Community Sense and Response Systems: Your Phone as Quake Detector

Matthew Faulkner, Robert Clayton, Thomas Heaton, K. Mani Chandy, Monica Kohler, Julian Bunn, Richard Guy, Annie Liu, Michael Olson, MingHei Cheng, Andreas Krause

The proliferation of smartphones and other powerful sensor-equipped consumer devices enables a new class of web applications: Community Sense and Response (CSR) systems. These applications are distinguished from standard web applications by the use of community-owned commercial sensor hardware. Just as social networks connect and share human-generated content, CSR systems work to gather, share, and act on sensory data from people's internet-enabled devices. In this article, we discuss our work building the Caltech Community Seismic Network as a prototypical CSR system harnessing accelerometers in smartphones and consumer electronics. We describe the systems and algorithmic challenges of designing, building and evaluating a scalable network for real-time awareness of dangerous earthquakes.

Nearly 2 million Android and iOS devices are activated every day, each carrying numerous sensors and a high-speed internet connection. Several recent sensing projects seek to partner with the owners of these and other consumer devices to collect, share, and act on sensor data about phenomena that impact the community. Coupled to cloud computing platforms, these networks can reach an immense scale previously beyond the reach of sensor networks [6]. [5] provides an excellent overview of how the Social and Mobile Web facilitate crowdsourcing data from individuals and their sensor devices. Additional applications of community and participatory sensing include: understanding traffic flows [14, 20, 16, 4]; identifying sources of pollution [2, 1], monitoring public health [18], and responding to natural disasters like hurricanes, floods, and earthquakes [8, 9, 11, 15]. These systems are made possible by volunteer sensors and low-cost web solutions for data collection and storage. However, as these systems mature, they will undoubtedly extend beyond data collection and begin to take real-time action on the community's behalf. For example, traffic networks may reroute traffic around an accident, or a seismic network may automatically slow trains to prevent derailing.

From collection to action

Acting on community sensor data is fundamentally different than acting on data from standard web applications or scientific sensors. The potential scale of raw data is vast, even by the standards of large web applications. Data recorded by community sensors often include signals produced by the people who operate them. And many of the desired applications, while far-reaching, push the limits of our current understanding of physical phenomena.

Scale. The volume of raw data that can be produced by a CSR network is astounding by any standard. Smartphones and other consumer devices often have multiple sensors, and can produce continuous streams of GPS position, acceleration, rotation, audio, and video data. While events of interest (e.g. traffic accidents, earthquakes, disease outbreaks) may be rare, devices must monitor continuously in order to detect them. Beyond obvious data heavyweights like video, rapidly monitoring even a single accelerometer or microphone produces hundreds of megabytes per day. Community sensing makes possible networks containing tens of thousands or millions of devices. For example, equipping taxi cabs with GPS devices or air quality sensors could easily yield a network of 50,000 sensors in a city like Beijing [22]. At these scales, even collecting a small set of summary statistics becomes daunting: if 500,000 sensors reported a brief status update once per minute, the total number of messages would rival the daily load in the Twitter network.

Non-traditional sensors. Community devices are also different than those used in traditional scientific and industrial applications. Beyond simply being lower in accuracy (and cost) than "professional" sensors, community sensors may be mobile, intermittently available, and affected by the unique environment of an individual's home or workplace. For example, the accelerometer in a smartphone could measure earthquakes, but will also observe user motion.

Complex phenomena. By enabling sensor networks that densely cover cities, community sensors make it possible to measure and act on a range of important phenomena, including traffic patterns, pollution, and natural disasters. However, due to the previous lack of fine-grained data about these phenomena, CSR systems must simultaneously learn about the phenomena they are built to act upon. For example, a community seismic network may need models learned from frequent, smaller quakes in order to estimate damage during rare, larger quakes.

These challenges are compounded by the need to make reliable decisions in real-time, and with performance guarantees. For example, choosing the best emergency response strategies after a natural disaster could be drastically aided by real-time sensor data. However, false alarms and inaccurate data can have high costs; rigorous performance estimates and system evaluations are prerequisites for automating real-world responses.

1. THE CALTECH COMMUNITY SEISMIC NETWORK

The *Community Seismic Network* project at Caltech seeks to rapidly detect earthquakes and provide real-time estimates of their impact using community-operated sensors. Large earthquakes are among the few scenarios that can threaten an entire city. The CSN project is built upon a vision of people sharing accelerometer data from their personal devices to collectively produce the information needed for effective real-time and post-event responses to dangerous earthquakes. To that end, CSN has partnered with more than a thousand volunteers in the Los Angeles area and cities around the world who contribute real-time acceleration data from their Android smartphones and low-cost USB-connected sensors.

Figure 1: CSN volunteers contribute data from low-cost accelerometers (above) and from Android smartphones via a CSN app (below).

