LANL Earthquake Prediction

Abhishek Rai Sharma, Sai Santosh Kumar Ganti, Mohit, Vijay Karigowdara
Department of Computer Science
Boston University
abhirs@bu.edu, gantis@bu.edu, beniwal@bu.edu, vijaykn@bu.edu

Abstract

Earthquakes have been a major cause for mass destruction. Although numerous earthquakes occur throughout the year, some of them are so destructive, they cause irrevocable damage to lives and resources. Researchers have been trying to solve the puzzle of Earthquake Prediction for years. In this project, given the segment of the acoustic data (seismic signal), we are building models to predict the time before the next laboratory earthquake.

1 Introduction

Earthquake prediction focuses on three major aspects of the earthquakes: when, where and magnitude. We are focusing on when the event will occur, the remaining time before the next laboratory earthquake.

Currently, scientists are trying to predict earthquakes using a recurrence interval for earthquakes that repeat periodically. However, as earthquake recurrence is not fixed for a given fault, prediction using this method results in large error bounds. Therefore, to better predict the earthquakes LANL (Los Alamos National Laboratory) is generating data set in the laboratory by simulating earthquakes. LANL team used steel blocks that were put together and applied stimulated pressure that resulted in shear stress. The recorded sounds (seismic signals) are then analysed for earthquake prediction.

Upon successful prediction, the laboratory data can be scaled to real-world data and researchers can provide an early warning system that could save numerous lives and infrastructures.

2 Signal Processing

	acoustic_data	time_to_failure
0	12	1.46909998
1	6	1.46909998
2	8	1.46909998
3	5	1.46909998
4	8	1.46909998

Figure 1: Data Apperance

The data are from an experiment conducted on a rock in a double direct shear geometry subjected to bi-axial loading, a classic laboratory earthquake model (fig. 2)

Two fault gouge layers are sheared simultaneously while subjected to a constant normal load and a prescribed shear velocity. The laboratory faults fail in repetitive cycles of stick and slip that is meant to mimic the cycle of loading and failure on tectonic faults. While the experiment is considerably simpler than a fault in Earth, it shares many physical characteristics. (fig. 2)

Los Alamos' initial work showed that the prediction of laboratory earthquakes from continuous seismic data is possible in the case of quasi-periodic laboratory seismic cycles. In this competition, the team has provided a much more challenging dataset with considerably more aperiodic earthquake failures.

The given data consists of a signal and time to failure. We had to go through signal processing concepts and apply transformations to generate features. Signal processing is of great significance in both scientific researches and engineering applications. A major purpose of signal processing is to extract physically meaningful information from a signal as much as possible.

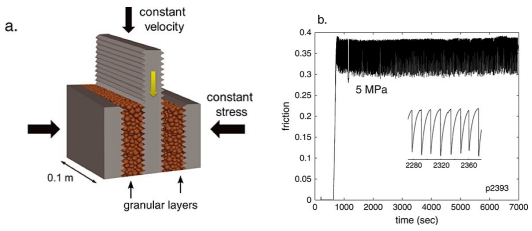


Figure 2: The seismic signal recorded in the laboratory

2.1 Fourier Transform

Since we are dealing with waves, we thought that our data could be better represented in the frequency domain, rather than using the raw or scaled data which are in the time domain. We separately transformed the data in each axis from the time domain to the frequency domain using the one dimensional Fourier transform. The Fourier transform is commonly used to convert a signal in the time spectrum to a frequency spectrum.

Fourier analysis is a method for expressing a function as a sum of periodic components, and for recovering the signal from those components. The Fourier transform generalized the Fourier coefficients of a signal over time. The Fourier coefficients are the measures of the signal amplitude as a function of frequency, here the time information will be totally lost.

The Fourier transform is commonly used to convert a signal in the time spectrum to a frequency spectrum. Examples of time spectra are sound waves, electricity, mechanical vibrations etc. This method makes use of the fact that every non-linear function can be represented as a sum of (infinite) sine waves.A Fourier Transform will break apart a time signal and will return information about the frequency of all sine waves needed to simulate that time signal. For sequences of evenly spaced values, the Discrete Fourier Transform (DFT) is defined as:

$$X_k = \sum_{n=0}^{N-1} x_n . e^{-i 2\pi \frac{k}{N} n}$$

Here,

- N = number of samples
- n = current sample
- x_n = value of the signal at time n
- k = current frequency (0 Hz to N-1 Hz)
- X_k = Result of the DFT (amplitude and phase)

The Fourier transform takes in n real numbers and returns complex numbers. The magnitude 2 n + 1 of these complex numbers became our new features. Since earthquakes are primarily composed of a Primary wave, a Secondary wave, and Surface wave, we could also use independent component analysis to separate the three wave sources recorded on the three axes of acceleration and define features of the SVM based on the separated waves.

