Imperial College London

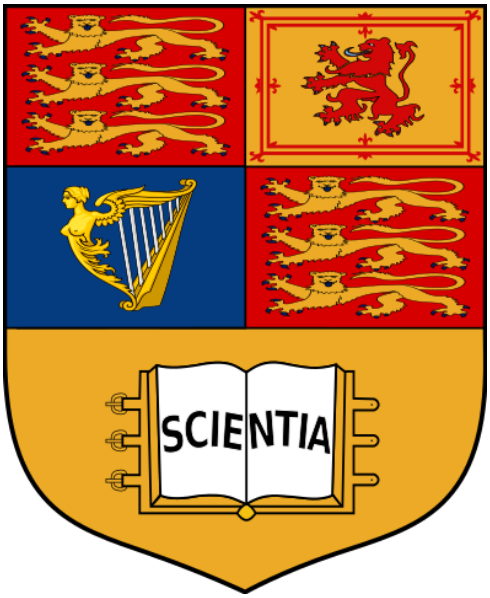
Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Forecasting induced seismicity in Oklahoma

Author:
Zhiyong Liu
zl1220@ic.ac.uk
Github: acse-liuzyon

Supervisor:
Dr. Stephen P. Hicks
s.hicks@imperial.ac.uk



July 2021

GitHub repository: https://github.com/acse-2020/acse2020-acse9-finalreport-liuzyon

Abstract

Human activities can cause minor earthquakes, which changes the stresses and strains on the earth's crust. These earthquakes are called induced earthquakes and most have a low magnitude. Multi-year research by the United States Geological Survey (USGS) published in 2015 showed that most of large earthquakes occured in Oklahoma may have been caused by deep injection of wastewater from the oil industry. In this project, a unique and rich dataset of industrial activities from regions in Oklahoma are used and correlations between these activities and seismicity are confirmed. With existing high-quality earthquake catalogues in these regions, possible signatures of human-induced seismicity are retrospectively forecasted through machine learning techniques. At the end of the project, a working forecasting model is generated and tested the permormance by the test dataset splitted from total dataset.

Key Words: induced earthquakes, machine learning method, forecasting model

1 Introduction

It is well known that humans can cause earthquakes through fluid injection and extraction. Ellsworth stated the understanding of the causes and mechanics of human-induced earthquakes. It includes wastewater injection, emerging oil and gas recovery technologies, and other indirect induced activities, including deep fluid injection (Ellsworth 2013). Norbeck and Rubinstein genreated a model based on fluid flow and seismic physics, which associate the past injection trends with the seismicity patterns (Norbeck & Rubinstein 2018). Such cases of induced seismicity have been recorded and proven in Oklahoma, where seismicity has been increased dramatically since 2010. In cases like Oklahoma, because the rate of earthquakes was very low before high-rate wastewater injection, it is relatively easy to determine the fundamental causes of induced seismicity. Therefore, due to the low background stress in intraplate regions, the human triggering signatures are clearly identifiable. However, in tectonically-active areas, it is more challenging to distinfuish between natural and triggered seismicity. In tectonically active regions in the US, such as California, although oil and gas extraction has taken place for many decades, only a handful of studies have been published with a focus on these areas (Hough et al. 2017). Big-data and statistical approaches will be crucial in separating natural causes from potential human triggering factors in these areas. In regions from Oklahoma, Hincks et al. developed a Bayesian network to for quantitative evaulation of correlations between well operational parameters, geological formation, and seismicity. The injection depth relative to crystalline basement was found to be the most significant parameter for seismic moment release (Hincks et al. 2018). Also, Wozniakowska and Eaton performed a machine learning estimation of the seismogenic activation potential for each well using logistic regression and also found that the injection depth influence greatly the seismogenic activation potential (Wozniakowska & Eaton 2020).

The correlations of induced earthquakes with fluid injection activities are well established, even in previous research some physics models are also developed for earthquake prediction. It is still not known why some areas are more susceptible to induced earthquakes. In this project, Several unique and rich datasets of industrial activities from regions in Oklahoma are used to statistically evaluate and retropectively forecast possible signatures of human-induced seismicity from existing high-quality earthquake in these regions. These datasets contains geological formation, injections, hydraulic fracturing activities and wells, which all have millions of items. They were selected subset from 2011 to 2018 and displayed on the local map of Oklahoma. Rough visual correlations between the earthquakes and these potential induced factors were discovered. Then, stepwise regression method was used to choose statistically significant features from all features. A logistic model was fitted before and after stepwise regression, which proved that the work of features selection indeed benefit to induced forecasting.