After an earthquake, fire fighters, medical teams and other first-responders must build *situational awareness* before they can effectively deploy their resources. Due to variations in ground structure, two points that are only a kilometer apart can experience significantly different levels of shaking and damage, as illustrated in Figure 2. Similarly, different buildings may receive differing amounts of damage due to the types of motion they experience. If communication has been lost in a city, it can take up to an hour for helicopter surveillance to provide the first complete picture of the damage a city has sustained. In contrast, a seismic network with fine spatial resolution could provide accurate measurements of shaking (and thus an estimate of damage) *immediately*. Because sensors can detect the moderate P-wave shaking that precedes the damaging S-wave shaking, sensors are expected to report data before network and power are lost, and before cellular networks are overloaded by human communication.

Another intriguing application of a community seismic network is to provide *early warning* of strong shaking. Early warning operates on the principle that accelerometers near the origin of an earthquake can observe initial shaking before locations further from the origin experience strong shaking. While the duration of warning that a person receives depends on the speed of detection and their distance from the origin, warning times of tens of seconds to a minute have been produced by early warning systems in Japan, Mexico, and Taiwan. These warning times can be used to evacuate elevators, stop trains, or halt delicate processes such as semiconductor processing or medical surgery. Additionally, warning of aftershocks alerted emergency workers involved in debris clearing during the 1989 Loma Prieta earthquake.

Partnering with the community. Community participation is ideal for seismic sensing for several reasons. First, community participation makes possible the densely distributed sensors needed for accurately measuring shaking throughout a city. For example, instrumenting the greater Los Angeles area at a spatial resolution of 1 sensor per square kilometer would require over 10,000 sensors. While traditional seismometer stations cost thousands of dollars *per sensor* to install and operate, the same number of sensors could be reached if 0.5% of the area's population volunteered data from their smartphones. In this way, community sensors can provide fine spatial coverage, and complement existing networks of sparsely deployed, high quality sensors.

Community sensors are also ideally situated for assisting the population through an emergency. In addition to collecting accelerometer data, community sensing software on a smartphone could be used to report the last-known location of family members, or give instructions on where to gather for help from emergency teams. In short, community sensing applications provide a new way for people to stay informed about the areas and people they care about.

CSN makes it easy for the community to participate by using low-cost accelerometers and sensors already present in volunteers' Android phones. A free Android application on the Google Play app store called CSN-Droid makes volunteering data as easy as installing a new app. The CSN project also partners with LA-area schools and city infrastructure to freely distribute 3000 low-cost accelerometers from Phidget, Inc. that interface via USB to a host PC, tablet, or other internet-connected device. Phidget sensors have also been installed in several high-rise buildings to measure structural responses to earthquakes. Figure 1 displays these sensors.

Fundamental challenges. Reliable, real-time inference of spatial events is a core task of seismic monitoring, and also a prototypical challenge for any application utilizing physical sensors. In the following, we outline a methodology devel-

oped to rapidly detect quakes from thousands of community sensors. As we will see, the computational power of community devices can be harnessed to overcome the cacophony of noise in community-operated hardware, and that on-device learning yields a decentralized architecture that is scalable and heterogeneous, while still provides rigorous performance guarantees.

2. DECENTRALIZED EVENT DETECTION

Suppose that a strong earthquake begins near a metropolitan area, and that a 0.1% of the population contributes accelerometer data from a personally-owned internet-enabled device. In Los Angeles county, this means data from 10,000 noisy sensors located on a coastal basin of rock and sediment, striped with fault lines, and cross-hatched with vibration-producing freeways. How could we detect the quake, and estimate its location and magnitude as quickly as possible?

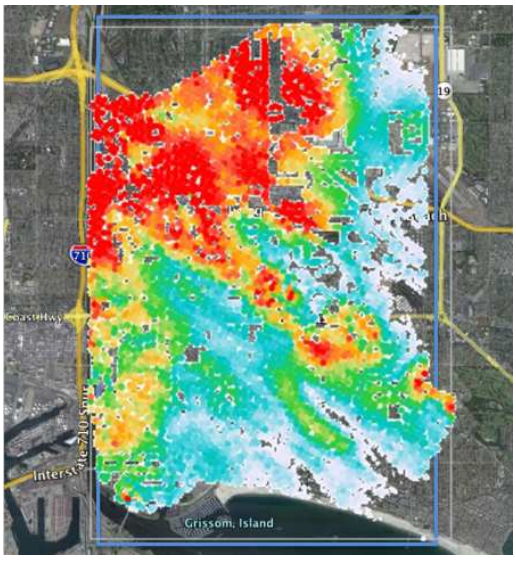

Figure 2: Differences in soil conditions and subsurface structures cause large variations in ground shaking. Data recorded by the Long Beach, CA network.

One direct approach from detection theory is to collect all data centrally, and perform classification using a likelihood ratio test,

$$\frac{\mathbb{P}\left[\text{ all measurements } \middle|\text{ strong quake }\right]}{\mathbb{P}\left[\text{ all measurements } \middle|\text{ no quake }\right]} > \tau \qquad (1)$$

This test declares a detection if the ratio exceeds a predetermined threshold τ. Unsurprisingly, this involves transmitting a daunting amount of data; a global network of 1M phones would be transmitting 30TB of acceleration data per day! Additionally, the likelihood ratio test requires the distribution of all sensor data, conditioned on the occurrence or non-occurrence of a strong earthquake. Each community sensor is unique, and so modeling these distributions requires modeling each sensor individually.

