Earthquake Prediction by Machine Learning Algorithm

Xuanzhang Liu
xzliu@tempel.edu

1. Introduction

Forecasting the occurrence of a future earthquake is one of the fundamental problems in earth science because of its severe consequences. Tremendous effort has been made in pursuing it, with occasional glimmers of hope but ultimately, disappointing results, leading many to conclude that short-term earthquake prediction is at best infeasible and perhaps impossible. However, With machine learning (ML), the earthquake science community has a new suite of tools to apply to this longstanding problem. Meanwhile, applying ML to the prediction problem raises multiple thorny issues, including how to properly validate performance on rare events, what to do with models that seem to have strong predictive value but may not generalize, and how to handle the output of opaque ML methods. Despite these challenges, recent work has shown that progress on some aspects of the prediction problem is possible. For example, ML has revealed that the time remaining before an earthquake in the laboratory and particular types of tectonic earthquakes known as slow slip events can be anticipated from statistical characteristics extracted from seismic data by employing the widely-used decision tree algorithm and its variance. [2, 4, 5]

However, current approaches failed to capture the continuous features of the seismic data, and their ML models are relatively simple, which is suitable for discrete data. Considering the scale of the dataset is large and Recurrent Neural Network has its advantages to learn the continuous pattern from seismic data, we utilize deep learning methods like RNN to mimic the faulting of Earth, and further predict the time remaining until the next earthquake. The experiment results shows that the neural network model can outperform the traditional machine learning models such as Random Forest and Support Vector Machine. And it has a more accurate prediction on the remaining time of the next earthquake.

2. Related Work

As most researches suggest that the raw data is not sufficient to accomplish the prediction task, different strategies of feature engineering are proposed. Fabio et al. [1] implement some statistic features including mean, standard deviation, max/min, kurtosis, skew and quantile. Base on these features, they employ different machine learning models like support vector machine, linear regression and random forest to show that the time-to-failure of earthquake is predictable. Apart from extracting features by hand, some researchers choose to do end-to-end learning and use 1D convolutional neural network to automatically get features from raw data [3]. The model start from 1D CNN to extract features from raw data with filter size 10. Through 3 convolution layers, the model gets 16 features, then these features are feed into several fully connected layers to get final output. Compared with these previous methods, our work uses Fourier transformation to extract more features that can not be seen in time domain.

