Predicting Bridge Performance under Earthquakes with Supervised Learning

Ziyang Jiang (zij004@stanford.edu), Xiao Zhang (zhx@stanford.edu)

Introduction

Since the 1970s, the seismic bridge design process has gone through a great change from "capacity-based" design to "performance-based" design, and the performance of bridges has become a great concern to engineers. The safety margins of the contributive factors vary from case to case, and the trends are still unclear, partially, if not all, because of the change in the design logics in the past decades. Therefore, having an on-hand trained model on bridge performance prediction would be to some extent helpful for knowing how well/badly an existing bridge would perform in a future earthquake as well as guiding the design of a new bridge to survive a future earthquake.

In this project, we are trying to train a prediction model for bridge performance under earthquakes with supervised learning. The inputs to our algorithm are the age of a bridge, the magnitude of the earthquake, and the distance between that bridge and the epicenter. We then use Logistic Regression, Quadratic Discriminative Analysis, and K-Nearest Neighbor Classifier to output a predicted performance, categorized to be positive (damaged) or negative (undamaged), of the bridge under the given earthquake.

Related Work

Prof. Kiremidjian's paper greatly inspires our interest on data-driven earthquake engineering. When trying to select the related features, we refer to a comprehensive study of U.S. bridge failures from MCEER technical report. We also learn the advantages of bootstrap resampling in managing unbalanced data from Dupret and Koda's paper.

The problem we encountered is actually a sub-problem of other projects on application of machine learning in earthquake engineering in the previous CS 229 projects. In this project, we implement some similar methodologies such as KNN but with a different approach to select the optimal K value.

Dataset and Features

The data collection is a big challenge for us because we have limited records of bridge failure due to earthquakes. Therefore, we make an assumption in advance: If an earthquake occurs near the

site of a bridge but there is no record showing that the bridge is damaged, then this should be identified as an example of "no damage" (negative examples).

We first search for bridge failure records in previous famous earthquakes in the reports published by the U.S. government, find the site location of the bridge, and mark it as a positive example. Next, we search for other earthquake records happened in bridge's history near the site location on USGS earthquake search catalog and mark them as negative examples. Finally, we split the dataset into training (70%) and test (30%) sets.

Because of the intrinsic scarcity of positive examples, our dataset is unbalanced. The positive-to-negative ratio is about 1:10. To mitigate this problem, we use bootstrap resampling with replacement to up-sample the positive class and generate training sets with more reasonable positive-to-negative ratio. In our project, this ratio varies from 0.2 to 1.0.

The raw features selected for the model are the age of the bridge, the magnitude of the earthquake, and the distance between the bridge and the epicenter, all of which are continuous variables. For future work, we plan to explore more features (continuous and discrete) such as the material type, the structure type, the annual average daily traffic, etc.

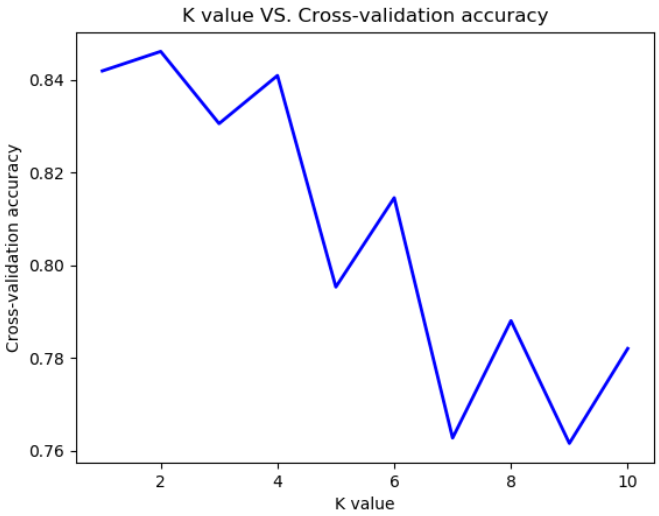
Methods

The goal of the algorithm for this binary classification problem is to predict the correct or more likely bridge performance given data on the earthquake and the bridge itself.

We use Logistic Regression, Quadratic Discriminative Analysis (QDA), and K-Nearest Neighbor Classifier (KNN) to train the model respectively and independently.