A natural next step is a decentralized approach. Suppose each device instead only transmits a finite summary of its current data, called a *pick message*. The central server again performs a hypothesis test, but now using the received pick messages instead of the entire raw data. Results from decentralized hypothesis testing theory state that *if* the sensors' measurements are independent conditional on whether there is an event or not, and if the probability of the measurements is known in each case *then* the asymptotically optimal strategy is to perform a hierarchical hypothesis test [21]: each sensor individually performs a hypothesis test, for some threshold τ, and picks only when

$$\frac{\mathbb{P}\left[\text{ one sensor's measurements } \middle|\text{ strong quake }\right]}{\mathbb{P}\left[\text{ one sensor's measurements } \middle|\text{ no quake }\right]} > \tau. \qquad (2)$$

Similarly, the Cloud server performs a hypothesis test on the number of picks S received at a given time, and declares a detection when a threshold τ' is exceeded:

$$\frac{\text{Bin}(S; r_T; N)}{\text{Bin}(S; r_F; N)} \geq \tau', \qquad (3)$$

The parameters r_T and r_F are the true positive and false positive pick rates for a single sensor, and $\text{Bin}(\cdot, p, N)$ is the probability mass function of the Binomial distribution. Asymptotically optimal decision performance can be obtained by using the decision rules (2) and (3) with proper choice of the thresholds τ and τ' [21]. Additionally, collecting picks instead of raw data may help preserve user privacy.

Challenges for the classical approach. Detecting rare events from community sensors presents three main challenges to this classical, decentralized detection approach:

1. How can we perform likelihood ratio tests on each sensor's data, when we do not have enough data (e.g. measurements of large, rare quakes) to accurately model sensor behavior during an event?
2. How can we model each sensor? Server-side modeling scales poorly, while on-device learning involves computational and storage limits.
3. How can we overcome the (strong) assumption of conditionally independent sensors, and incorporate spatial dependencies?

Next, we will consider how the abundance of normal data can be leveraged to detect rare events for which we lack training data. Then, we will see that new tools from computational geometry make it possible to compute the needed probabilistic models on resource-constrained devices. Finally, learning on the server-side adapts data aggregation according to spatial dependencies.

Leveraging "normal" data

The sensor-level hypothesis test in (2) requires two conditional probability distributions. The numerator models a particular device's acceleration during a strong quake, and due to the rarity of large quakes is impractical to obtain. In contrast, the denominator can be estimated from abundantly available "normal" data. Can we still hope to produce reliable picks?

It turns out that under mild conditions, a simple anomaly detection approach that uses only the probability of an acceleration time series in the absence of a quake can obtain

the same asymptotically optimal performance. A given sensor now picks when

$$\mathbb{P}[\text{ one sensor's measurements} \mid \text{no quake }] < \tau. \quad (4)$$

For an appropriate choice of threshold, this can be shown to produce the same picks as the full hypothesis test, without requiring us to produce a model of sensor data during future, unknown quakes. For details, see [11].

Learning on smartphones with Coresets

The above anomaly detection scheme makes use of the abundant "normal" data, but leaves us the challenge of computing the conditional distribution. In principle, each sensor could maintain a history of its observations, and periodically estimate a probabilistic model describing that data. On a mobile device, this means logging around 3GB of acceleration data per month. Storing and estimating models on this much data is a burden on volunteers' smartphone resources. Could we accurately model a sensor's data with (much) less storage?

In the CSN system, the local distribution is chosen to be a Gaussian Mixture Model (GMM) over a feature vector of acceleration statistics from short time windows (similar to phonemes in speech recognition). GMMs are flexible, multimodal distributions that can be practically estimated from data using the simple EM algorithm [3]. In contrast to estimating a single Gaussian, which can be fit knowing only the mean and variance of the data, estimating a GMM requires access to all the data; formally, GMMs do not admit finite sufficient statistics. This precludes, for example, our ability to compress the 3GB of monthly acceleration data and still recover the same GMM that would have been learned from the full data. Fortunately, it turns out that the picture is drastically different for approximating a GMM: a GMM can be fit to an arbitrary amount of data, with an arbitrary approximation guarantee, using a finite amount of storage!

A tool from computational geometry, called a *coreset*, makes such approximations possible. Roughly, a coreset for an algorithm is a (weighted) subset of the input, such that running the algorithm on the coreset gives a constant-factor approximation to running the algorithm on the full input. Coresets have been used to obtain approximations for a variety of geometric problems, such as k-means and k-medians clustering.

