Earthquake Waveform Recognition

Olivia Grubert and Bridget Vuong
Department of Computer Science, Stanford University

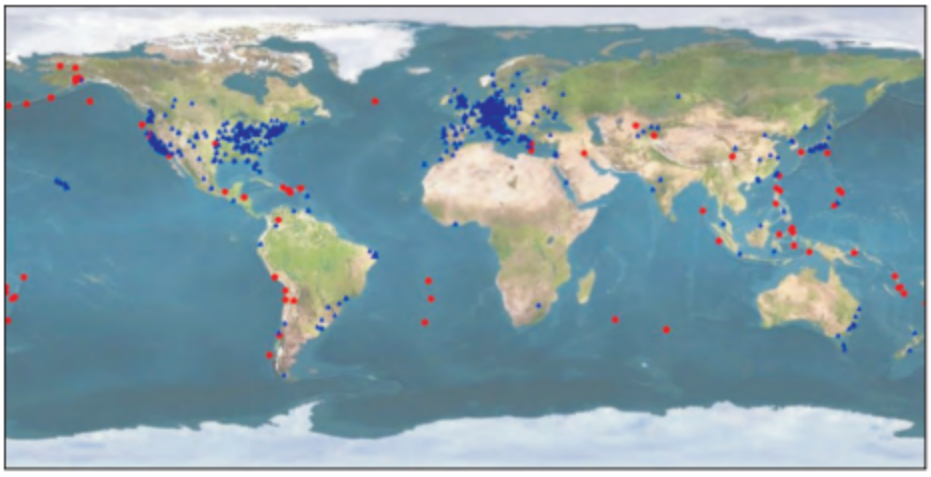
Abstract

When people think of applying machine learning to earthquake data, most seek to solve the long-elusive problem of predicting when and where an earthquake will happen and what magnitude it will be. Since earthquakes exhibit patterns that occur over hundreds or even thousands of years, however, there is just not enough data yet to achieve this goal. Instead of trying to predict the next earthquake, this paper aims at a more practical goal: to recognize earthquakes in real-time seismic signals, which could be useful in early warning systems such as Stanford's Quake Catcher Network (QCN). By applying algorithms such as SVM to data collected by QCN sensors, we distinguish seismic signals caused by an earthquake from signals caused by noise.

1 Introduction

Stanford's Quake Catcher Network is a network of inexpensive sensors that are connected to laptops and desktops in volunteer homes throughout the world. The purpose of the network is to detect and locate earthquakes as early as possible and to determine when and how badly the surrounding areas will be affected. Currently, the sensors are triggered to send data to QCN servers using a simple threshold on the instantaneous signal to long-term average signal ratio. Earthquake signals, however, have a distinct shape based on the arrivals of the Primary wave, Secondary wave, and Surface wave that is not captured by a simple threshold. This metric is also susceptible to other "ground-shaking" events, such as dropping the laptop, heavy footfall, or even setting down a cup of coffee on the table near the sensor. Because of the existence of these false positives, the triggered signals have to be manually verified to be an earthquake by a seismologist. The ability to confidently detect an earthquake automatically would be useful to the Quake Catcher Network.

Figure 1 Map of QCN sensor locations (blue) and earthquake distributions from USGS (red). [2]



2 Data

All QCN sensors report acceleration measured in the x, y, and z directions. The data stored on QCN servers come in two varieties: triggered and continuous. Triggered data is collected only when a

sensor detects a signal that exceeds a threshold, while continuous data is collected unconditionally every ten minutes. We collected data from 44 days (see Appendix A) on which a verified earthquake occurred. Earthquake data was collected from triggered data, and noise data was taken from continuous data sets on those same days. We filtered our data to only accept examples with a sample rate of 50 samples/second, then truncated each training example to have 9,000 points per *x, y,* and *z* axis. We converted our data from Seismic Analysis Code to text file so that our samples could be read by libsvm as a sparse matrix with each row corresponding to a training example and each column corresponding to a feature. In all, we selected 7,524 total examples to feed into our learning algorithms, with roughly 35% withheld for use as test data. Approximately 2.5% of the data represented verified earthquake events, while the rest of the samples were continuous noise measurements.

