Report

On

“FORECASTING EARTHQUAKES USING NEURAL NETWORK MODELS”

Submitted by-

Arnav Ahuja 2018B4A70619P

Study Oriented Project (MATH F266)

Supervised by-Prof. Sumanta Pasari

BIRLA INSTITUE OF TECHNOLOGY AND SCIENCE, PILANI

SEM-I, 2020

`

1. Objective

Building an **Earthquake Forecasting System** with the help of a neural network model that forecasts whether an earthquake of threshold magnitude or above is going to strike in the next 30 days or not.

2. Motivation

Earthquakes, when they strike are a highly devastating phenomenon, which cause a high loss to society and human life. Therefore, making a system which forecasts the occurrence of earthquakes becomes a very important task.

3. Introduction

In this world, earthquake is a major phenomenon which causes huge losses in both life as well as property. Since ancient times, we have always been looking for ways to prevent or control these natural disasters. The earthquake is a phenomenon which cause heavy losses of life and property. When it comes to India, the North-Eastern region is the most prone to earthquakes which has led to huge socio-economic damages. Shillong in the north-east region is situated amongst a maze of several crisscrossing faults like the Dauki Fault and Kopili Fault, which is the reason of so many earthquakes in the north-east. Improvement of seismic hazard assessment requires exhaustive catalogues.

Prediction of the time of occurrence, magnitude, and epicentral location of future large earthquakes has been the subject of several scientific efforts with distinctly different conclusions in recent years. Many methods have been proposed to forecast the occurrence of earthquakes like statistical methods and other modelling approaches. Such procedures are based on either the study of precursory phenomena before earthquakes such as seismic quiescence, changes in magnetic and electric signals recorded at seismic sites, and abnormal animal behaviour, or the analysis of historical earthquake data recorded in seismic catalogues.

The exponentially rising volume of seismic data has provided us with the opportunity to involve deep learning algorithms to detect and locate earthquakes reliably. By the universal approximation theorem, neural networks can model any complex function. Therefore, they must be able to model the complex relationship between seismic information and the occurrence of earthquakes.

Neural networks perform a key task in the prediction of earthquake. The neural network architecture is created to forecast these natural disasters with different deep learning optimization algorithms.

`

4. Literature Review

4.1 Related Work

This section explores the many methods recently published in the literature related to forecasting earthquakes. Most of methods proposed to address forecasting of earthquakes are divided into probabilistic approaches, which intend to discover the distribution of seismicity, and the deep learning techniques, which obtain a learning model from time series data.

Among the first ones, the method presented and developed by the U. S. Geological Survey and the California Geological Survey is worth mentioning. This paper assumes that occurrence of earthquakes follows a Poisson probability distribution.

Kagan et al. presented their research based on smoothed seismicity to forecast earthquakes of magnitude greater than or equal to 5.0 in South California.

Another probabilistic approach based on strain rate was proposed by Shen et al. In particular, the strain rate that is apparently proportional to the rate of occurrence was observed and averaged over ten years of time.

A multilayer perceptron neural network has been proposed to forecast earthquakes from total electron content (TEC) time series . This model considers the variations of ionospheric variables which are believed to be the precursors of earthquakes.

Neural networks are a powerful tool to do a variety of tasks. Along with the network the GUTENBERG RICHTER inverse power law is taken as the basis of forecasting the earthquakes.

4.2 Gutenberg Richter Inverse Power Law

Gutenberg-Richter's law is often referred to as one of the three basic empirically established laws of seismology apart from Omori's and Bath's laws. The Gutenberg-Richter law describes the power law magnitude distribution of earthquakes in a defined region and time interval. It is observed that the number of earthquakes of magnitude M is proportional to 10^{-bM}. The law states that earthquake magnitudes are distributed exponentially as

$$\log_{10}N(m) = a - bm,$$

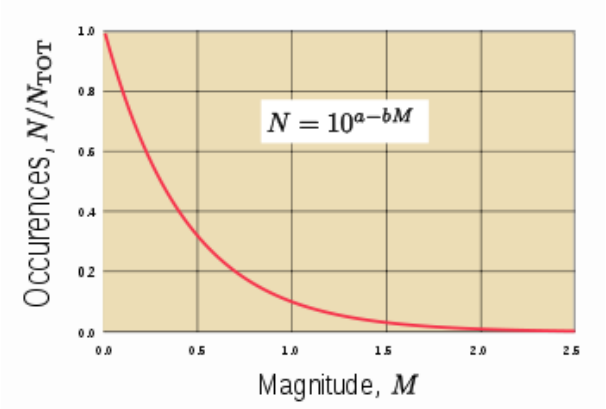
-where N(m) is the number of events with magnitude greater than or equal to m

-m is the earthquake magnitude

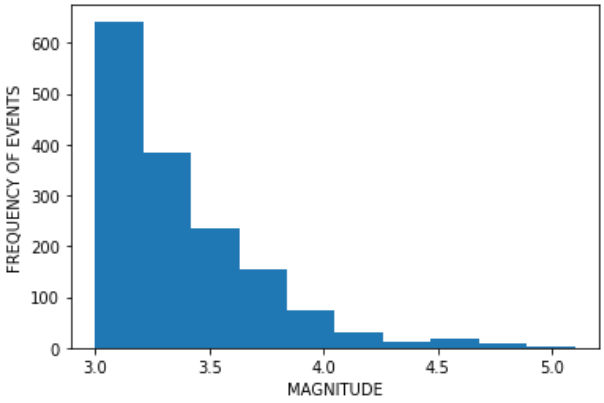
-a is called the "productivity"