It turns out that many geometric coreset techniques can also provide approximations for statistical problems. Given an input dataset D, we would like to find the maximum likelihood estimate for the means and variances of a Gaussian mixture model, collectively denoted θ. A weighted set C is a (k, ϵ)-coreset for GMMs if with high probability the log likelihood on $\mathcal{L}(C \mid \theta)$ is an ϵ approximation to the log likelihood on the full data $\mathcal{L}(C \mid \theta)$, for any mixture of k Gaussians:

$$(1-\varepsilon)\mathcal{L}(D \mid \theta) \leq \mathcal{L}(C \mid \theta) \leq \phi(D \mid \theta)(1+\varepsilon).$$

[12] showed that given input D, it is possible to sample such a coreset C whose size is *independent* of the size of input D (i.e. only depends polynomially on the dimension of the input, the number of Gaussians k, and parameters ε, δ), with probability at least $1-\delta$ for all (non-degenerate) mixtures θ of k Gaussians. This implies that learning mixture model parameters θ from a *constant size* coreset C can obtain approximately the same likelihood as learning the model from the entire, arbitrarily large D.

But how do we find C? [12] showed that efficient algorithms to compute coresets for projective clustering problems (e.g. k-means and generalizations) can provide coresets for GMMs. A key insight is that while uniformly subsampling the input may miss "important" regions of data, an adaptive sampling approach is likely to sample from "enough" regions to reliably estimate a mixture of k Gaussians; weighting the samples accounts for the sampling bias. Previous work [13] also identified that coresets for many optimization problems can be computed efficiently in the parallel or streaming model, and several of those results apply here. In particular, a stream of input data can be buffered to some constant size, and then compressed into a coreset. Careful merging and compressing of such coresets provides an approximation to the entire stream so far, while using space and update time polynomial in all the parameters, and logarithmic in n.

Learning spatial dependencies

Quake detection in community networks requires finding a complex spatio-temporal pattern in a large set of noisy sensor measurements. The start of a quake may only affect a small fraction of the network, so the event can easily be concealed in both single-sensor measurements and network-wide statistics. Data from recent high-density seismic studies, Figure 2, show that localized variations in ground structure significantly impact the magnitude of shaking at locations only a kilometer apart. Consequently, effective quake detection requires algorithms that can learn subtle dependencies among sensor data, and detect changes within groups of dependent sensors.

The "classical" approach described at the start of this section assumes that the sensors provide independent, identically distributed measurements conditioned on the occurrence or non-occurrence of an event. In this case, the fusion center would declare a detection if a sufficiently large number of sensors report picks. However, in many practical applications, the particular spatial configuration of the sensors matters, and the independence assumption is violated. Here, the natural question arises of how (qualitative) knowledge about the nature of the event can be exploited in order to improve detection performance.

Viewed as transmitting a vector $\mathbf{x} \in \mathbb{R}^p$ through a noisy channel, the signal is mostly zeros (sparse), but many bits in the received vector $\mathbf{y}$ are flipped due to noise. We should expect nearby sensors to be strongly correlated during a quake. If we knew groups of correlated sensors, detection could be improved by testing each group separately. This intuition (and some desirable analytic properties) can be captured by learning a orthonormal change-of-basis matrix that projects the binary messages received by the server onto a coordinate system that, roughly, aggregates groups of strongly correlated sensors. Given such a matrix $\mathbb{B}$ with columns $\mathbf{b}_i, \ldots, \mathbf{b}_p$, the server declares an event when

$$\max_i \mathbf{b}_i^T \mathbf{y} > \tau$$

To obtain reliable detection when the signal is weak (measured by the ℓ_0 pseudo-norm, $||\mathbf{x}||_0 < \sqrt{p}$), traditional hypothesis testing requires the error rate of each sensor (each element of $\mathbf{x}$) to *decrease* as the number of sensors p increases. This is in stark contrast to our intuition that more sensors should be better, and in contrast to the "numerous-but-noisy" approach of community sensing. However, [10] shows that if the matrix $\mathbf{B}$ is *sparsifying*, i.e. $||\mathbf{B}^T\mathbf{x}||_0 = p^\beta$, $||\mathbf{x}||_0 = p^\alpha$, $0 < \beta < \alpha < 1/2$, then the test $\max_i \mathbf{b}_i^T\mathbf{y} > \tau$ gives probability of miss and false alarm that decays to zero exponentially as a function of the "sparsification ratio" $||\mathbf{x}||_0/||\mathbf{B}^T\mathbf{x}||_0$, for any rate $r_F < 1/2$ of pick errors. Effectively, this allows large numbers of noisy sensors to contribute to reliable detection of signals that are observed only by a small fraction ($||\mathbf{x}||_0$) of sensors.

Learning to sparsify. The success of the above result depends on $\mathbf{B}$'s ability to concentrate weak signals. We could learn a basis $\mathbf{B}$ that optimizes $||\mathbf{B}^T\mathbf{x}||_0$ by solving

$$\min_{\mathbf{B}} ||\mathbf{B}^T\mathbf{X}||_0, \text{ subject to } \mathbf{B}\mathbf{B}^T = \mathbf{I} \quad (5)$$

where $\mathbf{X}$ is a matrix that contains binary observations as its columns and $||\cdot||_0$ is the sum of non-zero elements in the matrix. The constraint $\mathbf{B}\mathbf{B}^T = \mathbf{I}$ ensures that $\mathbf{B}$ remains orthonormal.

