Selective Data Gathering in Community Sensor Networks

Thesis by

Matthew Faulkner

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science

California Institute of Technology

Pasadena, California

2014

(Submitted April 14, 2014)

Contents

List of Figures

Abstract

Smartphones and other powerful sensor-equipped consumer devices make it possible to sense the physical world at an unprecedented scale. Nearly 2 million Android and iOS devices are activated every day, each carrying numerous sensors and a high-speed internet connection. Whereas traditional sensor networks have typically *deployed* a fixed number of devices to sense a particular phenomena, community networks can *grow* as additional participants choose to install apps and join the network. In principle, this allows networks of thousands or millions of sensors to be created quickly and at low cost. However, making reliable inferences about the world using so many community sensors involves several challenges, including scalability, data quality, mobility, and user privacy.

This thesis focuses on how learning at both the sensor- and network-level can provide scalable techniques for data collection and event detection. First, this thesis considers the abstract problem of distributed algorithms for data collection, and proposes a distributed, online approach to selecting which set of sensors should be queried. In addition to providing theoretical guarantees for submodular objective functions, the approach is also compatible with local rules or heuristics for detecting and transmitting potentially valuable observations. Next, the thesis presents a decentralized algorithm for spatial event detection, and describes its use detecting strong earthquakes within the Caltech *Community Seismic Network*. Despite the fact that strong earthquakes are rare and complex events, and that community sensors can be very noisy, our decentralized anomaly detection approach obtains theoretical guarantees for event detection performance while simultaneously limiting the rate of false alarms.

Chapter 1

Introduction

In the last several years, mobile computing has drastically reshaped the way that people produce and access information. While it is easy to think of mobile devices simply as *smaller* computers, smart devices like phones and tablets also possess an array of sensors - such as gyroscopes, cameras, accelerometers, and compasses - not present on traditional computers. These sensors are commonly used as additional forms of user input, but can be used more broadly to sense many aspects of the user's immediate environment. In fact, a variety of recent commercial apps and products make use of or supplement these mobile sensors, including Google Glass, the Jawbone activity tracking wristband, the Nest home thermostat, and the FitBit. While the immediate intent of these products is personal sensing, the same technology is equally applicable to sensing city-wide or global environments.

Recently, the field of *Community Sensing* has emerged to study how networks of community-owned or operated devices can be used to sense the physical environment. For example, several recent sensing projects seek to partner with the owners of smartphones and other consumer devices to collect, share, and act on sensor data about phenomena that impact the community. Coupled to cloud computing platforms, such projects can reach an immense scale previously beyond the reach of sensor networks [10]. Current applications of community and participatory sensing include:

- Understanding traffic flows and monitoring road conditions [33, 61, 39, 8, 44, 31]
- Identifying sources of pollution [4, 1, 64]
- Monitoring public health [45, 42, 51]
- Responding to natural disasters like hurricanes, floods, and earthquakes [14, 20, 22, 34]

Given that the most popular mobile apps can boast more than 100,000,000 downloads, it is exciting to imagine projects like these growing into vast community networks. However, community sensing applications face several challenges beyond those of more typical mobile apps. The potential scale of raw data is vast, even by the standards of large web applications. Community sensors are also exposed to noisy, dynamic environments. 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 community 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 [71]. 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 a different breed of sensor than those used in traditional scientific and industrial applications. Beyond simply being lower in accuracy (and cost) than "professional" sensors, community sensors are coupled to the individuals who own them. As a result, community sensors are often mobile, intermittently available, and directly experience the unique environment of an individual's home, workplace, or pocket.

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, these systems must simultaneously learn about the phenomena they are built to act upon. For example, a community seismic network may need to use measurements of frequent smaller quakes in order to obtain the models of ground composition needed to accurately estimate damage during rare, large quakes.

1.1 Main Contributions

This thesis presents two approaches to selective data gathering in community sensor networks. The first contribution is an online algorithm by which a set of devices (sensors) collectively learn to self-select a fixed size subset who will transmit their observations to a central server. This work appeared as *Online Distributed Sensor Selection* [27] in the 9^{th} International Conference on Information Processing and Sensor Networks (IPSN). We analyze sensor selection under the bandit feedback model, where the algorithm only receives feedback about the utility of the selected sensors. We prove very strong theoretical no-regret guarantees that apply whenever the (unknown) utility function satisfies a natural diminishing returns property called *submodularity*. These results are most appropriate when the utility function is fixed or slowly varying; however, a heuristic for observation-dependent sampling allows the algorithm to respond to rapid changes in environmental behavior, at the cost of additional communication.

The second contribution is a detailed study of earthquake detection in a community network of Android smartphones and other consumer sensors. Published in IPSN as *The Next Big One: Detecting Earthquakes and Other Rare Events from Community-based Sensors* [22], this work presents the Caltech *Community Seismic Network* (CSN), a distributed system comprised of an Android app, a desktop client, and a Google App Engine cloud application. In contrast to many environmental sensing networks, a seismic network may experience long periods of time without observing the object of interest (i.e. a dangerously large earthquake). Consequently, earthquake monitoring is a task of *detecting rare events*. We describe how CSN uses distributed anomaly detection to transmit observations of potential earthquakes, while limiting bandwidth usage during the usual quiescent periods. This concept is implemented by the CSN-Droid Android app, and is evaluated with shake table experiments and simulations of historic earthquakes.

1.2 Related work

The work in this thesis draws on several fields of research. In addition to the community sensing research outlined above, there is also a significant amount of research on sensor selection and submodular optimization. The Caltech Community Seismic Network should be viewed in the context of several recent community-oriented seismic sensing projects, and in the context of recent developments in earthquake early warning systems. The problem of detecting spatial events using community sensors is directly related to a large body of work in decentralized detection and anomaly detection.

Sensor selection. The problem of deciding when to selectively turn on sensors in sensor networks in order to conserve power was first discussed by [57] and [73]. Many approaches for optimizing sensor placements and selection assume that sensors have a fixed region [32, 29, 6]. These regions are usually convex or even circular. Further, it is assumed that everything within a region can be perfectly observed, and everything outside cannot be measured by the sensors. For complex applications such as environmental monitoring, these assumptions are unrealistic, and the direct optimization of prediction accuracy is desired. The problem of selecting observations for monitoring spatial phenomena has been investigated extensively in geostatistics [15], and more generally (Bayesian) experimental design [11]. Several approaches have been proposed to activate sensors in order to minimize uncertainty [73] or prediction error [19]. However, these approaches do not have performance guarantees. Submodularity has been used to analyze algorithms for placing [38] or selecting [68] a fixed set of sensors. These approaches however assume that the model is known in advance.

Submodular optimization. The problem of centralized maximization of a submodular function has been studied by [48], who proved that the greedy algorithm gives a factor $(1 - 1/e)$ approximation. Several algorithms have since been developed for maximizing submodular functions under more complex constraints (see [65] for an overview). Streeter and Golovin [59] developed an algorithm for online optimization of submodular functions, which we build on here.

Seismic networks. In addition to CSN, several other projects are exploring non-traditional seismic networks. Perhaps the most closely related system is the QuakeCatcher network [14]. While QuakeCatcher shares the use of MEMS accelerometers in USB devices and laptops, our system differs in its use of algorithms designed to execute efficiently on cloud computing systems and statistical algorithms for detecting rare events, particularly with heterogeneous sensors including mobile phones (which create far more complex statistical challenges). Kapoor et al. [34] analyze the increase in call volume after or during an event to detect earthquakes. Another related effort is the NetQuakes project [63], which deploys more expensive stand-alone seismographs with the help of community participation. Our CSN Phidget sensors achieve different tradeoffs between cost and accuracy. Several Earthquake Early Warning (EEW) systems have been developed to process data from existing sparse networks of high-fidelity seismic sensors (such as the Southern California Seismic Network). The Virtual Seismologist [16] applies a Bayesian approach to EEW, using prior information and seismic models to estimate the magnitude and location of an earthquake as sources of information arrive. ElarmS [3] uses the frequency content of initial P-wave measurements from sensors closest to the epicenter, and applies an attenuation function to estimate ground acceleration at further locations. We view our approach of community seismic networking as fully complementary to these efforts by providing a higher density of sensors and greater chance of measurements near to the epicenter. Our experiments provide encouraging results on the performance improvements that can be obtained by adding community sensors to an existing deployment of sparse but high quality sensors.

Decentralized detection. There has been a great deal of work in decentralized detection. The classical hierarchical hypothesis testing approach has been analyzed by Tsitsiklis [62]. Chamberland et al. [12] study classical hierarchical hypothesis testing under bandwidth constraints. Their goal is to minimizes the probability of error, under constraint on total network bandwidth. Wittenburg et al. [69] study distributed event detection in WSN. In contrast to the work above, their approach is distributed rather than decentralized: nearby nodes collaborate by exchanging feature vectors with neighbors before making decision. Their approach requires a training phase, providing examples of events that should be detected. Martinic et al. [43] also study distributed detection on multi-hop networks. Nodes are clustered into cells, and the observations within a cell are compared against a user-supplied "event signature" (a general query on the cell's values) at the cell's leader node (cluster head).

The communication requirements of the last two approaches are difficult to meet in community sensing applications, since sensors may not be able to communicate with their neighbors due to privacy and security restrictions. Both approaches require prior models (training data providing examples of events that should be detected, or appropriately formed queries) that may not be available in the seismic monitoring domain.

Anomaly Detection. There has also been significant amount of prior work on anomaly detection. Yamanishi et al. [70] develop the SmartSifter approach that uses Gaussian or kernel mixture models to efficiently learn anomaly detection models in an online manner. While results apply only in the centralized setting, they support

the idea of using GMMs for anomaly detection could be extended to learn, for each phone, a GMM that adapts to non-stationary sources of data.

Davy et al. [18] develop an online approach for anomaly detection using online Support Vector machines. One of their experiments is to detect anomalies in accelerometer recordings of industrial equipment. They use produce frequency-based (spectrogram) features, similar to the features we use. However, their approach assumes the centralized setting.

Subramaniam et al. [60] develop an approach for online outlier detection in hierarchical sensor network topologies. Sensors learn models of their observations in an online way using kernel density estimators, and these models are folded together up the hierarchy to characterize the distribution of all sensors in the network. Rajasegarar et al. [53] study distributed anomaly detection using one-class SVMs in wireless sensor networks. They assume a tree topology. Each sensor clusters its (recent) data, and reports the cluster descriptions to its parent. Clusters are merged, and propagated towards the root. The root then decides if the aggregate clusters are anomalous. Both approaches above are not suitable for the community sensing communication model, where each sensor has to make independent decisions. Zhang et al. [72] demonstrate online SVMs to detect anomalies in process system calls in the context of intrusion detection. Onat et al. [50] develop a system for detecting anomalies based on sliding window statistics in mobile ad hoc networks (MANETs). However, their approach requires for nodes to share observations with their neighbors.

Chapter 2

Online Learning for Distributed Sensor Selection

Community networks of consumer devices make it possible to build sensor networks at very large scale, yet efficiently operating such a network will require selectively gathering only a small percentage of the total sensor data. As an illustration, a single smartphone can easily produce hundreds of megabytes of sensor data, even if video data is not included. Transmitting this data from each device is both a burden on the device bandwidth and power, but also a burden on the central server that must process or store this information.

Sensor selection is one strategy for reducing the burden of data collection, whereby individual sensors are activated at specific points in time in order to obtain the most useful information from the network (e.g., accurate predictions at unobserved locations) while also minimizing power consumption. Sensor selection is a key challenge in deploying sensor networks for real-world applications such as environmental monitoring [38], building automation [56]. The problem has received considerable attention [2, 73, 19], and algorithms with performance guarantees have been developed [2, 35]. However, many of the existing approaches make simplifying assumptions. Many approaches assume (1) that the sensors can perfectly observe a particular sensing region, and nothing outside the region [2]. This assumption does not allow us to model settings where multiple noisy sensors can help each other obtain better predictions. There are also approaches that base their notion of utility on more detailed models, such as improvement in prediction accuracy w.r.t. some statistical model [19] or detection performance [37]. However, most of these approaches make two crucial assumptions: (2) The model, upon which the optimization is based, is known in advance (e.g., based on domain knowledge or data from a pilot deployment) and (3), a centralized optimization selects the sensors (i.e., some centralized processor selects the sensors which obtain highest utility w.r.t. the model). We are not aware of any approach that simultaneously addresses the three main challenges (1), (2) and (3) above and still provides theoretical guarantees.

This chapter presents an efficient algorithm, called *Distributed Online Greedy* (DOG), which addresses these three central challenges. Prior work [36] has shown that many sensing tasks satisfy an intuitive diminishing returns property known as *submodularity*, which states that activating a new sensor helps more if

few sensors have been activated so far, and less if many sensors have already been activated. Our algorithm applies to any setting where the true objective is submodular [48], thus capturing a variety of realistic sensor models. Secondly, our algorithm does not require the model to be specified in advance: it learns to optimize the objective function in an online manner. Lastly, the algorithm is distributed; the sensors decide whether to activate themselves based on local information. We analyze our algorithm in the no-regret model, proving convergence properties similar to the best bounds for any centralized solution.

