Prediction of Earthquakes using machine learning processing of multiRTL features*

P. Proskura, A. Zaytsev, I. Braslavsky, E. Egorov, E. Burnaev

March 25, 2019

Abstract

We want to solve a classification problem: if the magnitude of an earthquake is greater than a threshold at a given location and time. The goal is to predict if such an earthquake occurs in a time range $[30, 180]$ after the current date. We use data on Japan earthquakes 1992-2005 and consider predictions at locations given in this database. In our machine learning model, we use Region-Time-Length (RTL) features as inputs inspired by Gutenberg–Richter's relationship. As machine learning models we use various approaches as logistic regression, random forest, adaptive boosting and gradient boosting. The best model provides precision and recall as high as 0.95 and 0.98 correspondingly.

Keywords: Machine learning, RTL algorithm, Earthquakes prediction

1 Introduction

There is no way to build an accurate earthquake precursor because of the complex nonlinear behavior of seismicity. So, the approach using seismic data in combination with machine learning methods has become more popular to overcome this problem [7].

There are a number of different problem statements related to earthquake predictions: analyze all map and predict place and time of the next earthquake, split the map into several pieces and predict earthquake on the each

*Supported by Institute for Information Transmission Problems

of them, consider each earthquake as a separate event with given magnitude, location and time and predict the value of the magnitude [7]. In this paper we consider the third approach: we construct a model that predicts if the magnitude is greater than a given threshold value.

To solve applied problem scientists usually adopt several ideas. They start with some empirical relationships or mathematical model that provide precise enough representation of reality. However, often this representation is not good enough, and they adopt some kind of Machine learning on top of physics-driven description.

In the contest of prediction of large earthquakes scientists managed to obtain several important empirical statistical relationships. Let us note some of the most important relations. Gutenberg–Richter law [4] expresses the relationship between the magnitude and total number of earthquakes as follows

$$\log N = a - bM, \tag{1}$$

where N is the number of events with the magnitude greater than M, a and b (commonly referred to as the $\mathbf{b-value}$) are fitting coefficients.

Omori–Utsu (O–U) law [5] represents the decay of aftershock activity with time

$$\dot{N}(t) = \frac{C_1}{(C_2 + t)^p}, \tag{2}$$

where t is time, N is earthquake count, C_1, C_2 and decay exponent p (commonly referred to as the $\mathbf{p-value}$) are fitting coefficients.

On top of these features there are a lot of applications that consider prediction of earthquakes using machine learning. Authors of [6] considered the prediction of earthquakes as a problem of binary classification. They generated 51 meaningful seismic features calculated for our dataset based on well-known seismic facts such as "Standard Deviation of b-value" or "Time (T) of n events". As models were used various ensemble methods such as Random Forest, Rotation Forest and RotBoost. TODO: overview of large number of articles.

In this paper we consider slightly different problem statement, as we consider distant time interval for prediction of earthquakes. As inputs we use RTL features. To construct a better model we consider various machine learning techniques as wel as additional heuristics: normalization of RTL features and combination of RTL with various hyperparameters.

2 Description of the problem

Let t is the index of time. Y_t is a time series, where Y_i is a target event. We also observe features $X_{t,k}$. Our proposal is to predict the target event Y using feature description X. So, We have a history $X_{t,k}, Y_t$ up to the time T, $t \in [0, T]$. We have to raise the alarm in the window $[T, T_{min}]$, in this case, the target event occurs in the window $[T + T_{min}, T + T_{max}]$. In our approach we use different combinations of RTL Features.

2.1 Formal problem statement

To solve classification problem we need to create a matrix of input features $\mathbf{X}$ and target labels $\mathbf{Y}$. So, we want to make labels of the target events Y for features X^j, each of which is indexed in space-time by vector (x, y, t). Thus, we will create a labels for the indices (x, y, t).
The target earthquakes are everything that falls into the space-time cylinder, i.e. for the statistics calculated for the index (x, y, t) the value 1 ($\mathbf{Y}$) is assigned if there is at least one earthquake of magnitude $M > Mc$ with coordinates (x_e, y_e, t_e) satisfying the following constraints:

$$||(x, y) - (x_e, y_e)||_2 \leq R_c, \quad \delta_c < t_e - t < T_c \tag{3}$$

Thus, we can determine the optimal working conditions for the predictive model by choosing the hyperparameters (R_c, δ_c, T_c) of the labels.
Also, it should be noted that for a **RTL** precursor, we can obtain a series of precursors by specifying various hyperparameters (r_0 and t_0). Let's describe our approach to earthquake prediction:

- Make a grid of hyperparameters (r_0^i, t_0^i). For each set of parameters counts RTL statistics for each earthquake that constructs $\mathbf{X}$ matrix of features.
- Consider a grid of hyperparameters (M^i, R_c^i, δ_c^i, T_c^i). For each parameter set, form the labels of the target events that shapes into $\mathbf{Y}$ labels.
- Evaluate quality of the machine learning model for the sample ($\mathbf{X}$, $\mathbf{Y}$), defined by hyperparameters.