(5) can be impractical to compute, and can be sensitive to noise or outliers in the data. Instead, we may wish to find a basis that sparsely represents "most of" the observations. More formally, we introduce a latent matrix $\mathbf{Z}$, which can be thought of as the "cause", in the transform domain, of the noise-free signals $\mathbf{X}$. In other words $\mathbf{X} = \mathbf{B}\mathbf{Z}$. We desire $\mathbf{Z}$ to be sparse, and $\mathbf{B}\mathbf{Z}$ to be close to the observed signal $\mathbf{Y}$. This suggests the next optimization, originally introduced for text modeling [7], as a heuristic for (5):

$$\min_{\mathbf{B},\mathbf{Z}} ||\mathbf{Y} - \mathbf{B}\mathbf{Z}||_F^2 + \lambda||\mathbf{Z}||_1, \text{ subject to } \mathbf{B}\mathbf{B}^T = \mathbf{I} \quad (6)$$

where $||\cdot||_F$ is the matrix Frobenius norm, and $\lambda > 0$ is a free parameter. (6) essentially balances the difference between $\mathbf{Y}$ and $\mathbf{X}$ with the sparsity of $\mathbf{Z}$: increasing λ more strongly penalizes choices of $\mathbf{Z}$ that are not sparse. For computational efficiency, the ℓ_0-norm is replaced by the convex and heuristically "sparsity-promoting" ℓ_1-norm.

Although (6) is non-convex, fixing either $\mathbf{B}$ or $\mathbf{Z}$ makes the objective function with respect to the other convex. The objective can then be efficiently solved (to a local minima) via an iterative two-step convex optimization process.

3. BUILDING CSN

Managing a community sensor network and processing its data in real-time leads to challenges in scalability and data security. Cloud computing platforms, such as Amazon EC2, Heroku, or Google App Engine provide practical and cost-effective resources for reliably scaling web applications. The CSN network is built upon Google App Engine (GAE). Figure 3 presents an overview of the CSN architecture. Heterogeneous sensors include cell phones, stand-alone sensors, and accelerometers connected via USB to host computers to the cloud. The cloud, in turn, performs event detection and issues notifications of potential seismic events. An advantage of the cloud computing system is that sensors anywhere in the world can connect merely by specifying a URL.

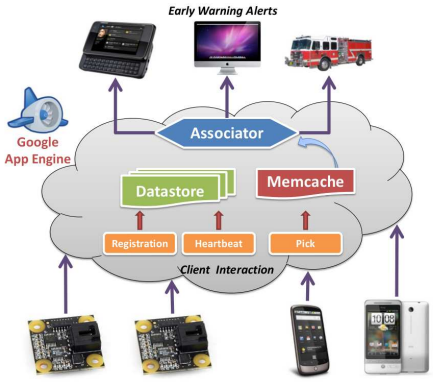

Figure 3: The CSN cloud maintains the persistent state of the network in Datastore, performs real-time processing of pick data via Memcache, and serves notifications and data products.

3.1 CSN Clients

The CSN network is comprised of two kinds of sensor clients: a desktop client with USB accelerometer, and an Android app for phones and tablets, Figure 1. Figure 4 shows the internal data flow and the messaging between the cloud and an Android client; desktop clients differ primarily in their picking algorithm and lack of GPS. At the core of the application is a suite of sensors, including the 3-axis accelerometer and GPS. The raw stream of accelerometer data is continuously tested for anomalies, which are reported as *pick* messages. The raw data is also stored (temporarily) in a local database. This both allows the server to issue *data requests* for specific intervals of data, and allows updates to the GMM anomaly detection model. Clients listen for push notifications from the server, implemented via Google's Cloud Messaging services.

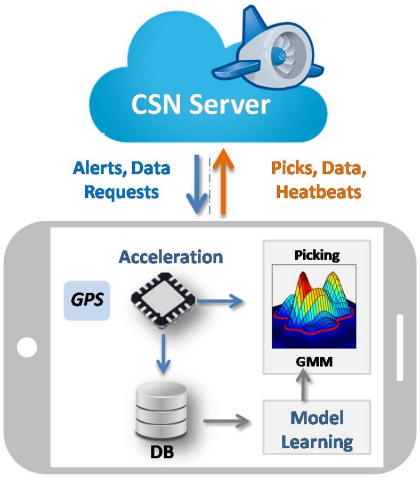

Figure 4: CSN-Droid stores and processes sensor data locally on an Android phone or tablet; sends *pick* messages during potential quakes; receives alerts; and responds to data requests.