3 SVM Algorithm

We used a regularized SVM algorithm in order to train a model for earthquake waveform prediction. The following optimization over data points $x^{(i)}$ and labels $y \in \{-1, 1\}$ for parameters ω, b, ξ_i defines our model [4, 5]:

$$\begin{aligned} min_{\ \omega, b, \xi} \quad & \tfrac{1}{2}||w||^2 + C \sum_{i=1}^{m} \xi_i \\ s.t. \quad & y^{(i)}(w^T \phi(x^{(i)}) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & \qquad\qquad i = 1, \ldots, m \end{aligned}$$

We also tuned several kernels in testing, defined as $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$. Their descriptions and relevant parameters are given in the following sections.

3.1 Linear Kernel $K(x_i, x_j) = x_i^T x_j$

3.2 Polynomial Kernel $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$ $for\ \gamma > 0$

3.3 RBF Kernel $K(x_i, x_j) = e^{(-\gamma ||x_i - x_j||^2)}$ $for\ \gamma > 0$

3.4 Sigmoid Kernel $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$

See our Results section for the effect of using each of these kernels in our predictive models.

4 Methods

4.1 Raw Data

Our first method for learning to recognize earthquake waveforms was to simply run the raw data collected from QCN sensors through the SVM algorithm with default parameters. Each feature vector is a concatenation of 9,000 points on each of the 3 axes to give a full feature size of 27,000.

Figure 2 Plot of raw noise data.

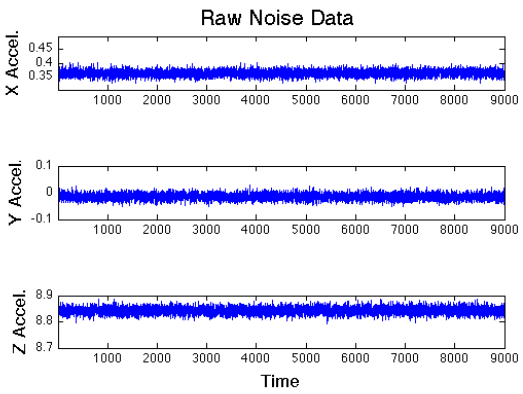
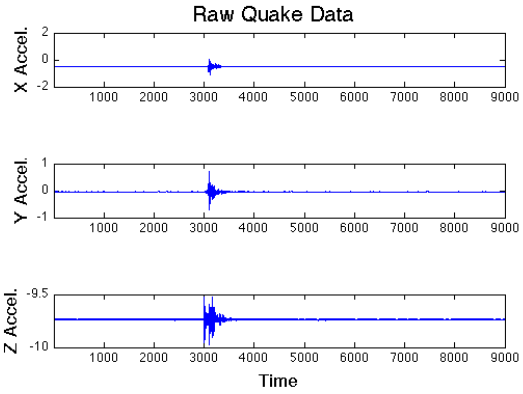


Figure 3 Plot of raw earthquake data.



4.2 Scaled Data

We realized that we were interested only in learning the shape of earthquake waveforms and did not want SVM to be sensitive to the specific amplitudes of the waveforms, which could vary based on distance from the earthquake epicenter or even slight differences in each of the sensors. We separately normalize the data in each axis to the interval [-1, 1] by subtracting the mean of the points for each axis and dividing by the maximum deviation from the mean.

$$x' = \frac{(x - \mu)}{max_{x \in X}(x - \mu)}$$

Scaling our data still results in a feature vector of size 27,000 with 9,000 scaled points for each of the 3 axes. Again, we ran the scaled data through SVM with default parameters.

Figure 4 Plot of scaled noise data.