-b is the slope of the line, called the "b-value", and is typically in the range of 0.8-1.1

`



Theoretical Gutenberg Richter Relationship



Actual Gutenberg Richter Relationship

4.3 Seismicity Indicators

We take 8 seismicity indicators as the input to our network as they capture a large portion of the seismic information of the earthquake event

i. The T value

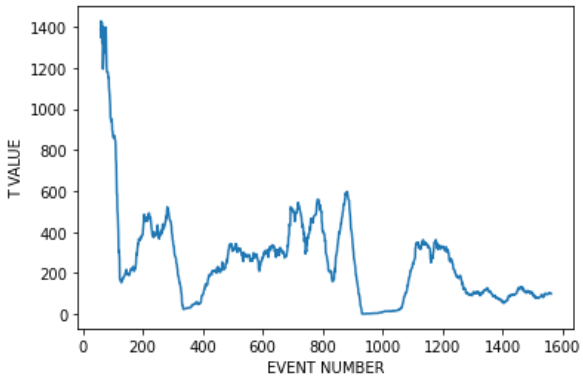
The time elapsed over the last n events of magnitude greater than a predefined threshold value is defined as $T=t_n - t_1$

t_n = Time of occurrence of the n^{th} event

t_1 = Time of occurrence of the 1^{st} event

A large T value indicates a lack of foreshocks which indicates a lower chance of occurrence of a forthcoming large seismic event. On the other hand, a small T value indicates a high foreshock frequency and a higher chance of occurrence of a forthcoming large seismic event.

`

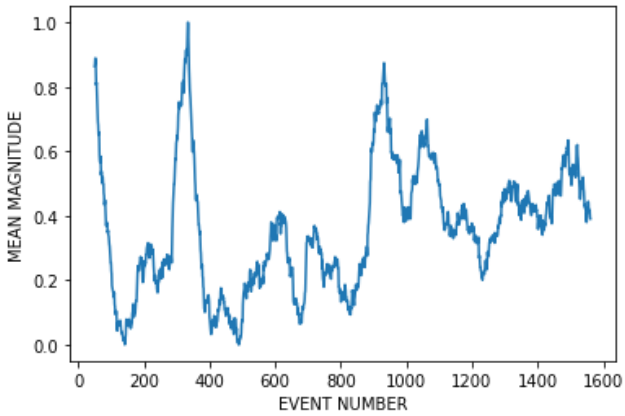


ii. The Mean Magnitude

The mean of the Richter magnitudes of the last n events is defined as

$$M_{mean} = \Sigma M_i / n$$

This is because the observed magnitudes of foreshocks increase immediately before the occurrence of a major earthquake. Therefore, a high mean magnitude means a high probability of a forthcoming seismic event.



iii. The rate of square root of seismic energy released

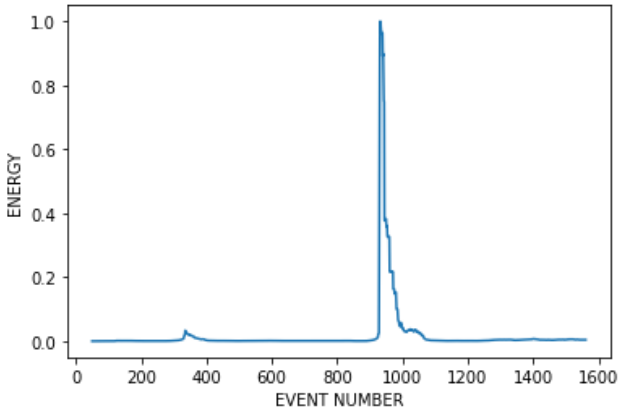
The rate of square root of seismic energy released over time T is defined as

$dE^{1/2} = \Sigma E^{1/2}/T$

$E = 10^{(11.8+1.5M)}$ ergs

Sometime the physical system accumulates energy that will be released abruptly in the form of a major seismic event when the stored energy reaches a threshold. A higher energy release suggests a forthcoming event.

`



iv. Slope of the log of the earthquake frequency versus magnitude curve (b value)

This parameter is based on the so-called Gutenberg-Richter inverse power law for earthquake magnitude and frequency which is expressed as

$$\log_{10}N = a - bM$$

N = number of events of magnitude greater than or equal to M

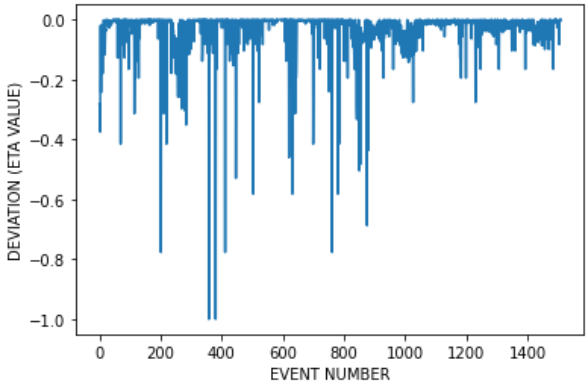
a and b are constants

v. Summation of the mean square deviation from the regression line based on the Gutenberg Richter inverse power law (η value)

