Earthquake Detection from Seismological Data

Lucien Michaël Iseli, Florian Maxime Charles Ravasi and Jules Eliot Gottraux
Master of Data Science, EPFL, Switzerland

I. Introduction

In seismology, the detection of earthquakes is a pretty active research. The detection of severe earthquake, those that make your house tremble and are actually dangerous, is not a particularly tricky task but when talking about earthquakes of small magnitude that's a different story. In fact, as of today the detection and classification of earthquakes is still done by hand by specialists. That is they manually inspect the measurements taken by sensors to produce the catalogs, the tables containing the earthquakes's informations. This process is error-prone and tedious, this is why we try to address this issue in this project. We'll use data from sensors that capture the vibration present in the earth, the same data that specialists in detection use, to create a machine learning model able to detect if an earthquake has happened. Most of the work we will present will be prior to the creation and optimization of the machine learning algorithm. That is we spent most of our time preparing the data and doing featuring engineering. As we will see, this task revealed to be very difficult and one would need more time and substantial expertise to obtain satisfying performances for small earthquake detection. However by simplifying the task we managed to get acceptable result. In fact this work is inspired by a research [1] conducted by three researchers, they worked hard to achieve that using neural networks and clever memory management.

II. Dataset Characteristics

We get our data using the ObsPy library [2]. ObsPy grants us access to stations, which have sensors, which record the seismological data over time on different channels. Those sensors record constantly the activity, or vibration, of the earth at their location. This include seismic waves which are the waves of energy released when a seism occur nearby. This data is thus one dimensional and filled with noise as it is a completely raw measure, it is a giant time serie containing the amplitude of waves captured by the sensor. It has the information of the seism and earthquake but this information is aggregated with all the vibration happening near the sensor. The goal will be to create, from this time series, meaningful features for the model containing sufficient information to classify a certain window of time. We'll thus fix the duration of a time window, for example one minute, then compute features on this time window and feed these features to the machine learning model.

So the basis for the features is this time series, for the labels we use a hand-made catalog of earthquake that contains, among other properties, the location, magnitude and time of the earthquake.

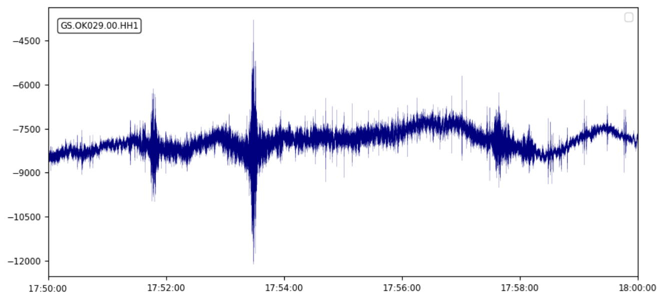

Fig. 1. 10 minute of data example, no earthquake

Before diving into the machine learning model construction, we have to take care of several difficulties inherent in the dataset.
First the data collection, after having chosen the location, station and channel from which we want to get the time series we have to download it and store that to a usable format for the next steps. The frequency of the sensors is 100 data point per second, so the amount of data increases quite quickly. Then, the data has holes in it, since it is sometimes missing for some period. Those holes are not regular and unpredictable, so we have to take care of them with caution.

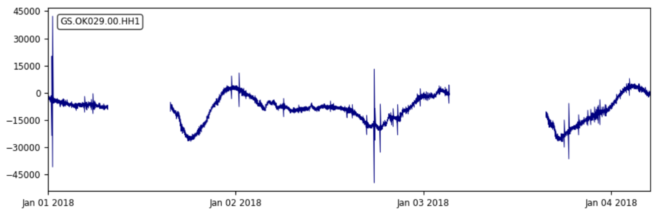

Fig. 2. Example of missing values in the dataset

We already mentionned that the data obtained from ObsPy is very noisy, we want to emphasize this and as we will see in examples, some earthquakes and non-earthquakes are undistinguishable for the untrained eye. The catalog is noisy in a sense as well, since the time series are parsed by humans and thus some earthquakes are missed and as the magnitude decreases the number of missed earthquake increases, being the majority for small magnitudes. Another difficulty in the catalog is the delay that is induced by the difference of location between the detection of the earthquake and the location of our station. All the catalog has to be calibrated to ensure that the time of the earthquake corresponds to a peak in our time series. However, the speed of propagation of an earthquake isn't constant. The so-called p-waves's speed of an earthquake varies quite a bit, depending on the rocks it travels through for example.