1. Logistic Regression
   The Logistic Regression updates a set of parameters that maximize the log likelihood:
   $$l(\theta) = \sum_{i=1}^{m} y^{(i)} \log h\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h\left(x^{(i)}\right)\right)$$
2. Quadratic Discriminative Analysis (QDA)
   QDA is one of the most commonly used generative models, where we assume that the results from each class are normally distributed. In binary case, with means μ_0, μ_1 and Σ_0, Σ_1, the log-likelihood can be computed as:
   $$l(x; \mu, \Sigma) = \frac{\frac{1}{(2\pi)^{n/2}|\Sigma_1|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_1)^T \Sigma_1^{-1}(x-\mu_1)\right)}{\frac{1}{(2\pi)^{n/2}|\Sigma_0|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_0)^T \Sigma_0^{-1}(x-\mu_0)\right)}$$
3. K-Nearest Neighbor Classifier (KNN)

The K-Nearest Neighbor Classifier considers k neighbors of the current data point, compares the number of positives and negatives within the k neighbors, and outputs the label (i.e. positive or negative) of the one that has a greater number than the other. To determine the optimal value of k, we use a cross-validation set and plot its accuracy with respect to the value of k from 1 to 10 as shown below.



Based on the figure shown above, the optimal K value in this case is $k = 2$.

Results and Discussion

The following training and testing accuracy are the average results from 5 cases with different positive-to-negative ratio after resampling:

Model	Training Accuracy	Testing Accuracy
Logistic Regression	95.4%	73.8%
QDA	97.8%	94.8%
KNN	99.1%	80.6%

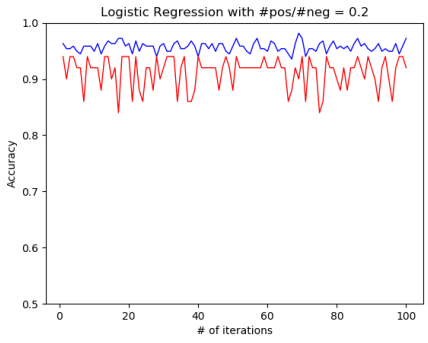
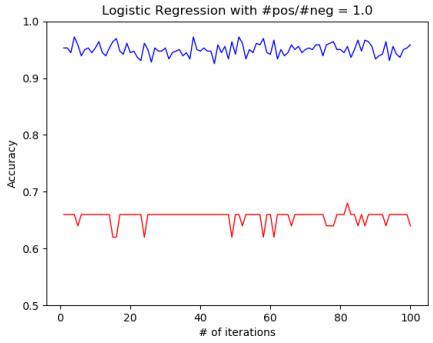
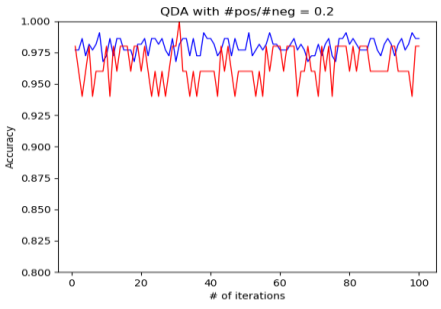
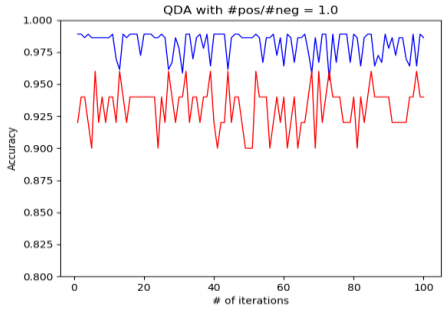
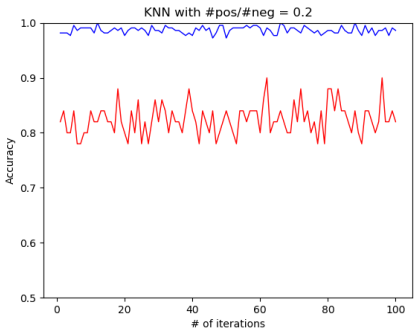
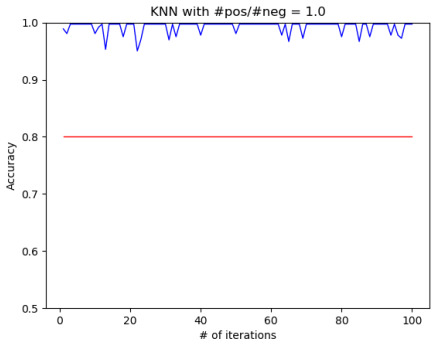
From table above, **QDA** appears to have the highest testing accuracy (94.8%, around 28% higher than logistic regression and 18% higher than KNN).