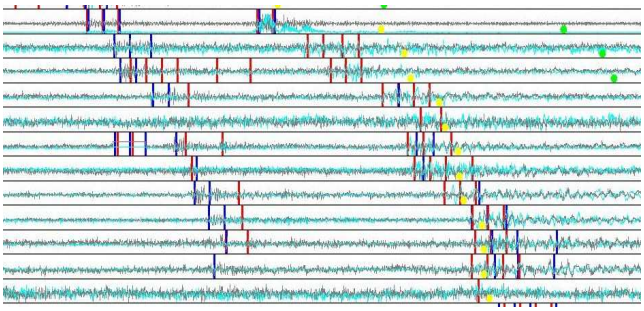

Figure 5: CSN sensors produced picks (blue and red bars) for both P-wave and S-wave of the Anza M3.6 earthquake. Time series plots are arranged by distance to quake epicenter.

3.2 CSN in the Cloud

Cloud computing services are well-suited for the network maintenance and real-time response tasks of CSR systems. Figure 3 depicts the main data flows through the cloud. First, client registration and heartbeat messages are persisted to the geographically-replicated Datastore. Next, incoming picks are spatially aggregated via geographic hashing into Memcache (a distributed in-memory data cache). While memcache is not persistent (objects can be ejected from the cache due to memory constraints), it is much faster than the datastore. Memcache is also ideal for computations that need to occur quickly, and, because memcache allows values to set an expiry time, it is also perfect for data whose usefulness expires after a period of time. Finally, an Associator performs the final event detection and issues notifications. Implementing this architecture on App Engine offers several practical advantages:

Dynamic scaling. Incoming requests are automatically load-balanced between instances that are created and destroyed based on current demand levels. This both simplifies algorithmic development, and reduces costs during idle periods.

Robust data. Datastore writes are automatically replicated to geographically separate data centers. This is prudent for any application, but especially important to CSN, where we may lose data hosted at Caltech due to a large earthquake in Los Angeles.

Easy deployment. Deploying applications on App Engine is comparatively straightforward as individual server instances do not need to be configured and coordinated. Additionally, by utilizing the same front ends that power Google's search platform, we can expect low latency from any point in the world. Together, these facts allow the network to encompass new cities or countries as soon as volunteers emerge.

4. EXPERIMENTAL EVALUATION

Large earthquakes are rare and unpredictable, which makes evaluating a system like CSN challenging. First, we must assess whether community hardware is capable of detecting strong quakes. Second, we need to evaluate detection algorithms on their ability to detect future quakes which we are unable to model or predict. Our approach must be efficient to implement on mobile devices and cloud platforms.

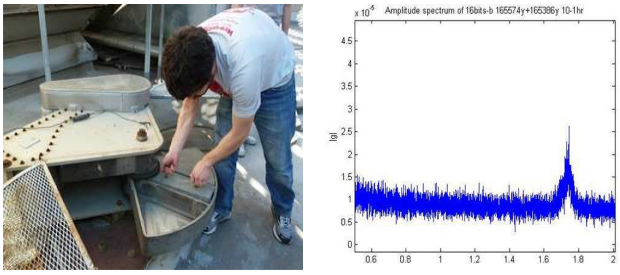

Figure 6: Eccentric weights oscillate Millikan Library, demonstrating that CSN hardware can observe resonant frequencies in buildings.

Our experiments start with an evaluation of whether community hardware is adequate for seismic detection. Several results indicate that this is indeed the case. Experiments with a large actuator called a "shake table" allow us to expose sensors to accurate reproductions of historic, moderately large (M4.5-5.5) earthquakes. The shake table demonstrates that both USB sensors and the lower quality phone accelerometers can detect the smaller initial shaking (P-wave) and stronger secondary shaking (S-wave) that produce the characteristic signature of an earthquake, as shown in Figure 7. These laboratory experiments are confirmed by measurements of actual earthquakes observed by the CSN network; similar signatures are visible in Figure 5, which shows a subset of measurements of a M3.6 quake.

A second experiment assesses whether community sensors can detect changes in the motion of buildings caused by earthquakes. We oscillated the ten-story Millikan Library on the Caltech campus, using a large eccentric weight on the roof of the building. CSN sensors in the library measured the resonant frequency of the building (around 1.7Hz), Figure 6, confirming that low-cost sensors can perform structure monitoring.

Next, we evaluate the ability of community sensors to detect future quakes for which no training data is available. While earthquakes are rare, data gathered from community sensors can be plentiful. To characterize "normal" (background) data, seven volunteers carried Android phones throughout their daily routines to gather over 7GB of phone accelerometer data, and 20 USB accelerometers recorded 55GB of acceleration. From this data, we estimated models for each sensor type's normal operating behavior. We evaluated anomaly detection performance on 32 historic records of moderately large (M5-5.5) events (as recorded by the Southern California Seismic Network). Figure 8 summarizes the ability of individual sensors to transmit "event" or "no event" to the cloud server, in the form of Receiver Operating Characteristic curves, and shows anomaly detection outperforming several standard baselines: the vertical axis is the attainable detection (pick) rate of a single sensor, against the horizontal axis of allowable false detection (pick) rate. Figure 8(c) shows that anomaly detection performance does not degrade when the Gaussian mixture model is estimated from a small coreset, but that uniformly subsampling the training data does cause a significant decrease in accuracy. Combining the accuracy results for USB and Android sensors, Figure 8(d)