3. Algorithm Evaluation

3.1. Feature Engineering

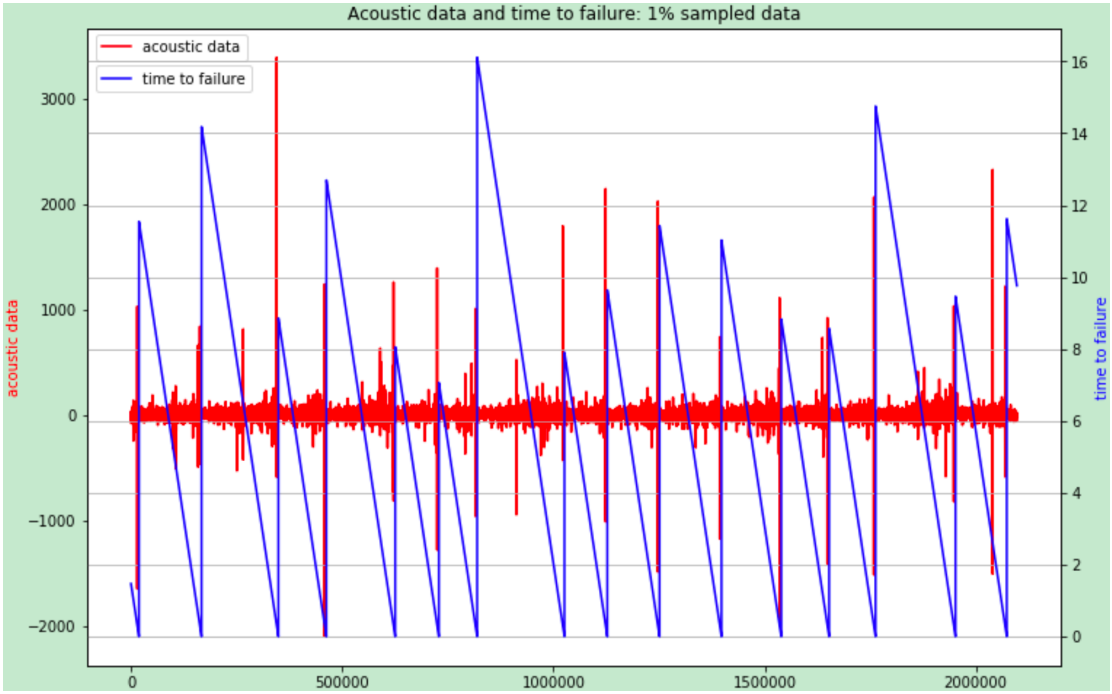

Figure 1. Raw data visualization

The data for this challenge comes from a classic laboratory earthquake experiment. The training data comprised a single continuous time segment of the recorded seismic data exhibiting multiple laboratory earthquakes as shown in Figure 1. The red curve shows the seismic signal recorded on a piezoceramic transducer located on the biaxial apparatus side block. Each burst in amplitude corresponds to a laboratory earthquake. The blue curve shows the time to failure derived from the earthquakes and the measured shear stress on the experimental apparatus. The test data consisted of individual small segments of a different portion of the same experimental data. Thus, the predictions from the test data could not be assumed by contestants to follow the same regular pattern seen in the training data, making the prediction challenging. In addition, there was no overlap between the training and testing sets

In our work, we also use the features proposed in most kernels, like mean, standard deviation, kurtosis, skew, and quantile (explained below). Besides, since the raw data is time related, we come up with the idea of transferring data into frequency domain, in which we can extract more features. If earthquake will not happen in a short time, there will only be some low frequency data. However, when earthquake is about to happen, high frequency data appears.

The feature engineering steps are the following:

1. Read in the training data as chunks of 150,000 lines because the test data given has a size of 150,000 lines

2. For a chunk of 150,000 lines, it is further divided into 3 parts and features are generated separately for each part.

3. For each part, the following statistical measurements are used as features:
    - Mean
    - Standard deviation
    - Maximum and minimum
    - Kurtosis: measure of tailedness of data
    - Skew: measure of asymmetry of data
    - Quantiles(0.01 / 0.05 / 0.95 / 0.99): cut point that dividing distribution
    - Mean at frequency domain
    - Standard deviation at frequency domain
4. Fourier transformation is applied to each chunk of data, after transferring to frequency domain we can extract some more features that can not be seen in time domain.