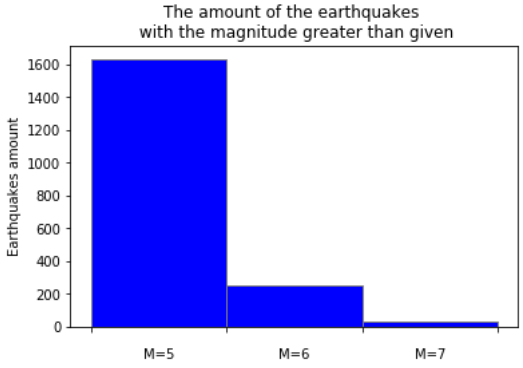

Figure 1: Histogram of magnitudes: earthquakes with greater magnitude consist a small part of the sample.

2.2 Data

We study the prediction of strong earthquakes in the middle-term horizon. Strong earthquake is an earthquake with the magnitude higher than $M_c = 5$. Predictions of earthquakes are related with the difficulties:

- The sample is unbalanced. In Japan, from 1990 to 2016, there were $247,204$ earthquakes. Consider a distribution by the magnitudes which is shown in Figure 1. The most part of classifiers and their metrics are common with balanced samples, where the amount of target class is approximately the half of the total. Thus in our case, the tuning of classifier and choice of metrics are needed to make them more sensitive to the target class.

- The dataset is nonhomogeneous as the network of seismic stations changes as time passes. So, the process of features generation is harder, because every couple measurements are needed to be evaluate for approximate equality.

- There is a lag in time between available and desired prediction interval, because exact point is unavailable according our measurements.

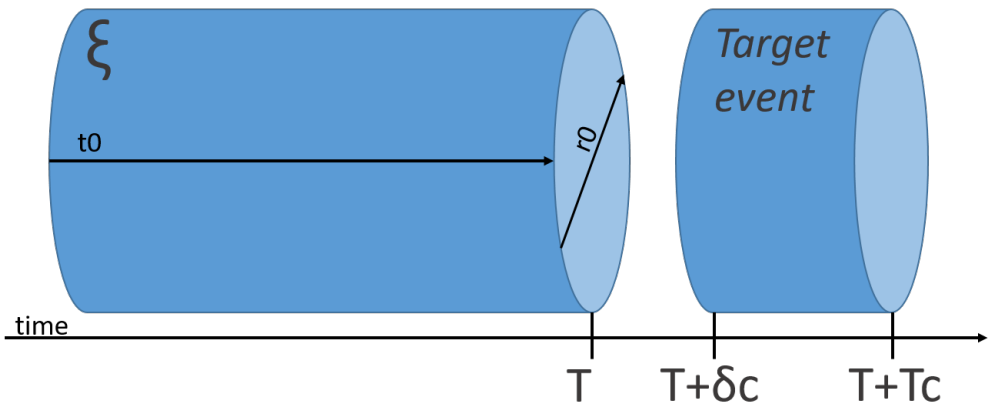

Figure 2: Space-time cylinder used for generation of RTL features and interval considered for prediction

3 Methods

3.1 RTL Features

As X-input for ML model we use RTL features. The basic assumption of the **Region – Time – Length**(**RTL**) algorithm [1] is that the influence weight of each prior event on the main event under investigation may be quantified in the form of a weight. Weights become larger when an earthquake is larger in magnitude or is closer to the investigated place or time. Thus, **RTL** characterizes the level of seismicity at the point of space in the certain time.