In previous studies for induced earthquakes in Oklahoma, forecasting works were almost always implemented using regression models. Compared to them, after selecting statistically significant features, this project originally designed and implemented a neural network for induced seismicity forecasting, which effectively improves the performance of model fitting and prediction.

2 Software Description

2.1 Visualization: Matplotlib & GeoPandas

GeoPandas can help researcher make working with geospatial data in Python easier. It extends the datatypes in Pandas, which allows users perform spatial operations on geometric types. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. In the initial phase of project, we visualized the geological formation (depth to basement) and the various activities (seismicity, injection, hydraulic fracturing) in interest area on the Oklahoma map by

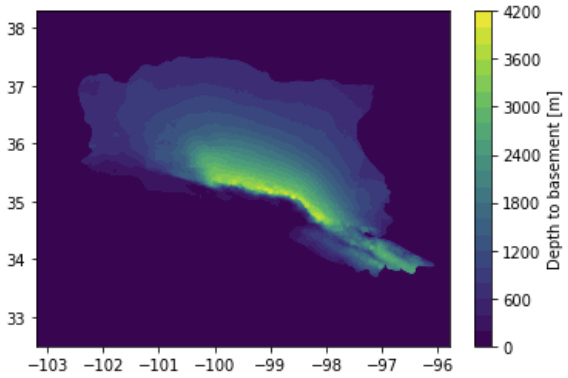


Figure 1: Depth to basement in Oklahoma.

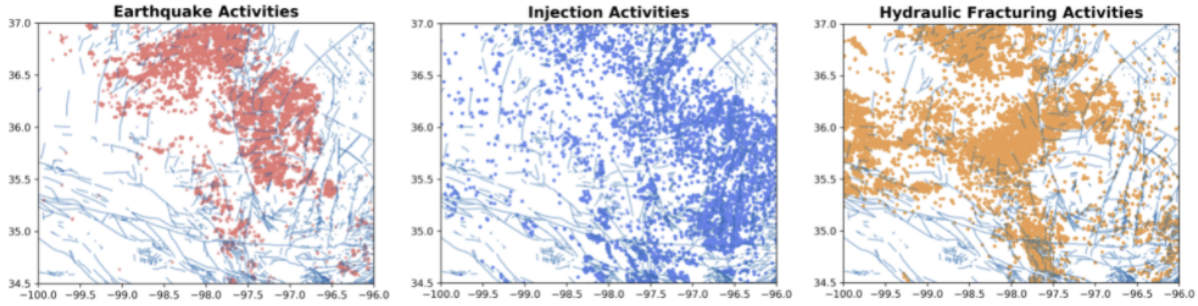


Figure 2: Activities during 2011 to 2018 in interest area from Oklahoma.

GeoPandas and Matplotlib as shown in Fig 1 and Fig 2. By comparing these pictures, we found a rough spatial correlation between them.

2.2 Stepwise Regression

In any phenomenon, some factors will play a more important role in determining the outcome. Stepwise Regression is method which is widely used to find which factors are important and which are not. The factors that have a rather high p-value do not make a meaningful contribution to the accuracy of our predictions. Therefore, only the important factors are retained to ensure that the model make predictions based on the factors that will help it produce the most accurate results.

P-values are often used to see if the patterns they measured were statistically significant. If the p-value of a statistical test is small enough, we are able to reject the null hpytothesis of the test and the pattern measured is statistically significant. The most common threshold for p is 0.05. For the variable whose p-value is greater than 0.05, we can assume that it is not statistically significant as shown in Fig 3. Here is step in stepwise logistic regression process. The 'injection_vol', 'HF_Base_Water_Volume', 'HF_Base_NoWater_Volume' both have a p-value than threshold 0.05 in logistic regression results, which are not statistically significant and should be eliminated in later process.