A bandit approach toward sensor selection. At the heart of our approach is a novel distributed algorithm for multiarmed bandit (MAB) problems. In the classical multiarmed bandit [54] setting, we picture a slot machine with multiple arms, where each arm generates a random payoff with unknown mean. Our goal is to devise a strategy for pulling arms to maximize the total reward accrued. The difference between the optimal arm's payoff and the obtained payoff is called the *regret*. Known algorithms can achieve average per-round regret of $\mathcal{O}(\sqrt{n \log n}/\sqrt{T})$ where n is the number of arms, and T the number of rounds (see e.g. the survey of [25]). Suppose we would like to, at every time step, select k sensors. The sensor selection problem can then be cast as a multiarmed bandit problem, where there is one arm for each possible set of k sensors, and the payoff is the accrued utility for the selected set. Since the number of possible sets, and thus the number of arms, is exponentially large, the resulting regret bound is $\mathcal{O}(n^{k/2}\sqrt{\log n}/\sqrt{T})$, i.e., exponential in k. However, when the utility function is submodular, the payoffs of these arms are correlated. Recent results [59] show that this correlation due to submodularity can be exploited by reducing the n^k-armed bandit problem to k separate n-armed bandit problems, with only a bounded loss in performance.

Existing bandit algorithms, such as the widely used EXP3 algorithm [5], are centralized in nature. Consequently, the key challenge in distributed online submodular sensing is how to devise a distributed bandit algorithm. In Sec. 2.3 and 2.4, we develop a distributed variant of EXP3 using novel algorithms to sample from and update a probability distribution in a distributed way. Roughly, we develop a scheme where each sensor maintains its own weight, and activates itself *independently from all other sensors* purely depending on this weight.

Observation specific selection. A shortcoming of centralized sensor selection is that the individual sensors' current measurements are not considered in the selection process. In many applications, obtaining sensor measurements is less costly than transmitting the measurements across the network. For example, cell phones used in participatory sensing [9] can inexpensively obtain measurements on a regular basis, but it is expensive to constantly communicate measurements over the network. In Sec. 2.5, we extend our distributed selection algorithm to activate sensors depending on their observations, and analyze the tradeoff between power consumption and the utility obtained under observation specific activation.

Communication models. We analyze our algorithms under two models of communication cost: In the *broadcast* model, each sensor can broadcast a message to all other sensors at unit cost. In the *star network*

model, messages can only be between a sensor and the base station, and each message has unit cost. In Sec. 2.3 we formulate and analyze a distributed algorithm for sensor selection under the simpler broadcast model. Then, in Sec. 2.4 we show how the algorithm can be extended to the star network model.

Our main contributions.

- Distributed EXP3, a novel distributed implementation of the classic multiarmed bandit algorithm.
- Distributed Online Greedy (DOG) and LAZYDOG, novel algorithms for distributed online sensor selection, which apply to many settings, only requiring the utility function to be submodular.
- OD-DOG, an extension of DOG to allow for observation-dependent selection.
- We analyze our algorithm in the no-regret model and prove that it attains the optimal regret bounds attainable by any efficient centralized algorithm.
- We evaluate our approach on several real-world sensing tasks including monitoring a 12,527 node network.

2.1 Sensor Selection

We now formalize the sensor selection problem. Suppose a network of sensors has been deployed at a set of locations V with the task of monitoring some phenomenon (e.g., temperature in a building). Constraints on communication bandwidth or battery power typically require us to select a subset $\mathcal{A}$ of these sensors for activation, according to some utility function. The activated sensors then send their data to a server (base station). We first review the traditional offline setting where the utility function is specified in advance, illustrating how submodularity allows us to obtain provably near-optimal selections. We then address the more challenging setting where the utility function must be learned from data in an online manner.

2.1.1 The Offline Sensor Selection Problem

A standard offline sensor selection algorithm chooses a set of sensors that maximizes a known sensing quality objective function $f(\mathcal{A})$, subject to some constraints, e.g., on the number of activated sensors. One possible choice for the sensing quality is based on prediction accuracy (we will discuss other possible choices later on). In many applications, measurements are correlated across space, which allows us to make predictions at the unobserved locations. For example, prior work [19] has considered the setting where a random variable $\mathcal{X}_s$ is associated with every location $s \in V$, and a joint probability distribution $P(\mathcal{X}_V)$ models the correlation between sensor values. Here, $\mathcal{X}_V = [\mathcal{X}_1, \ldots, \mathcal{X}_n]$ is the random vector over all measurements. If some measurements $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$ are obtained at a subset of locations, then the conditional distribution $P(\mathcal{X}_{V\setminus\mathcal{A}} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$ allows predictions at the unobserved locations, e.g., by predicting $\mathbb{E}[\mathcal{X}_{V\setminus\mathcal{A}} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}]$. Furthermore, this conditional distribution quantifies the *uncertainty* in the prediction: Intuitively, we would like to select sensors that minimize the predictive uncertainty. One way to quantify the predictive uncertainty

is the mean squared prediction error,

$$\mathrm{MSE}(\mathcal{X}_{V\setminus\mathcal{A}} \mid \mathbf{x}_{\mathcal{A}}) = \frac{1}{n}\sum_{s\in V\setminus\mathcal{A}} \mathbb{E}[(\mathcal{X}_s - \mathbb{E}[\mathcal{X}_s \mid \mathbf{x}_{\mathcal{A}}])^2 \mid \mathbf{x}_{\mathcal{A}}].$$

In general, the measurements $\mathbf{x}_{\mathcal{A}}$ that sensors $\mathcal{A}$ will make is not known in advance. Thus, we can base our optimization on the *expected mean squared prediction error*,

$$\mathrm{EMSE}(\mathcal{A}) = \int dp(\mathbf{x}_{\mathcal{A}})\, \mathrm{MSE}(\mathcal{X}_{V\setminus\mathcal{A}} \mid \mathbf{x}_{\mathcal{A}}).$$

Equivalently, we can maximize the *reduction* in mean squared prediction error,

$$f_{\mathrm{EMSE}}(\mathcal{A}) = \mathrm{EMSE}(\emptyset) - \mathrm{EMSE}(\mathcal{A}).$$

By definition, $f_{\mathrm{EMSE}}(\emptyset) = 0$, i.e., no sensors obtain no utility. Furthermore, f_{EMSE} is monotonic: if $\mathcal{A} \subseteq \mathcal{B} \subseteq V$, then $f_{\mathrm{EMSE}}(\mathcal{A}) \leq f_{\mathrm{EMSE}}(\mathcal{B})$, i.e., adding more sensors always helps. That means, f_{EMSE} is maximized by the set of all sensors V. However, in practice, we would like to only select a small set of, e.g., at most k sensors due to bandwidth and power constraints:

$$\mathcal{A}^* = \arg\max_{\mathcal{A}} f_{\mathrm{EMSE}}(\mathcal{A}) \text{ s.t. } |\mathcal{A}| \leq k.$$

Unfortunately, this optimization problem is NP-hard, so we cannot expect to efficiently find the optimal solution. Fortunately, it can be shown [17] that in many settings[1], the function f_{EMSE} satisfies an intuitive diminishing returns property called submodularity. A set function $f : 2^V \to \mathbb{R}$ is called *submodular* if, for all $\mathcal{A} \subseteq \mathcal{B} \subseteq V$ and $s \in V \setminus \mathcal{B}$ it holds that $f(\mathcal{A} \cup \{s\}) - f(\mathcal{A}) \geq f(\mathcal{B} \cup \{s\}) - f(\mathcal{B})$. Many other natural objective functions for sensor selection satisfy submodularity as well [36]. For example, the *sensing region* model where $f_{REG}(\mathcal{A})$ is the total area covered by all sensors $\mathcal{A}$ is submodular. The *detection* model where $f_{DET}(\mathcal{A})$ counts the expected number of targets detected by sensors $\mathcal{A}$ is submodular as well.

A fundamental result of Nemhauser et al. [48] is that for monotone submodular functions, a simple greedy algorithm, which starts with the empty set $\mathcal{A}_0 = \emptyset$ and iteratively adds the element

$$s_k = \arg\max_{s\in V\setminus\mathcal{A}_{k-1}} f(\mathcal{A}_{k-1} \cup \{s\}); \;\; \mathcal{A}_k = \mathcal{A}_{k-1} \cup \{s_k\}$$

which maximally improves the utility obtains a near-optimal solution: For the set $\mathcal{A}_k$ it holds that

$$f(\mathcal{A}_k) \geq (1 - 1/e) \max_{|\mathcal{A}|\leq k} f(\mathcal{A}),$$

i.e., the greedy solution obtains at least a constant fraction of $(1 - 1/e) \approx 63\%$ of the optimal value.

[1] For Gaussian models and conditional suppressorfreeness [17]

One fundamental problem with this offline approach is that it requires the function f to be specified in advance, i.e., before running the greedy algorithm. For the function f_{EMSE}, this means that the probabilistic model $P(\mathcal{X}_V)$ needs to be known in advance. While for some applications some prior data, e.g., from pilot deployments, may be accessible, very often no such prior data is available. This leads to a "chicken-and-egg" problem, where sensors need to be activated to collect data in order to learn a model, but also the model is required to inform the sensor selection. This is akin to the "exploration–exploitation tradeoff" in reinforcement learning [5], where an agent needs to decide whether to explore and gather information about effectiveness of an action, or to exploit, i.e., choose actions known to be effective. In the following, we devise an online monitoring scheme based on this analogy.

2.1.2 The Online Sensor Selection Problem

We now consider the more challenging problem where the objective function is not specified in advance, and needs to be learned during the monitoring task. We assume that we intend to monitor the environment for a number T of time steps (rounds). In each round t, a set S_t of sensors is selected, and these sensors transmit their measurements to a server (base station). The server then determines a sensing quality $f_t(S_t)$ quantifying the utility obtained from the resulting analysis. For example, if our goal is spatial prediction, the server would build a model based on the previously collected sensor data, pick a random sensor s, make prediction for the variable $\mathcal{X}_s$, and then compare the prediction μ_s with the sensor reading x_s. The error $f_t = \sigma_s^2 - (\mu_s - x_s)^2$ is an unbiased estimate of the reduction in EMSE. In the following analysis, we will only assume that the objective functions f_t are bounded (w.l.o.g., take values in $[0, 1]$), monotone, and submodular, and that we have some way of computing $f_t(S)$ for any subset of sensors S. Our goal is to maximize the total reward obtained by the system over T rounds, $\sum_{t=1}^{T} f_t(S_t)$.

We seek to develop a protocol for selecting the sets S_t of sensors at each round, such that after a small number of rounds the average performance of our online algorithm converges to the same performance of the offline strategy (that knows the objective functions). We thus compare our protocol against all strategies that can select a fixed set of k sensors for use in all of the rounds; the best such strategy obtains reward $\max_{S\subseteq V:|S|\leq k} \sum_{t=1}^{T} f_t(S)$. The difference between this quantity and what our protocol obtains is known as its *regret*, and an algorithm is said to be *no-regret* if its average regret tends to zero (or less)[2] as $T \to \infty$.

When $k = 1$, our problem is simply the well-studied *multiarmed bandit* (MAB) problem, for which many no-regret algorithms are known [25]. For general k, because the average of several submodular functions remains submodular, we can apply the result of Nemhauser *et al.* [48] (*cf.*, Sec. 2.1.1) to prove that a simple greedy algorithm obtains a $(1 - 1/e)$ approximation to the optimal offline solution. This is closely related to the problem of maximizing a monotone submodular function subject to a cardinality constraint. Nemhauser *et al.* [48] showed that for the latter problem the simply greedy algorithm obtains a $(1 - 1/e)$ approximation, and Feige [24] showed that this is optimal in the sense that obtaining a

[2]Formally, if R_T is the total regret for the first T rounds, no-regret means $\limsup_{T\to\infty} R_T/T \leq 0$.

$(1 - 1/e + \epsilon)$ approximation for any $\epsilon > 0$ is NP-hard. These facts suggest that we cannot expect any efficient online algorithm to converge to a solution better than
$(1 - 1/e) \max_{S \subseteq V : |S| \leq k} \sum_{t=1}^{T} f_t(S)$. We therefore define the $(1 - 1/e)$-regret of a sequence of (possibly random) sets $\{S_t\}_{t=1}^{T}$ as

$$R_T := (1 - 1/e) \cdot \max_{S \subseteq V : |S| \leq k} \sum_{t=1}^{T} f_t(S) \; - \; \sum_{t=1}^{T} \mathbb{E}\left[f_t(S_t)\right]$$

where the expectation is taken over the distribution for each S_t. We say an online algorithm producing a sequence of sets has *no-*$(1 - 1/e)$*-regret* if $\limsup_{T \to \infty} \frac{R_T}{T} \leq 0$.

2.2 Centralized Online Sensor Selection

Before developing the distributed algorithm for online sensor selection, we will first review a centralized algorithm which is guaranteed to achieve no $(1 - 1/e)$-regret. In Sec. 2.3 we will show how this centralized algorithm can be implemented efficiently in a distributed manner. This algorithm starts with the greedy algorithm for a known submodular function mentioned in Sec. 2.1.1, and adapts it to the online setting. Doing so requires an online algorithm for selecting a *single* sensor as a subroutine, and we review such an algorithm before discussing the centralized algorithm for selecting multiple sensors in Sec. 2.2.1.

Let us first consider the case where $k = 1$, i.e., we would like to select one sensor at each round. This simpler problem can be interpreted as an instance of the multiarmed bandit problem (as introduced in Sec. 2.1.2), where we have one arm for each possible sensor. In this case, the EXP3 algorithm [5] is a centralized solution for no-regret single sensor selection. EXP3 works as follows: It is parameterized by a *learning rate* η, and an *exploration probability* γ. It maintains a set of weights w_s, one for each arm (sensor) s, initialized to 1. At every round t, it will select each arm s with probability

$$p_s = (1 - \gamma) \frac{w_s}{\sum_{s'} w_{s'}} + \frac{\gamma}{n},$$

i.e., with probability γ it explores, picking an arm uniformly at random, and with probability $(1 - \gamma)$ it exploits, picking an arm s with probability proportional to its weight w_s. Once an arm s has been selected, a feedback $r = f_t(\{s\})$ is obtained, and the weight w_s is updated to

$$w_s \leftarrow w_s \exp(\eta r / p_s).$$

Auer et al. [5] showed that with appropriately chosen learning rate η and exploration probability γ it holds that the cumulative regret R_T of EXP3 is $\mathcal{O}(\sqrt{Tn \ln n})$, i.e., the average regret R_T/T converges to zero.