This parameter is defined based on the Gutenberg-Richter magnitude-frequency relationship as follows:

$$\eta = \Sigma(\log_{10}N_i - (a - bM_i))^2 / (n - 1)$$

This is a measure of the deviation of the observed seismic data to the Gutenberg-Richter inverse power-law relationship. The lower the η value, the more likely that the observed distribution can be estimated using the inverse power law whereas a high η value indicates higher randomness and the inappropriateness of the power-law for describing the magnitude-frequency distribution.



`

vi. Magnitude deficit or the difference between the largest observed magnitude and the largest expected magnitude based on the Gutenberg Richter relationship (ΔM value)

$$\Delta M = M_{max,observed} - M_{max,expected}$$

$M_{max,observed}$ = the maximum observed magnitude in the last n events

$M_{max,expected}$ = the maximum magnitude in the last n events based on the inverse power-law relationship.

Since an event of the largest magnitude will likely occur only once among the n events, N = 1, log N = 0 we get

$$M_{max,expected} = a/b$$

vii. Mean time between characteristic or typical events (μ value)

This is the average time or gap observed between characteristic or typical events among the last n events. Characteristic events should ideally be separated by approximately equal time periods. The mean time μ is then given by

$$\mu = \Sigma(t_{i\ characterisic})/n_{characteristic}$$

viii. Coefficient of variation of the mean time between characteristic events (μ), also known as the aperiodicity of the mean (c value)

This parameter is a measure of the closeness of the magnitude distribution of the seismic region to the characteristic distribution and is defined mathematically as

c = standard deviation of the observed times/μ

A high c value indicates a large difference between the calculated mean time and the observed mean time between characteristic events and vice versa

`

5. Methodology

5.1 Reasoning for Using Specific Networks

The networks used have been found out after a lot of hit and trial and systematic reasoning. The code for training the networks is given in the appendix. The reasoning follows-

- If the training accuracy is high but the validation accuracy is low then it means that the model is overfitting, which means that the model is so complex that it is learning the training data extremely well. Hence it is overfitting to the training data and giving low accuracy on the validation data. In such a case decreasing the number of layers is the right approach. Regularization can also be used here. For example, while training the model for the Asian Dataset the training accuracy was 80% whereas the validation accuracy was around 60% which clearly suggested overfitting and hence, I decreased the number of layers and applied l2 regularization.
- If the training accuracy as well as the validation accuracy is low, then it means that the model is underfitting which suggests that the model is not complex enough to capture the original relationship between the input and output. In such a case increasing the complexity of the model by increasing the number of layers is the right approach.
- If the size of the dataset is very big then sometimes the total number of features can be less as compared to the size of the dataset. For example, using only 2 input features and training with 30,000-40,000 points. This can be adapted by splitting the data into random batches and then training them separately and ultimately assembling them or training a smaller subset of the dataset of random points(but we have to be careful while choosing the subset) once and then testing it on random batches of data.

5.2 Himalayan Region

5.2.1 Earthquake Catalogue

The historical earthquake data of India from the year 1980 to 2020 are collected from ISC (International Seismological Centre) Bulletin from the Himalayan region. The catalogue consists of 1562 events with the preliminary data of latitude, longitude etc as shown in table.

EVENTID	DATE	TIME	LAT	LONG	DEPTH	MAGNITUDE
589363	1982-11-21	08:10:21	28.8000	81.0500	0	3.6
495770	1986-05-16	05:16:13	34.0000	72.5800	15.0	4.0
490430	1986-07-10	07:56:12	34.1500	72.6900	2.0	4.5
.	.	.	.	.	.	.
.	.	.	.	.	.	.
617005962	2019-12-31	00:52:46.57	35.3951	74.8048	0.0	3.3
617439537	2019-12-31	04:27:55.79	35.5289	74.4619	0.0	3.1

`

5.2.2 Calculation of Seismic Indicators for Training

The Seismic indicators are mathematically calculated values from the earthquake catalogue using formulas given above. These inputs are normalized using the min-max scaling to achieve faster convergence of the model We have taken 8 seismic indicators as a part of this work.

INDICATOR	DESCRIPTION
T value	Time elapsed over last N events
M_{mean}	Mean of richter magnitudes of last N events
$dE^{1/2}$	Rate of square root of seismic energy released
b	Slope of the Gutenberg Richter Law
η	Deviation from Gutenberg Richter line
ΔM	Magnitude deficit
μ	Mean time between characteristic events
c	Coefficient of variation of mean time between typical events

5.2.3 Network Used

The neural network proposed is as defined in table. It is trained using 1132 data points i.e. earthquake events.

The purpose of the training is to get the loss value and accuracy of the prediction model of the given seismic indicators. Finally, the loss value of the prediction model is used to backpropagate the network and change the weights of the network according to the Adam optimizer. The loss is calculated through the Binary Cross Entropy function as we have only two classes.