In this project, we performed a stepwise logistic regression that used a backwards elimination approach on all interest potential factors (Ryan 2020). All factors were initially included and the least statistically significant factor is eliminated in each step. In other words, the most 'useless' factor is dropped. This process continued untill all factors kept are statistically important.

```
Optimization terminated successfully.
         Current function value: 0.631770
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                  class   No. Observations:                 1600
Model:                          Logit   Df Residuals:                     1593
Method:                           MLE   Df Model:                            6
Date:                Wed, 28 Jul 2021   Pseudo R-squ.:                0.04317
Time:                        17:38:31   Log-Likelihood:                -1010.8
converged:                       True   LL-Null:                       -1056.4
Covariance Type:            nonrobust   LLR p-value:                 1.699e-17
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
injection_vol    0.1912      0.119      1.609      0.108      -0.042       0.424
injection_psi   -0.6618      0.206     -3.219      0.001      -1.065      -0.259
injection_depth_sum    0.3869      0.130      2.976      0.003       0.132       0.642
depth_to_basement   -0.4143      0.059     -7.030      0.000      -0.530      -0.299
HF_number        0.7876      0.149      5.299      0.000       0.496       1.079
HF_Base_Water_Volume   -0.0275      0.135     -0.204      0.838      -0.292       0.237
HF_Base_NoWater_Volume   -0.1966      0.119     -1.659      0.097      -0.429       0.036
==============================================================================
```

Figure 3: Activities during 2011 to 2018 in interest area from Oklahoma.

2.3 Neural Network Construction: Pytorch

Pytorch is an open source machine learning library for Python and is completely based on Torch library, used for applications such as computer vision and natural language processing. It is mostly developed by Facebook's AI Research lab (FAIR) (Patel 2018). PyTorch provides two features:

- Tensor computing (like Numpy) with strong acceleration via graphics processing units (GPU)
- Deep neural networks built on a type-based automatic differentiation system.

Pytorch provides a class called Tensor (torch.Tensor) to store and operate on homogeneous multidimensional rectangular arrays of numbers. They are tailored for datasets in machine learning. Tensors in Pytorch are similar to Numpy Arrays, but it supports to be operated on a CUDA-capable Nvidia GPU (Paszke et al. 2019).

To define a neural network in Pytorch, it is pretty convenient to create a class that inherits from nn.Module as below:

```
# Get cpu or gpu device for training.
device = "cuda" if torch.cuda.is_available() else "cpu"
print("Using {} device".format(device))

# Define model
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(4, 8),
            nn.ReLU(),
            nn.Linear(8, 8),
            nn.ReLU(),
            nn.Linear(8, 2),
            nn.ReLU()
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits

model = NeuralNetwork().to(device)
```