The **RTL** takes into account weighted quantities associated with three parameters (time, place and magnitude) of earthquakes. A **RTL** parameter is defined as the product of the following three functions:

$$\mathsf{RTL}(x, y, z, t) = \mathsf{R}(x, y, z, t) \cdot \mathsf{T}(x, y, z, t) \cdot \mathsf{L}(x, y, z, t),$$

where $\mathsf{R}(x, y, z, t)$ is an epicentral distance, $\mathsf{T}(x, y, z, t)$ is a time distance and $\mathsf{L}(x, y, z, t)$ is a rupture length. They depends on the size of the space-time cylinder $\mathcal{E}$, defined by radius r_0 and time length t_0. Figure 2 demonstrates used space-time cylinder and the target interval for prediction.

$$\mathsf{R}(x,y,z,t) = \left[\sum_{r_i \in \mathcal{E}} \exp\left(-\frac{r_i}{r_0}\right)\right],$$

$$\mathsf{T}(x,y,z,t) = \left[\sum_{t_i \in \mathcal{E}} \exp\left(-\frac{t-t_i}{t_0}\right)\right],$$

$$\mathsf{L}(x,y,z,t) = \left[\sum_{l_i \in \mathcal{E}} \left(\frac{l_i}{r_i}\right)\right]$$

For l_i we use an empirical relationship for Japan $\log l_i = 0.5M_i - 1.8$. We use only earthquakes with magnitude at least $M_0 > 5.0$

RTL is a very unstable statistics. Therefore in the article [2] author proposed to normalize the parameters on the variances.
Normalization is one of the steps of the preprocessing. For our case we use the proper type of normalization - standardization. The purpose is to make each feature have zero-mean and unit-variation [3]. It takes a few steps:

- Determine distribution mean and standard deviation for each feature.
- Subtract the mean from each feature.
- Divide the values of each feature calculated previous by its standard deviation.

Also we can subtract the moving average instead of mean. It will help to take into account time trend. The use of normalization increases qualtiy metrics by about 3%.

Thus the negative **RTL** means a lower seismicity compared to the background rate around the investigated place, and the positive **RTL** represents a higher seismicity compared to the background Figure 3. We are interested in both types of anomalies.

3.2 Classifiers

As classifiers we used the following machine learning methods:

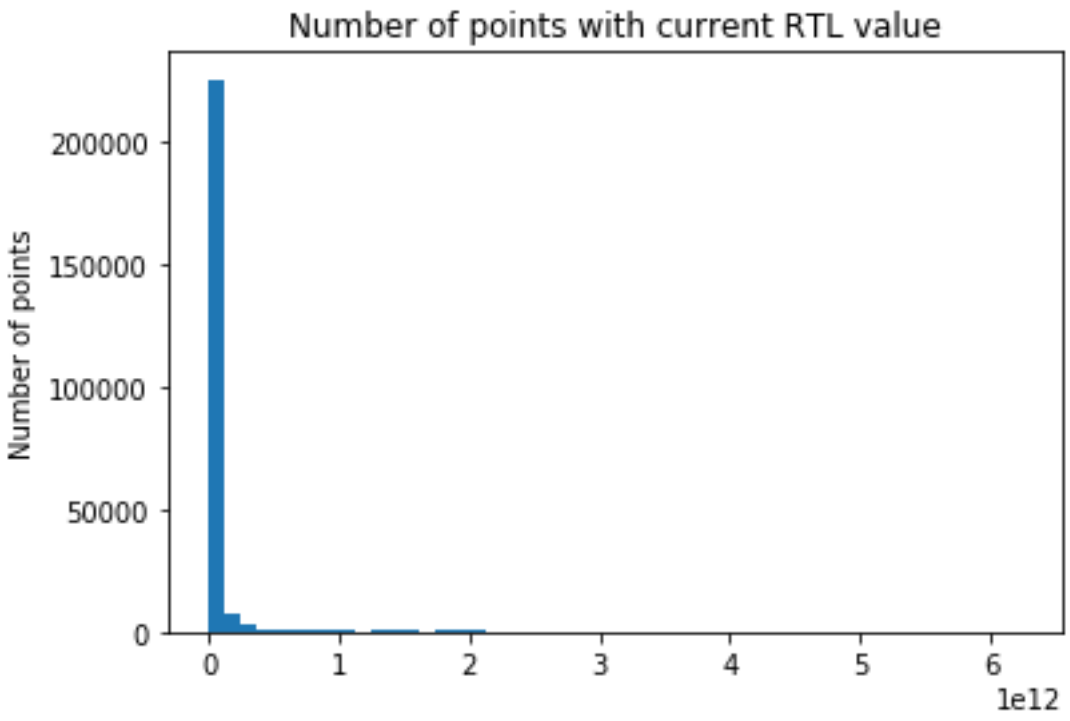

Figure 3: Histogram of RTL values for each point, smaller values prevail.

- **Logistic regression**. This is a statistical model used to predict the probability of occurrence of an event. The model is based on the following assumption:
$$Pr(y = 1|x) = f(\theta^T x),$$
where $f(z) = \frac{1}{1+e^{-z}}$ and θ are the parameters of the model [8].