2.2 Hilbert Huang Transform

The Hilbert Huang transform (HHT) is a time series analysis technique that is designed to handle nonlinear and non-stationary time series data.

In the case of earthquake prediction, the applications have shown that HHT can give much finer energy–time–frequency distribution properties and extract more valuable nonlinear and non-stationary features than the traditional signal processing methods

The application of the improved HHT method in dynamic feature extraction for vibration based damage detection was expected as a promising area for future research in earthquake engineering.

The analytic signal $x_a(t)$ of signal $x(t)$ is:

$$x_a = F^{-1}(F(x)2U) = x + iy$$

Where,

- F is the Fourier transform
- U is the step function
- y is the Hilbert transform of x

In other words, the negative half of the frequency spectrum is zeroed out, turning the real-valued signal into a complex signal.

We have used these signal processing techniques to experiment and create features to better predict the results.

3 Data Analysis

The training data is a single, continuous segment of experimental data having 'acoustic_data' (input signal) and 'time_to_failure' in other words the time remaining before the next laboratory earthquake. Whereas, the test data consists of a folder containing many small segments. The data within each test file is continuous acoustic_data, here the test files do not represent a continuous segment of the experiment; thus, the predictions cannot be assumed to follow the same regular pattern seen in the training file. We have to predict a single time_to_failure corresponding to each seg_id in the test folder.

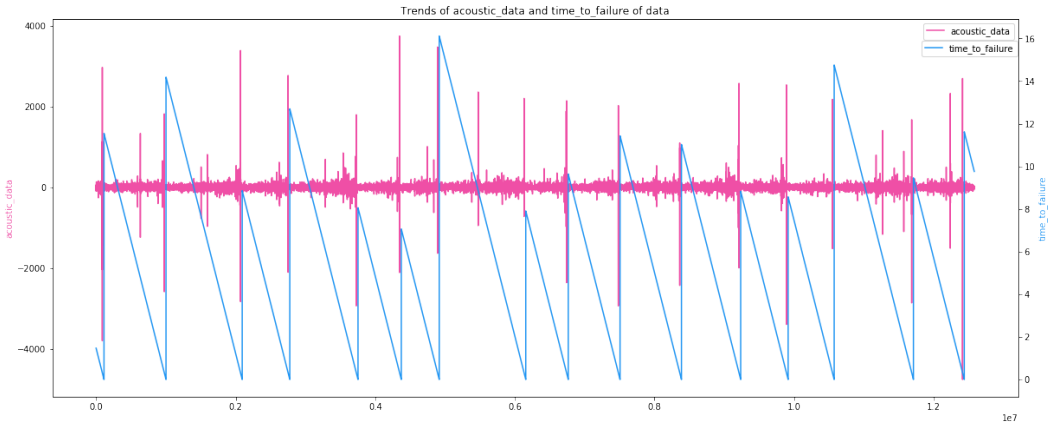
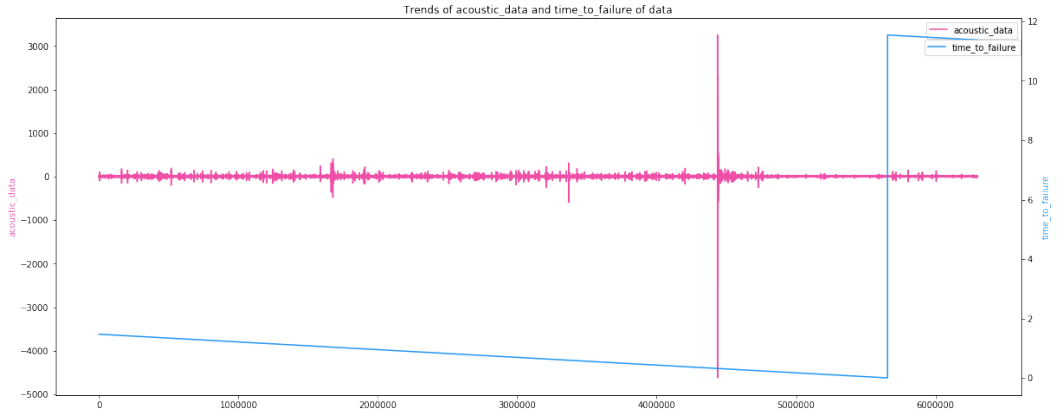


Figure 3: Trend of 2% of the acoustic data

The plot shows only 2% of the full data. We observed that acoustic data exhibits complex oscillations with variable amplitude. Moreover, we found an increase in the amplitude of the acoustic data just before each failure. These large amplitudes are also obtained at different moments in time.