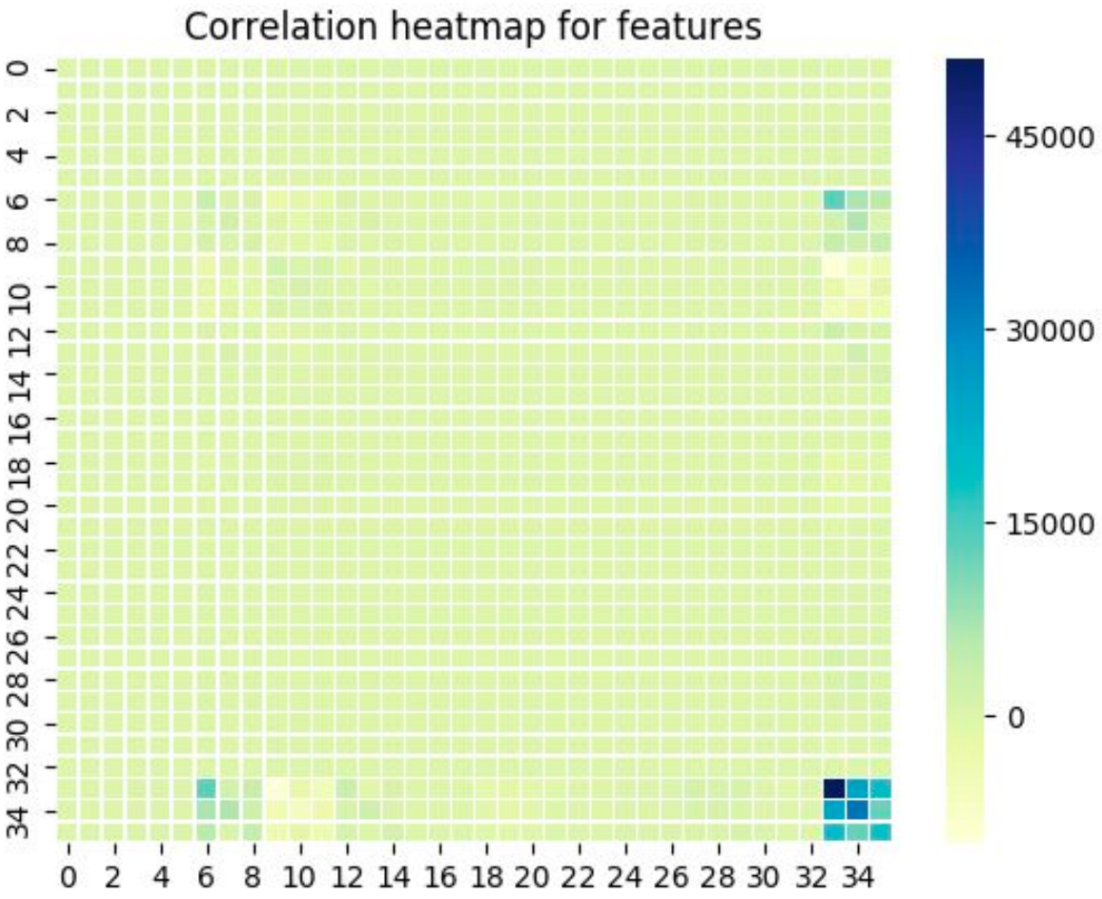

Figure 2. Feature correlation

Figure 2 shows the correlations of each features. The number represents the metric we mentioned above. If the correlation between two features are high, it means that one feature could be deduce from another. Therefore, the lower correlation between two features, the better features they are. Here it shows our features mostly all have relative low correlations.

3.2. Random Forest

Random forest is an ensemble method that multiple decision trees are constructed during training time and regression results are the average voting of each tree. In this project, we used sklearn to implement random forest regressors.

```
grid_model = RandomForestRegressor()

param_list = {'n_estimators': [50, 100, 150, 200, 250, 300, 320, 340, 360, 380, 400, 420, 440, 460, 480, 500],
              'bootstrap': [True, False],
              'max_features': ['auto', 'sqrt', 'log2'],
              'max_depth': [10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36]}

grid_search = GridSearchCV(grid_model, param_grid=param_list, n_jobs=16)
```

```
Best parameters
{'bootstrap': True, 'max_depth': 32, 'max_features': 'log2', 'n_estimators': 440}
```

```
MAE is 3.6664974583532604
```

Figure 3. Running Random Forest

Explanations:

- bootstrap: a method to pick a subset of the training set with replacement. The purpose is to use different to construct each individual decision tree.
- max_depth: the maximum number of depth of each decision tree
- max_features: number of features used to generate trees
- n_estimators: total number of trees inside this RF

Result: In this project, a common used metric called Mean Absolute Error (MAE) is applied to measure the performances of different models. It is defined as: $MAE = \sum(y_{pred} - y_{test})/n$. The MAE is conceptually the easiest evaluation metric for regression problems. It answers the question, "How far were you off in your predictions, on average?" If the value of MAE is smaller, the more accurate the model is. From the Figure 3, we can see the result by RF is 3.66.

3.3. Support Vector Machine

Support vector machine is a deterministic classification model, using support vector to get the decision boundary that has a largest margin to nearest data. It can also be used as a regression method, maintaining all the main features that characterize the algorithm.

Original support vector regression is a linear model. To extend it to fit nonlinear functions, kernel functions are introduced. What kernels do is to map the data to higher dimension from which it can be differentiate by a single line or a hyperplane. In scikit-learn there are two kinds of SVR implement, one is SVR, a normal support vector regressor, the other one is NuSVR, which has an restrict on number of support vectors. This limitation is a way to do regularization, but the partition of support vectorsshould

be carefully chosen. So we do grid search on some important hyperparameters: number of support vectors, penalty parameter, kernel and degree of polynomial kernel, and 5-fold validation is performed. Table 1 shows the parameter sets and results.

Table 1. Grid search parameter sets and results

nu \ C	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**
0.25	2.94	2.82	2.76	2.73	2.71
0.5	2.69	2.62	2.58	2.55	2.53
0.75	2.58	2.53	2.50	2.47	2.45
1.0	2.57	2.51	2.47	2.44	2.42

3.4. Recurrent Neural Network

Recurrent neural network is a neural network structure aiming at dealing with sequential data. Like reading a book, human will keep knowledge about previous chapters in mind while reading. So as the way how recurrent neural network works, during learning all neural cells share the same parameters and update together.

