

Earthquake Prediction

Syed Zain Raza

May 19, 2019

1 Summary

I participated in an active competition of predicting earthquakes on Kaggle. The final model I choose is RNN-LSTM stacked, a deep recurrent neural network architecture, which takes given features as inputs and outputs the time remaining before the next earthquake. I implemented RNN-LSTM stacked using Keras and ran the code on a cloud platforms i-e AWS-EC2-Instance and Google Cloud Instance. Amazon EC2 instance had a CPU 4 CPU's and GPU of NVIDIA TeslaM60 with 30 GB memory and Google Cloud Instance had CPU 4 CPU's and GPU of NVIDIA TeslaP100 with 15 GB memory. Performance is evaluated using mean absolute error. The competition is still active so in the public leaderboard, my score is 1.443; I rank 691 among the 4063 teams.

2 Problem Description

Problem. In Earth sciences, forecasting earthquakes is one the great challenges. Due to their destructive nature, earthquakes have always a big impact on our Earth. In this competition, the goal is to predict when the next earthquake will take place. This is a regression and time series prediction problem where we have to predict the time before the next laboratory earthquake happens. The competition is at <https://www.kaggle.com/c/LANL-Earthquake-Prediction/overview>.

Data. The data for this competition was generated using laboratory methods and setup. The data has one feature "acoustic data" which are the seismic signals which will help us to predict earthquakes. The labels are "time to failiure" which is the time remaining before the next earthquake. Training data is of 9GB size. The number of training samples is $n = 629.1million$. Test data has 2624 segments, each of which has seismic signals and we have to predict the "time to failure" for each signal.

Challenges. One of the challenges for this project is the imbalance data. Only 13 times an earthquake happened in almost 630 million samples. Another interesting thing is that the training data is gathered using single continuous experiment. On the other hand, test data has different segments, for each of which we don't know whether they were captured within same and continuous experiment or maybe each segment was gathered in a different set of experiments. The predictions can not be assumed to follow the same pattern as that of the training data.

3 Solution

Model. The model I finally choose is the RNN-LSTM stacked, a standard deep recurrent neural network. I used two layers of LSTM with dropout. 10 dense layers and then one dense layer for output values.

Implementation. I implemented the RNN-LSTM model stacked using Keras with TensorFlow as the backend. My codes are available at https://github.com/ZainRaza14/kaggle_EarthquakePrediction. I ran the code on a cloud platforms i-e AWS-EC2-Instance and Google Cloud Instance. Amazon EC2 instance had a CPU 4 CPU's and GPU of NVIDIA TeslaM60 with 30 GB memory and Google Cloud Instance had CPU 4 CPU's and GPU of NVIDIA TeslaP100 with 15 GB memory. I used jupyter notebook on the instances for running my code. It took almost 3 hours to train the model on both instances.

Settings. The loss function is mean absolute error. The optimizer is adam with a learning rate of 0.0005. I used 100 epochs with 500 steps per epoch. Batch size was set as 32 while generating training and validation data.

Cross-validation. I tuned the parameters using a 5-fold cross-validation for random forest regressor and xgb regressor. I did not use cross fold validation for the RNN versions. I divided the data after analyzing the data. I kept the validation within a range which only had one earthquake and train data had 12 earthquakes in total.

4 Compared Methods

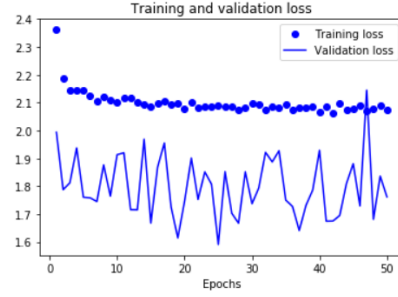
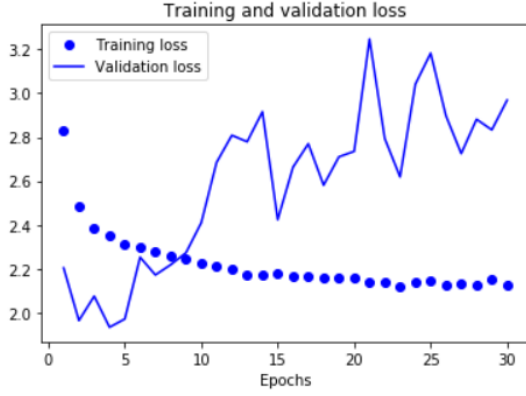
I tried different machine learning and deep learning models during this project.