We analysed the data set and observed huge fluctuations just before the failure and we saw a cyclical pattern. Moreover, we can see the trains of intense oscillations preceding the large signals and also some oscillations with smaller peaks after the large one. Then the failure occurs after some minor oscillations.

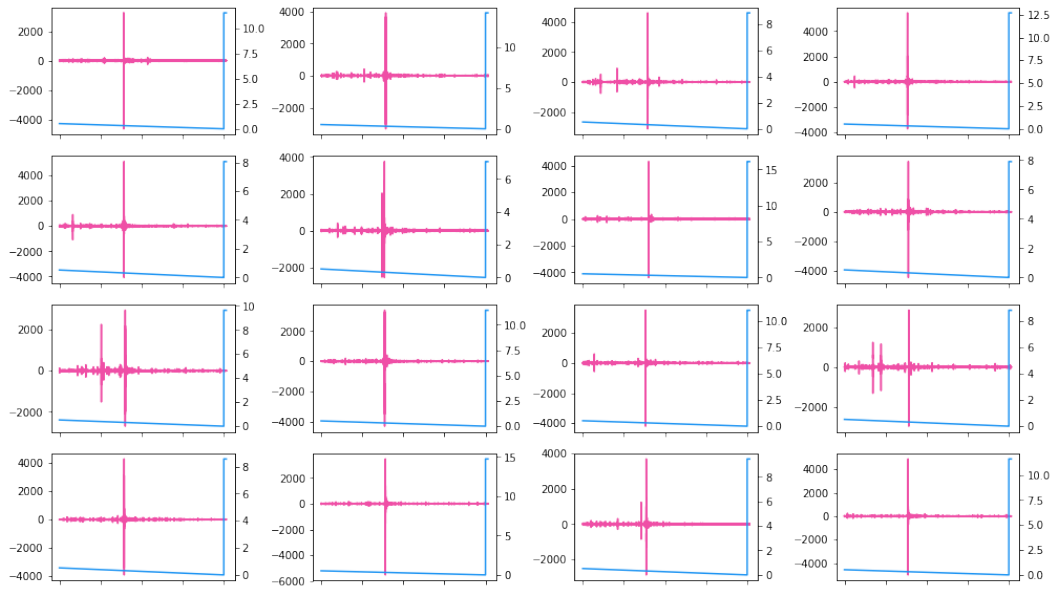


Figure 4: Close trend of of the acoustic data.

Additionally, after we expanded the time to failure plots and found a remarkably consistent big spike in the signal before failure. The main concern was that the 150,000 sample slices are very short compared to the overall time between quakes. Therefore, we can't be sure about the presence of meaningful signal spikes in many training or test samples. It is also possible that the analysis of these spikes could add a predictive capability to any small number of test samples.

4 Feature Engineering

Feature Engineering is the process of using domain knowledge of the data to create features that work well with the machine learning algorithms, usually, these involve statistical features that help us analyze trends and find correlations of these generated features that help us with predictions. Feature engineering is fundamental to applications of machine learning and comes at great difficulty and cost. These days we obviate manual feature generation with automated feature learning. Though an informal topic, it is highly essential in applied machine learning.

Since the features in the data are very important to the predictive models and will most likely influence the results. The quality and quantity of the features will have a great influence on whether the model is good or bad. In essence, the better the features are, the better the result will be. Though it's not so black and white because results achieved also depend on the model and the data provided, that said, choosing the right features is still a paramount factor for better results and better features often produce simpler and flexible models and would likely yield better results.

Below was our process for feature engineering:

1. Brainstorming or Testing features.
2. Deciding what features to create.
3. Creating features.
4. Checking how the features work with our model.
5. Improving your features if needed.
6. Go back to brainstorming/creating more features until the work is done.