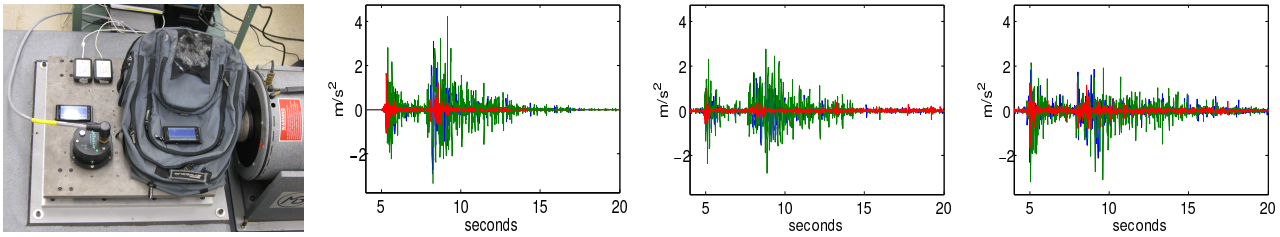

Figure 7: Android accelerometers accurately record strong shaking during a shake table experiment. (a) Shake table experimental setup. (b) Ground truth. (c) Android phone. (d) Android phone in backpack.

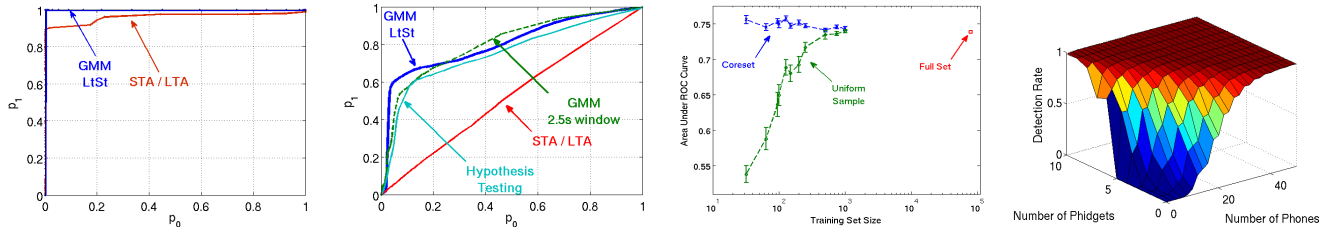

Figure 8: Attainable true positive and false positive pick rates for (a) USB accelerometer, (b) Android accelerometer. (c) Coresets allow drastic reduction in data storage, without sacrificing pick performance. (d) Estimated quake detection rates for a mixture of USB and mobile phone sensors in a given area.

shows the tradeoff of detecting with a mix of sensor types, while constraining to one false alarm per year. Our results indicate that approximately 50 phones or 10 Phidgets should be enough to detect a nearby magnitude 5 or larger event with close to 100% success.

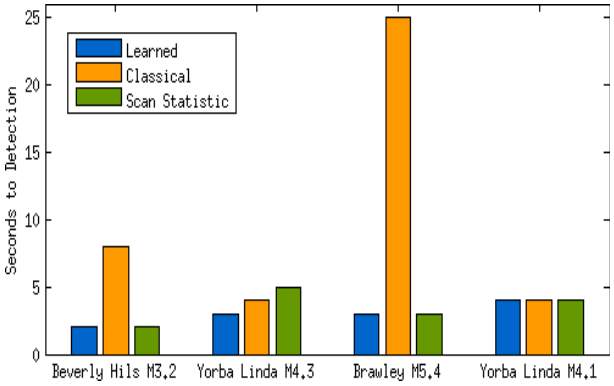

Figure 9: The learned sparsifying basis is faster than a network aggregate or a spatial scan statistic [19] for detecting four quakes recorded by the CSN network.

While earthquakes are inherently unpredictable, simulations provide a qualitative idea of spatial dependencies among sensors that can be used to train detectors. Using a prior distribution constructed from historic earthquakes available in the USGS database, and a simulator for community sensors similar to that in [17], we simulated picks from 128 CSN sensors during 1000 simulated quakes. These picks are used as training data for a sparsifying basis, a network-wide hypothesis test, and a spatial scan statistic. After training, each algorithm is then evaluated on its ability to detect four recent events using real measurements recorded by the network. Figure 9 summarizes detection performance: for each of the four events, the vertical bars give the time to detection for the learned bases, classical hypothesis testing, and a competitive scan statistic algorithm. The bases learned from simple simulations in general achieve faster detection, e.g. 8 seconds faster than competitive algorithms in detecting the Beverly Hills event.

5. CONCLUSION

In this article, we have outlined several algorithmic and systems principles that facilitate detecting rare and complex spatial signals using large numbers of low-cost community sensors. We have found that employing machine learning at each stage of a decentralized architecture allows efficient use of sensor-level and cloud-level resources, and is essential to providing performance guarantees when little can be said about a particular community sensor, or when little is known about the events we seek to detect. Community sensing is applicable to a variety of application domains, including disasters like fires, floods, radiation, epidemics, and traffic accidents, as well as monitoring the pollution, pedestrian traffic, and acoustic noise levels in urban environments. In all of these cases "responding" may range from taking physical action to merely devoting additional resources to an event of interest. While the CSN project is motivated by detecting and reacting to strong earthquakes, we believe that community sense and response systems for these domains and others will require a similar blueprint of machine learning and scalable systems.