2.2.1 Selecting Multiple Sensors?

In principle, we could interpret the sensor selection problem as a $\binom{n}{k}$-armed bandit problem, and apply existing no-regret algorithms such as EXP3. Unfortunately, this approach does not scale, since the number of arms grows exponentially with k. However, in contrast to the traditional multiarmed bandit problem, where the arms are assumed to have independent payoffs, in the sensor selection case, the utility function is submodular and thus the payoffs are correlated across different sets. Recently, Streeter and Golovin showed how this submodularity can be exploited, and developed a no-$(1 - 1/e)$-regret algorithm for online maximization of submodular functions [59].

The key idea behind their algorithm, OG_{unit}, is to turn the offline greedy algorithm into an online algorithm by replacing the greedy selection of the element s_k that maximizes the benefit $s_k = \arg\max_s f(\{s_1, ..., s_{k-1}\} \cup \{s\})$ by a bandit algorithm. As shown in the pseudocode below, OG_{UNIT} maintains k bandit algorithms, one for each sensor to be selected. At each round t, it selects k sensors according to the choices of the k bandit algorithms $\mathcal{E}_i$ [3]. Once the elements have been selected, the i^{th} bandit algorithm $\mathcal{E}_i$ receives as feedback the incremental benefit $f_t(s_1, \ldots, s_i) - f_t(s_1, \ldots, s_{i-1})$, i.e., how much additional utility is obtained by adding sensor s_i to the set of already selected sensors. Below we define $[m] := \{1, 2, \ldots, m\}$.

Algorithm OG_{UNIT} from [59]:

Initialize k multiarmed bandit algorithms $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_k$,
each with action set V.
For each round $t \in [T]$
  For each stage $i \in [k]$ in parallel
    $\mathcal{E}_i$ selects an action v_i^t
  For each $i \in [k]$ in parallel
    feedback $f_t(\{v_j^t : j \leq i\}) - f_t(\{v_j^t : j < i\})$ to $\mathcal{E}_i$.
  Output $S_t = \{a_1^t, a_2^t, \ldots, a_k^t\}$.

In [58] it is shown that OG_{UNIT} has a $\left(1 - \frac{1}{e}\right)$-regret bound of $\mathcal{O}(kR)$ in this feedback model assuming each $\mathcal{E}_i$ has expected regret at most R. Thus, when using EXP3 as a subroutine, OG_{UNIT} has no-$(1 - 1/e)$-regret.

Unfortunately, EXP3 (and in fact all MAB algorithms with no-regret guarantees for non-stochastic reward functions) require sampling from some distribution with weights associated with the sensors. If n is small, we could simply store these weights on the server, and run the bandit algorithms $\mathcal{E}_i$ there. However, this solution does not scale to large numbers of sensors. Thus the key problem for online sensor selection is to develop a multiarmed bandit algorithm which implements *distributed sampling* across the network, with minimal overhead of communication. In addition, the algorithm needs to be able to maintain the distributions (the weights) associated with each $\mathcal{E}_i$ in a distributed fashion.

[3] Bandits with duplicate choices are handled in Sec. 4.6.1 of [59]

2.3 Distributed Online Sensor Selection

We will now develop DOG, an efficient algorithm for distributed online sensor selection. For now we make the following assumptions:

1. Each sensor $v \in V$ is able to compute its contribution to the utility $f_t(S \cup \{v\}) - f_t(S)$, where S are a subset of sensors that have already been selected.

2. Each sensor can broadcast to all other sensors.

3. The sensors have calibrated clocks and unique, linearly ordered identifiers.

These assumptions are reasonable in many applications: (1) In target detection, for example, the objective function $f_t(S)$ counts the number of targets detected by the sensors S. Once previously selected sensors have broadcasted which targets they detected, the new sensor s can determine how many additional targets have been detected. Similarly, in statistical estimation, one sensor (or a small number of sensors) randomly activates each round and broadcasts its value. After sensors S have been selected and announced their measurements, the new sensor s can then compute the improvement in prediction accuracy over the previously collected data. (2) The assumption that broadcasts are possible may be realistic for dense deployments and fairly long range transmissions. In Sec. 2.4 we will show how assumptions (1) and (2) can be relaxed.

As we have seen in Sec. 2.2, the key insight in developing a centralized algorithm for online selection is to replace the greedy selection of the sensor which maximally improves the total utility over the set of previously selected sensors by a bandit algorithm. Thus, a natural approach for developing a distributed algorithm for sensor selection is to first consider the single sensor case.

2.3.1 Distributed Selection of a Single Sensor

The key challenge in developing a distributed version of EXP3 is to find a way to sample exactly one element from a probability distribution p over sensors in a distributed manner. This problem is distinct from randomized leader election [47], where the objective is to select exactly one element but the element need not be drawn from a specified distribution. We note that under the multi-hop communication model, sampling one element from the uniform distribution given a rooted spanning tree can be done via a simple random walk [40], but that under the broadcast and star network models this approach degenerates to centralized sampling. Our algorithm, in contrast, samples from an arbitrary distribution by allowing sensors to individually decide to activate. Our bottom-up approach also has two other advantages: (1) it is amenable to modification of the activation probabilities based on local observations, as we discuss in Sec. 2.5, and (2) since it does not rely on any global state of the network such as a spanning tree, it can gracefully cope with significant edge or node failures.

A naive distributed sampling scheme. A naive distributed algorithm would be to let each sensor keep track of all activation probabilities p. Then, one sensor (e.g., with the lowest identifier) would broadcast a single random number u uniformly distributed in $[0, 1]$, and the sensor v for which $\sum_{i=1}^{v-1} p_i \leq u < \sum_{i=1}^{v} p_i$ would activate. However, for large sensor network deployments, this algorithm would require each sensor to store a large amount of global information (all activation probabilities p). Instead, each sensor v could store only their own probability mass p_v; the sensors would then, in order of their identifiers, broadcast their probabilities p_v, and stop once the sum of the probabilities exceeds u. This approach only requires a constant amount of local information, but requires an impractical $\Theta(n)$ messages to be sent, and sent sequentially over $\Theta(n)$ time steps.

Distributed multinomial sampling. In this section we present a protocol that requires only $\mathcal{O}(1)$ messages in expectation, and only a constant amount of local information.

For a sampling procedure with input distribution p, we let $\hat{p}$ denote the resulting distribution, where in all cases at most one sensor is selected, and nothing is selected with probability $1 - \sum_v \hat{p}_v$. A simple approach towards distributed sampling would be to activate each sensor $v \in V$ *independently* from each other with probability p_v. While in expectation, exactly one sensor is activated, with probability $\prod_v (1 - p_v) > 0$ no sensor is activated; also since sensors are activated independently, there is a nonzero probability that more than one sensor is activated. Using a synchronized clock, the sensors could determine if no sensor is activated. In this case, they could simply repeat the selection procedure until at least one sensor is activated. One naive approach would be to repeat the selection procedure until exactly one sensor is activated. However with two sensors and $p_1 = \varepsilon, p_2 = 1 - \varepsilon$ this algorithm yields $\hat{p}_1 = \varepsilon^2/(1 - 2\varepsilon + 2\varepsilon^2) = \mathcal{O}(\varepsilon^2)$, so the first sensor is severely underrepresented. Another simple protocol would be to select exactly one sensor uniformly at random from the set of activated sensors, which can be implemented using few messages.

The Simple Protocol:

For each sensor v in parallel

Sample $X_v \sim \text{Bernoulli}(p_v)$.

If $(X_v = 1)$, X_v activates.

All active sensors S coordinate to select a single sensor uniformly at random from S, e.g., by electing the minimum ID sensor in S to do the sampling.

It is not hard to show that with this protocol, for all sensors v,

$$\hat{p}_v = p_v \cdot \mathbb{E}\left[\frac{1}{|S|} \,\middle|\, v \in S\right] \geq p_v / \mathbb{E}\left[|S| \mid v \in S\right] \geq p_v/2$$

by appealing to Jensen's inequality. Since $\hat{p}_v \leq p_v$, we find that this simple protocol maintains a ratio $r_v := \hat{p}_v/p_v \in [\frac{1}{2}, 1]$. Unfortunately, this analysis is tight, as can be seen from the example with two sensors and $p_1 = \varepsilon, p_2 = 1 - \varepsilon$.

To improve upon the simple protocol, first consider running it on an example with $p_1 = p_2 = \cdots = p_n = 1/n$. Since the protocol behaves exactly the same under permutations of sensor labels, by symmetry we have $\hat{p}_1 = \hat{p}_2 = \cdots = \hat{p}_n$, and thus $r_i = r_j$ for all i, j. Now consider an input distribution p where there exists integers N and $k_1, k_2, \ldots, k_n$ such that $p_v = k_v/N$ for all v. Replace each v with k_v fictitious sensors, each with probability mass $1/N$, and each with a label indicating v. Run the simple protocol with the fictitious sensors, selecting a fictitious sensor v', and then actually select the sensor indicated by the label of v'. By symmetry this process selects each fictitious sensor with probability $(1-\beta)/N$, where β is the probability that nothing at all is selected, and thus the process selects sensor v with probability $k_v(1-\beta)/N = (1-\beta)p_v$ (since at most one fictitious sensor is ever selected).

We may thus consider the following improved protocol which incorporates the above idea, simulating this modification to the protocol exactly when $p_v = k_v/N$ for all v.

The Improved Protocol(N):

For each sensor v in parallel

Sample $X_v \sim \text{Binomial}(\lceil N \cdot p_v \rceil, 1/N)$.

If $(X_v \geq 1)$, then activate sensor v.

From the active sensors S, select sensor v with probability $X_v / \sum_{v' \in S} X_{v'}$.

This protocol ensures the ratios $r_v := \hat{p}_v/p_v$ are the same for all sensors, provided each p_v is a multiple of $1/N$. Assuming the probabilities are rational, there will be a sufficiently large N to satisfy this condition. To reduce $\beta := \mathbf{Pr}[S = \emptyset]$ in the simple protocol, we may sample each X_v from $\text{Bernoulli}(\alpha \cdot p_v)$ for any $\alpha \in [1, n]$. The symmetry argument remains unchanged. This in turn suggests sampling X_v from $\text{Binomial}(\lceil N \cdot p_v \rceil, \alpha/N)$ in the improved protocol. Taking the limit as $N \to \infty$, the binomial distribution becomes Poisson, and we obtain the desired protocol.

The Poisson Multinomial Sampling (PMS) Protocol(α):

Same as the improved protocol, except each

sensor v samples $X_v \sim \text{Poisson}(\alpha p_v)$

Straight-forward calculation shows that

$$\mathbf{Pr}[S = \emptyset] = \prod_v \exp\{-\alpha \cdot p_v\} = \exp\{-\sum_v \alpha \cdot p_v\} = e^{-\alpha}$$

Let C be the number of messages. Then

$$\mathbb{E}[C] = \sum_v \mathbf{Pr}[X_v \geq 1] = \sum_v (1 - e^{-\alpha p_v}) \leq \sum_v \alpha p_v = \alpha$$

Here we have used linearity of expectation, and $1 + x \leq e^x$ for all $x \in \mathbb{R}$. In summary, we have the following result about our protocol:

Proposition 1. *Fix any fixed p and $\alpha > 0$. The PMS Protocol always selects at most one sensor, ensures*

$$\forall v : \mathbf{Pr}[v \text{ selected}] = (1 - e^{-\alpha})p_v$$

and requires no more than α messages in expectation.

In order to ensure that exactly one sensor is selected, whenever $S = \emptyset$ we can simply rerun the protocol with fresh random seeds as many times as needed until S is non-empty. Using $\alpha = 1$, this modification will require only $\mathcal{O}(1)$ messages in expectation and at most $\mathcal{O}(\log n)$ messages with high probability in the broadcast model. We can combine this protocol with EXP3 to get the following result.

Theorem 2. *In the broadcast model, running* EXP3 *using the PMS Protocol with $\alpha = 1$, and rerunning the protocol whenever nothing is selected, yields exactly the same regret bound as standard* EXP3*, and in each round at most $e/(e-1) + 2 \approx 3.582$ messages are broadcast in expectation.*

The regret bound for EXP3 is $\mathcal{O}(\sqrt{\mathsf{OPT} n \log n})$, where OPT is the total reward of the best action. Our variant simulates EXP3, and thus has identical regret.

Remark. Running our variant of EXP3 requires that each sensor know the number of sensors, n, in order to compute its activation probability. If each sensor v has only a reasonable estimate of n_v of n, however, our algorithm still performs well. For example, it is possible to prove that if all of the sensors have the same estimate $n_v = cn$ for some constant $c > 0$, then the upper bound on expected regret, $R(c)$, grows as $R(c) \approx R(1) \cdot \max\{c, 1/c\}$. The expected number of activations in this case increases by at most $\left(\frac{1}{c} - 1\right)\gamma$. In general underestimating n leads to more activations, and underestimating or overestimating n can lead to more regret. This graceful degradation of performance with respect to the error in estimating n holds for all of our algorithms.