Based on the feature it could be highly relevant or weakly relevant. It is important to create a lot of features even if some end up being irrelevant because they can be discarded later if not required. And this can also be used later in order to prevent overfitting.

One of the problems that can occur with feature generation is Feature explosion. It can be caused by feature combination or feature templates, both leading to a quick growth in the total number of features.

- Feature templates - implementing features templates instead of coding new features
- Feature combinations - combinations that cannot be represented by the linear system

There are a few solutions to help stop feature explosion such as: regularization, kernel method, feature selection.

Since the training data contains only one column having seismic signal, we have generated several features from this data so that we can better analyze the data and find patterns that will lead us to predict the time before next laboratory earthquake. This is one of the most important tasks in this project. Various statistical features are generated along with below are the different groups of features that are generated.

4.1 Statistical Features

We generated the following statistical features

- Aggregations:
  - mean
  - median
  - standard deviation
  - minimum
  - maximum
  - absolute minimum and maximum
  - exponentially weighted moving average
- Quantiles (95, 97, 99, 1, 3, 5)
- Rolling window features - We generated rolling features with the windows 10,50,100,200,1000 and with the same statistical features mentioned above.
- Trend features - We used linear regression to generate trend feature.
- Aforementioned aggregations for first and last 10000 and 50000 values.
- Average difference between the consecutive values in absolute and percent values.
- Max value to min value and their differences also count of values greater than some arbitrary threshold.

- Inter-Quartile Range
- Generating statistical features after applying signal processing techniques like Fast-Fourier Transform, Hilbert Transform.

5 Models

Below are the models we used built for our prediction. We use the same features that we extracted above and use these features in our model. We then compared the predicted values with the actual values provided by Kaggle and calculated a mean absolute error, which is the metric we are using to evaluate our model.

5.1 Light GBM

Light GBM is a gradient boosting framework that uses tree based learning algorithm. How this algorithm differs from a tree based algorithm is that while other algorithms grow trees horizontally Light GBM grows trees vertically meaning ti grows tree leaf-wise while other algorithms grow level-wise. It chooses the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than level wise algorithm.

Below diagrams explain the LightGBM algorithm compared to other boosting algorithms.

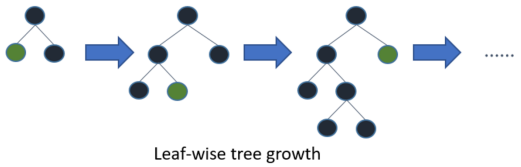


Figure 5: Light GBM

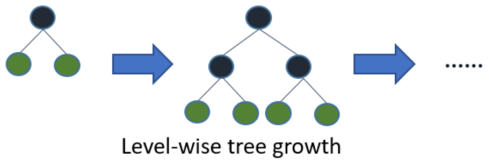


Figure 6: Other Boosting Algorithms

Light GBM can handle the large size of data and takes lower memory to run. Another reason why Light GBM is popular is that it focuses on accuracy of results. LGBM also supports GPU learning and thus data scientists are widely using LGBM for data science application development.

Light GBM is sensitive to overfitting and can easily overfit small data.

5.2 Extreme Gradient Boosting

XGBoost (Extreme Gradient Boosting) belongs to a family of boosting algorithms and uses the gradient boosting (GBM) framework at its core. It is an optimized distributed gradient boosting library. XGBoost is one of the implementations of Gradient Boosting concept, but what makes XGBoost unique is that it uses "a more regularized model formalization to control over-fitting, which gives it better performance," according to the author of the algorithm, Tianqi Chen. Therefore, it helps to reduce overfitting.

It is an implementation of gradient boosting machines created by Tianqi Chen, now with contributions from many developers. It belongs to a broader collection of tools under the umbrella of the Distributed

Machine Learning Community or DMLC who are also the creators of the popular mxnet deep learning library. Tianqi Chen provides a brief and interesting back story on the creation of XGBoost in the post Story and Lessons Behind the Evolution of XGBoost. The implementation of the algorithm was engineered for efficiency of compute time and memory resources. A design goal was to make the best use of available resources to train the model. Some key algorithm implementation features include:

- Sparse Aware implementation with automatic handling of missing data values.
- Block Structure to support the parallelization of tree construction.
- Continued Training so that you can further boost an already fitted model on new data.

6 Experiments and Future work

Firstly we generated the basic statistics without features relating to signal processing techniques. With these features and training the model (Light GBM and Extreme Gradient Boost) combined we got the mean absolute error as 1.530.

