Energy Measurement in EXO-200 using Boosted Regression Trees

Mike Jewell , Alex Rider

June 6, 2016

1 Introduction

The EXO-200 experiment uses a Liquid Xenon (LXe) time projection chamber (TPC) to search for neutrinoless-double beta decay($0\nu\beta\beta$), an extremely rare hypothetical decay that would indicate the Majorana nature of neutrinos.[1, 2, 3] The EXO-200 experiment has been taking data for over 2 years and has published one of the most sensitive limits on $0\nu\beta\beta$ half-life.

Events deposit energy in the LXe through both scintillation light (175nm) and free ionization charge. The scintillation light is detected at either end of the EXO-200 detector by large area avalanche photodiodes (APDs). The ionized charge is drifted along the z-axis of the detector where it first passes a shielding/induction wire grid (V-wires) and is than collected by a second wire grid of collection wires (U-wires). Each wire is 3mm in pitch and wires are ganged into groups of three before being readout and saved. The total charge energy of an event is then calculated by determining the sum amplitude of all channels which collected charge.

In order to accurately reconstruct the energy of the waveform signals have to be accurately identified as either collection or induction signals. This is currently done by performing a χ^2 fit to both a collection and induction signal and than classifying the waveform based on the ratio of these scores. Waveforms classified as induction are flagged and then not included into the sum when determining event energy. This current technique achieves reasonable efficiency at identifying collection and induction signals but energy deposits spanning multiple channels present a slight challenge because waveforms will contain both collection and induction signals. In addition, the waveforms are shaped before being saved to disk making identification and energy estimation somewhat more complicated. In this study an alternative technique for reconstruction of event energy in EXO-200 using Boosted Regression Trees from sklearn is explored.[4]

2 Simulation and Data Compression

2.1 Monte Carlo

The EXO-200 Monte Carlo has been described in detail elsewhere [1]. For waveform generation energy deposits in the detector are used to simulate waveforms on each of the U-wire channels using the Shockley-Ramo theorem to calculate induced signal. To simplify this analysis Monte Carlo energy deposits were not generated with GEANT4 as in the standard analysis. Instead events with uniformly distributed energy and position were simulated in the detector. For each deposit the waveform from the collection channel is saved and tagged as collection while the waveform from the two neighboring channels are tagged as induction and saved. For each event the true energy of the deposit was also saved to be used as the target when training and testing.

Each waveform in the detector consists of 2048 samples taken at a sampling rate of 1MHz with a fixed trigger time at 1024. In addition, standard Monte Carlo includes added noise, sampled from

real data into the waveforms. In order to simplify the first stage of the analysis no noise was added. The second stage of this analysis included added noise into the waveforms. This was done by using a database of noise waveforms seen in real data and adding these into the generated waveforms at random an example is shown in Figure 5b.

2.2 Template Generation

Initially a template waveform for both induction and collection was created by simulating an event in the exact center of the detector. The resulting collection and the average of the induction signals from this event was than used as a template to represent the typical collection and induction waveform. The results of these templates in Time Space are shown in Figure 1.

Although most induction and collection signals have roughly the same shape, there are some differences due to the exact position and energy of the initial deposit. Using these two templates offered a reasonable projection into collection and induction space but this did not fully capture the detailed shaped information. For the final analysis a more sophisticated template generating algorithm was implemented. This involved using a K-means clustering algorithm to find a larger subset of template waveforms that better samples the waveform space.

Results for template generation using the sklearn K-means clustering algorithm with 15 clusters are shown in Figure 3. Waveforms are created using the same Monte Carlo technique. Events with varying energy are uniformly generated within the detector volume and a set of 20k WFs was generated. Each waveform is than normalized by the maximum amplitude and time shifted so the peaks all occur at the same time. This was necessary so that the clusters represented different shapes as opposed to different time off sets of the same shapes. The time offset is handled in the optimal filter stage. In addition, a windowing from sample 900 to 1250 was implemented. This was required because the clustering algorithm seemed to fail when the large pre/post trigger segments with 0 signal were included as this is identical for all pulses. The algorithm succeeds in finding both collection and induction like clusters. Typical templates are shown in Figure 3.