2.3.2 Distributed Online Greedy

We now use our single sensor selection algorithm to develop our main algorithm, the Distributed Online Greedy algorithm (DOG). It is based on the distributed implementation of EXP3 using the PMS Protocol. Suppose we would like to select k sensors at each round t. Each sensor v maintains k weights $w_{v,1}, \ldots, w_{v,k}$ and normalizing constants $Z_{v,1}, \ldots, Z_{v,k}$. The algorithm proceeds in k stages, synchronized using the common clock. In stage i, a single sensor is selected using the PMS Protocol applied to the distribution $(1-\gamma)w_{v,i}/Z_{v,i} + \gamma/n$. Suppose sensors $S = \{v_1, \ldots, v_{i-1}\}$ have been selected in stages 1 through $i-1$. The sensor v selected at stage i then computes its local rewards $\pi_{v,i}$ using the utility function $f_t(S \cup \{v_i\}) - f_t(S)$. It then computes its new weight

$$w'_{v,i} = w_{v,i} \exp(\eta \pi_{v,i}/p_{v,i}),$$

and broadcasts the difference between its new and old weights $\Delta_{v,i} = w'_{v,i} - w_{v,i}$. All sensors then update their i^{th} normalizers using $Z_{v,i} \leftarrow Z_{v,i} + \Delta_{v,i}$. Fig. 1 presents the pseudo-code of the DOG algorithm. Thus given Theorem 12 of [58] we have the following result about the DOG algorithm:

Theorem 3. *The* DOG *algorithm selects, at each round* t *a set* $S_t \subseteq V$ *of* k *sensors such that*

$$\frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} f_t(S_t)\right] \geq \frac{1-\frac{1}{e}}{T} \max_{|S|\leq k} \sum_{t=1}^{T} f_t(S) - O\left(k\sqrt{\frac{n\log n}{T}}\right).$$

In expectation, only $\mathcal{O}(k)$ *messages are exchanged each round.*

Algorithm 1: The Distributed Online Greedy Algorithm

Input: $k \in \mathbb{N}$, a set V, and $\alpha, \gamma, \eta \in \mathbb{R}_{>0}$. Reasonable defaults are any $\alpha \in [1, \ln|V|]$, and $\gamma = \eta = \min\left(1, (|V|\ln|V|/g)^{1/2}\right)$, where g is a guess for the maximum cumulative reward of any single sensor [5].

Initialize $w_{v,i} \leftarrow 1$ and $Z_{v,i} \leftarrow |V|$ for all $v \in V$, $i \in [k]$. Let $\rho(x,y) := (1-\gamma)\frac{x}{y} + \frac{\gamma}{|V|}$.

for each round $t = 1, 2, 3, \ldots$ **do**

  Initialize $S_{v,t} \leftarrow \emptyset$ for each v in parallel.

  for each stage $i \in [k]$ **do**

    for each sensor $v \in V$ in parallel **do**

      repeat

        Sample $X_v \sim \text{Poisson}(\alpha \cdot \rho(w_{v,i}, Z_{v,i}))$.

        if $(X_v \geq 1)$ **then**

          Broadcast $\langle \text{sampled } X_v, \text{id}(v)\rangle$; Receive messages from sensors S. (Include $v \in S$ for convenience).

          if $\text{id}(v) = \min_{v' \in S} \text{id}(v')$ **then**

            Select exactly one element v_{it} from S such that each v' is selected with probability $X_{v'} / \sum_{u \in S} X_u$.

            Broadcast $\langle \text{select } \text{id}(v_{it})\rangle$.

          Receive message $\langle \text{select } \text{id}(v_{it})\rangle$.

          if $\text{id}(v) = \text{id}(v_{it})$ **then**

            Observe $f_t(S_{v,t} + v)$; $\pi \leftarrow f_t(S_{v,t} + v) - f_t(S_{v,t})$;

            $\Delta_v \leftarrow w_{v,i}(\exp\{\eta \cdot \pi / \rho(w_{v,i}, Z_{v,i})\} - 1)$; $Z_{v,i} \leftarrow Z_{v,i} + \Delta_v$;

            $w_v \leftarrow w_v + \Delta_v$; Broadcast $\langle \text{weight update } \Delta_v, \text{id}(v)\rangle$.

        if receive message $\langle \text{weight update } \Delta, \text{id}(v_{it})\rangle$ **then** $S_{v,t} \leftarrow S_{v,t} \cup \{v_{it}\}$;

        $Z_{v,i} \leftarrow Z_{v,i} + \Delta$;

        ;

      until v receives a message of type $\langle \text{select } \text{id}\rangle$;

Output: At the end of each round t each sensor has an identical local copy $S_{v,t}$ of the selected set S_t.

2.4 Communication to the cloud: The Star Network Model

In some applications, the assumption that sensors can broadcast messages to all sensors may be unrealistic. Furthermore, in some applications sensors may not be able to compute the marginal benefits $f_t(S \cup \{s\}) - f_t(S)$ (since this calculation may be computationally complex). In this section, we analyze LAZYDOG, a variant of

our DOG algorithm, which replace the above assumptions by the assumption that there is a dedicated base station[4] available which computes utilities and which can send non-broadcast messages to individual sensors.

We make the following assumptions:

1. Every sensor stores its probability mass p_v with it, and can only send messages to and receive messages from the base station.

2. The base station is able, after receiving messages from a set S of sensors, to compute the utility $f_t(S)$ and send this utility back to the active sensors.

These conditions arise, for example, when cell phones in participatory sensor networks can contact the base station, but due to privacy constraints cannot directly call other phones. We do not assume that the base station has access to all weights of the sensors – we will only require the base station to have $\mathcal{O}(k + \log n)$ memory. In the fully distributed algorithm DOG that relies on broadcasts, it is easy for the sensors to maintain their normalizers $Z_{v,i}$, since they receive information about rewards from all selected sensors. The key challenge when removing the broadcast assumption is to maintain the normalizers in an appropriate manner.

2.4.1 Lazy Renormalization & Distributed EXP3

EXP3 (and all MAB with no-regret guarantees against arbitrary reward functions) must maintain a distribution over actions, and update this distribution in response to feedback about the environment. In EXP3, each sensor v requires only $w_v(t)$ and a normalizer $Z(t) := \sum_{v'} w_{v'}(t)$ to compute $p_v(t)$[5]. The former changes only when v is selected. In the broadcast model the latter can simply be broadcast at the end of each round. In the star network model (or, more generally in multi-hop models), standard flooding echo aggregation techniques could be used to compute and distribute the new normalizer, though with high communication cost. We show that a lazy renormalization scheme can significantly reduce the amount of communication needed by a distributed bandit algorithm *without altering its regret bounds whatsoever*. Thus our lazy scheme is complementary to standard aggregation techniques.

Our lazy renormalization scheme for EXP3 works as follows. Each sensor v maintains its weight $w_v(t)$ and an estimate $Z_v(t)$ for $Z(t) := \sum_{v'} w_{v'}(t)$, Initially, $w_v(0) = 1$ and $Z_v(0) = n$ for all v. The central server stores $Z(t)$. Let

$$\rho(x, y) := (1 - \gamma)\frac{x}{y} + \frac{\gamma}{n}.$$

Each sensor then proceeds to activate as in the sampling procedure of Sec. 2.3.1 as if its probability mass in round t were $q_v = \rho(w_v(t), Z_v(t))$ instead of its true value of $\rho(w_v(t), Z(t))$. A single sensor is selected by the server with respect to the true value $Z(t)$, resulting in a selection from the desired distribution. Moreover,

[4]Though the existence of such a base station means the protocol is not completely distributed, it is realistic in sensor network applications where the sensor data needs to be accumulated somewhere for analysis.

[5]We let $x(t)$ denote the value of variable x at the start of round t, to ease analysis. We do not actually need to store the historical values of the variables over multiple time steps.

v's estimate $Z_v(t)$ is only updated on rounds when it communicates with the server under these circumstances. This allows the estimated probabilities of all of the sensors to sum to more than one, but has the benefit of significantly reducing the communication cost in the star network model under certain assumptions. We call the result *Distributed EXP3*, give its pseudocode for round t in Fig. 2.

Since the sensors underestimate their normalizers, they may activate more frequently than in the broadcast model. Fortunately, the amount of "overactivation" remains bounded.

Theorem 4. *The number of sensor activations in any round of the Distributed* EXP3 *algorithm is at most* $\alpha + (e-1)$ *in expectation and* $\mathcal{O}(\alpha + \log n)$ *with high probability, and the number of messages is at most twice the number of activations.*

Unfortunately, there is still an $e^{-\alpha}$ probability of nothing being selected. To address this, we can set $\alpha = c \ln n$ for some $c \geq 1$, and if nothing is selected, transmit a message to each of the n sensors to rerun the protocol.

Corollary 5. *There is a distributed implementation of* EXP3 *that always selects a sensor in each round, has the same regret bounds as standard* EXP3*, ensures that the number of sensor activations in any round is at most* $\ln n + \mathcal{O}(1)$ *in expectation or* $\mathcal{O}(\log n)$ *with high probability, and in which the number of messages is at most twice the number of activations.*

2.4.2 Lazy Renormalization

Once we have the distributed EXP3 variant described above, we can use it for the bandit subroutines in the OG_{UNIT} algorithm (*cf.* Sec. 2.2.1). We call the result the LAZYDOG algorithm, due to its use of lazy renormalization. The lazy distributed EXP3 still samples sensors from the same distribution as the regular distributed EXP3, so LAZYDOG has precisely the same performance guarantees with respect to $\sum_t f_t(S_t)$ as DOG. It works in the star network communication model, and requires few messages or sensor activations. Corollary 5 immediately implies the following result.

Corollary 6. *The number of sensors that activate each round in* LAZYDOG *is at most* $k \ln n + \mathcal{O}(k)$ *in expectation and* $\mathcal{O}(k \log n)$ *with high probability, the number of messages is at most twice the number of activations, and the* $(1 - 1/e)$*-regret of* LAZYDOG *is the same as* DOG*.*

2.5 Observation-dependent sampling

Theorem 3 states that DOG is guaranteed to do nearly as well as the offline greedy algorithm run on an instance with objective function $f_\Sigma := \sum_t f_t$. Thus the reward of DOG is asymptotically near-optimal on average. In many applications, however, we would like to perform well on rounds with "atypical" objective functions. For example, in an outbreak detection application, we would like to get very good data on rounds

Algorithm 2: Distributed EXP3: the PMS Protocol(α) with lazy renormalization, applied to EXP3

Input: Parameters $\alpha, \eta, \gamma \in \mathbb{R}_{>0}$, sensor set V.
Let $\rho(x, y) := (1-\gamma)\frac{x}{y} + \frac{\gamma}{|V|}$.
Sensors:
foreach sensor v in parallel **do**
  Sample r_v uniformly at random from $[0, 1]$.
  if $(r_v \geq 1 - \alpha \cdot \rho(w_v(t), Z_v(t))$ **then**
    Send $\langle r_v, w_v(t)\rangle$ to the server.
    Receive message $\langle Z, w\rangle$ from server.
    $Z_v(t+1) \leftarrow Z$; $w_v(t+1) \leftarrow w$.
  else $Z_v(t+1) \leftarrow Z_v(t)$; $w_v(t+1) \leftarrow w_v(t)$.
  ;
Server:
Receive messages from a set S of sensors.
if $S = \emptyset$ **then** Select nothing and wait for next round.;
else foreach sensor $v \in S$ **do**
  $Y_v \leftarrow \min\{x : \mathbf{Pr}[X \leq x] \geq r_v\}$, where $X \sim \text{Poisson}(\alpha \cdot \rho(w_v(t), Z(t)))$.
  Select v with probability $Y_v / \sum_{v' \in S} Y_{v'}$.
  Observe the payoff π for the selected sensor v^*; $w_{v^*}(t+1) \leftarrow w_{v^*}(t) \cdot \exp\{\eta\pi/\rho(w_{v^*}(t), Z(t))\}$;
  $Z(t+1) \leftarrow Z(t) + w_{v^*}(t+1) - w_{v^*}(t)$;
  for each $v \in S \setminus v^*$ **do** $w_v(t+1) \leftarrow w_v(t)$;
  ;
  for each $v \in S$ **do** Send $\langle Z(t+1), w_v(t+1)\rangle$ to v.
  ;
;

with significant events, even if the nearest sensors typically report "boring" readings that contribute very little to the objective function. For now, suppose that we are only running a single MAB instance to select a single sensor in each round. If we have access to a black-box for evaluating f_t on round t, then we can perform well on atypical rounds at the cost of some additional communication by having each sensor v take a local reading of its environment and estimate its payoff $\bar{\pi} = f_t(\{v\})$ if selected. This value, which serves as a measure of how interesting its input is, can then be used to decide whether to boost v's probability for reporting its sensor reading to the server. In the simplest case, we can imagine that each v has a threshold τ_v such that v activates with probability 1 if $\bar{\pi} \geq \tau_v$, and with its normal probability otherwise. In the case where we select $k > 1$ sensors in each round, each sensor can have a threshold for each of the k stages, where in each stage it computes $\bar{\pi} = f_t(S \cup \{v\}) - f_t(S)$ where S is the set of currently selected sensors. Since the activation probability only goes up, we can retain the performance guarantees of DOG if we are careful to adjust the feedback properly.

