Using Low-Cost Remote Sensing Data to Detect Building Collapse in Post-Earthquake Environments

CS 229 Final Project Report

Shaun Benjamin Henry Corrigan-Gibbs Steven Wong

1 Introduction

The scale and suddenness of natural disasters render post-disaster relief efforts difficult. For earthquakes in particular, being able to map the distribution of damage quickly and with confidence can help channel appropriate aid to the most-severely impacted regions. Accurate mapping can also aid in determining whether citizens can return safely to their homes, so as to prevent casualties from delayed building collapses.

There are generally two broad types of post-disaster damage surveys for earthquakes: *field-based* surveys and *remote-sensing-based* surveys. Field-based surveys involve having experts physically visit each individual building to assign it a damage level. Field-based surveys are thus time-consuming and resource-intensive. In contrast, remote-sensing-based surveys usually involve a panel of experts looking over aerial or satellite imageries to estimate if buildings in the images have collapsed or not. Remote-sensing-based surveys thus save time but have a greater degree of error.

In this project we attempt to use machine learning techniques to identify collapsed buildings in satellite photos by applying supervised learning techniques to improve remote-sensing damage estimates. Our target variable is the field-based damage assessment, and the features are the satellite-based damage assessment combined with relevant external data (e.g., earthquake epicenter, elevation, population density, per-capita income).

Using the machine learning techniques we develop here, future disaster relief professionals might be able to use a more limited field-based damage assessment, in combination with remote-sensing-based data, to identify highly damaged areas more quickly and at lower cost.

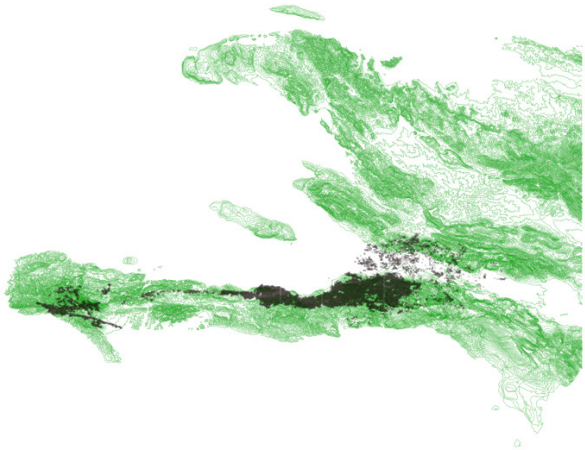

Figure 1: A topographical map of Haiti with the geographical areas included in our data set marked in black.

2 Data Sources

The 2010 earthquake in Haiti devastated the areas in and around the Haitian capital of Port-au-Prince. In the wake of the earthquake, disaster relief agencies collected extensive data on building collapse and we have obtained two such data sets from colleagues at Stanford. These data sets divide the affected geographical area into geograhpical grid cells, and the data tables contain information on the fraction of buildings in each cell which collapsed in the earthquake.

One data set contains information drawn from the expensive field-based damage survey. Another data set contains remote-sensing data derived from satellite imagery. These two data sets contain information on a total of 28,094 geographical cells, with each cell measuring 50 meters $\times$ 50 meters. Since the field-based and remote-sensing data sets use slightly different grid layouts, we had to merge the two disparate data sets into a resized geographical grid, which left us with 13,389 adjusted grid cells.

Figure 1 shows the geographical areas of Haiti covered by our data sets, which are also the areas most affected by the earthquake.

In addition to these closed-source datasets, we collected features on Haiti's physical characteristics and on seismic measurements from the 2010 earthquake event. From the World Bank, via HaitiData.org [3], we obtained estimates of the probability that the peak ground velocity (earthquake intensity) will exceed a given level within the next 50 years. In other words, this data set is an estimate of the earthquake risk of each geographical grid cell, based on the path of fault lines and other geological features (e.g., Figure 2). We also collected data on the soil capacity, level of ground erosion, flood potential, and geomorphology for each area of our geographical grid. From the U.S. Geological Survey [7], we collected measurements of the actual peak ground velocity and acceleration during the 2010 Haiti earthquake.

Standardizing these disparate data sets—each of which uses different formatting conventions, geographical projections, and cell sizes—took much longer than we had anticipated. We used Esri's ArcMap software package to approximate their spatial relationships between the different grid cells and to spatially-join their values. In particular, we

1. scaled and translated the imported data cell's coordinates to approximately match our coordinate system, and
2. matched imported rows to the nearest training example (geographical grid cell) in our data set.

3 Methods

After combining our closed-source data set obtained from colleagues at Stanford with open-source geographical data sets available online, we imported the harmonized "master" data set into MATLAB for analysis.