- **Random Forest**. This is an ensemble classifier that is developed on the basis of majority voting of decision trees. Various number of decision trees are generated over bootstrap samples of the training dataset. The final decision is made by aggregating the predictions obtained by all the decision trees. Thus, a Random Forest allows to find complicated relationships in the data, but at the same time more resistant to retraining [9].

- **AdaBoost**. This is another method that combines classifiers into ensembles. The key feature of this is that it introduces weights for all objects. At each iteration, the weights of each incorrectly classified object increase, so a new classifier ensemble focuses attention on these objects. AdaBoost is sensitive to noise and rejects in data. However, it is less prone to overfitting compared with other algorithms of machine learning [10].

- **Gradient Boosting**. This is an ensemble method that builds an ensemble of trees one-by-one, then the predictions of the individual trees are summed. The next decision tree tries to cover the discrepancy between the target function $f(x)$ and the current ensemble prediction by reconstructing the residual. Thus, an iteration process is constructed, in each iteration of which the loss function is minimized by gradient descent [11].

- **Major RTL**. This is the method which estimates the threshold for RTL features. We evaluate our model to count optimal threshold from the train sample. If the value less than that threshold than the label is 0, 1 otherwise. According to the threshold labels for test are counted.

3.3 Metrics

We solve Imbalanced Classification Problem, so we chose appropriate metrics. In addition to common Precision, Recall and ROC AUC we look at F1-score

and PR AUC. See definitions of the metrics in Appendix A.

3.4 Hyperparameters and quality metrics

The following grid was created for **RTL**:

$$\mathbf{r_0} = [10, 25, 50, 100],\ \mathbf{t_0} = [30, 90, 180, 365]$$

. The grid to find the best way to predict earthquakes was multidimensional and included (r0, t0) array, different values of adjusted magnitude, number of test samples and different classifiers.

Results of the grid with the best test size and best $t0$ for every $r0$ are given in Table 1. The best result is for Gradient Boosting and its realization as XGBoost [12], for the greater size of cylinder and for the smaller gap between time of measurements and the target event. Gradient Boosting works better than Logistic Regression because of non-linear behaviour of dependence of the magnitude of earthquakes on RTL features. Both these approaches work better than Major RTL because of difficulties of the RTL model.

3.5 Result for a set of RTL features

In this section we used 16 calculated RTL features as inputs for a machine learning models. Quality of obtained models are available in Table 2. The improvement compared to the single best RTL feature with $M_c = 50, R_c = 50, \delta_c = 10, T_c = 180$ is small.

4 Conclusion

We considered the problem of middle term earthquakes prediction. For this problem we obtained high quality results for Gradient Boosting. Usage of Machine learning provide an improvement compared to the state-of-the-art major_RTL method. However, for most of the cases one suitable RTL feature is not worse than a set of them generated using different hyperparameters.

References

[1] Sobolev G. A. and Y. S. Tyupkin 1997: Low-seismicity precursors of large earthquakes in Kamchatka. Volc. Seismol., 18, 433-446

r0	best t0	Algorithm	Precision	Recall	F1	ROC AUC	PR AUC
10	180	Logistic Regression	0.54	0.36	0.43	0.64	0.53
		Random Forest	0.62	0.51	0.56	0.76	0.69
		AdaBoost	0.63	0.50	0.56	0.77	0.83
		Gradient Boosting	0.62	0.52	0.56	0.80	0.69
		Major_RTL	0.57	0.47	0.47	0.52	0.77
25	90	Logistic Regression	0.72	0.58	0.64	0.70	0.44
		Random Forest	0.79	0.51	0.62	0.74	0.68
		AdaBoost	0.75	0.67	0.71	0.77	0.53
		Gradient Boosting	0.82	0.6	0.69	0.79	0.77
		Major_RTL	0.67	0.52	0.62	0.70	0.60
50	180	Logistic Regression	0.91	0.84	0.88	0.83	0.67
		Random Forest	0.91	0.84	0.87	0.80	0.70
		AdaBoost	0.83	0.96	0.89	0.81	0.74
		Gradient Boosting	0.89	0.92	0.91	0.87	0.73
		Major_RTL	0.60	0.74	0.74	0.80	0.71
100	180	Logistic Regression	0.94	0.94	0.94	0.80	**0.94**
		Random Forest	**0.97**	0.90	0.94	0.89	0.90
		AdaBoost	0.96	0.96	0.96	0.92	0.90
		Gradient Boosting	0.95	**0.98**	**0.97**	**0.93**	**0.94**
		Major_RTL	0.90	0.86	0.89	0.83	0.89