Ideally, we wish that the sensors learn what their thresholds τ_v should be. We treat the selection of τ_v in each round as an online decision problem that each v must play. We construct a particular game that the sensors play, where the strategies are the thresholds (suitably discretized), there is an *activation cost* c_v that v pays if $\bar{\pi}_v \geq \tau_v$, and the payoffs are defined as follows: Let $\pi_v = f_t(S \cup \{v\}) - f_t(S)$ be the marginal benefit of selecting v given that sensor set S has already been selected. Let A be the set of sensors that activate in

the current iteration of the game, and let $\max\left(\pi_{(A\setminus v)}\right) := \max\left(\pi_{v'} : v' \in A \setminus \{v\}\right)$. The particular reward function ψ_v we choose for each sensor v for each iteration of the game is

$$\psi_v(\tau) = \begin{cases} c_v - \max\left(\pi_v - \max\left(\pi_{(A\setminus v)}\right), 0\right) & \text{if } \bar{\pi} < \tau \\ \max\left(\pi_v - \max\left(\pi_{(A\setminus v)}\right), 0\right) - c_v & \text{if } \bar{\pi} \geq \tau \end{cases}$$

based on empirical performance. Thus, if a sensor activates ($\bar{\pi} \geq \tau$), its payoff is the improvement over the best payoff $\pi_{v'}$ among all sensors $v' \in A$ minus its activation cost. In case multiple sensors activate, the highest reward is retained.

In the broadcast model where each sensor can compute its marginal benefit, we can use any standard no-regret algorithm for combining expert advice, such as *Randomized Weighted Majority* (WMR) [41], to play this game and obtain no regret guarantees[6] for selecting τ_v. In our context a sensor using WMR simply maintains weights $w(\tau_i) = \exp\left(\eta \cdot \psi_{\text{total}}(\tau_i)\right)$ for each possible threshold τ_i, where $\eta > 0$ is a learning parameter, and $\psi_{\text{total}}(\tau_i)$ is the total cumulative reward for playing τ_i in every round so far. On each step each threshold is picked with probability proportional to its weight. In the more restricted star network model, we can use a modification of WMR that feeds back unbiased estimates for $\psi_t(\tau_i)$, the payoff to the sensor for using a threshold of τ_i in round t, and thus obtains reasonably good estimates of $\psi_{\text{total}}(\tau_i)$ after many rounds. We give pseudocode in Fig. 3. In it, we assume that an activated sensor can compute the reward of playing any threshold.

Algorithm 3: Selecting activation thresholds for a sensor

Input: parameter $\eta > 0$, threshold set $\{\tau_i : i \in [m]\}$
Initialize $w(\tau_i) \leftarrow 1$ for all $i \in [m]$.
for each round $t = 1, 2, \ldots$ **do**
  Select τ_i with probability $w(\tau_i) / \sum_{j=1}^{m} w(\tau_j)$.
  if sensor activates **then**
    Let $\psi(\tau_i)$ be the reward for playing τ_i in this round of the game. Let $q(\tau_i)$ be the total probability of activation conditioned on τ_i being selected (including the activation probability that does not depend on local observations.)
    for each threshold τ_i **do**
      $w(\tau_i) \leftarrow w(\tau_i) \exp\left(\eta\psi(\tau_i)/q(\tau_i)\right)$.

We incorporate these ideas into the DOG algorithm, to obtain what we call the *Observation-Dependent Distributed Online Greedy* algorithm (OD-DOG). In the extreme case that $c_v = 0$ for all v the sensors will soon set their thresholds so low that each sensor activates in each round. In this case OD-DOG will exactly simulate the offline greedy algorithm run on each round. In other words, if we let $G(f)$ be the result of running the offline greedy algorithm on the problem

$$\arg\max\left\{f(S) : S \subset V,\ |S| \leq k\right\}$$

[6] We leave it as an open problem to determine if the outcome is close to optimal when all sensors play low regret strategies (i.e., is the *price of total anarchy* [7] small in any variant of this game with a reasonable way of splitting the value from the information?)

then OD-DOG will obtain a value of $\sum_t f_t(G(f_t))$; in contrast, DOG gets roughly $\sum_t f_t(G(\sum_t f_t))$, which may be significantly smaller. Note that Feige's result [24] implies that the former value is the best we can hope for from efficient algorithms (assuming P $\neq$ NP). Of course, querying each sensor in each round is impractical when querying sensors is expensive. In the other extreme case where $c_v = \infty$ for all v, OD-DOG will simulate DOG after a brief learning phase. In general, by adjusting the activation costs c_v we can smoothly trade off the cost of sensor communication with the value of the resulting data.

2.6 Experiments in sensor selection

In this section, we evaluate our DOG algorithm on several real-world sensing problems.

2.6.1 Data Sets

Temperature data. In our first data set, we analyze temperature measurements from the network of 46 sensors deployed at Intel Research Berkeley. Our training data consisted of samples collected at 30 second intervals on 3 consecutive days (starting Feb. 28th 2004), the testing data consisted of the corresponding samples on the two following days. The objective functions used for this application are based on the expected reduction in mean squared prediction error f_{EMSE}, as introduced in Sec. 2.1.

Precipitation data. Our second data set consists of precipitation data collected during the years 1949 - 1994 in the states of Washington and Oregon [67]. Overall 167 regions of equal area, approximately 50 km apart, reported the daily precipitation. To ensure the data could be reasonably modeled using a Gaussian process we applied preprocessing as described in [38]. As objective functions we again use the expected reduction in mean squared prediction error f_{EMSE}.

Water network monitoring. Our third data set is based on the application of monitoring for outbreak detection. Consider a city water distribution network for delivering drinking water to households. Accidental or malicious intrusions can cause contaminants to spread over the network, and we want to install sensors to detect these contaminations as quickly as possible. In August 2006, the Battle of Water Sensor Networks (BWSN) [21] was organized as an international challenge to find the best sensor placements for a real metropolitan water distribution network, consisting of 12,527 nodes. In this challenge, a set of intrusion scenarios is specified, and for each scenario a realistic simulator provided by the EPA is used to simulate the spread of the contaminant for a 48 hour period. An intrusion is considered detected when one selected node shows positive contaminant concentration. The goal of BWSN was to minimize impact measures, such as the expected population affected, which is calculated using a realistic disease model. For a security-critical sensing task such as protecting drinking water from contamination, it is important to develop sensor selection schemes that maximize detection performance even in adversarial environments (i.e., where an adversary

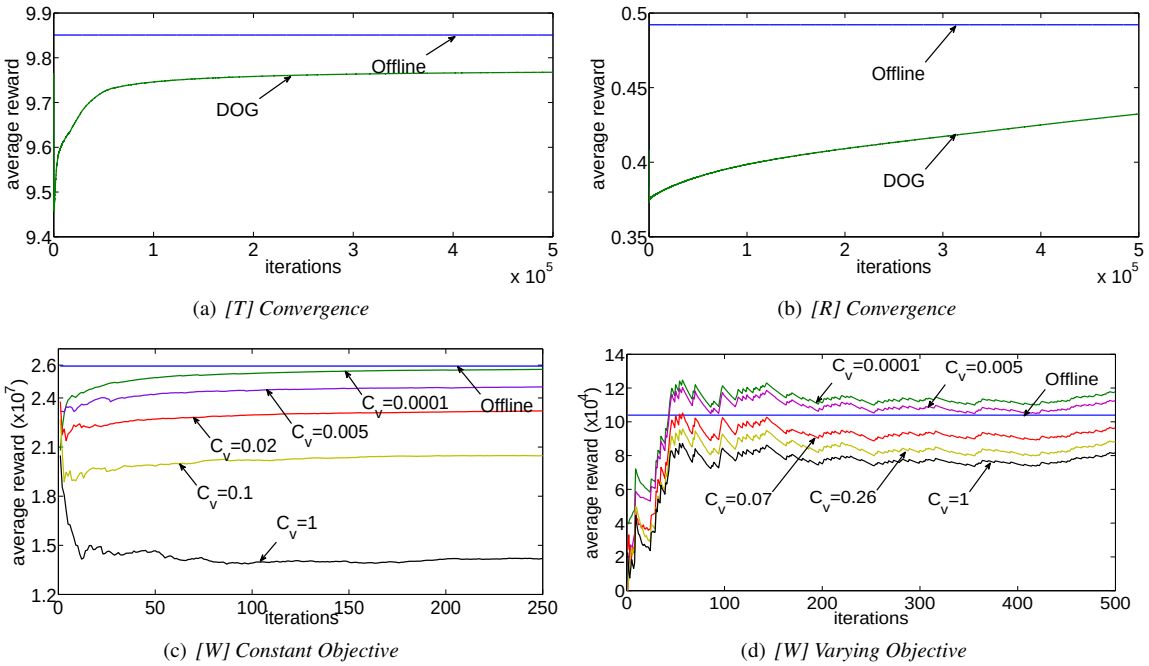
picks the contamination strategy knowing our network deployment and selection algorithm). The algorithms developed here apply to such adversarial settings. We reproduce the experimental setup detailed in [37]. For each contamination event i, we define a separate submodular objective function $f_i(S)$ that measures the expected population protected when detecting the contamination from sensors S. In [37], Krause et al. showed that the functions $f_i(A)$ are monotone submodular functions.

2.6.2 Convergence Experiments

In our first set of experiments, we analyzed the convergence of our DOG algorithm. For both the temperature [T] and precipitation [R] data sets, we first run the offline greedy algorithm using the f_{EMSE} objective function to pick $k = 5$ sensors. We compare its performance to the DOG algorithm, where we feed back the same objective function at every round. We use an exploration probability $\gamma = 0.01$ and a learning rate inversely proportional to the maximum achievable reward $f_{\mathrm{EMSE}}(V)$. Fig. 2.1(a) presents the results for the temperature data set. Note that even after only a small number of rounds (≈ 100), the algorithm obtains 95% of the performance of the offline algorithm. After about 13,000 iterations, the algorithm obtains 99% of the offline performance, which is the best that can be expected with a .01 exploration probability. Fig. 2.1(b) show the same experiment on the precipitation data set. In this more complex problem, after 100 iterations, 76% of the offline performance is obtained, which increases to 87% after 500,000 iterations.

2.6.3 Observation Dependent Activation

We also experimentally evaluate our OD-DOG algorithm with observation specific sensor activations. We choose different values for the activation cost c_v, which we vary as multiples of the total achievable reward. The activation cost c_v lets us smoothly trade off the average number of sensors activating each round and the average obtained reward. The resulting activation strategies are used to select a subset of size $k = 10$ from a collection of 12,527 sensors. Fig. 2.1(c) presents rates of convergence using the OD-DOG algorithm under a fixed objective function which considers all contamination events. In Fig. 2.1(d), convergence rates are presented under a varying objective function, which selects a different contamination event on each round. For low activation costs, the performance quickly converges to or exceeds the performance of the offline solution. Even under the lowest activation costs in our experiments, the average number of extra activations per stage in the OD-DOG algorithm is at most 5. These results indicate that observation specific activation can lead to drastically improved performance at small additional activation cost.



(a) *[T] Convergence* (b) *[R] Convergence*

(c) *[W] Constant Objective* (d) *[W] Varying Objective*

Figure 2.1: Experimental results on [T] Temperature data, [R] precipitation data and [W] water distribution network data.

Chapter 3

Decentralized Anomaly Detection in Community Networks

In contrast to the previous chapter which focused on repeatedly selecting sets of sensors to observe the same (or similar) environment, this chapter considers the problem of detecting sudden and *rare* events such as earthquakes or other disasters using data from a large community network. Due to the unavailability of data characterizing the rare events, the approach described here is based on anomaly detection. First, each sensor learns a model of normal sensor data (e.g., acceleration patterns experienced by smartphones under typical manipulation). Each sensor then independently detects unusual observations (which are considered unlikely with respect to the model), and notifies a central server, or *fusion center*. The fusion center then decides whether a rare event has occurred or not based on the received messages. Our approach is grounded in the theory of decentralized detection, and we characterize its performance accordingly. In particular, we show how sensors can learn decision rules that allow us to control system-level false positive rates and bound the amount of required communication in a principled manner while simultaneously maximizing the detection performance.

To better study the challenges and opportunities of community sensing, we implement our approach in the *Community Seismic Network (CSN)*. The goal of the CSN system is to detect seismic motion using accelerometers in smartphones and other consumer devices (Figure 3.4), and issue real-time early-warning of seismic hazards. The duration of the warning is the time between a person or device receiving the alert and the onset of significant shaking; this duration depends on the distance between the location of initial shaking and the location of the receiving device, and on delays within the network and fusion center. Warnings of up to tens of seconds are possible [3], and even warnings of a few seconds help in stopping elevators, slowing trains, and closing gas valves. Since false alarms can have high costs, it is important to limit the false positive rate of the system.

Using community-based sensors for earthquake early warning is particularly challenging due to the large variety of sensor types, sensor locations, and ambient noise characteristics. For example, a sensor near a construction site will have different behavior than a sensor in a quiet zone. Moreover, sensor behavior may

change over time; for example, construction may start in some places and stop in others. With thousands of sensors, one cannot expect to know the precise characteristics of each sensor at each point in time; these characteristics have to be deduced by algorithms. A system that scales to tens of thousands or millions of sensors must limit the rate of message traffic so that it can be handled efficiently by the network and fusion center. For example, one million phones would produce approximately 30 Terabytes of accelerometer data each day. Another key challenge is to develop a system infrastructure that has low response time even under peak load (messages sent by millions of phones during an earthquake). Moreover, the Internet and computers in a quake zone are likely to fail with the onset of intensive shaking. So, data from sensors must be sent out to a distributed, resilient system that has data centers outside the quake zone. The CSN uses cloud-computing based sensor fusion to cope with these challenges. We report our initial experience with the CSN, and experimentally evaluate our detection approach based on data from a pilot deployment. Our results, including data from shaketable experiments that allow us to mechanically play back past earthquakes, indicate the effectiveness of our approach in distinguishing seismic motion from accelerations due to normal daily manipulation. They also provide evidence for the feasibility of earthquake early warning using a dense network of cell phones.