TRAINING ARCHITECTURE	SIZE/ NUMBER OF NEURONS
Input Layer	8
ReLU Activation	-
Hidden Layer 1	24
ReLU Activation	-
Hidden Layer 2	24
ReLU Activation	-
Hidden Layer 3	12
Sigmoid Activation	-
Output Layer	1

`

The backpropagation algorithm used is Adam algorithm along with mini batch technique to get the maximum accuracy. The minibatch size used is 64 which is determined empirically and gives the maximum results.

The loss function used is the Binary Cross Entropy Loss which defines a loss values of the output class when there are two output class as is the case here.

The code for training the networks is given in the appendix.

OUTPUT

The output dataset consists of 1 for events which have an event of magnitude 4.3 or greater in the next 30 days otherwise 0.

5.3 Sumatra Region (Indonesia)

5.3.1 Dataset

The dataset of the Sumatra region consisted of 21858 data points and is taken from the year 1970 to 2020. These events contain preliminary information like latitude, longitude etc as shown in table.

DATE	TIME	LAT	LONG	DEPTH	MAGNITUDE
01-01-1970	03:29:17	-4.93	102.733	58	5.4
06-01-1970	23:26:47	-7.683	107.069	90	4.8
07-01-1970	13:53:52	-1.846	100.399	40	5.2
.	.	.	.	.	.
.	.	.	.	.	.
29-08-2020	05:35:40	-4.1794	101.1399	0	4.9
30-08-2020	18:27:53	-8.431	111.0805	0	4.2

5.3.2 Network Used

The neural network proposed is as defined in table. It is trained using 18,579 data points i.e. earthquake events.

The purpose of the training is to get the loss value and accuracy of the prediction model of the given seismic indicators. Finally, the loss value of the prediction model is used to backpropagate the network and change the weights of the network according to the Adam optimizer.

TRAINING ARCHITECTURE	SIZE/ NUMBER OF NEURONS
Input Layer	8

`

ReLU Activation	-
Hidden Layer 1	4
ReLU Activation	-
Hidden Layer 2	4
Sigmoid Activation	-
Output Layer	1

The loss is calculated through the Binary Cross Entropy function as we have only two classes. The backpropagation algorithm used is Adam algorithm along with mini batch technique to get the maximum accuracy. The minibatch size used is 128 which is determined empirically and gives the maximum results. A weight decay of 10^{-5} is added for the purpose of regularization.

The network was trained with 500 epochs to achieve a good accuracy. Its performance was evaluated on the predictions of the validation dataset which consists of 2786 earthquake events.

OUTPUT

The output dataset consists of 1 for events which have an event of magnitude 5.7 or greater in the next 30 days otherwise 0.

The code for training the networks is given in the appendix.

5.4 Central Java Region (Indonesia)

5.4.1 Dataset

The dataset of the Central Java region consisted of 15,682 data points and is taken from the year 19760 to 2020. These events contain preliminary information like latitude, longitude etc as shown in table.

DATE	LAT	LONG	DEPTH	MAGNITUDE
21-02-1963	-6.3	106.7	38	4.8
21-02-1963	-6.3	106.8	33	5.2
22-02-1963	-6.1	106.3	172	4.9
.	.	.	.	.
.	.	.	.	.
14-05-2020	-9.24	119.2	21	4.7
15-05-2020	-6.41	103.62	10	4.5

`

5.4.2 Network Used

The neural network proposed is as defined in table. It is trained using 13,329 data points i.e. earthquake events.

The purpose of the training is to get the loss value and accuracy of the prediction model of the given seismic indicators. Finally, the loss value of the prediction model is used to backpropagate the network and change the weights of the network according to the Adam optimizer.

TRAINING ARCHITECTURE	**SIZE/ NUMBER OF NEURONS**
Input Layer	8
ReLU Activation	-
Hidden Layer 1	24
ReLU Activation	-
Hidden Layer 2	24
ReLU Activation	-
Hidden Layer 3	12
Sigmoid Activation	-
Output Layer	1

The loss is calculated through the Binary Cross Entropy function as we have only two classes. The backpropagation algorithm used is Adam algorithm along with mini batch technique to get the maximum accuracy. The minibatch size used is 128 which is determined empirically and gives the maximum results. A weight decay of 10^{-5} is added for the purpose of regularization.

The network was trained with 200 epochs to achieve a good accuracy. Its performance was evaluated on the predictions of the validation dataset which consists of 2352 earthquake events.

OUTPUT

The output dataset consists of 1 for events which have an event of magnitude 5.7 or greater in the next 30 days otherwise 0.

The code for training the networks is given in the appendix.

5.5 Sulawesi Region (Indonesia)

5.5.1 Dataset

The dataset of the Sumatra region consisted of 34,628 data points and is taken from the year 1960 to 2020. These events contain preliminary information like latitude, longitude etc as shown in table.

`

DATE	LAT	LONG	DEPTH	MAGNITUDE
06-02-1963	0	124	139	4.2
10-02-1963	4.1	126.3	40	4.3
14-02-1963	-7.4	128.2	197	5.3
.	.	.	.	.
.	.	.	.	.
29-12-2019	4.67	125.45	22	4.8
29-12-2019	4.75	125.19	57	4.7

5.5.2 Network Used

The neural network proposed is as defined in table. It is trained using 26,883 data points i.e. earthquake events.

