

CS 6190: Probabilistic Machine Learning Spring 2023

Midterm Report

Handed out: 21 Feb, 2023
Due: 11:59pm, 21 March, 2023

1. Introduction

The midterm report is about earthquake prediction. I am going to use the dataset from Kaggle and knowledge from the class to get the prediction. The problem is the earthquake magnitude prediction from given certain conditions. For example, by giving some observations and assumptions, we can know the degree of the next earthquake.

2. Motivation

Many countries are engaging in earthquake prediction. However, current earthquake prediction systems can only warn people a few seconds earlier than the catastrophe. Though the short period gives people time to escape or find a place to shield themselves, it seems insufficient to protect everyone living in the high-frequent earthquake region. For example, Turkey happened several significant earthquakes this year, but it is not the first nor the last disaster there. Everyone knows there are following earthquakes in the future, but no one knows how severe they will be.

Learning techniques, compared with the current system giving people only seconds to run, can give people sufficient time slots to prepare for the next disaster.

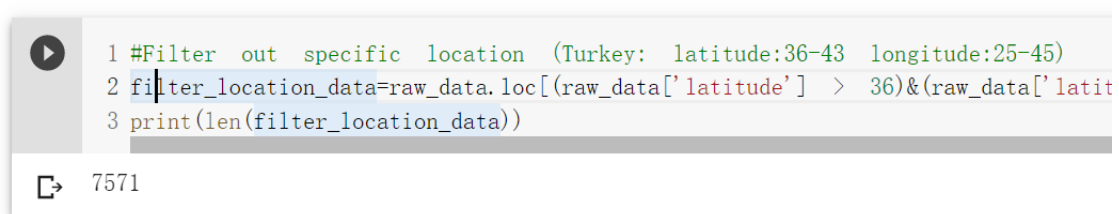
3. Works

I have done data analysis, training a model with bayesian linear regression, scoring by mean square error, and comparing it with sklearn linear model.

There are more than six hundred thousand rows in the data set (fig.1), but I only focus on the specific region, Turkey. Because the data set includes the earthquake this year. The data amount goes to 7571 after filtering by location. (fig.2)

```
Index(['Unnamed: 0', 'time', 'latitude', 'longitude', 'depth', 'mag',  
      'magType', 'nst', 'gap', 'dmin', 'rms', 'net', 'id', 'updated', 'place',  
      'type', 'horizontalError', 'depthError', 'magError', 'magNst', 'status',  
      'locationSource', 'magSource'],  
      dtype='object')  
613787
```

Figure 1: All columns



```
1 #Filter out specific location (Turkey: latitude:36-43 longitude:25-45)  
2 filter_location_data=raw_data.loc[(raw_data['latitude'] > 36)&(raw_data['latit  
3 print(len(filter_location_data))
```

7571

Figure 2: Filter Turkey

There are 23 columns in the data overall(fig.1). Because of time reasons and some columns needing data processing, I only used 9 as my input data and mag as my output data. Inputs include depth, magType, gap, dmin, rms, horizontalError, depthError, magError and magNst. I analyzed these columns and magnitudes by covariance and correlation to view their relationship. Some included missing data, so I padded the data(e.g., max/minimum/median) to them. It can be told that besides the depth, the rest of column was padded.(fig.3) Because this lecture is not about data processing or analysis, let's pretend the padding does not affect the training.