Table 1: Results for different values of hyperparameters for generation of RTL features: the better results are for bigger size of cylinder

Algorithm	Precision	Recall	F1	ROC AUC	PR AUC
Gradient Boosting (best single RTL)	0.97	0.96	0.96	0.95	0.94
Logistic Regression	0.94	0.95	0.94	0.81	0.94
Random Forest	0.97	0.90	0.94	0.89	0.90
AdaBoost	0.97	0.96	0.97	0.92	0.90
Gradient Boosting	0.95	0.98	0.97	0.93	0.93
Major_RTL	0.90	0.85	0.89	0.82	0.89

Table 2: Result for 16 features parameters for X and Y generation

[2] Huang Q. Seismicity pattern changes prior to large earthquakes-An approach of the RTL algorithm //TERRESTRIAL ATMOSPHERIC AND OCEANIC SCIENCES. – 2004. – T. 15. – №. 3. – C. 469-492.

[3] Aksoy S., Haralick R., Feature Normalization and Likelihood-based Similarity Measures for Image Retrieval, Pattern Recognit. Lett., Special Issue on Image and Video Retrieval, 2000

[4] Gutenberg B. and Richter C., Seismicity of the earth and associated phenomena: Princeton University Press, 1954.

[5] Utsu T. and Ogata Y., "The centenary of the Omori formula for a decay law of aftershock activity," Journal of Physics of the Earth, vol. 43, pp. 1-33, 1995.

[6] Asim K. M., Idrisb A., Martínez-Álvarezc F., Iqbala T., Short Term Earthquake Prediction in Hindukush Region using Tree based Ensemble Learning, 2016 International Conference on Frontiers of Information Technology

[7] Asim K. M., Awais M., Martınez–Alvarez F., Iqbala T., Seismic activity prediction using computational intelligence techniques in northern Pakistan, 2017, Acta Geophysica, Volume 65, Issue 5, pp.919-930

[8] Friedman J., Hastie T., Additive logistic regression: a statistical view of boosting, 1990, The Annals of Statistics

[9] Tin Kam Ho, Random Decision Forests, Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282.

[10] Freund Y., Schapire R., A Short Introduction to Boosting, Journal of Japanese Society for Artificial Intelligence, September, 1999

[11] Friedman J., Greeedy Function Approximation: A Gradient Boosting Machine, IMS 1999 Reitz Lecture

[12] Chen T., Guestrin C., XGBoost: A Scalable Tree Boosting System, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. ACM. pp. 785–794

A Quality metrics for classification problem

Introduce necessary definitions: Classification problem can be formulated as whether this object belongs to the target class or not.

- True Positive — if the object belongs to the target class and we predict that it belongs.
- True Negative — if the object doesn't belong to the target class and we predict that it doesn't.
- False Positive — if the object doesn't belong to the target class but we predict that it does.
- False Negative — if the object belongs to the target class but we predict that it doesn't.

The **precision** score quantifies the ability of a classifier to not label a negative example as positive. The is the probability that a positive prediction made by the classifier is positive. The score is in the range $[0, 1]$ with 0 is the worst, and 1 is perfect. The precision score can be defined as:

$$\textbf{Precision} = \frac{TruePositive}{TruePositive + FalsePositive}$$

The **recall** score quantifies the ability of the classifier to find all the positive samples. It defines what part of positive samples have been chosen by classifier as positive. The score is in the range $[0, 1]$ with 0 is the worst, and 1 is perfect.

$$\textbf{Recall} = \frac{TruePositive}{TruePositive + FalseNegative}$$

The **F1 - score** is a single metric that combines both precision and recall via their harmonic mean. It measures the test accuracy and reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$\textbf{F1} = 2\frac{PrecisionRecall}{Precision + Recall}$$

ROC AUC score counts the curve area under the Roc_curve. Roc_curve

is the plot True Positive rate from the False Positive rate, which defines as

$$TruePositiveRate = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$FalsePositiveRate = \frac{FalsePositive}{FalsePositive + TrueNegative}$$

ROC AUC score measures the quality of binary classifier. The best value is 1, value 0.5 is equal to random classification.

PR AUC score counts the curve area under the Precision_Recall_curve: Precision from Recall. Precision-Recall is a useful measure of success of prediction when the classes are very imbalanced. The perfect classifier curve ends in $(1.0, 1.0)$ and has area under it that equals 1.