The purpose of the training is to get the loss value and accuracy of the prediction model of the given seismic indicators. Finally, the loss value of the prediction model is used to backpropagate the network and change the weights of the network according to the Adam optimizer.

TRAINING ARCHITECTURE	SIZE/ NUMBER OF NEURONS
Input Layer	8
ReLU Activation	-
Hidden Layer 1	14
ReLU Activation	-
Hidden Layer 2	12
Sigmoid Activation	-
Output Layer	1

The loss is calculated through the Binary Cross Entropy function as we have only two classes. The backpropagation algorithm used is Adam algorithm along with mini batch technique to get the maximum accuracy. The minibatch size used is 64 which is determined empirically and gives the maximum results. A weight decay of 10^{-5} is added for the purpose of regularization.

The network was trained with 500 epochs to achieve a good accuracy. Its performance was evaluated on the predictions of the validation dataset which consists of 5194 earthquake events.

OUTPUT

The output dataset consists of 1 for events which have an event of magnitude 5.7 or greater in the next 30 days otherwise 0.

`

The code for training the networks is given in the appendix.

5.6 Asian Region

5.6.1 Dataset

The dataset of the Sumatra region consisted of 65,336 data points and is taken from the year 1970 to 2020. These events contain preliminary information like latitude, longitude etc as shown in table.

DATE	TIME	LAT	LONG	DEPTH	MAGNITUDE
01-01-1970	29:16.8	-4.93	102.733	58	5.4
03-01-1970	51:48.6	-3.684	118.749	38	5.2
04-01-1970	00:40.2	24.139	102.503	31	7.5
.	.	.	.	.	.
.	.	.	.	.	.
30-06-2018	43:27.3	13.306	50.8974	12	4.7
30-06-2018	56:29.7	-1.1879	97.9378	20.3	4

5.6.2 Network Used

The neural network proposed is as defined in table. It is trained using 55,535 data points i.e. earthquake events.

The purpose of the training is to get the loss value and accuracy of the prediction model of the given seismic indicators. Finally, the loss value of the prediction model is used to backpropagate the network and change the weights of the network according to the Adam optimizer.

TRAINING ARCHITECTURE	SIZE/ NUMBER OF NEURONS
Input Layer	8
ReLU Activation	-
Hidden Layer 1	12
ReLU Activation	-
Hidden Layer 2	12
ReLU Activation	-
Hidden Layer 3	12
Sigmoid Activation	-
Output Layer	1

`

The loss is calculated through the Binary Cross Entropy function as we have only two classes. The backpropagation algorithm used is Adam algorithm along with mini batch technique to get the maximum accuracy. The minibatch size used is 256 which is determined empirically and gives the maximum results. A weight decay of 10^{-5} is added for the purpose of regularization.

The network was trained with 8000 epochs to achieve a good accuracy. Its performance was evaluated on the predictions of the validation dataset which consists of 9800 earthquake events.

OUTPUT

The output dataset consists of 1 for events which have an event of magnitude 5.7 or greater in the next 30 days otherwise 0.

The code for training the networks is given in the appendix.

6. Results

6.1 F1 Score

Accuracy can be used when the class distribution is similar while F1-score is a better metric when there are imbalanced classes as is the case here. In the case of earthquakes, the dataset is imbalanced as the number of high magnitude earthquakes is less and the number of low magnitude earthquakes is high. Hence this being an imbalanced dataset, as in most real-life applications, f1 score is a better metric.

The traditional F-score or balanced F-score (**F_1 score**) is the harmonic mean of the precision as well as recall-

$$\mathrm{F1} = \frac{\mathrm{TP}}{\mathrm{TP} + 0.5(\mathrm{FP} + \mathrm{FN})}$$

- TP is the true positives

- FP is the false positives

- FN is the false negatives

F1 SCORES OF THE DATASETS

REGION	F1 SCORE
Himalayan Region	0.8928571428
Sumatra Region	0.81291895185
Central Region	0.9011832427
Sulawesi Region	0.8692946058
Asian Region	0.8557046979