We expected the labels of our data set to have a non-linear relationship with many of the features. For example, the probability that a particular grid cell contains a collapsed building dramatically increases when conditioned on the fact that the cell lies inside of Port-au-Prince, though the relationship between grid latitude and collapse probability is non-linear.

Since we wanted to perform supervised learning on a data set with many non-linearities, we concluded

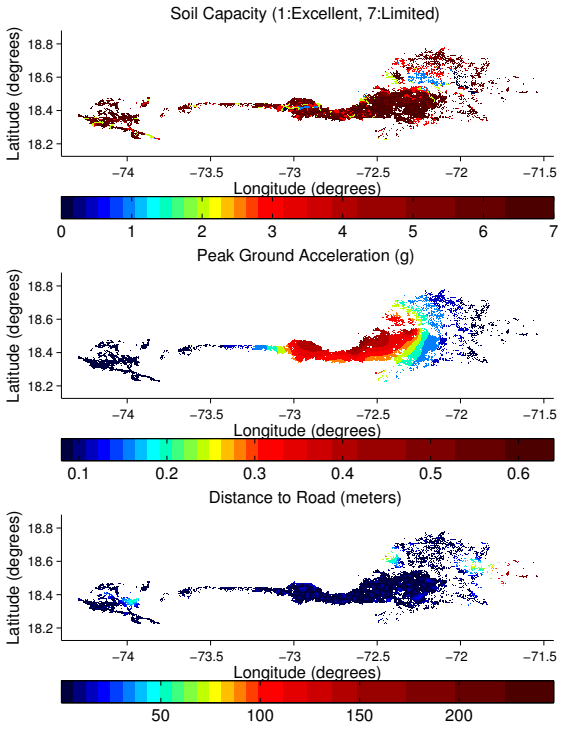

Figure 2: Soil capacity (top), peak ground acceleration (middle), and distance to nearest road (bottom), as recorded in our data set

that support-vector machine (SVM) [6] and random forest (RF) [4] classifiers would both be applicable algorithms. These two algorithms are particularly convenient to use because there are a number of open-source SVM and RF implementations which are easy to integrate into our MATLAB application in an "off-the-shelf" manner.

Our experiments used the built-in MATLAB SVM classifier, the LIBSVM [5] classifier, and the `randomforest-matlab` [2] classifier. To tune the parameters of our classifiers, we used hold-out cross validation: we trained the classifier on roughly 70% of our examples and tested on the remaining 30%.

We also experimented with using PCA and ICA to reduce the dimensionality of our examples to improve the performance of our classifiers. To implement PCA and ICA, we used the MATLAB PCA implementation and the FastICA [1] ICA toolkit.

Since our data set actually contains the *number* of collapsed buildings in each geographical grid cell, we converted these multi-class labels into binary labels by marking geographical cells with at least one collapsed building with a "1" and cells without any collapsed buildings with a "0." Since some

of our geological features used numerical codes to indicate `enum`-like values (`0=grassland`, `1=forest`, `2=urban`, etc.), we converted each of these features into an array of binary indicator values (i.e., `0=not_grassland`, `1=grassland`). Finally, to speed up the training and testing process, we removed cells from the data set that were so far from the earthquake-affected area that they could not possibly have sustained damage from it.

4 Results

This section summarizes the results of the experimental evaluation of our classifiers.

4.1 Support-Vector Machines versus Random Forests

We first compare the two SVM classifiers (LIBSVM and MATLAB) and the random forest classifier. Since we expect that many of our features will have have non-linear correlations to our labels, we chose to use a Gaussian kernel with the SVMs. For the sake of completeness, we also experimented with using a linear kernel, but found that the SVM classifier often failed to converge upon a feasible decision boundary, even when using a relatively aggressive soft-margin classifier.

Figure 3 shows the training and test error of our three classifiers when trained on a varying number of the examples in our data set (selected at random) and when tested on the remaining training examples. Even after tuning the parameters of both SVM algorithms, the built-in MATLAB SVM implementation consistently outperforms the LIBSVM implementation by roughly 10%. The random forest classifier outperforms the MATLAB SVM classifier at all training set sizes by slightly over 5%. At the largest training set size (8,000 examples), the test error of the random forest classifier is just under 30%, though in some cases the error of the random forest classifier dipped down to closer to 25%.