III. Data pre-processing

We first have to take of the anomalies in the data: the delay induced by the physical distance separiating our sensors and the location of measurement and the temporal holes.
For the missing data, we take care of that when loading the data. We interpolate it with gaussian noise so that it represent the whole space. This way it will just be taken as a moment where nothing interesting happened and it should not worsen the performance, as it is quite representative to some part of the dataset where no earthquake has happened. - Fix delay of catalog
Convert latitude, longitude, elevation to a cartesian coordinate system in order to compute accuracte distance easily, we do this using ECEF coordinates[3]
Now that we have distance, we manually find the start time of some earthquake and see the time difference. With time difference and distance we computed the average speed of propagation or earthquake/p-waves and got $5676.611\frac{m}{s}$ which is very close to what they found in [4].

IV. Labeling of the Data

Since the dataset lives in a continuous space, we have to discretize it. We have to choose time windows that will be classified as earthquake or not earthquake. The choice of the time is crucial and it is hard to predict what would be a good choice. Also, that gives rise to a fundamental question for the creation of the labels: when does an earthquake end? It is possible, in fact rather probable depending on the length of the discretization process, that an earthquake spans over multiple time windows thus knowing the duration of the earthquake would permit to overcome this issue. This is important, because not being able to accurately and correctly label our data will of course be disastrous for the machine learning model. Unfortunately, the duration of an earthquake is an open question in geology, the only option simple enough is to label as earthquake only the window that contains the moment of an earthquake.
The problem with this approach is illustrated in figure 3. In the catalog the earthquake will be at T_1. We can see that the earthquake indeed trigger a big instant augmentation in the variation of the amplitude, that is typical to the earthquakes that are quite easy to classify. The problem here is that when discretizing the time series in windows, the partitioning of the data will change

V. Feature Creation

From that seismic waves we have to choose what processing and transformations will have to be done to be able to feed the machine learning algorithm. We have two main options: leave it as it is and opt for an algorithm such as convolutional neural networks, the approach they took in ConvNetQuake (the paper [1] that inspired this project), or perform some feature

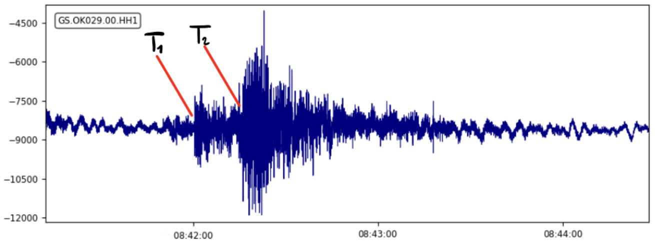

Fig. 3. Illustration of the problem of time window labeling

engineering by hand to be able to use some more standard algorithm, this is the approach we chose. So rather than having an gigantic input dimension we choose to compress it in a smart way, creating meaningful features for the algorithm. By taking time window of 10 seconds, this compression is on the order of 100 if we create tens of features. This permits to accelerate the training process and thus allow to span more time or take more channel.
We begin by adding the first measures that comes to mind: the standard deviation, the maximum and minimum amplitude **ask lucien**. For the next ones, let's take an example to see what characterise an earthquake:
- Examples and presentation of the two first features - Transition to fourier and presentation of those features

VI. Model Selection

- Tried svm with gaussian kernel $\rightarrow$ shitty
- Elaborate on the unbalance problem $\rightarrow$ trees
- Difficult so we move to a simpler problem $\rightarrow$ present richter's magnitude scale
- Still difficult

VII. Conclusion

VIII. Further improvements

- Better features, signals, seismology
- Undersampling
- Oversampling
- More robust way to compute speed of P-waves, for example compute average speed in each different direction or get ground information to have a better idea of how fast p-waves will go through the ground
- Finding a better way to decide if a time-window has an earthquake or not
- Multiple stations
- Better way to choose time window?
- Better knowledge of deep nets or random forests - Try another place with more earthquake as we focus on notiecable earthquake and those are rare in oklahoma

References

[1] Convolutional neural network for earthquake detection and location, 14 February 2018.
`https://advances.sciencemag.org/content/4/2/e1700578`

[2] ObsPy: python library to collect seismological data.
`https://docs.obspy.org/`

[3] J. Zhu, (1994), Conversion of Earth-centered Earth-fixed coordinates to geodetic coordinates.
`https://ieeexplore.ieee.org/abstract/document/303772/`

[4] Guy T. Kuster and M. Nafi Toksöz, (1974), ”Velocity and Attenuation of Seismic Waves in two-phase Media: Part I. Theoretical Formulations,” geophysics 39: 587-606.
`https://library.seg.org/doi/abs/10.1190/1.1440450`