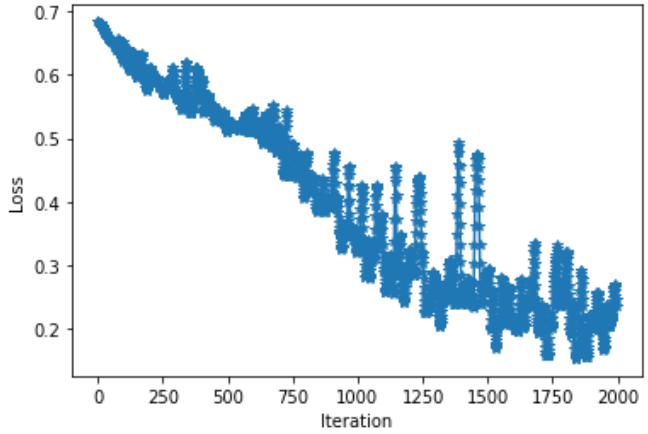
`

6.2 Learning Curve For The Models

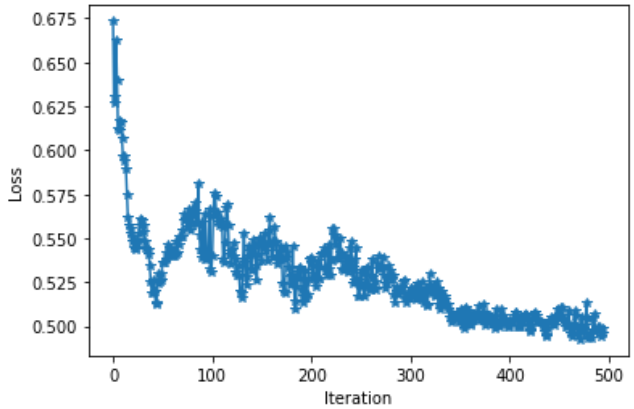
The models were trained using the Adam optimizer as it gives a faster and better convergence than the traditional stochastic gradient descent.

The following are the loss curves for all the model-

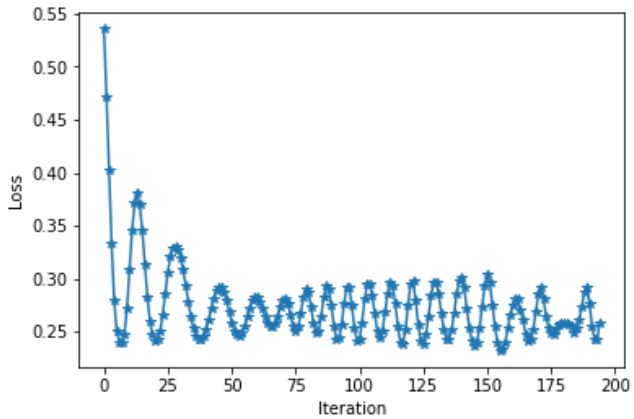
6.2.1 Himalayan Region



6.2.2 Sumatra Region (Indonesia)

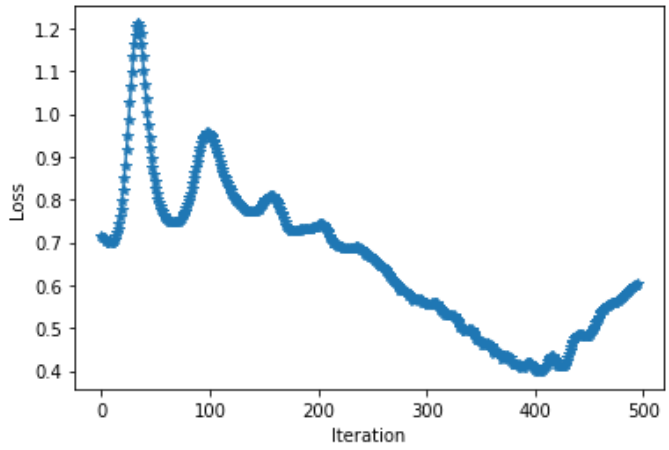


6.2.3 Central Java Region (Indonesia)

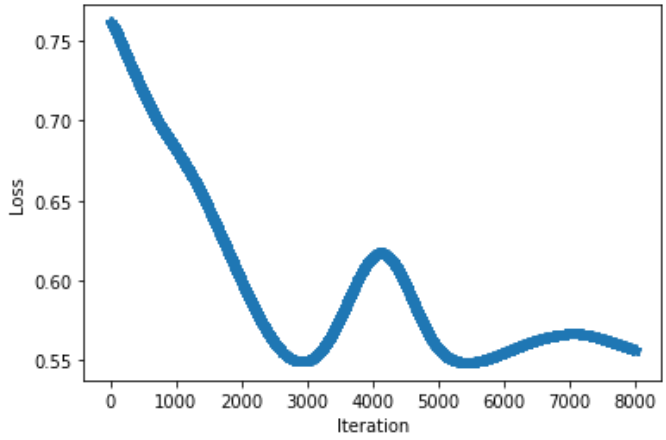


`

6.2.4 Sulawesi Region (Indonesia)



6.2.5 Asian Region



`

7. Conclusion

The prediction of earthquakes in the Himalayan region has been addressed in this paper by the means of neural networks. The Gutenberg Richter Law is a great tool to estimate the relationship between the number of earthquakes and their magnitudes. The 8 seismicity indicators have essentially captured most of the information of earthquake events. We can see that neural networks are an effective tool for forecasting earthquakes due to the universal approximation theorem which states that the neural networks can represent a wide variety of interesting functions when given appropriate weights.

The results achieved can be considered good since an f1 accuracy above 80% is achieved in all the datasets which shows that the method of neural networks has a high performance. Different neural networks are used for different dataset as their distributions are different.

However, a high imbalance can be seen in the datasets as earthquakes of higher magnitudes do not occur as frequently as those of lower magnitude. In this sense, the future work should be focussed on either using techniques to predict imbalanced dataset or use data augmentation techniques to generate artificial events to make the model more robust to higher magnitude earthquakes.