However, the original version of recurrent neural network is facing some problems, like losing what has been learned before in the situation that the time sequence is long enough and gradient vanishing. Thats why we come up with the long-short term memory structure. In the cell of LSTM, there are three "gates" to determine what to forget, what to update and what to output. After each LSTM cell there will be an output to system and a status output to next cell, and the status output will help the neural network model to keep a long-term memory.

We combine convolution layers and LSTM to catch both local and temporal information from features. Our model is shown in Figure 4, the first convolution layer will generate 64 features based on the manually extracted features and the second layer will extend them to 128 features. Then the whole sequence of features will be feeded to LSTM and outputs have dimensionality of 64. To make our model have more representation power, a two layers fully connected network is added to each output from LSTM. Two dropoutlayers are added to the fully connected layers to avoid overfitting, the first one has 0.5 dropping rate while 0.2 for the second one.

In Figure 5 it shows the loss curve. We set the learning rate to 0.0045 and it will decay exponentially every 850 epochs. The loss after 2000 epochs is 2.14, and the mean absolute error on test data is 2.11. But we are not pretty sure about the reason the loss increase in a sudden at about 500 epochs. One possible explanation is our model reaches the local minimum before the sudden increase. However, our optimizer is Adam, which can escape from the local optima and the loss will raise in a short time.

4. Future Work

Since the raw data only provides us limited features, our model can be improved by adding other powerful features, including the depth of the source, the crust structure, and geographical data. Apart from bagging (e.g., random forest algorithm), there is another ensemble method - boosting that we can use to get better models. Several support vector regressors can be assembled to outperform a single model by giving samples that are not fitted perfectly higher weights.