Figure 4 demonstrates the performance of each of the three classifiers, as measured by the time taken to train on the given number of rows and to test on the remainder of the data set. At training set sizes of under 4,000 examples, the MATLAB SVM outperforms both classifiers. As the training set size increases past 4,000 examples, the MATLAB

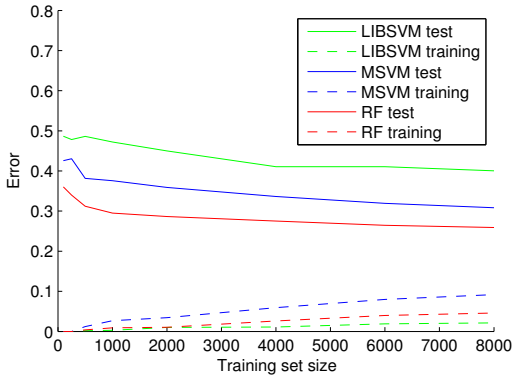

Figure 3: Training and test error of the classifiers when trained on a varying number of examples and tested on the remaining examples.

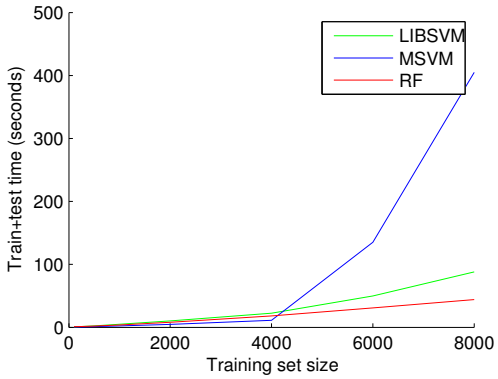

Figure 4: Running time for each of the three classifiers as training set size increases.

SVM algorithm exhibits a dramatic slowdown—taking more than 8× as long to run than the random forest classifier an more than 4× as long as the LIBSVM classifier.

Since the random forest classifier performed the best in terms of both test error and running time, we provide an additional analysis of the how the random forest parameters affect the accuracy of the classifier. Figure 5 demontrates how the test error of the classifier varies as the number of trees in the forest and the number of splitting features used at each tree node vary. As the number of trees and splitting dimensions increases, the marginal benefit of increasing these parameters decreases and the running time increases. We set both values to 500 as a compromise between training speed and classifier accuracy.

Table 1 summarizes the best observed performance of each of the three classifiers. The random forest

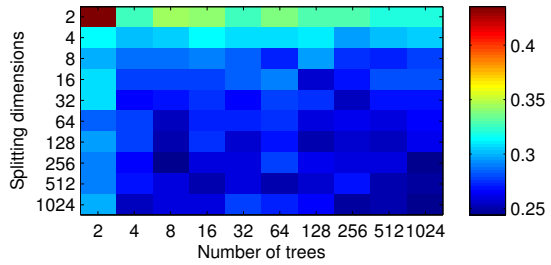

Figure 5: Test error while varying the number of trees and number of features used to split at each tree node.

Classifier	Accuracy	Advantage
Naïve remote-sensing	65.9%	0%
LIBSVM	60.5%	-5.4%
MATLAB SVM	68.9%	3.0%
Random forest	73.9%	8.0%

Table 1: Summary of accuracy of the classifiers used in this study.

classifier achieves an 8% improvement over using only remote-sensing data to predict building collapse.

4.2 Feature Selection

We implemented forward search to select a subset of features which has the greatest impact on test error. Figure 6 shows that using a single feature, the number of buildings collapsed as detected by the remote-sensing-based survey, the classifier is able to achieve a test error of just above 35%. The subsequent features which reduced the test error the most were (in order):

1. soil quality,
2. the "dense forest" indicator variable,
3. distance to the nearest road, and
4. geology types.

Since the classifier performance was acceptable when using all 69 features, and since using all features slightly reduced the test error (to nearly 26%), we decided to train on all features.

4.3 PCA and ICA

We investigated using PCA and ICA for improving the performance of our classifier in case it were to be used on a much larger data set. Such techniques might be useful for post-earthquake analysis of an area in which high-resolution grid data is available

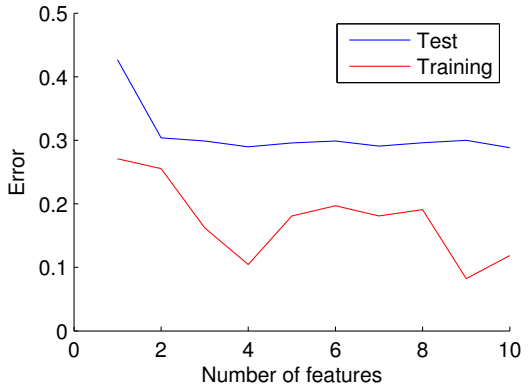

Figure 6: Test error on a fixed-size subset of the data set while performing forward search to identify a useful subset of features.