`

8. Appendix

HIMALAYAN DATA MODEL TRAINING

In [ ]:
```
class earthquake_model(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(8,24),
            nn.ReLU(),
            nn.Linear(24,24),
            nn.ReLU(),
            nn.Linear(24,12),
            nn.ReLU(),
            nn.Linear(12,1),
            nn.Sigmoid()
        )
    def forward(self,x):
        return self.net(x)
```

SPLIT DATA INTO TRAIN AND VALIDATION SETS

In [ ]:
```
arr_in = np.zeros((len(final_dataf),8))
arr_out = np.zeros((len(final_dataf),1))
for i in range(len(final_dataf)):
    arr_in[i] = np.array(final_dataf.iloc[i])
    arr_out[i] = dataf.iloc[i+N]["OUTPUT"]
```

In [ ]:
```
X_train,X_val,Y_train,Y_val = train_test_split(arr_in,arr_out,random_state=0)
```

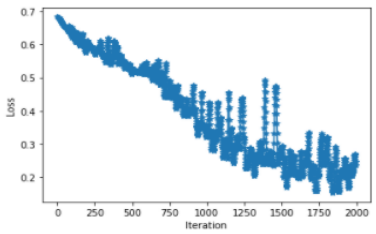
In [ ]:
```
X_train,X_val,Y_train,Y_val = map(torch.Tensor,(X_train,X_val,Y_train,Y_val))
```

TRAINING MODEL

In [ ]:
```
model = earthquake_model()
opt = optim.Adam(model.parameters(),lr=0.0001)
loss_fn = nn.BCELoss()
```

In [ ]:
```
l,bm = fit_optim(model,2000)
```

```
Iteration 1995 Loss 0.22672905027866364
```



In [ ]:
```
torch.save(bm2,'Earthquake_Model_4(90.4%)')
```

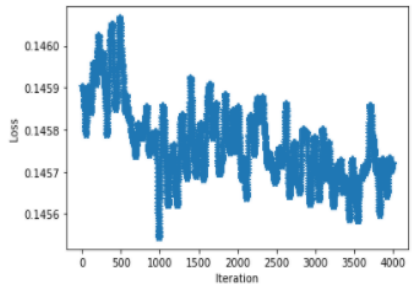
In [ ]:
```
saved_model = torch.load("Earthquake_Model_4(90.4%)")
```

In [ ]:
```
calc_metric(saved_model,X_val,Y_val)
```

```
The accuracy after 2000 epochs is  90.47619047619048
[[192  15]
 [ 21 150]]
0.8928571428571428
```

In [ ]:
```
#l2,bm2 = fit_optim(model,4000)
```

```
Iteration 3995 Loss 0.1457221359014511
```




```
The loss before training is0.14590245485305786
The loss after training is0.1457282155752182
```

`

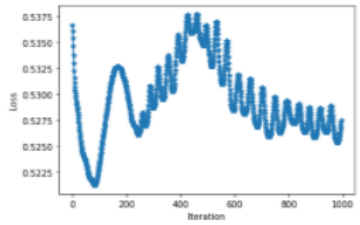
SUMATRA

In [89]:
```
class earthquake_model_sumatra(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(8,4),
            nn.ReLU(),
            nn.Linear(4,4),
            nn.ReLU(),
            nn.Linear(4,1),
            nn.Sigmoid()
        )
    def forward(self,x):
        return self.net(x)
```

In [90]:
```
model_sumatra = earthquake_model_sumatra()
```

In [93]:
```
l_sumatra,b_sumatra = fit_optim(model_sumatra,X_train_sumatra,Y_train_sumatra,1000,0.0001,128)
```

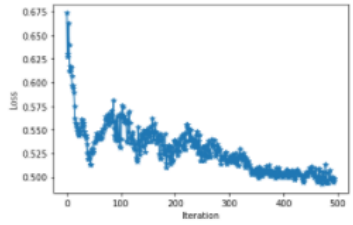
```
Iteration 995 Loss 0.5277016162872314
```




```
The loss before training is0.53666752576828
The loss after training istensor(0.4502, grad_fn=<BinaryCrossEntropyBackward>)
```

In [ ]:
```
l_sumatra,b_sumatra = fit_optim(model_sumatra,X_train_sumatra,Y_train_sumatra,500,0.0001,128)
```

```
Iteration 495 Loss 0.5033876895904541
```




```
The loss before training is0.6733927130699158
The loss after training istensor(0.4289, grad_fn=<BinaryCrossEntropyBackward>)
```

In [94]:
```
calc_metric2(b_sumatra,X_val_sumatra,Y_val_sumatra)
```

```
The f1 accuracy is 0.8129189518586227
```

In [100]:
```
# torch.save(b_sumatra,"sumatra_model_3(f81%)")
```

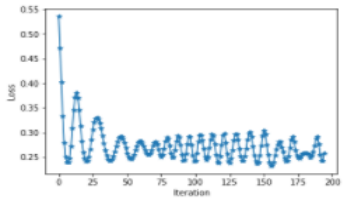
CENTRAL

In [ ]:
```
class earthquake_model_central(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(8,24),
            nn.ReLU(),
            nn.Linear(24,24),
            nn.ReLU(),
            nn.Linear(24,12),
            nn.ReLU(),
            nn.Linear(12,1),
            nn.Sigmoid()
        )
    def forward(self,x):
        return self.net(x)
```

