Machine Learning Engineer Nanodegree

Capstone Proposal (LANL Earthquake Prediction from Kaggle Competition) [1]

Tobias Steidle

(tobias.steidle@softwaredev.de)

April 24th, 2019

Proposal

Domain Background

Due to their devastating consequences, earthquakes and their prediction are one of the most important problems in the geosciences. The National Earthquake Information Center registers between 12000-14000 earthquakes per year[2]. The number of fatalities varies greatly. Between 2000 and 2015, an average of 50100 people dies each year[3]. The economic losses, which amount to billions every year, have not yet been taken into account.

Even if earthquakes cannot be prevented, a more precise prediction of when, where and how strong an earthquake will be would be of great help. This would increase the warning time and reduce the number of fatalities.

Problem Statement

So far there are no reliable predictions for the occurrence of earthquakes. So far it has been possible to circle the location and strength of earthquakes, but not the time. After an earthquake, data records usually show small aftershocks and certain patterns. The problem is that the sensors often measure the same thing without a dangerous earthquake following. No method is so sophisticated that earthquakes such as thunderstorms can be predicted.

Datasets and Inputs

The data sets are freely available for download on the Kaggle competition website[4].

The data consists of a train.csv file containing a single, continuous training segment of experimental data. And test data in the test.zip file.

Input Data fields

- acoustic_data - the seismic signal [int16]
- time_to_failure - the time (in seconds) until the next laboratory earthquake [float64]
- seg_id - the test segment ids for which predictions should be made (one prediction per segment)

Test data set

- test.zip containing many small segments of test data.

Solution Statement

There are several possible options to solve the problem. It is possible to solve approaches with a "classical" ML-algorithm like Random Forrest or Linear Regression as well as with a neural net. Since the seismic signals are similar to audio signals, methods of audio signal analysis can also be used. First, an MFCC (Mel Frequency Cepstral Coefficients)[5] is used to check whether some MFCC values have a linear relationship to time_to_failure. These values can be used for training. The predictions determined are then executed and evaluated on the test data set.

Benchmark Model

As this is a Kaggle Competition, the Public Leaderboard[6] is a good benchmark for the planned solution. Due to the restriction of a maximum of 2 submissions per day, part of the training data can be used as test data to check the effectiveness of the solution. My personal goal would be to be among the top 20% on the leaderboard. Currently, the best value is a Mean Absolute Error (lower is better) of 1.303. The value on the Leaderboard is calculated with approximately 13% of the test data.

Evaluation Metrics

The used metric is the Mean Absolute Error[7]. The Mean Absolute Error measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

n = Number of predicted values

y_j = real value

$\hat{y}_j$ = predicted value

For each record oft he test dataset the prediction error is calculated. Convert each error to a positive figure by taking the absolute value for each error. Finally, calculate the mean value for all recorded absolute errors.

Project Design

I am planning the following workflow to solve the problem:

- Data preprocessing
  - Visualize a section of the data to improve the understanding of the data (using plots)
  - Apply a mel-frequency cepstral coefficients (MFCC) to identify relationships between the acoustic data and time_to_failure
  - Use Prinipal Component Analysis (PCA)[8] to reduce the number of trainable features
  - Splitting the data into a training and test set

- Training
  - testing different ML algorithms (regression, trees, etc.) possibly also a neural network
  - Evaluate the results to select the best algorithm

- Optimization / Finetuning
  - GridSearch to determine the best hyper parameters[9]
  - Improve performance with Cross Validation[10]

[1] https://www.kaggle.com/c/LANL-Earthquake-Prediction/
[2] https://www.iris.edu/hq/inclass/fact-sheet/how_often_do_earthquakes_occur
[3] https://www.statista.com/statistics/263108/global-death-toll-due-to-earthquakes-since-2000/
[4] https://www.kaggle.com/c/LANL-Earthquake-Prediction/data
[5] https://en.wikipedia.org/wiki/Mel-frequency_cepstrum
[6] https://www.kaggle.com/c/LANL-Earthquake-Prediction/leaderboard
[7] https://www.statisticshowto.datasciencecentral.com/absolute-error/
[8] https://towardsdatascience.com/a-step-by-step-explanation-of-principal-component-analysis-b836fb9c97e2
[9] https://scikit-learn.org/stable/modules/grid_search.html
[10] https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f