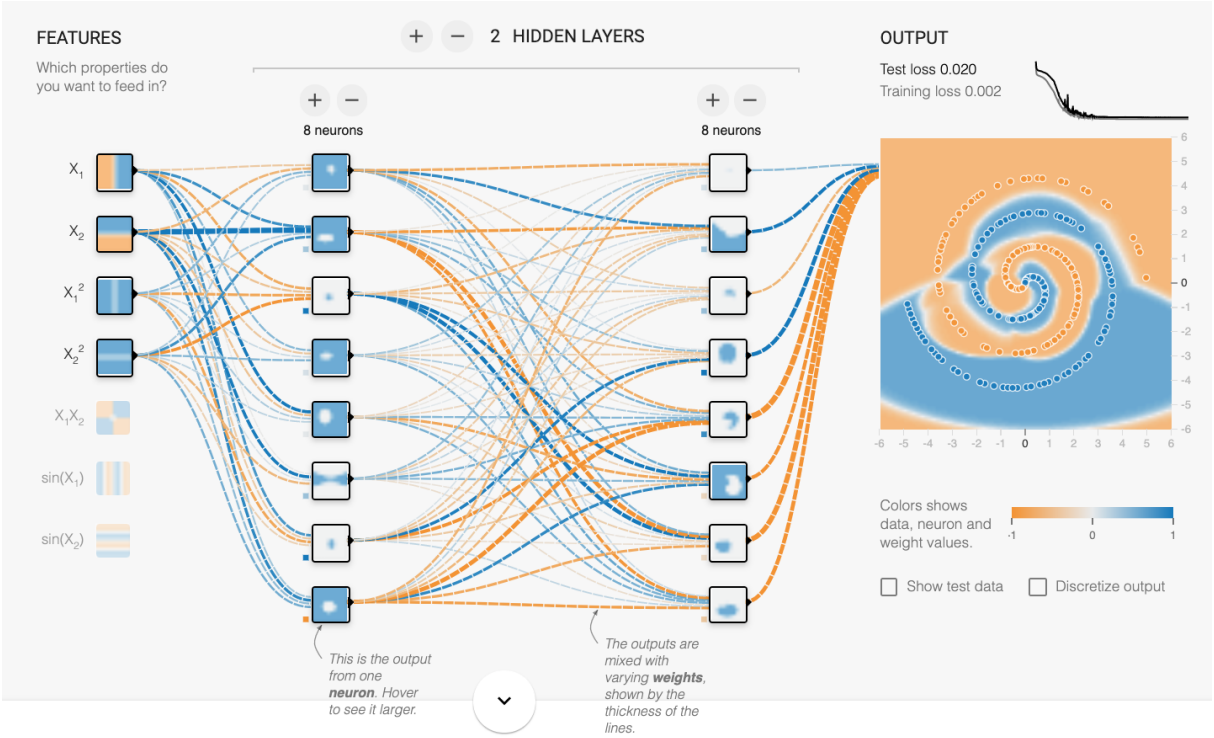


Figure 4: The model created simulated in playground.tensorflow.org

User can define the layers of the network in the *__init__* function and indicates how data will pass through the network in the *forward* function. With Pytorch, user can move it to the GPU to accelate operations in the neural netwok if available. Figure 4 is the model created above trained on the playground online through Tensorflow. The neural network prediction model used in this project is implemented by this way.

2.4 Feature Attribution: Captum

Feature attribution is the technique to investgiate how much each feature in the model contributes to the prediction for given dataset fitting. With the increase in model complexity and the resulting lack of transparency, model interpretability methods have become increasingly important in machine learning field. In this project, we implemented a neural network model with the PyTorch library (PyTorch 2021) for improving the accuracy of prediction. Therefore, we use the Captum for feature attribution in neural network model, which is a model interpretability and understanding library for Pytorch. It includes general purpose implementations of integrated gradients, saliency maps, smoothgrad, vargrad and others for PyTorch models.

Captum provides advanced algorithms, including Intergarated gradientes used in this project, to provides users an simple way to understand which features are contributing to a models' output. Integrated gradients represents the integral of gradients with respect to inputs along the path from a given baseline to input. The integral can be approximated using a Riemann Sum or Gauss Legendre quadrature rule. Formally, it can be described as follows in Equation (1):

$$IntegratedGrads_i(x) ::= (x_i - x_i^{'}) \times \int_{\alpha=0}^{1} \frac{\partial F(x^{'} + \alpha \times (x - x^{'}))}{\partial x_i} d\alpha \tag{1}$$

Intergarated Gradients along the i-th dimension of input X. Alpha is the scaling coffecient (Sundararajan

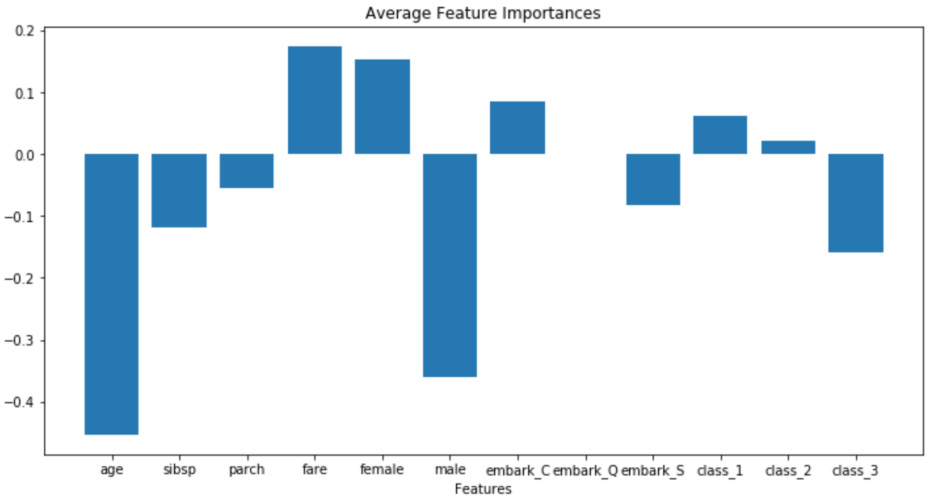


Figure 5: Feature Importance Example in Captum Tutorial.

et al. 2017).