In [ ]:
```
model_central = earthquake_model_central()
```

In [ ]:
```
l_central,b_central = fit_optim(model_central,X_train_central,Y_train_central,200,0.0001,64)
```

```
Iteration 195 Loss 0.2824236750602722
```




```
The loss before training is0.5357078909873962
The loss after training istensor(0.1993, grad_fn=<BinaryCrossEntropyBackward>)
```

In [ ]:
```
calc_metric(b_central,X_val_central,Y_val_central)
```

```
The accuracy is  82.01396973224679
The confusion matrix is
 [[   0  309]
 [   0 1409]]
The f1 score is 0.9011832427246562
```

In [88]:
```
calc_metric2(b_central,X_val_central,Y_val_central)
```

```
The f1 accuracy is 0.9011832427246562
```

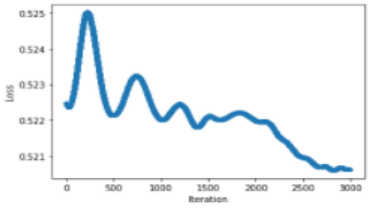
In [ ]:
```
#torch.save(b_central,"central_model_1(82%)")
```

`

SULAWESI

In [ ]:
```
l_sulawesi,b_sulawesi = fit_optim(model_sulawesi,X_train_sulawesi,Y_train_sulawesi,3000,0.00001,4096)
```

```
Iteration 2995 Loss 0.52061527967453
```




```
The loss before training is0.522469162940979
The loss after training istensor(0.5006, grad_fn=<BinaryCrossEntropyBackward>)
```

In [ ]:
```
calc_metric(b_sulawesi,X_val_sulawesi,Y_val_sulawesi)
```

```
The accuracy is  74.20072608033728
The confusion matrix is
 [[2513 1314]
 [ 889 3823]]
The f1 score is 0.776322469286222
```

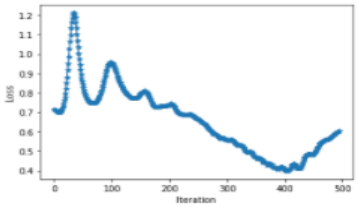
In [ ]:
```
torch.save(b_sulawesi,"sulawesi_model_1(74%)")
```

In [95]:
```
class earthquake_model_sulawesi(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(8,14),
            nn.ReLU(),
            nn.Linear(14,12),
            nn.ReLU(),
            nn.Linear(12,1),
            nn.Sigmoid()
        )
    def forward(self,x):
        return self.net(x)
```

In [96]:
```
model_sulawesi = earthquake_model_sulawesi()
```

In [97]:
```
l_sulawesi,b_sulawesi = fit_optim(model_sulawesi,X_train_sulawesi,Y_train_sulawesi,500,0.0001,64)
```

```
Iteration 495 Loss 0.6037675738334656
```




```
The loss before training is0.7128465175628662
The loss after training istensor(0.3021, grad_fn=<BinaryCrossEntropyBackward>)
```

In [98]:
```
calc_metric(b_sulawesi,X_val_sulawesi,Y_val_sulawesi)
```

```
The accuracy is  76.88073394495413
The confusion matrix is
 [[  0 250]
 [  2 838]]
The f1 score is 0.8692946058091287
```

In [99]:
```
calc_metric2(b_sulawesi,X_val_sulawesi,Y_val_sulawesi)
```

```
The f1 accuracy is 0.8692946058091287
```

`

9. References

1. Ashif Panakkat and Hojjat Adeli, “Neural Network Models For Earthquake Magnitude Prediction Using Multiple Seismicity Indicators”, International Journal of Neural Systems, Vol. 17, No. 1 (2007) 13–33
2. Asencio-Corte, G.; Martı´nez- A´lvarez, F., Troncoso, A., Morales-Esteban, A.(2015), “Medium–large earthquake magnitude prediction in Tokyo with artificial neural networks”.Springer Neural computing and Applications, 28(5), pp.1043-1055.
3. Kagan YY, Jackson DD, Rong Y (2007) A testable five-year forecast of moderate and large earthquakes in southern California based on smoothed seismicity. Seismol Res Lett 78(1):94–98
4. Petersen MD, Cao T, Campbell KW, Frankel AD (2007) Timeindependent and time-dependent seismic hazard assessment for the state of California: uniform California earthquake rupture forecast model 1.0. Seismol Res Lett 78(1):99–109
5. Shen ZZ, Jackson DD, Kagan YY (2007) Implications of geodetic strain rate for future earthquakes, with a five-year forecast of M5 earthquakes in southern California. Seismol Res Lett 78(1):116–120
6. Azam F, Sharif M, Yasmin M, Mohsin S (2014) Artificial intelligence based techniques for earthquake prediction: a review. Sci Int 26(4):1495–1502
7. Asim, K. M., Martínez-Álvarez, F., Basit, A., Iqbal, T. (2017), “Earthquake magnitude prediction in Hindukush region using machine learning techniques”. Springer Natural Hazards, 85(1), pp.471-486.

`