MC induction signals are all so similar and probably easily confined in parameter space, so when those classifiers moved onto the LB set where inductions signals are more varied, they were expected to block induction with worse performance. This interesting pattern in the clustering model's induction blockage implies that there is a significant variation in the LB induction signals, and so the clustering ended up finding centroids that actually preferred blocking types of induction signals more often seen in the LB data than the MC data, which other models could not do because of how they were trained.

VII. Validation and Comparison

The first sanity check we actually did was look at the coefficient vector learnt by logistic regression to see which attributes it put the most weight on. Taking the coefficient vector from our classifier and multiplying each entry by the mean value of that attribute in the unscaled testing data, we found that the most importance was placed on the chi-2 fit to the template induction signal, and intermediate importance was given to UWire pulse energy and pulse width, all of which we expected from physical understanding to be strongly characteristic for distinguishing our two classes of signals. We also saw that no weight was placed on channel number, as we would also expect, because induction happens on all wire channels. Other models may have slightly different weights, but this shows that what our classifiers are learning is consistent with physical expectations.

As we have been already mentioning, for each model we get an estimate of the generalization error by testing on the

hand-labeled set. This serves as a check for satisfying the necessary criterion on deposition passage. Also recall that the MC deposition signals are good models of depositions seen in LB data, so they too give good verification of classifier performance on deposition signals. Another sanity check that we do is run the classifier on the LB data set and collect the waveforms of the signals classified as induction, which we then randomly trace through and view to make sure that they all look like signals that we indeed wish to exclude. The acceptable candidate models are defined as having $> 99\%$ deposition passage on both the MC and hand-labelled data sets, while also passing the sanity check, and those models are Logistic Regression, Linear SVM, Gaussian SVM, and Random Forests.

Now for directly comparing the acceptable models, along with the previously proposed cuts-model. Here's a table giving the deposition passages and induction blockages for the 4 classifiers developed by this work and the previously proposed discriminator.

	Previous Discriminator	Logistic Regression	Linear SVM	Gaussian SVM	Random Forest
Deposition Passage (MC)	0.998610597	0.996943312	0.995739163	0.995276	1
Deposition Passage (HL)	0.99695122	0.99695122	0.993902439	1	0.993902439
Induction Blockage (MC)	0.52958153	0.994949495	0.956709957	0.778499	1
Induction Blockage (HL)	0.307623077	0.576923077		0.692308	0.807692308

Fig. 3. Deposition Passages and Induction Blockages for 4 new classifiers and one previously proposed classifier. Tested on MC data set and hand-labeled (HL) data set.

As can be seen from the table, we now have several competitive and appealing alternative classifiers for separating deposition and induction signals. Particularly, our Random Forest classifier is the most appealing, with perfect separation on the training data, and very consistent generalization error. The Gaussian SVM is also impressively close to the performance of the Random Forest. All classifiers proposed by this work show significant improvement in induction blockage over the previously proposed discriminator.

On top of our sanity check on the induction-classified signals in the LB data, we were interested in examining the similarity or overlap of the sets of signals that each classifier called induction. This sort of similarity measure describes the extent of agreement on which signals are induction in the LB data between each pair of classifiers, and can also be thought of as a measure of similarity between the classifier geometries. In the following table, each entry is defined as $\frac{|S_i \cap S_j|}{|S_i|}$, where i, j represent row, column indices and the indices range from 1 to 5 for the 5 classifiers in the above table, and S_i is the set of signal ID's that are called induction by classifier i, so that each entry is the fraction of signals that classifiers i and j both call induction out of the total number of signals that classifier i calls induction.

	Previous Discriminator	Logistic Regression	Linear SVM	Gaussian SVM	Random Forest
Previous Discriminator	1	0.09012406	0.11482547	0.09296527	0.12297678
Logistic Regression	0.56699862	1	0.46687821	0.81277809	0.54623253
Linear SVM	0.75589021	0.48852054	1	0.44325679	0.68469136
Gaussian SVM	0.58869727	0.81809173	0.4263892	1	0.53169297
Random Forest	0.79232451	0.55939205	0.67012275	0.54096558	1

Fig. 4. Similarity matrix for the classifiers

The hope of this table was to perhaps see that one classifier's set of induction signals was a subset of another classifier's set of induction signals, implying that we can more absolutely claim that the second classifier is better than the first since the second classifier agrees with and expands on the first classifier's definition of induction and both have at least 99% deposition passage. We do not end up seeing that, and that is likely because the geometry of the data is so complex in the 14-dimensional parameter space that the subspaces that the classifiers call induction are quite varied. Regardless of not seeing full superset-subset encapsulation, we do see reasonable agreement percentages among the classifiers and we acknowledge that we won't get consistently perfect induction subspace definitions.

VIII. Conclusion

We have successfully generated several competitive and appealing models for classifying induction signals in the EXO-200 detector. Our criterion-passing models show significant improvement in induction blockage over the previously proposed discriminator, at a factor of about 2. We have taken several measures to validate the generalization error of our models, and have taken steps to help us understand the geometry of the data and the differences in results between our models. Our Random Forest classifier will succeed the previously proposed discriminator and will be included in the Reconstruction module that will remove induction signals from our Low-Background data. The EXO-200 experiment hopes to complete another analysis and release another paper in the coming year, and we hope to see the improvements in Reconstruction efficiency and Single-Site/Multi-Site discrimination in this next analysis due to this addition.

Acknowledgment

While both authors contributed to all of the above classifiers and discussion, Jason Chaves was the primary contributor to the data formatting, the EXO connections, logistic regression, one-class svm, K-Means, and Validation and Kevin Shin was the primary contributor of creating various data subsets, linear SVM, Gaussian SVM, Decision Trees, Random Forests, and the gridsearch parameter optimization implementation. The authors would like to thank Dave Moore, a PostDoc working on the EXO-200 experiment, who graciously helped us collect the Monte Carlo and Low Background data with calculated features. We would also like to thank Andrew Maas for helpful suggestions along the way, and the CS 229 staff for this course.

References

[1] Jaakkola, Tommi. "Machine Learning: Lecture 7." *AI.MIT.edu*. MIT.
[2] "Sci Kit Learn." Classifiers. SciKit Learn. Web. 14 Dec 2012. *http://scikit-learn.org/stable/*.
[3] EXO Collaboration
[4] EXO Reconstruction Workshop
[5] EXO Week Conference