In summary, this chapter describes:

- a novel approach for online decentralized anomaly detection,
- a theoretical analysis, characterizing the performance of our detection approach,
- an implementation of our approach in the Community Seismic Network, involving smartphones, USB MEMS accelerometers and cloud-computing based sensor fusion, and
- a detailed empirical evaluation of our approach characterizing the achievable detection performance when using smartphones to detect earthquakes.

3.1 Problem Statement

We consider the problem of decentralized detection of rare events, such as earthquakes, under constraints on the number of messages sent by each sensor. Specifically, a set of N sensors make repeated observations $\mathbf{X}_t = (X_{1,t}, \ldots, X_{N,t})$ from which we would like to detect the occurrence of an event $E_t \in \{0, 1\}$. Here, $X_{s,t}$ is the measurement of sensor s at time t, and $E_t = 1$ iff there is an event (e.g., an earthquake) at time t. $X_{s,t}$ may be a scalar (e.g., acceleration), or a vector of features containing information about Fourier frequencies, moments, etc. during a sliding window of data (see Section 3.4.2 for a discussion of features that we use in our system).

We are interested in the decentralized setting, where each sensor s analyzes its measurements $X_{s,t}$, and sends a message $M_{s,t}$ to the fusion center. Here we will focus on binary messages (i.e., each sensor gets to vote on whether there is an event or not). In this case, $M_{s,t} = 1$ means that sensor s at time t estimates that an event happened; $M_{s,t} = 0$ means that sensor s at time t estimates that no event happened at that time. For large networks, we want to minimize the number of messages sent. Since the events are assumed to be rare,

we only need to send messages (that we henceforth call *picks*) for $M_{s,t} = 1$; sending no message implies $M_{s,t} = 0$. Based on the received messages, the fusion center then decides how to respond: It produces an estimate $\widehat{E}_t \in \{0, 1\}$. If $\widehat{E}_t = E_t$, it makes the correct decision (*true positive* if $E_t = 1$ or *true negative* if $E_t = 0$). If $\widehat{E}_t = 0$ when $E_t = 1$, it missed an event and thus produced a *false negative*. Similarly, if $\widehat{E}_t = 1$ when $E_t = 0$, it produced a *false positive*. False positives and false negatives can have very different costs. In our earthquake example, a false positive could lead to incorrect warning messages sent out to the community and consequently lead to inappropriate execution of remedial measures. On the other hand, false negatives could lead to missed opportunities for protecting infrastructure and saving lives. In general, our goal will be to minimize the frequency of false negatives while constraining the (expected) frequency of false positives to a tolerable level (e.g., at most one false alarm per year).

Classical Decentralized Detection. How should each sensor, based on its measurements $X_{s,t}$, decide when to *pick* (send $M_{s,t} = 1$)? The traditional approach to decentralized detection assumes that we know how likely particular observations $X_{s,t}$ are, in case of an event occurring or not occurring. Thus, it assumes we have access to the conditional probabilities $\mathbb{P}[X_{s,t} \mid E_t = 0]$ and $\mathbb{P}[X_{s,t} \mid E_t = 1]$. In this case, under the common assumptions that the sensors' measurements are independent conditional on whether there is an event or not, it can be shown that the optimal strategy is to perform *hierarchical hypothesis testing* [62]: we define two thresholds τ, τ', and let $M_{s,t} = 1$ iff

$$\frac{\mathbb{P}[X_{s,t} \mid E_t = 1]}{\mathbb{P}[X_{s,t} \mid E_t = 0]} \geq \tau. \tag{3.1}$$

i.e., if the likelihood ratio exceeds τ. Similarly, the fusion center sets $\widehat{E}_t = 1$ iff

$$\frac{\mathrm{Bin}(S_t; p_1; N)}{\mathrm{Bin}(S_t; p_0; N)} \geq \tau', \tag{3.2}$$

where $S_t = \sum_s M_{s,t}$ is the number of picks at time t; $p_\ell = \mathbb{P}[M_{s,t} = 1 \mid E_t = \ell]$ is the sensor-level true ($\ell = 1$) and false ($\ell = 0$) positive rate respectively; and $\mathrm{Bin}(\cdot, p, N)$ is the probability mass function of the Binomial distribution. Asymptotically optimal decision performance in either a Bayesian or Neyman-Pearson framework can be obtained by using the decision rules (3.1) and (3.2) with proper choice of the thresholds τ and τ' [62].

There has also been work in *quickest detection* or *change detection* (*cf.*, [52] for an overview), where the assumption is that there is some time point t_0 at which the event occurs; $X_{s,t}$ will be distributed according to $\mathbb{P}[X_{s,t} \mid E_t = 0]$ for all $t < t_0$, and according to $\mathbb{P}[X_{s,t} \mid E_t = 1]$ for all $t \geq t_0$. In change detection, the system trades off waiting (gathering more data) and improved detection performance. However, in case of rare *transient* events (such as earthquakes) that may occur repeatedly, the distributions $\mathbb{P}[X_{s,t} \mid E_t = 1]$ are expected to change with t for $t \geq t_0$.

Challenges for the Classical Approach. Detecting rare events from community-based sensors has three main challenges:

(i) Sensors are highly heterogeneous (i.e., the distributions $\mathbb{P}[X_{s,t} \mid E_t]$ are different for each sensor s)

(ii) Since events are rare, we do not have sufficient data to obtain good models for $\mathbb{P}[X_{s,t} \mid E_t = 1]$

(iii) Bandwidth limitations may limit the amount of communication (e.g., number of picks sent).

Challenge (i) alone would not be problematic – classical decentralized detection can be extended to heterogeneous sensors, as long as we know $\mathbb{P}[X_{s,t} \mid E_t]$. For the case where we do not know $\mathbb{P}[X_{s,t} \mid E_t]$, but we have training examples (i.e., large collections of sensor data, annotated by whether an event is present or not), we can use techniques from nonparametric decentralized detection [49]. In the case of rare events, however, we may be able to collect large amounts of data for $\mathbb{P}[X_{s,t} \mid E_t = 0]$ (i.e., characterizing the sensors in the no-event case), while still collecting extremely little (if any) data for estimating $\mathbb{P}[X_{s,t} \mid E_t = 1]$. In our case, we would need to collect data from cell phones experiencing seismic motion of earthquakes ranging in magnitude from three to ten on the Gutenberg-Richter scale, while resting on a table, being carried in a pocket, backpack, etc. Furthermore, even though we can collect much data for $\mathbb{P}[X_{s,t} \mid E_t = 0]$, due to challenge (iii) we may not be able to transmit all the data to the fusion center, but have to estimate this distribution locally, possibly with limited memory. We also want to choose decision rules that minimize the number of messages sent.

3.2 Online Decentralized Anomaly Detection

We now describe our approach to online, decentralized detection of anomalous events.

Assumptions. In the following, we adopt the assumption of classical decentralized detection that sensor observations are conditionally independent given E_t, and independent across time (i.e., the distributions $\mathbb{P}[X_{s,t} \mid E_t = 0]$ do not depend on t). For earthquake detection this assumption is reasonable (since most of the noise is explained through independent measurement noise and user activity). While spatial correlation may be present, e.g., due to mass events such as rock concerts, it is expected to be relatively rare. Furthermore, if context about such events is available in advance, it can be taken into account. We defer treatment of spatial correlation to future work. We do *not* assume that the sensors are homogeneous (i.e., $\mathbb{P}[X_{s,t} \mid E_t = 0]$ may depend on s). Our approach can be extended in a straightforward manner if the dependence on t is periodic (e.g., the background noise changes based on the time of day, or day within week).

Overview. The key idea behind our approach is that since sensors obtain a massive amount of normal data, they can accurately estimate $\mathbb{P}[X_{s,t} \mid E_t = 0]$ purely based on their local observations. In our earthquake monitoring example, the cell phones can collect data of acceleration experienced under normal operation (lying on a table, being carried in a backpack, etc.). Further, if we have hope of detecting earthquakes, the signal $X_{s,t}$ must be sufficiently different from normal data (thus $\mathbb{P}[X_{s,t} \mid E_t = 0]$ must be low when $E_t = 1$). This suggests that each sensor should decide whether to pick or not based on the likelihood $L_0(x) = \mathbb{P}[x \mid E_t = 0]$

only; sensor s will pick ($M_{s,t} = 1$) iff, for its current readings $X_{s,t} = x$ it holds that

$$L_0(x) < \tau_s \tag{3.3}$$

for some sensor specific threshold τ_s. See Figure 3.6(c) for an illustration. Note that using this decision rule, for a pick it holds that $\mathbb{P}\left[M_{s,t} = 1 \mid E_t = e\right] = \mathbb{P}\left[L_0(X_{s,t}) < \tau_s \mid E_t = e\right] = p_e$. This anomaly detection approach hinges on the following fundamental anti-monotonicity assumption: that

$$L_0(x) < L_0(x') \Leftrightarrow \frac{\mathbb{P}\left[x \,|\, E_t = 1\right]}{\mathbb{P}\left[x \,|\, E_t = 0\right]} > \frac{\mathbb{P}\left[x' \,|\, E_t = 1\right]}{\mathbb{P}\left[x' \,|\, E_t = 0\right]}, \tag{3.4}$$

i.e., the less probable x is under normal data, the larger the likelihood ratio gets in favor of the anomaly. The latter is the assumption on which most anomaly detection approaches are implicitly based. Under this natural anti-monotonicity assumption, the decision rules (3.3) and (3.1) are equivalent, for an appropriate choice of thresholds.

Proposition 7. *Suppose Condition* (3.4) *holds. Then for any threshold* τ *for rule* (3.1)*, there exists a threshold* τ_s *such that rule* (3.3) *makes identical decisions.*

Since the sensors do not know the true distribution $\mathbb{P}\left[X_{s,t} \,|\, E_t = 0\right]$, they use an online density estimate $\widehat{\mathbb{P}}\left[X_{s,t} \mid E_t = 0\right]$ based on collected data. The fusion center will then perform hypothesis testing based on the received picks $M_{s,t}$. In order for this approach to succeed we have to specify:

(i) How can the sensors estimate the distribution $\widehat{\mathbb{P}}\left[X_{s,t} \mid E_t = 0\right]$ in an online manner, while using limited resources (CPU, battery, memory, I/O)?

(ii) How should the sensors choose the thresholds τ_s?

(iii) Which true positive and false positive rates p_1, p_0 and threshold τ', (3.2), should the fusion center use?

We now discuss how our approach addresses these questions.

Online Density Estimation. For each sensor s, we have to, over time, estimate the distribution of *normal* observations $\widehat{L}_0(X_{s,t}) = \widehat{\mathbb{P}}\left[X_{s,t} \mid E_t = 0\right]$, as well as the activation threshold τ_s. There are various techniques for online density estimation. Parametric approaches assume that the density $\mathbb{P}\left[X_{s,t} \mid E_t = 0\right]$ is in some parametric family of distributions:

$$\mathbb{P}\left[X_{s,t} \mid E_t = 0\right] = \phi(X_{s,t}, \theta).$$

The goal then is to update the parameters θ as more data is obtained. In particular, mixture distributions, such as mixtures of Gaussians, are a flexible parametric family for density estimation. If access to a batch of training data is available, algorithms such as Expectation Maximization can be used to obtain parameters

that maximize the likelihood of the data. However, due to memory limitations, it is rarely possible to keep all data in memory; furthermore, model training would grow in complexity as more data is collected. Fortunately, several effective techniques have been proposed for incremental optimization of the parameters, based on Variational Bayesian techniques [55] and particle filtering [23]. Online nonparametric density estimators (whose complexity, such as the number of mixture components, can increase with the amount of observed data) have also been developed [28]. Here, we use Gaussian mixture models for density estimation.

Online Threshold Estimation. Online density estimators as introduced above allow us to estimate $\widehat{\mathbb{P}}[X_{s,t} \mid E_t = 0]$. The remaining question is how the sensor-specific thresholds τ_s should be chosen. The key idea is the following. Suppose we would like to control the per-sensor false positive rate p_0 (as needed to perform hypothesis testing in the fusion center). Since the event is assumed to be extremely rare, with very high probability (close to 1) *every* pick $M_{s,t} = 1$ will be a false alarm. Thus, we would like to choose our threshold τ_s such that, if we obtain a measurement $X_{s,t} = x$ at random, with probability $1 - p_0$, it holds that $\widehat{L}_0(x) \geq \tau_s$.

This insight suggests a natural approach to choosing τ_s: For each training example $x_{s,t}$, we calculate its likelihood $\widehat{L}_0(x_{s,t}) = \widehat{\mathbb{P}}[x_{s,t} \mid E_t = 0]$. We then choose τ_s to be the p_0-th percentile of the data set $\mathfrak{L} = \{\widehat{L}_0(x_{s,1}), \ldots, \widehat{L}_0(x_{s,t})\}$. As we gather an increasing amount of data, as long as we use a consistent density estimator, this procedure will converge to the correct decision rule.