Baseline I used a simple fully connected neural network as my baseline. It had only dense layers. It was giving me a mean absolute error of 2.137.

Random Forest I also tried implementing random forest regressor. I added a lot of features for this technique. Random forest had a max depth of 10, min samples split of 2 and n estimator of 100. This gave me a mean absolute error of 1.573.

XGB-Regressor This is another regressor technique that I used. Adding a number of features for this was also important. I tested with different learning rates i-e 0.01, 0.05, 0.5 and a max depth of 5. This method was giving me a mean absolute error of 1.665.

Simple LSTM I used a simple LSTM with only one LSTM layer. Also used a dense layer after it to generate a single output value. This was giving me a mean absolute error of 1.683.



(a) Training and Validation loss for GRU without dropout (b) Training and Validation loss for GRU with dropout











Figure 1: Losses.

RNN-GRU In this scenario, I used GRU layer without a dropout. I used two stacked GRU layers with a dense layer of 10 with relu activation function and then a dense layer of 1 at the end. This was giving me 1.647 error. The training and validation losses can be seen in the figure 1(a). It can be seen that the model is overfitting.

RNN-GRU with dropout This method was same as previous, the only thing I changed was to add recurrent dropout and dropout in the GRU layers. This was giving me 1.543 error. The use of dropouts significantly removed overfitting. It can be seen in the figure 1(b).

RNN-LSTM Stacked The last technique I used is a stacked LSTM with dropout. It also had a dense layer 10 with relu activation and then one dense layer to output the value for time to failure for next earthquake. I also tried multiple learning rates and hidden states for the layers. The final model was giving me a absolute error of 1.443. So that's why I chose it as my final model.

Method	Mean Absolute Error	Leaderboard
Baseline	2.137	2800
Random Forest Regressor	1.573	1900
XGB Regressor	1.665	2142
Simple LSTM	1.683	2300
RNN-GRU	1.647	2032
RNN-GRU with dropout	1.543	1400
RNN-LSTM Stacked	1.491	691/4063

682	noctiaplus		1.442	2	4d
683	chi_k		1.442	13	1mo
684	cainiaoluo		1.442	6	8h
685	zachanton		1.442	6	3mo
686	o933		1.442	61	20h
687	fb chem		1.442	19	5d
688	goodman		1.443	4	6d
689	Serge		1.443	51	21h
690	Maxim		1.443	10	2d
691	Syed Zain Raza		1.443	16	-10s

(a) Public leaderboard.

Figure 2: My rankings on the public leaderboard is as above !

5 Outcome

I participated in an active competition. My score is 1.443 in the public leaderboard. I rank 691/4063 in the public leaderboard. The screenshots are in Figure 2.

References

- [1] Li, Y., Li G, Zhang B, Wu G, Constructive ensemble of RBF neural networks and its application to earthquake prediction, in: Advances in neural networks-ISNN Springer, Berlin, Heidelberg, pp 532-537 (2005).
- [2] Alves E.I. Earthquake forecasting using neural networks: results and future work. Nonlinear Dyn 44, 341-349 (2006).
- [3] Alexandridis, A., Chondrodima, E., Efthimiou, E., Papadakis, G.. Large earthquake occurrence estimation based on radial basis function neural networks. IEEE Trans. Geosci. Remote Sens. 52 5443-5453 (2014).
- [4] DeVries, P. M. R., Viégas, F., Wattenberg, M., Meade, B. J. (2018). Deep learning of aftershock patterns following large earthquakes. Nature, 560(7720), 632–634.
- [5] Corbi, F., Sandri, L., Bedford, J., Funiciello, F., Brizzi, S., Rosenau, M., Lallemand, S. (2019). Machine Learning Can Predict the Timing and Size of Analog Earthquakes. Geophysical Research Letters