As shown in Fig 5, this is the result of feature importance analysis for the model trained in official tutorial of Captum. The official tutorial used an example model trained on the titanic survival data. It first trained a deep neural network on the data using Pytorch and used Captum to understand which of the features were most important and how the network reached its prediction. The features involved Age, Sibsp, Parch, Fare, Gender, Embark, Class and the target is a binary indicating whether passager is survived. After normalization and one-hot encodings, the neural network architecture was defined and trained. The neural network had a simple architecture using 2 hidden layers, the first with 12 hidden units and the second with 8 hidden units, each with Sigmoid non-linearity. The final layer performed a softmax operation and had 2 units, corresponding to the outputs of either survived or not survived. The accuracy of this model on dataset achieved to 81.6%. To perform feature attribution on this model, this example applied integrated Gradients, which is one of the Feature Attribution methods included in Captum. The feature attribution method code is shown in below.

```
    ig = IntegratedGradients(model)
    test_input_tensor.requires_grad_()
    attr, delta = ig.attribute(test_input_tensor,target=1,
    return_convergence_delta=True)
    attr = attr.detach().numpy()

    # Helper method to print importances and visualize distribution
    def visualize_importances(feature_names, importances, title="Average Feature
    Importances", plot=True, axis_title="Features"):
        print(title)
        for i in range(len(feature_names)):
            print(feature_names[i], ": ", '%.3f'%(importances[i]))
        x_pos = (np.arange(len(feature_names)))
        if plot:
            plt.figure(figsize=(12,6))
            plt.bar(x_pos, importances, align='center')
            plt.xticks(x_pos, feature_names, wrap=True)
            plt.xlabel(axis_title)
            plt.title(title)
    visualize_importances(feature_names, np.mean(attr, axis=0))
```

The result of feature attribution was listed in Table 1 and shown in Fig 5. From the feature attribution information, we obtain some interesting insights regarding the importance of various features. We see that the strongest features appear to be age and being male, which are negatively correlated with

Table 1: Average Feature Importances for each feature

age	sibsp	parch	fare	male	embark_C	embark_Q	embark_S	class_1	class_2	class_3
-0.454	-0.119	-0.056	0.175	-0.359	0.086	-0.001	-0.082	0.062	0.021	-0.159

survival. Embarking at Queenstown and the number of parents / children appear to be less important features generally.

References

Ellsworth, W. L. (2013), 'Injection-induced earthquakes', *Science* **341**(6142).

Hincks, T., Aspinall, W., Cooke, R. & Gernon, T. (2018), 'Oklahoma's induced seismicity strongly linked to wastewater injection depth', *Science* **359**(6381), 1251–1255.

Hough, S., Tsai, V., Walker, R. & Aminzadeh, F. (2017), 'Was the mw 7.5 1952 kern county, california, earthquake induced (or triggered)?', *Journal of Seismology* **21**, 1613 – 1621.

Norbeck, J. & Rubinstein, J. L. (2018), 'Hydromechanical earthquake nucleation model forecasts onset, peak, and falling rates of induced seismicity in oklahoma and kansas', *Geophysical Research Letters* **45**(7), 2963–2975.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. et al. (2019), 'Pytorch: An imperative style, high-performance deep learning library', *Advances in neural information processing systems* **32**, 8026–8037.

Patel, M. (2018), 'When two trends fuse: Pytorch and recommender systems'.

PyTorch (2021), 'Stepwise regression tutorial in python'.
URL: *https://pytorch.org/docs/stable/index.html*

Ryan, K. (2020), 'Stepwise regression tutorial in python'.
URL: *https://towardsdatascience.com/stepwise-regression-tutorial-in-python-ebf7c782c922*

Sundararajan, M., Taly, A. & Yan, Q. (2017), Axiomatic attribution for deep networks, *in* 'International Conference on Machine Learning', PMLR, pp. 3319–3328.

Wozniakowska, P. & Eaton, D. W. (2020), 'Machine learning-based analysis of geological susceptibility to induced seismicity in the montney formation, canada', *Geophysical Research Letters* **47**(22), e2020GL089651.