In practice, due to memory and computation constraints, we cannot keep the entire data set $\mathfrak{L}$ of likelihoods and reestimate τ_s at every time step. Unfortunately, percentiles do not have sufficient statistics as the mean and other moments do. Moreover, Munro and Paterson [46] show that computing rank queries exactly requires $\Omega(n)$ space. Fortunately, several space-efficient online ε-approximation algorithms for rank queries have been developed. An algorithm that selects an element of rank r' from N elements for a query rank r is said to be *uniform ε-approximate* if

$$\frac{|r' - r|}{N} \leq \varepsilon$$

One such algorithm which requires logarithmic space is given by [30]. We do not present details here due to space limitations. We summarize our analysis in the following proposition:

Proposition 8. *Suppose that we use a uniformly consistent density estimator (i.e.,* $\limsup_x \{\widehat{\mathbb{P}}[x \mid E_t = 0] - \mathbb{P}[x \mid E_t = 0]\} \to 0$ *a.s.). Further suppose that $\tau_{s,t}$ is an ε-accurate threshold obtained through percentile estimation for p_0. Then for any $\varepsilon > 0$, there exists a time t_0 such that for all $t \geq t_0$, it holds that the false positive probability $\widehat{p}_0 = \mathbb{P}\left[\widehat{L}_0(x_{s,t}) < \tau_s\right]$ is $|\widehat{p}_0 - p_0| \leq 2\varepsilon$.*

The proof of Proposition 8, which we omit for space limitations, hinges on the fact that if the estimate $\widehat{L}_0(x)$ converges uniformly to $L_0(x)$, the p_0-th percentiles (for $0 < p_0 < 1$) converge as well. Furthermore, the use of an ε-approximate percentile can change the false positive rate by at most ε.

Algorithm 4: Threshold Optimization procedure

Data: Estimated sensor ROC curve, N sensors, communication constraints $\bar{p}$, bound on fusion-level false positives $\bar{P}$

Result: sensor operating point (p_0, p_1)

for i^{th} *operating point* (p_0^i, p_1^i) *s.t.* $p_0^i \leq \bar{p}$ **do**

//Do Neyman-Pearson hypothesis testing to evaluate p_0^i;

Compute $N(p_0^i) = \min\{N' : \sum_{S>N'} \text{Bin}(S; p_0^i; N) \leq \bar{P}\}$;

Compute $P_D^i = \sum_{S>N(p_0^i)} \text{Bin}(S; p_1^i; N)$;

Compute $P_F^i = \sum_{S>N(p_0^i)} \text{Bin}(S; p_0^i; N)$;

Choose $\ell = \arg\max_i P_D^i$ and set $(p_0, p_1) = (p_0^\ell, p_1^\ell)$;

Uniform convergence rates for density estimation have been established as well [26], allowing us to quantify the time required until the system operates at ε-accurate false positive rates. Since we assume that communication is expensive, we may impose an upper bound on the expected number of messages sent by each sensor. This can be achieved by imposing an upper bound $\bar{p}$ on p_0, again relying on the fact that events are extremely rare. We present more details in the next section.

Hypothesis Testing for Sensor Fusion. Above, we discussed how we can obtain local decision rules that allow us to control the sensor-level false positive rate p_0 in a principled manner, and in the following we assume that the sensors operate at this false positive rate. However, in order to perform hypothesis testing as in (3.2), it appears that we also need an estimate of the sensor-level true-positive rate p_1.

Suppose that we would like to maximize the detection rate P_D at the fusion center while guaranteeing a false positive rate P_F that is bounded by $\bar{P}$. It can be shown that the optimal decision rule (3.2) is equivalent to setting $\widehat{E}_t = 1$ iff $S_t \geq N(p_0)$, for some number $N(p_0)$ that *only* depends on the total number N of sensors, and the sensor false-positive rate p_0. Thus, to control the fusion-level false positive rate P_F we, perhaps surprisingly, *do not need to know* the value for p_1, since P_F does not depend on p_1:

$$P_F = \sum_{S>N(p_0)} \text{Bin}(S; p_0; N) \text{ and } P_D = \sum_{S>N(p_0)} \text{Bin}(S; p_1; N).$$

Thus, our online anomaly detection approach leads to decision rules that provide guarantees about the fusion-level false positive rate.

Our goal is not just to bound the false positive rate, but also to maximize detection performance. The detection performance P_D above depends on the sensor-level true positive rate p_1. If we have an accurate estimate of p_1, all sensors are homogeneous and the anti-monotonicity condition (3.4) holds, the following result, which is a consequence of [62], holds:

Theorem 9. *Suppose condition* (3.4) *holds and the sensors are all homogeneous (i.e.,* $\mathbb{P}[X_{s,t} \mid E_t]$ *is independent of* s*). Further suppose that for each sensor-level false-positive rate* p_0 *we know its true-positive rate* p_1*. Then one can choose an operating point* (p_0^*, p_1^*) *that is asymptotically optimal (as* $N \to \infty$*).*

Unfortunately, without access to training data for actual events (e.g., sensor recordings during many large earthquakes), we cannot obtain an accurate estimate for p_1. However, in Section 3.5, we show how we can obtain an empirical estimate $\widehat{p}_1$ of p_1 by performing shaketable experiments. Suppose now that we have an estimate $\widehat{p}_1$ of p_1. How does the detection performance degrade with the accuracy of $\widehat{p}_1$? Suppose we have access to an estimate of the sensors' Receiver Operator Characteristic (ROC) curve, i.e., the dependency of the achievable true positive rates $\widehat{p}_1(p_0)$ as a function of the false positive rate (see Figure 3.8(a) for an example). Now we can view both the estimated rates $\widehat{P}_D \equiv \widehat{P}_D(\widehat{p}_1(p_0)) \equiv \widehat{P}_D(p_0)$ and $\widehat{P}_F = \widehat{P}_F(p_0)$ as functions of the sensor-level false positive rate p_0. Based on the argument above, we have that $\widehat{P}_F(p_0) = P_F(p_0)$, i.e., the estimated false positive rate is exact, but in general $\widehat{P}_D(p_0) \neq P_D(p_0)$. Fortunately, it can be shown that if the estimated ROC curve is *conservative* (i.e., $\widehat{p}_1(p_0) \leq p_1(p_0)$ for all rates p_0), then it holds that $\widehat{P}_D(p_0) \leq P_D(p_0)$ is an *underestimate* of the true detection probability. Thus, if we are able to obtain a pessimistic estimate of the sensors' ROC curves, we can make guarantees about the performance of the decentralized anomaly detection system. We can now choose the optimal operating point by

$$\max_{p_0 \leq \bar{p}} \widehat{P}_D(p_0) \text{ s.t. } \widehat{P}_F(p_0) \leq \bar{P},$$

and are guaranteed that the optimal value of this program is a pessimistic estimate of the true detection performance, while $\widehat{P}_F$ is in fact the exact false alarm rate. Algorithm 4 formalizes this procedure. We summarize our analysis in the following theorem:

Theorem 10. *If we use decentralized anomaly detection to control the sensor false positive rate p_0, and if we use a conservative estimated ROC curve $(p_0, \widehat{p}_1)$, then Algorithm 4 chooses an operating point p_0 to maximize a lower bound on the true detection performance, i.e., $\widehat{P}_D(p_0) \leq P_D(p_0)$.*

3.3 The Community Seismic Network

To better understand the possibilities and challenges of community sensing with commercial hardware, the Caltech Community Seismic Network project was created to detect and measure earthquakes using the accelerometers in smartphones and other low-cost commercial sensors.

Several factors make earthquake monitoring a suitable application for community sensing. Current smartphones have the necessary sensors (such as an accelerometer, GPS, gyroscope, and compass) to measure strong shaking. Historically, large earthquakes have had catastrophic consequences for a large number of people, and so many people in active seismic areas may be willing to participate in an effort to better understand earthquakes. Further, earthquakes are spatial events that benefit from the fine spatial resolution of massive sensor networks.

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

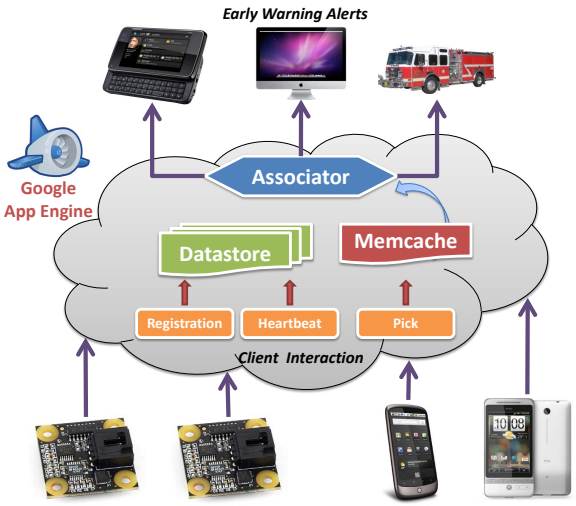


Figure 3.1: The CSN system collects performs decentralized anomaly detection by processing pick messages from low-cost sensors and smartphone accelerometers. Data is processed in a web application built on Google's App Engine. Data products, such as alerts and shake maps, may be issued to the community or emergency responders.

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.

3.4 Applications

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. 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*. Similarly, accelerometers in buildings could inform decisions about which buildings are safe to re-enter after a quake.

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

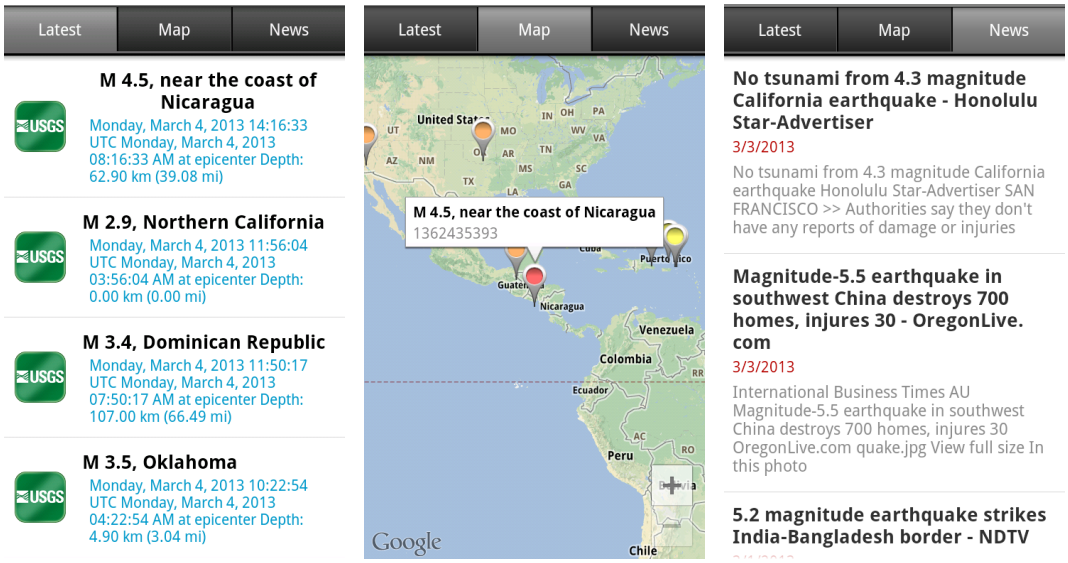


Figure 3.2: CSN volunteers contribute data from low-cost accelerometers (above) and from Android smartphones via a CSN app.

involved in debris clearing during the 1989 Loma Prieta earthquake.

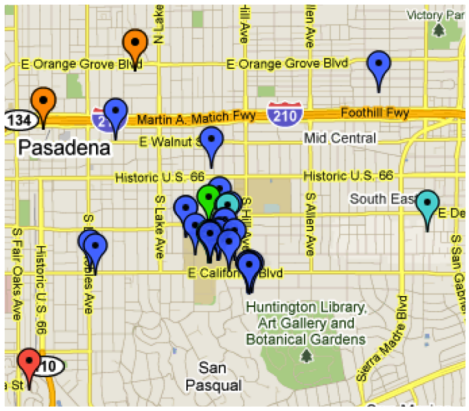


Figure 3.3: Map of locations where measurements have been reported from during our pilot deployment of CSN.

3.4.1 Community Sensors: Android and USB Accelerometers

The Community Seismic Network currently uses two types of sensors: accelerometers in Google Android smartphones (see Figure 3.2), and 16-bit MEMS accelerometers manufactured by Phidgets, Inc., used as USB-accessories to laptops and desktop computers (see Figure 3.4). Each of the sensors has unique advantages: The USB sensors provide higher fidelity measurements. By firmly affixing them to a non-carpeted floor (preferably) or a wall background noise can be drastically removed. However, their deployment relies on the community purchasing a separate piece of hardware (currently costing roughly USD 150 including custom housing). In

Figure 3.4: A low-cost USB accelerometer manufactured by Phidget, Inc.

contrast, the Google Android platform has a large market share, currently approximately 16.3% (more than the Apple iOS platform, and slightly less than Research in Motion), and is projected to grow further [13]. Android based smartphones typically contain 3-axes accelerometers, and integration of an Android phone into the CSN only requires download of a free application. On the other hand, the built-in accelerometers are of lower quality (our experiments showed a typical resolution of approximately 13 bits), and phones are naturally exposed to frequent acceleration during normal operation. We have also built early prototypes of stand-alone devices on top of Arduino boards that connect through USB or WiFi to computing systems with access to the cloud.

Are inexpensive accelerometers sensitive enough to detect seismic motion? We performed experiments to assess the fidelity of the Phidgets and a variety of Android phones (the HTC Dream, HTC Hero and Motorola Droid). We placed the sensors on a flat surface and recorded for an hour. We found that when resting, the phones experienced noise with standard deviation $\approx 0.08 m/s^2$, while the Phidgets experienced noise with standard deviation of $\approx 0.003 m/s^2$. Earthquakes with magnitude 4 on the Gutenberg-Richter scale achieve an acceleration of approximately $.12 m/s^2$ close to the epicenter, which can be detected with the Phidgets, but barely exceeds the background noise level of the phones. However, earthquakes of magnitude 5 already achieve acceleration of .5 m/s^2, increasing to roughly 1.5 m/s^2 for magnitude 6 events. These phones sample their accelerometers at between 50Hz and 100Hz, which is comparable to many high fidelity seismic senors. These numbers suggest that cell phone accelerometers should be sensitive enough to be able to detect large earthquakes.