2.3 Optimal Filter

The goal of this analysis is to learn a mapping from the space of digitized waveforms to the amount of charge deposited on a particular channel. Early work showed that learning in the full space of 2048 length vectors that comprise the raw waveforms was impractical. We devised a method for compressing some of the information stored in each pulse into many fewer than 2048 real parameters using optimal filters.

Optimal filters take as inputs a noisy time series $v(t)$ and a template $s(t)$ and return the amplitude of s in v. Under certain assumptions it can be shown that an optimal filter is the best estimator of of the amplitude of s in v. The optimal filtering we employed is equivalent to LMS fitting of the template to the signal in the frequency domain. The χ^2 (Cost function) for a particular amplitude, A, is defined as follows

$$\chi^2 = \int df \frac{|\tilde{v}(f) - A\tilde{s}(f)|^2}{J(f)} \tag{1}$$

Where $\tilde{v}(f)$, and $\tilde{s}(f)$ are the Fourier transforms of the signal and the template, respectively. $J(f)$ is the power spectral density (PSD) which is used as the weight for each frequency in the cost function. In practice, the waveforms are always sampled discretely, so that discrete Fourier transforms are implemented using FFTs and the integral in equation 1 is carried out as a sum. For waveforms generated without added noise, a uniform PSD was used but for waveforms with added noise, the average PSD was estimated using real data.

As previously stated, we initially started by compressing waveforms into 2 component vectors where the first component is the amplitude of the optimal filter with a typical collection pulse shape used as a template and the second component is the amplitude of the optimal filter with a typical induction pulse used as a template. The results of doing this for a sample of 10000 Monte Carlo Pulses is shown in Figure 2 Later analysis increased the parameter space to a 15 component vector where each component represented the amplitude of the optimal filter with each of the 15 K-means templates.

3 Energy Estimation

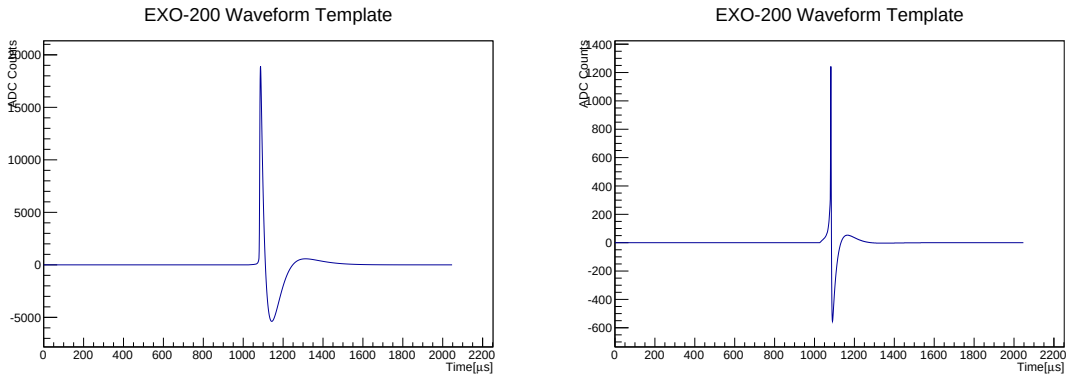
The compressed pulse data with 2 components shown in Figure 2 was used to train a boosted regression tree implemented in Python's Scikit Lear Library to predict the energy in a sample of test pulses reserved from the training set. The boosted regression tree learned the mapping from the representation of the pulses in $\mathbb{R}^2$ to the energy of the pulse by boosting 500 trees of depth 30 together. The boosted tree was trained using 10k noiseless MC WFs including both collection and induction signals. An additional 2781 WFs were used as a test set to determine the test error. A plot of the predicted energy versus the true energy for the pulses in the test set are shown in figure 4a. The average error in predicted energy for collection pulses is 13% Along the y-axis it is possible to see the 7 out of 1238 induction pulses that our model erroneously gave energies to. There are also 3 out of 1543 collection pulses that our model erroneously assigned 0 energy. The scatter in predicted energy for collection pulses is likely due to template mismatch and should be improved by expanding the space of templates.