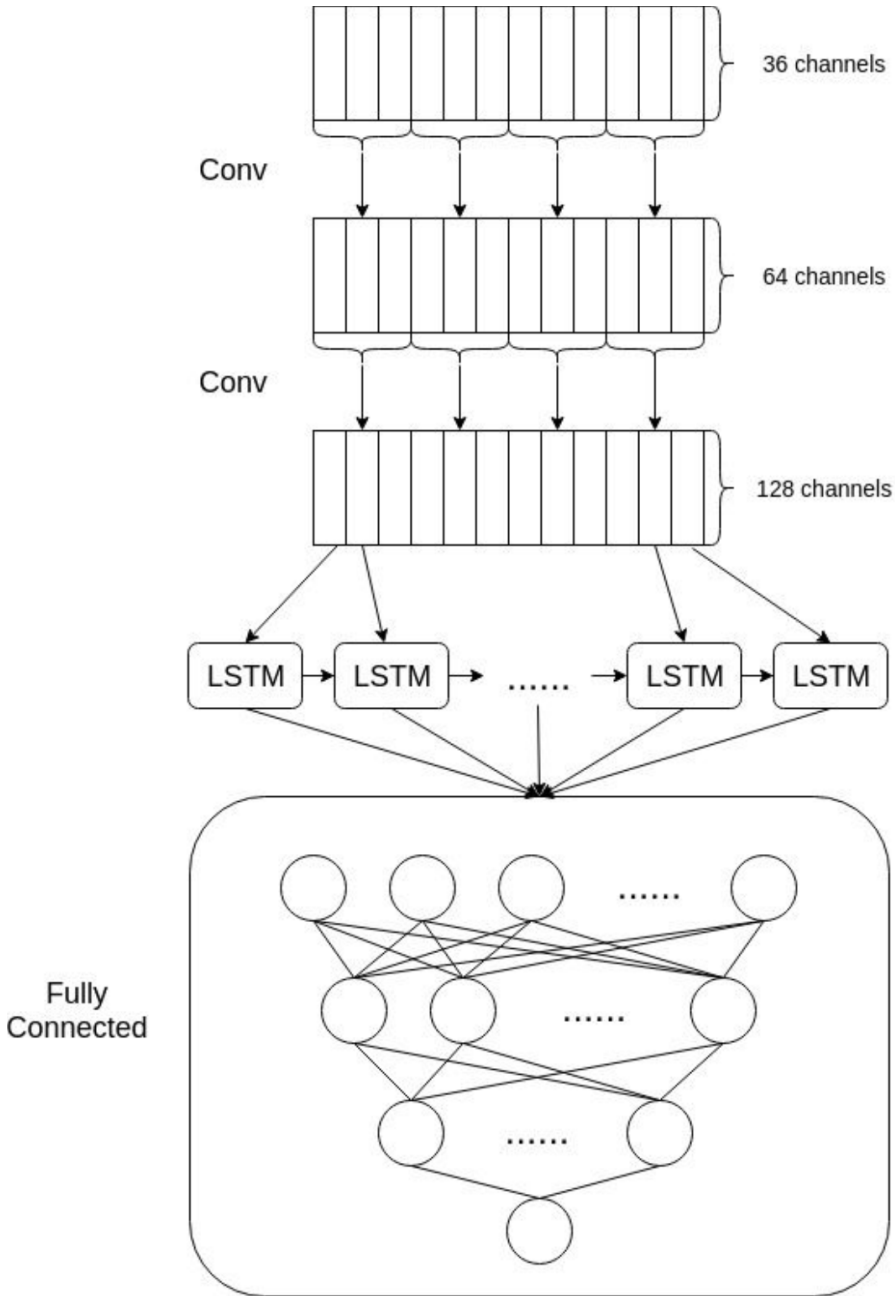

Figure 4. RNN prediction structure

5. Conclusion

In this work, we get the laboratorial earthquake data with time-related feature acoustic data and sequential outcome time to failure.To predict time to failure given acoustic data, we first analyze the data and extract features based on Fourier transformation. Two baseline algorithm RF and SVR are trained in this project. To improve the performance of prediction, we design a deep neural network combining convolution layers, recurrent neural network and fully connected layers to make sure that our model has strong representation power. After 2,000 epochs of training, mean square error is reduced to 2.14 and

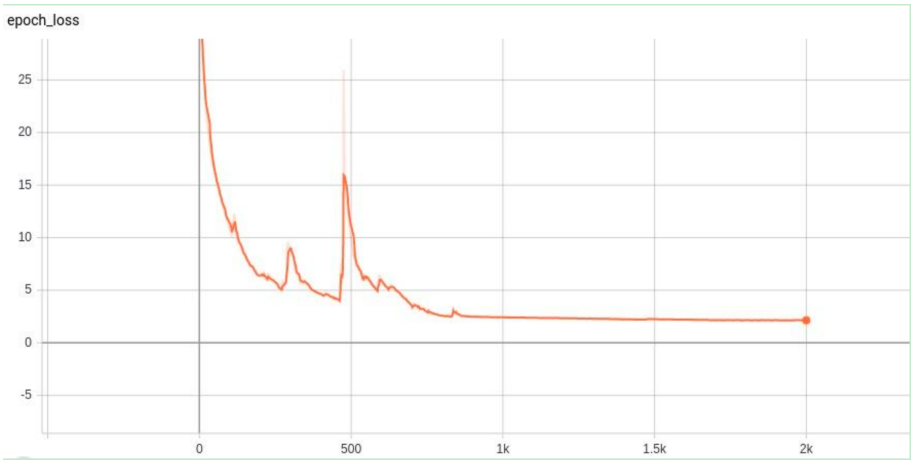

Figure 5. RNN prediction result

this model is applied to test data where mean absolute error is 2.11.

References

[1] F. Corbi, L. Sandri, J. Bedford, F. Funiciello, S. Brizzi, M. Rosenau, and S. Lallemand. Machine learning can predict the timing and size of analog earthquakes. *Geophysical Research Letters*, 46(3):1303–1311, 2019.

[2] C. Hulbert, B. Rouet-Leduc, P. A. Johnson, C. X. Ren, J. Rivière, D. C. Bolton, and C. Marone. Similarity of fast and slow earthquakes illuminated by machine learning. *Nature Geoscience*, 12(1):69–74, 2019.

[3] H. A. Jasperson, D. C. Bolton, P. A. Johnson, C. Marone, and M. Dehoop. Unsupervised classification of acoustic emissions from catalogs and fault time-to-failure prediction. In *AGU Fall Meeting Abstracts*, volume 2019, pages S53A–06, 2019.

[4] B. Rouet-Leduc, C. Hulbert, and P. A. Johnson. Continuous chatter of the cascadia subduction zone revealed by machine learning. *Nature Geoscience*, 12(1):75–79, 2019.

[5] B. Rouet-Leduc, C. Hulbert, N. Lubbers, K. Barros, C. J. Humphreys, and P. A. Johnson. Machine learning predicts laboratory earthquakes. *Geophysical Research Letters*, 44(18):9276–9282, 2017.