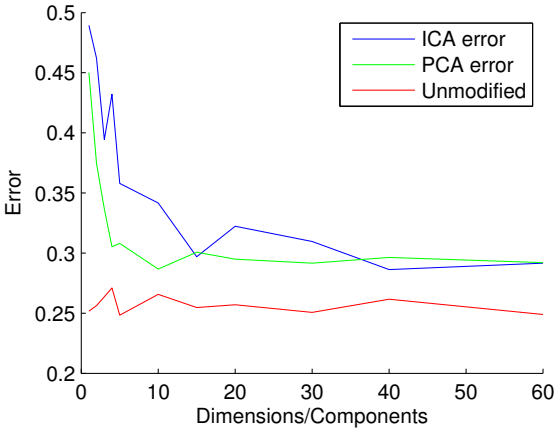

Figure 7: Test error while varying the number of components used to train the classifier. The "unmodified" line shows the baseline error without PCA or ICA.

and thus, the data set contains many training examples. For example, a 100×100 kilometer region with grid cell resolution of 10 meters would have 100,000,000 grid cells.

Figure 7 shows how the test error when using a random forest classifier trained on 70% of the data set which is preprocessed using PCA or ICA with a varying number of dimensions/components. Applying either PCA or ICA reduces the accuracy of the classifier by roughly 5% relative to using no preprocessing.

However, using PCA and ICA to reduce the dimensionality of the data does improve performance. As Figure 8 demonstrates, the running time of the classifier is roughly linear in the number of features (while holding all else constant) so using PCA or

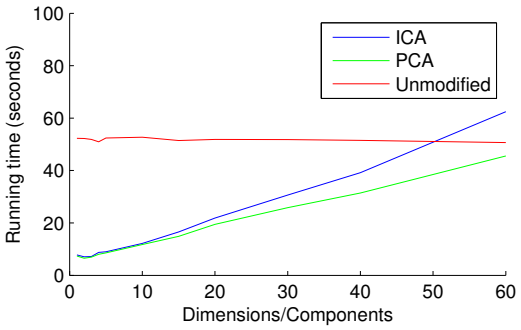

Figure 8: PCA timing

		Predicted	
		False	True
Actual	False	2115	795
	True	792	2298

Table 2: Results of one run of our classifier when trained with 8000 randomly selected examples and tested on 4000 examples.

ICA to reduce the number of features does offer a performance improvement in terms of running time.

4.4 Asymmetric Error Analysis

So far we have measured the performance of our classifier by the test error, but in reality, false positives and false negatives might have dramatically different costs in a real-world post-earthquake scenario. For example, if our classifier does not detect a set of collapsed buildings (false negative), relief workers might not be dispatched to that grid cell and people in need of help might not get it. In contrast, the cost of false positives might be relatively low, since relief workers arriving at a scene without collapsed buildings would quickly see the lack of damage and move on to another site.

Table 2 breaks classifier error into false positives and false negatives. When using the random forest classifier, the number of false positives and false negatives is nearly equal.

5 Conclusion

In the event of future earthquakes, this classifier can be used in conjunction with remote-sensing-based survey for rapid disaster response. In particular, this classifier could enable disaster relief teams

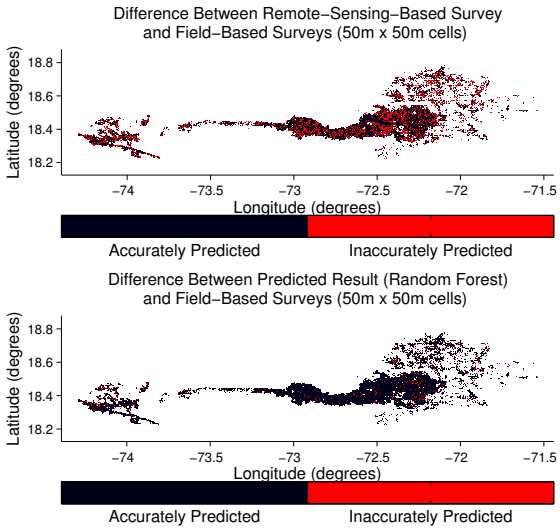

Figure 9: Damage estimates using only remote-sensing data (left) and using our classifier (right).

to incorporate available data sets, especially those known empirically to be correlated with damage severity. That said, we do not claim that disaster response teams can abandon field-based surveys entirely. Rather, by highlighting areas of likely damage, the classifier can help relief workers channel time-sensitive aid more efficiently, and to reduce the overall amount of field-based survey required.

References

[1] FastICA package for MATLAB. `http://research.ics.aalto.fi/ica/fastica/`.

[2] randomforest-matlab. `https://code.google.com/p/randomforest-matlab/`.

[3] The World Bank. Haitidata. `http://haitidata.org/`.

[4] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[5] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[7] U.S. Geological Survey. Shakemap us2010rja6. `http://earthquake.usgs.gov/earthquakes/shakemap/global/shake/2010rja6/`.