In addition this same procedure was repeated for this same testing and training set using the 15 template WFs generated in clustering instead of the 2 typical templates input by hand. The results of this method is shown in Figure 4b here no induction signals were assigned non-zero energy (>10keV) and the energy reconstruction was 0.1%. This is a huge improvement over the original 2 template technique.

Finally to test the generalization of this method to MC with added noise we repeated the same procedure using the 15 template WFs and 90k training WFs generated with noise and reported the error on a test set including 10k WFs with added noise. The results are shown in Figure 5. This resulted in an error of 17% in the energy estimation of collection signals and many induction signals (80%) had energy $> 10keV$. Given the amplitude of the noise, the intrinsic energy resolution should be $\sim 5\%$. The drastic decrease in performance with the presence of noise indicates that we are likely over fitting. Future work will focus on determining the optimal depth for the regression trees and the optimal number of regression trees to boost together. Furthermore, simply using a larger training set may improve the performance of this analysis. We will also experiment with weak learners different from threshold functions.

4 Conclusions

Using the Boosted Regression Tree algorithm from sklearn an initial Energy Reconstruction algorithm has been implemented to predict the energy associated with U-wire Collection Signals in the EXO-200 detector. The current algorithm uses a set of n=15 templates to represent pulses as vectors in $\mathbb{R}^n$ using an optimal filter. A boosted regression tree learns the mapping between these vectors and the energy of the pulses. This resulted in 0.1% error in the energy estimate for WFs with no added noise but 17% error for WFs with realistic noise. Future work by the EXO-Collaboration to improve this analysis will focus on understanding the large error of this estimation.



(a) Collection Signal.

(b) Induction Signal.

Figure 1: Templates of Induction and collection signals used in the optimal filter in the time domain. Signals represent pure collection and induction pulses.

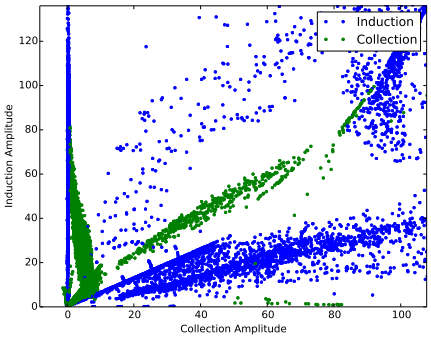


Figure 2: Projection of waveforms into collection and induction space using the 2 templates generated with a charge deposit in the detector center.