Since we are dealing with signals, we wanted to generate more features and improve our accuracy and improve our mean absolute error. Hence, we applied signal processing techniques upon our data to generate features like Fast Fourier transform and took the statistical features again after applying these signal processing techniques but here our mean absolute was 1.538 which was a slight increase compared to the previous result. So, we wanted to see the correlation between the features and the target variable, the right statistics which gives us the highest correlation with the output.

Along with Fourier transform we generated features after applying Hilbert transform on the data. We also generated time series features like autocorrelation, percentile, binned entropy, number of peaks in the signal, etc so as to get as much information as possible. In the future we want to apply feature elimination techniques to select the best features and to ignore less relevant features to reduce the training complexity of the model and improve the accuracy.

We will be exploring training the models with different regression methods like CatBoostRegressor, Support Vector Regressor, etc. Apart from experimenting with these models, different feature elimination techniques will be applied like removing features with low variance, univariate feature selection, recursive feature elimination, etc., to detect the best features and train the model.

7 Results

We ended up having the lowest MSE 1.530 via ensembling Light GBM and Extreme Gradient Boosting models.

The result above is generated while we trained our models on the statistical features without signal processing and this the mean absolute error is in the figure above.

As we have discussed above, we now used signal processing features and trained our model on the statistical attributes along with signal processing attributes and realized that it's not helping us with the accuracy of the model. Hence, we ended up having a higher mean absolute error than expected. The generated result is presented by the figure below.

8 Conclusion

In this project, we leveraged our knowledge of Machine Learning to better predict laboratory earthquakes. Although we did get a low mean absolute error, the Kaggle tests the model for only 13% of the data. Therefore, the actual error may vary for our model when testing against 100% of the data. Moreover, there were very few features in the signal to correlate with the time to the next laboratory earthquake.

Code

The repository containing all code for the project can be found at the following Github Repository. https://github.com/santoshganti/MachineLearningFinal

Acknowledgments

We appreciate Prof. Sang (Peter) Chin for his excellent teachings in the class. He taught the concepts of Machine Learning in detail, which helped us complete this project. Moreover, we would like to thank our Teaching Assistant Peilun Dai for his tremendous efforts to help us for understanding the concepts related to Machine Learning.

References

[1] Bertrand Rouet-Leduc1, Claudia Hulbert, Nicholas Lubbers, Kipton Barros, Colin Humphreys, Paul A. Johnson , Machine Learning Predicts Laboratory Earthquakes, https://arxiv.org/pdf/1702.05774.pdf

[2] "Analysis of Earthquake Ground Motions Using an Improved Hilbert–Huang Transform." Soil Dynamics and Earthquake Engineering, Elsevier, 3 July 2007, www.sciencedirect.com/science/article/pii/S0267726107000620.

[3] Chen, T., & Guestrin C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, 785–794, New York, NY, USA.

[4] Demir, N., PhD. (undated). Ensemble Methods: Elegant Techniques to Produce Improved Machine Learning Results. Toptal. Retrieved from: https://www.toptal.com/machine-learning/ensemble-methods-machine-learning

[5] Ke, G., Meng, Q., Findlay, T., Wang, T., Chen, W., Ma, W., … Liu, T. (2017, December). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Guyon, I. & von Luxburg, U. (General Chairs), Thirty-first Conference on Neural Information Processing Systems (NIPS 2017). Long Beach, CA. Retrieved from: http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree

[6] Singhal, G., (2018). Multiprocessing in Python on Windows and Jupyter/Ipython.Making it work. Medium. Retrieved from: https://medium.com/@grvsinghal/speed-up-your-python-code-using-multiprocessing-on-windows-and-jupyter-or-ipython-2714b49d6fac

[7] Jaidev Deshpande, Motivation for Hilbert Spectral Analysis: https://pyhht.readthedocs.io/en/latest/tutorials/hilbert_view_nonlinearity.html

[8] Olivia Grubert and Bridget Vuong, Earthquake Waveform Recognition: http://cs229.stanford.edu/proj2012/GrubertVuong-EarthquakeWaveformPrediction.pdf

[9] Kan Nishida, Introduction to Extreme Gradient Boosting in Exploratory: https://blog.exploratory.io/introduction-to-extreme-gradient-boosting-in-exploratory-7bbec554ac7

[10] Jason Brownlee,A Gentle Introduction to XGBoost for Applied Machine Learning: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/