6. ACKNOWLEDGEMENTS

We acknowledge the support of the Gordon and Betty Moore Foundation, NSF awards CNS0932392, IIS0953413 and ERC StG 307036. Andreas Krause was supported in part by a Microsoft Research Faculty Fellowship. We thank Signal Hill Petroleum and Nodal Seismic for data from the Long

Beach Network, and SCSN for data from the permanent earthquake network in southern California.

7. REFERENCES

[1] Karl Aberer, Saket Sathe, Dipanjan Chakraborty, Alcherio Martinoli, Guillermo Barrenetxea, Boi Faltings, and Lothar Thiele. Opensense: Open community driven sensing of environment. In *ACM SIGSPATIAL International Workshop on GeoStreaming (IWGS)*, 2010.

[2] Paul M Aoki, RJ Honicky, Alan Mainwaring, Chris Myers, Eric Paulos, Sushmita Subramanian, and Allison Woodruff. A vehicle for research: using street sweepers to explore the landscape of environmental community action. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 375–384. ACM, 2009.

[3] Jeff A Bilmes et al. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. *International Computer Science Institute*, 4(510):126, 1998.

[4] Paul Borokhov, Sebastien Blandin, Samitha Samaranayake, Olivier Goldschmidt, and Alexandre Bayen. An adaptive routing system for location-aware mobile devices on the road network. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1839–1845. IEEE, 2011.

[5] Maged N Kamel Boulos, Bernd Resch, David N Crowley, John G Breslin, Gunho Sohn, Russ Burtner, William A Pike, Eduardo Jezierski, and Kuo-Yu S Chuang. Crowdsourcing, citizen sensing and sensor web technologies for public and environmental health surveillance and crisis management: trends, ogc standards and application examples. *International journal of health geographics*, 10(1):67, 2011.

[6] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M.B. Srivastava. Participatory sensing. In *World Sensor Web Workshop*, pages 1–5. Citeseer, 2006.

[7] Xi Chen, Yanjun Qi, Bing Bai, Qihang Lin, and Jaime G Carbonell. Sparse latent semantic analysis. In *SIAM 2011 International Conference on Data Mining*, 2011.

[8] E.S. Cochran, J.F. Lawrence, C. Christensen, and R.S. Jakka. The Quake-Catcher Network: Citizen Science Expanding Seismic Horizons. *Seismological Research Letters*, 80(1):26, 2009.

[9] Mari Ervasti, Shideh Dashti, Jack Reilly, Jonathan D Bray, Alexandre Bayen, and Steven Glaser. ishake: mobile phones as seismic sensors–user study findings. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, pages 43–52. ACM, 2011.

[10] M. Faulkner, A. Liu, and A. Krause. A fresh perspective: Learning to sparsify for detection in massive noisy sensor networks. In *Proceedings of the 12th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 2013.

[11] M. Faulkner, M. Olson, R. Chandy, J. Krause, K. M. Chandy, and A. Krause. The next big one: Detecting earthquakes and other rare events from community-based sensors. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 2011.

[12] Dan Feldman, Matthew Faulkner, and Andreas Krause. Scalable training of mixture models via coresets. In *Advances in Neural Information Processing Systems 24*, pages 2142–2150. NIPS, 2011.

[13] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300. ACM, 2004.

[14] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.C. Herrera, A.M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proc. of the International conference on Mobile systems, applications, and services*, pages 17–20. Citeseer, 2008.

[15] A. Kapoor, N. Eagle, and E. Horvitz. People, Quakes, and Communications: Inferences from Call Dynamics about a Seismic Event and its Influences on a Population. In *Proceedings of AAAI Symposium on Artificial Intelligence for Development*, 2010.

[16] Andreas Krause, Eric Horvitz, Aman Kansal, and Feng Zhao. Toward community sensing. In *Proc. ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2008.

[17] Annie Liu, Michael Olson, Julian Bunn, and K Mani Chandy. Towards a discipline of geospatial distributed event based systems. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, pages 95–106. ACM, 2012.

[18] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 55–68. ACM, 2009.

[19] Daniel B. Neill. Fast subset scan for spatial pattern detection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2), 2012.

[20] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. VTrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98. ACM, 2009.

[21] J.N. Tsitsiklis. Decentralized detection by a large number of sensors. *Mathematics of Control, Signals, and Systems (MCSS)*, 1(2):167–182, 1988.

[22] Xiaoxiao Yu, Qiao Fu, Lin Zhang, Wenzhu Zhang, Victor OK Li, and Leo Guibas. Cabsense: Creating high-resolution urban pollution maps with taxi fleets (Preprint). *In preparation*, In preparation.