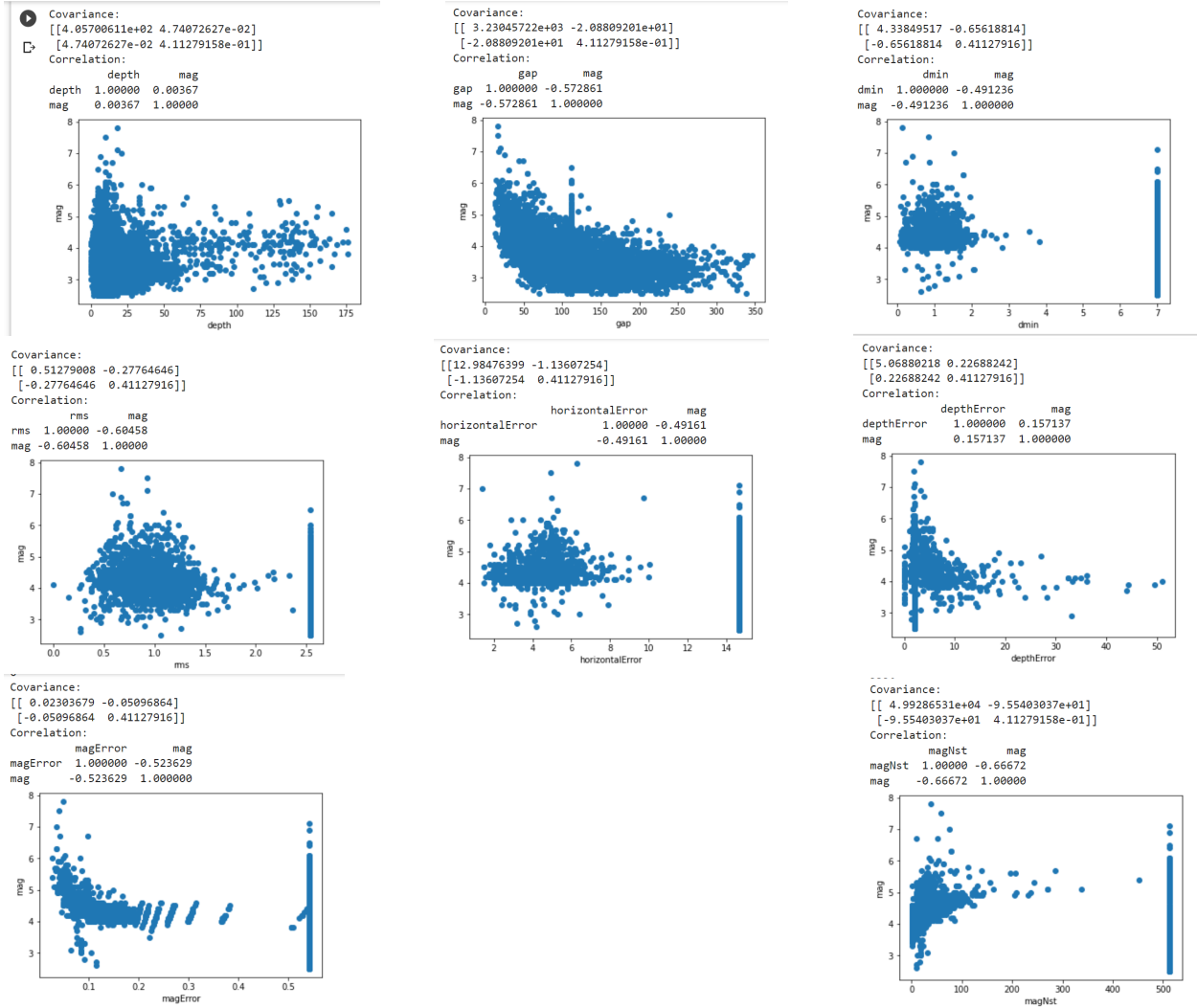


Figure 3: Correlation

Afterward, I use bayesian linear regression to train my model.

$f(x, a) : a[0]*x[0] + a[1]*x[1] + a[2]*x[2] + a[3]*x[3] + a[4]*x[4] + a[5]*x[5] + a[6]*x[6] + a[7]*x[7] + a[8]$
The a here are the weights and x are inputs. I then use the parameter used in HW2, but the result is terrible.(shown in fig. 4 left side) It is not converged and has a bad prediction. After tuning the parameter, alpha and beta, the result seems more reasonable.(shown in fig. 4 right side) Then, I need a score function to evaluate the model. Here I use mean square error function, and the result shown as fig. 5. The result seems not satisfying, because 9.4 is way too high. My expectation is less than 1. Then, I use sklearn linear model to confirm if the bad performance is applied to every linear model. From the result shown in fig.6, it seems like my model has a significant improvement space. I think

by tuning the parameters, it will meet the same performance as regular linear model.

<pre>No convergence alpha: 2 beta 25.0 After 6812 iterations mN: [[-5.17634161e+07] [7.01411965e+07] [-6.63853137e+07] [-7.26386788e+07] [-5.72095521e+07] [-7.09027362e+07] [-7.36887139e+07] [3.96124383e+08] [-7.36771682e+07]] Predict: [1.52454175e+10] Ground Truth: 3.7</pre>	<hr/> <pre>Break for convergence. alpha: 0.0001 beta 0.0007 After 1220 iterations mN: [[-0.01246334] [0.01623005] [-0.01602455] [-0.01750595] [-0.01371692] [-0.0174646] [-0.01797261] [0.0977219] [-0.01783445]] Predict: [3.6563896] Ground Truth: 3.7</pre>
--	--

Figure 4: Baysien linear regression

```
Baysian linear Mean Square Error:
9.445365719225485
```

Figure 5: Mean square error result

```
Regular linear Mean Square Error:
0.21100232341172254
slope: [ 8.14561606e-06 -3.47080350e-03  9.15624033e-03 -2.59369422e-01
 1.81207083e-02 -8.02094786e-03 -3.16099147e-01 -1.04114138e-03]
```

Figure 6: Regular linear result

4. Following plan

I believe the model can be better after fine-tuning. Hence, tuning the parameters is not my goal. Instead, **using other distributions to see the difference** is my goal. Bayesian linear model is based on Gaussian distribution, and I am curious to use other distributions, i.e., Wishart and student t, to see their properties in these scenarios. Afterward, I will **analyze the output from different models** rather than only mean square error. For example, student t can avoid the noise effect, and let see if I can see it in this project.

5. Reference to literature

- Kaggle data set: <https://www.kaggle.com/datasets/grigol1/earthquakes-2000-2023>
- Columans meaning: <https://earthquake.usgs.gov/data/comcat/data-eventterms.php>
- Baysien linear learning: <https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf>
- Regular linear learning: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html