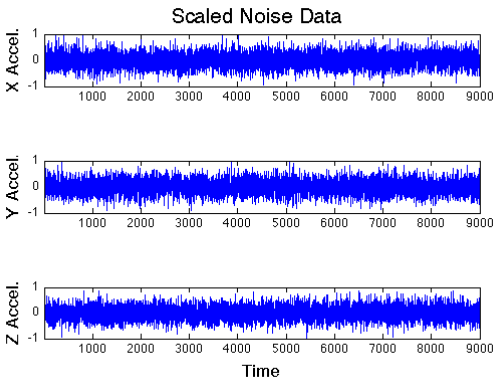
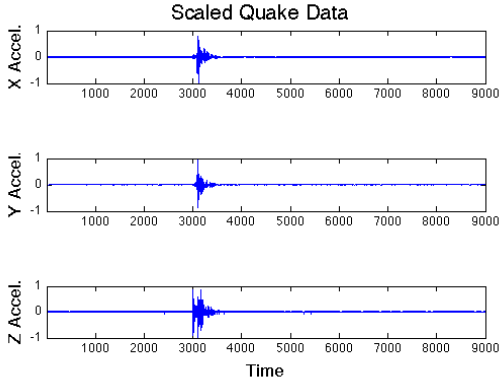


Figure 5 Plot of scaled earthquake data



4.3 Fourier Transformed Data

Since we are dealing with waves, we thought that our data could be better represented in the frequency domain, rather than using the raw or scaled data. We separately transformed the data in each axis from the time domain to the frequency domain using the one-dimensional Fourier transform.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i 2\pi \frac{k}{N} n}$$

The Fourier transform takes in n real numbers and returns $\frac{n}{2} + 1$ complex numbers. The magnitude of these complex numbers became our new features. The resulting feature vector has 4501 features for each of the 3 axes for a total of 13503 features. The smaller feature set size turned out to be an added bonus since the SVM did not take as long to run.
We applied the Fourier transform on both raw and scaled data. For Fourier-transformed and scaled data, we additionally scaled the magnitude returned by the Fourier transform by dividing by the maximum.

Figure 6 Plot of scaled, FFT noise data.

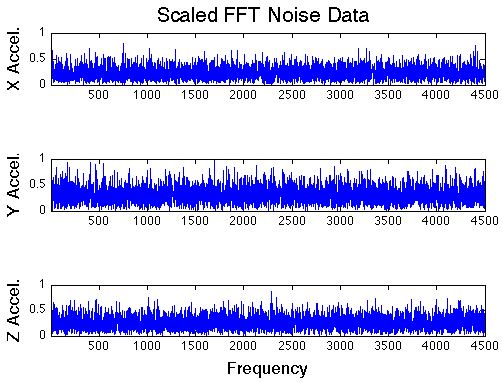
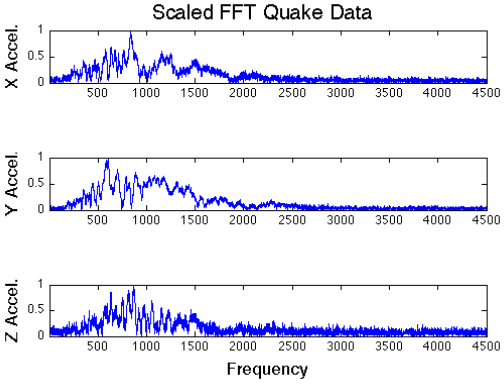


Figure 7 Plot of scaled, FFT earthquake data.



4 Results

The results of our experiments are summarized in Figure 8 below. Running SVM with a linear kernel on data that was both Fourier-transformed and scaled performed the best with a training accuracy of 99.979% and a test accuracy of 99.562%. Varying parameter C on our top-performing case did not yield better results.

Figure 8 Accuracy of SVM with default parameters $C = 1$, $\gamma = 1/numfeatures$, $r = 0$ using different kernels and on different datasets.