In addition, since the testing set is also unbalanced, we compute the average testing accuracy for both positive and negative classes as shown below:

Model	Testing Accuracy (Positive class)	Testing Accuracy (Negative class)
Logistic Regression	82.7%	73.4%
QDA	99.5%	94.5%
KNN	29.9%	83.8%

From the table above, we observe that the result from **QDA** is still more accurate than other 2 models for both positive and negative classes. Also, KNN preforms badly when predicting positive classes in the test set (only about 30% accuracy).

Since bootstrap resampling is a randomized sampling algorithm, we run 100 iterations for each model and plot both training and testing accuracy VS numbers of iterations with different positive-to-negative ratio as tabulated below:

Algorithm	**#pos/#neg = 0.2**	**#pos/#neg = 1.0**
Logistic Regression		
QDA		
KNN		

There are a few observations from these plots.

1. The training accuracy is relatively stable in terms of number of iterations and only fluctuates about ±2.5% for each algorithm.
2. The accuracy decreases as the size of resampling increases. However, since the test set is also unbalanced, a very high testing accuracy may not be meaningful as it always tends to

predict the result to be negative. Additionally, simply resampling the size does not truly add any valuable data point into the data set, so the scarcity of positive examples is not improved.

3. The testing accuracies of KNN for a positive-to-negative ratio of 1.0 appear to be a straight line (or closely), which indicates that KNN algorithm tends to make the same predictions for random input of positive examples. This also makes sense because in case of $k = 2$, a larger positive example size ($\#positive\ examples \gg k$) will make the output of KNN more stable as it only uses the nearest 2 neighbors for prediction.

From all the plots and tables shown above, QDA seems to be the most accurate one among the 3 models we choose for this project. Also, its accuracy does not decrease too much as we expand the size of our positive class. The testing accuracy is even more stable than that of QDA (always around 80%), but it behaves poorly when making predictions on the positive class.

Conclusion and Future Work

In this project, we pre-process the raw data set with bootstrap resampling and implement 3 supervised learning models on the training set. During training process, we observe that the accuracy decreases as we increase the size of resampling. However, since the test set is also unbalanced, a very high testing accuracy may not be meaningful as it always tends to predict the result to be negative.

For future work, we expect to expand the size of dataset (more explicitly, to increase the number of positive examples), to run tests on other generative models, and to implement multi-class classification, so we may obtain a more meaningful and practical model as we desired.

Thinking about the nature of the problem helps us understand the observations above. In a civil engineering perspective, seismic performance of structures is highly uncertain and most of them are hard to predict. To view this study in a broader scope, it may be observed that there are usually lots of different constraints (on features, data size, physical meanings of results, etc.) in civil engineering scenarios, which may impact the practicality of machine learning in such kind of studies.

Contributions

Xiao and Ziyang came up with the topic and scope of the project together. We both engaged in data collection and feature selection. Ziyang mainly ran the algorithms to train the model and test the model. Xiao and Ziyang interpreted the results and made observations together. Ziyang prepared the proposal and the milestone whereas Xiao made much of the poster and the final report.

References

1. G. Dupret and M. Koda, “Bootstrap re-sampling for unbalanced data in supervised learning,” Jul. 2000.
2. C. Kellerman, “Earthquakes,” *NIST*, 28-Feb-2017. https://www.nist.gov/topics/disaster-failure-studies/studies-hazard-types/earthquakes.
3. Kiremidjian A. and Basöz N, “Evaluation of Bridge Damage Data from Recent Earthquakes,” *NCEER*, vol. 11, no. 2, 1997.
4. Ferguson M. and Martin A. (2015) “Earthquake-Induced Structural Damage Classification Algorithm,” *CS 229 Class Project*
5. George C. Lee, Satish B. Mohan, Chao Huang and Bastam N. Fard, “A Study of U.S. Bridge Failures,” *MCEER-13-0008*, 2013.

GitHub Repository

The link to the GitHub repository containing all the datasets, codes and output are given below: https://github.com/jzy95310/CS229-Fall-2018-Final-Report