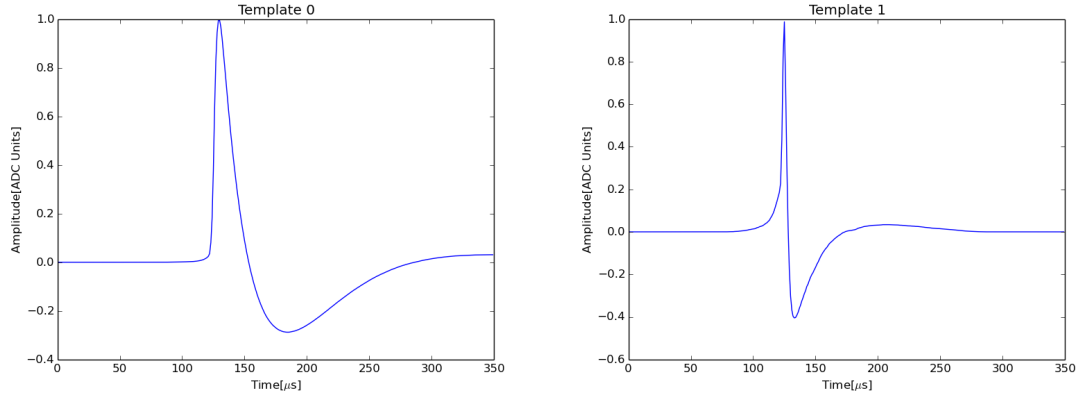
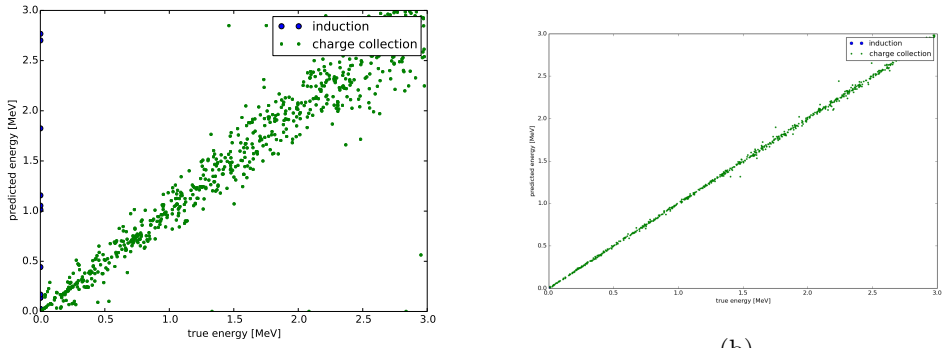
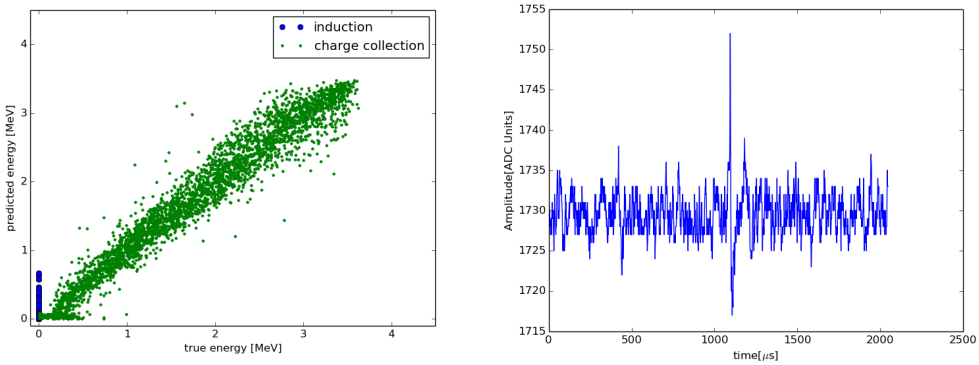


Figure 3: Templates generated using K-means clustering with 10 clusters. Appears to find both collection like and induction like signals.



(a) (b)

Figure 4: Energy predicted by regression tree versus true energy for a sample of test pulses using 2 template method (13% error).4a. Predicted energy using 15 templates generated with K-means Clustering (0.1% error).



(a) (b)

Figure 5: WFs with noise Figure 5b and the Energy prediction for these WFs Figure 5a. Observed 17% error on 10k WF test set in measurement of energy for collection signals. In addition many of the induction signals were assigned large energies.

References

[1] J. B. Albert et al. Improved measurement of the $2\nu\beta\beta$ half-life of ^{136}Xe with the EXO-200 detector. *Phys. Rev.*, C89(1):015502, 2014.

[2] J. B. Albert et al. Search for Majorana neutrinos with the first two years of EXO-200 data. *Nature*, 510:229, 2014.

[3] M. Auger et al. The EXO-200 detector, part I: Detector design and construction. *JINST*, 7:P05010, 2012.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.