SVM Kernel	**Raw Data**	**Scaled Data**	**1D Fourier Transformed Data**	**1D Fourier Transformed Data, Scaled**
Linear	Train = 99.645% Test = 2.371%	Train = 100.000% Test = 88.763%	Train = 84.361% Test = 82.707%	Train = 99.979% **Test = 99.562%**
3rd Degree Polynomial	Train = 95.045% Test = 5.327%	Train = 96.383% Test = 99.015%	Train = 96.948% Test = 79.241%	Train = 96.383% Test = 99.015%
RBF	Train = 97.449% Test = 8.646%	Train = 96.383% Test = 99.015%	Train = 100.000% Test = 99.197%	Train = 96.383% Test = 99.015%
Sigmoid	Train = 93.961% Test = 11.529%	Train = 96.383% Test = 99.015%	Train = 96.383% Test = 98.504%	Train = 96.383% Test = 99.015%

5 Conclusions and Future Work

There are still many alternate methods that we could try, given more time. Instead of using a one-dimensional Fourier transform separately on each of the axes, we could use a three-dimensional Fourier transform on the *x*, *y*, and *z* components of the acceleration. Since earthquakes are primarily composed of a Primary wave, a Secondary wave, and a Surface wave, we could also use independent component analysis to separate the three wave sources recorded on the three axes of acceleration and define features of the SVM based on the separated waves.
We are satisfied with the results of our experiments overall and hope that it can be used as a stepping stone for developing more sophisticated earthquake detection methods that can be used by the Quake Catcher Network. One such extension to this project would be to recognize an earthquake waveform given real-time data using Hidden Markov Models.

6 Acknowledgements

We would like to thank Associate Professor Andrew Ng for teaching us all that we know about machine learning. We would also like to thank Assistant Professor Jesse Lawrence at the Department of Geophysics and Project Leader of the Quake Catcher Network for his guidance in getting this project started, for providing us with access to QCN data, and for helping us understand the basics of earthquakes as well as some of the technical challenges that face QCN. We also thank Andrew Maas for answering our questions on the Machine Learning aspects of our project.

7 References

1. Chang, C.C. and Lin, C.J.. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27, 2011.
2. Cochran E., Lawrence J., Christensen C., Chung A. A novel strong-motion seismic network for community participation in earthquake monitoring. IEEE Instrumentation and Measurement, 12:6:8-15, 2009.
3. Frigo, M. and Johnson, S. The design and implementation of FFTW3. IEEE 93:2:216–231, 2005.
4. Hsu, C.-W., Chang, C.-C., and Lin, C.-J. A practical guide to support vector classification. Tech. rep., Department of Computer Science, National Taiwan University.
5. Ng, A. Lecture Notes 3 - Support Vector Machines. November 2012.
6. The Quake-Catcher Network. Data retrieved November 2012 from QCN Database.

8 Appendix A

Dates from which data was collected

Oct 31, 2012	Sep 21, 2012	May 27, 2012	Jan 14, 2012	Dec 29, 2011	Jul 21, 2011
Oct 23, 2012	Sep 07, 2012	May 25, 2012	Jan 07, 2012	Dec 23, 2011	Jun 21, 2011
Oct 09, 2012	Aug 08, 2012	May 20, 2012	Jan 06, 2012	Oct 21, 2011	Jun 17, 2011
Oct 04, 2012	Jul 10, 2012	May 12, 2012	Jan 04, 2012	Oct 09, 2011	Jun 15, 2011
Oct 03, 2012	Jul 06, 2012	Mar 17, 2012	Jan 02, 2012	Sep 20, 2011	Jun 14, 2011
Sep 28, 2012	Jun 29, 2012	Mar 05, 2012	Jan 01, 2012	Sep 19, 2011	Jun 13, 2011
Sep 23, 2012	Jun 11, 2012	Jan 27, 2012	Dec 31, 2011	Sep 10, 2011	Jun 05, 2011