Figure 3.4 presents the locations where messages have been reported from in our network.

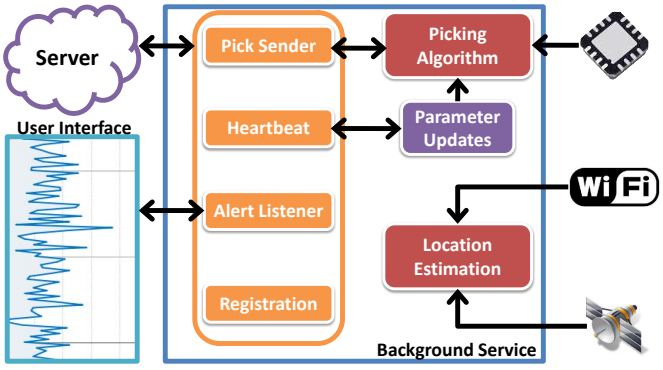


Figure 3.5: CSN-Droid app architecture

3.4.2 Android Client

Figure 3.4.2 presents an overview of our Android client application. It consists of several components, which we explain in the following. The client for the Phidget sensors follows a similar implementation, and a detailed discussion is omitted due to space limitations.

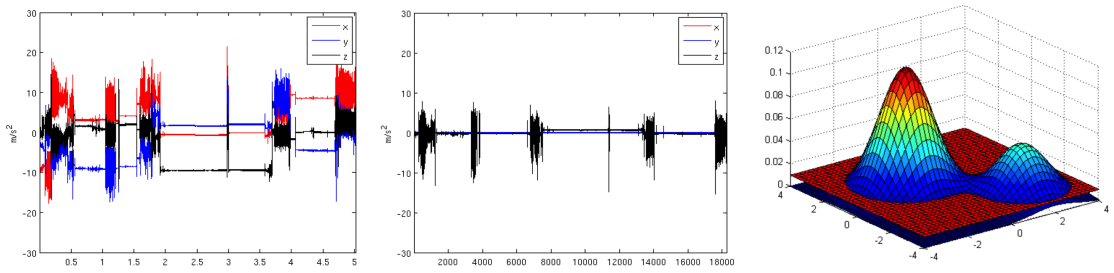
A central policy decision of the system was that the only manner in which information is exchanged between a client computer and the cloud computing system is for the client to send a message to the cloud and await a reply: in effect to execute a remote procedure call. All information exchanges are initiated by a client, never by the cloud. This helps ensure that participants in the CSN are only sent information at points of their choosing.

Registration. Upon the first startup, the application registers with the Cloud Fusion Center (CFC). The CFC responds with a unique identifier for the client, which will be used in all subsequent communications with the CFC.

Picking Algorithm. A background process runs continuously, collecting data from the sensor. The picking algorithm generates ”pick” messages by analyzing raw accelerometer data to determine if the data in the recent past is anomalous. The algorithm executes in the background without a user being aware of its existence. It implements the approach discussed in Section 3.2.

For density estimation, we use a Gaussian mixture model for $\mathbb{P}[X_{s,t} \mid E]$. The most important design choice is the representation $X_{s,t}$ of the sensor data. Our approach is to compute various features from short time windows (similar to phonemes in speak recognition). The idea is that normal acceleration, e.g., due to walking, or manual phone operation, lead to similar signatures of features.

A first challenge is that phones frequently change their orientation. Since accelerometers measure gravity, we first determine (using a decaying average) and subtract out the gravity component from the $[X, Y, Z]$-components of the signal. We then rotate the centered signal so that the estimated gravity component points in the negative Z direction $[0, 0, -1]$. Figures 3.6(a) and 3.6(b) illustrate this process. Since we cannot



(a) *Before gravity correction* (b) *After gravity correction* (c) *GMM based picking*

Figure 3.6: (a) 5 hours of recording three-axis accelerometer data during normal cell phone use. (b) Data from (a) after removing gravity and appropriate signal rotation. (c) Illustration of the density estimation based picking algorithm. The red plane shows an operating threshold. Acceleration patterns for which the density does not exceed the threshold result in a pick.

consistently orient the other axes, we use features that are invariant under rotation around the vertical (Z) axis, by replacing the $[X, Y]$ component by its Euclidean norm $||[X, Y]||_2$.

We consider time windows of 2.5 seconds length and, for both the Z and $||[X, Y]||_2$ components calculate 16 Fourier coefficients, the second moment, and the maximum absolute acceleration. This procedure results in a 36-dimensional feature vector. To avoid the curse of dimensionality we perform linear dimensionality reduction by projection on the top 16 components. These principal components can be computed using online algorithms [66]. While PCA captures most variance in the training data (normal acceleration patterns), it is expected that unusual events may carry energy in directions not spanned by the principal components. We therefore add the projection error (amount of variance not captured by the projection) as an additional feature. We arrived at this choice of features, as well as the number $k = 6$ of mixture components through careful cross-validation experiments, using our experimental setup discussed in Section 3.5.

Our threshold for picking is obtained using online percentile estimation, as detailed in Section 3.2. In order to bootstrap the deployment of Gaussian mixture models to new phones, our phone client has the capability of updating its statistical model via messages from the CFC. The threshold by which the algorithm on a client computer determines whether an anomaly is present can also be changed by a message from the cloud computer. This allows the CFC to throttle the rate at which a given sensor generates pick messages.

Pick Reporting. Whenever the picking algorithm declares a pick, a message is sent to the CFC, which includes the time, location, and estimated amplitude of the data which triggered the pick. Including the location is important for two reasons. First, for mobile clients, it is more efficient than receiving regular location updates. Second, sending the location is helpful in order to facilitate faster association by avoiding database lookups for every stationary sensor pick. While it should be possible to improve detection performance at the CFC by sending more information or additional rounds of messages, it is unclear if the cost of this communication is acceptable. Electricity and the Internet may be lost shortly after a large quake, and so our system is designed to use minimal messages to report crucial information as quickly as possible.

Heartbeats. At some prespecified interval, "heartbeat" messages are sent to the CFC, allowing the CFC to keep track of which phones are currently running the program. The heartbeats contain the time, location, waveform logs, and a parameter version number. Using the parameter version number, the CFC can determine whether to send updated parameters to each phone or not. This mechanism allows modifications to the picking algorithm without requiring changes to the underlying client software.

User interface. While the main application runs in the background using Android's multitasking capability, the application provides a user interface to display the recorded waveforms. We are currently collaborating with a USGS led effort in earthquake early warning. Our application will connect to the early warning system and be able to issue warnings about when shaking will occur, as well as the estimated epicenter of the event. While the application is currently a research prototype and not yet deployed in public use, we anticipate that the capability of real-time early warning may encourage users to download and install the application.

Power Usage. Battery drain is an important factor in users' decisions to install and run our application. In our experiments on the Motorola Droid, the battery life exceeded 25 hours while continuously running the client (but without any further operation of the phone). This runtime would not inconvenience a user who charges their phone each night. However, prior to public release of the client, power optimizations or duty cycling will need to be performed.

3.4.3 Cloud Fusion Center

The Cloud Fusion Center (CFC) performs the fusion-level hypothesis test defined in (3.2) and administers the network. In devising a system to serve as a logically central location, we evaluated building our own server network, using virtualized remote servers, having collocated servers, and building our application to work on Google App Engine. App Engine was chosen as the platform for the CFC for several reasons: easy scalability, built in data security, and ease of maintenance.

The App Engine platform is designed from the ground up to be scalable to an arbitrary number of clients. As we expect to grow our sensor network to a very high sensor density, this element of the platform's design is very important. What makes the scalability of the platform easily accessible is the fact that incoming requests are automatically load-balanced between instances that are created and destroyed based on current demand levels. This reduces algorithmic complexity, as the load balancing of the network is handled by the platform rather than by code written for our CSN system.

A second consideration in our selection was data security. With the other solutions we had available to us, if the data we collected was stored on the server network we were using, then, without redundant servers in separate geographies, we risked losing all of our data to the very earthquakes we hoped to record. App Engine solves this problem for us by automatically replicating datastore writes to geographically separate data centers as the writes are processed. This achieves the level of data redundancy and geographical separation we require, without forcing us to update our algorithms. Other network storage solutions would have been possible as

well, but having it built into the platform meant that latency for code accessing the storage would be lower.

A final compelling reason to select the App Engine was its ease of maintenance. Rather than spending time building server images and designing them to coordinate with each other, we were able to immediately begin working on the algorithms that were most important to us. Server maintenance, security, and monitoring are all handled by the App Engine team and do not take away from the time of the research team members.

App Engine also includes a number of other benefits we considered. First, it utilizes the same front ends that drive Google's search platform, and, consequently, greatly reduces latency to the platform from any point in the world. Since we plan to expand this project beyond Southern California, this is very useful. Second, the platform supports live updates to running applications. Rather than devising server shutdown, update, and restart mechanisms as is commonly required, we can simply redeploy the application that serves our sensors and all new requests to the CFC will see the new code instead of the old code with no loss of availability.

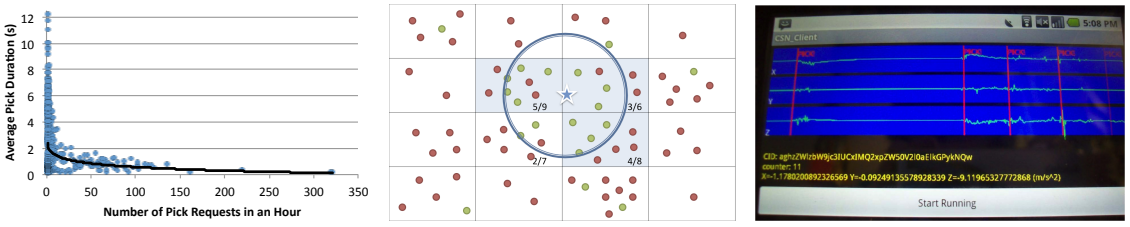
All of these features do not come without a price, however. We will discuss what we perceive as the two largest drawbacks of the platform: loading requests and design implications.

Loading Requests. Because App Engine dynamically scales the number of available instances available to serve a given application as the volume of requests per unit time changes, it creates a phenomenon known as a loading request. This request is named in this manner because it is the first request to a new instance of the application. That is, when App Engine allocates a new instance to serve increasing traffic demands, it sends an initial live request to that instance. In Java, this results in the initialization of the Java Virtual Machine, including loading all of the appropriate libraries.

Over the last three months, we experienced loading requests with a median frequency of 9.52% of all requests. While this means that 90.48% of requests did not experience increased latency as a result of the platform, the remaining requests experienced a median increased processing duration of 5,400 ms. Because of the extreme penalty paid by loading requests, when examining average request duration, their presence dominates the figures. This results in a unique property of App Engine, which is that the system performs much better at higher request loads.

Fig. 3.7(a), shows that, as the request volume increases, the average duration of each request decreases. This is a result of a reduced impact of loading requests. This data leads to the conclusion that if we avoid potential bottleneck points such as datastore writes, we can expect our performance to stay the same or get better for any increased future load imposed on the system (e.g., as the number of sensors scales up).

Design Implications. When designing an algorithm to run on App Engine, the algorithm has to fit inside of the constraints imposed by the architecture. There are a few factors to consider. First, as a result of the automatic scaling done by App Engine, every request to the system exists in isolation. That is, the running requests maintain no shared state, nor do they have any inter-process communication channels. Additionally, since there are no long running background processes, maintaining any form of state generated as a result of successive calls is more difficult. In order to accurately ascertain the number of incoming picks in a unit time



(a) *Pick duration* (b) *Associator* (c) *Android Client*

Figure 3.7: (a) Average duration of a pick request as a function of system load. (b) Model of dispersed sensors using a hash to a uniform grid to establish proximity. (c) A picture of the CSN android client in debug mode, capturing picks.

over a specified geography, we had to surmount these hurdles.

The only common read/write data sources provided are memcache (a fast key value store) and datastore. The datastore is a persistent object store used for permanent data archiving for future analysis or retrieval. Long term state which changes infrequently, such as the number of active sensors in a given region, is stored and updated in the datastore, but cached in the memcache for quick access. Due to its slower performance, particularly in aggregating writes for immediate retrieval by other processes, it is unsuitable for short term state aggregation.

Short term state, such as the number of picks arriving in an interval of time in a particular region, is stored in memcache. While memcache is not persistent, as objects can be ejected from the cache due to memory constraints, operations that utilize the memcache are much faster. Memcache is 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. That is, after a long enough period of time has passed since a pick arrived, it can no longer be used in detecting an event; therefore, its contributed value to the memcache can be safely expired.

Memcache operates as a key value store, effectively a distributed hash table. In order to determine how many sensors sent picks in a given period of time, we devised a system of keys which could be predictably queried to ascertain the number of reporting sensors. We used a geography hashing scheme to ascribe an integer value to every latitude/longitude pair, which generates a uniform grid of cells whose size we can control, with each sensor fitting into one cell in the grid (see Fig. 3.7(b)). Incoming picks then update the key corresponding to a string representation of the geographical hash and a time bucket derived by rounding the arrival time of the pick to the nearest second.

In this manner, independent processes aggregate their state, and each process runs the hypothesis testing algorithm of Section 3.2 in the cell whose state it updated to determine the value of $\widehat{E}_t$. If $\widehat{E}_t = 0$, then no action needs to be taken. If $\widehat{E}_t = 1$ a task queue task is launched to initiate the alert process; the task is named using the hash values that generated the alert. Each named task creates a 'tombstone' (a marker in the system) on execution which prevents additional tasks with the same name from being created, so even if successive picks also arrive at the $\widehat{E}_t = 1$ conclusion, we know that only one alert will be sent out for a given set of

inputs.

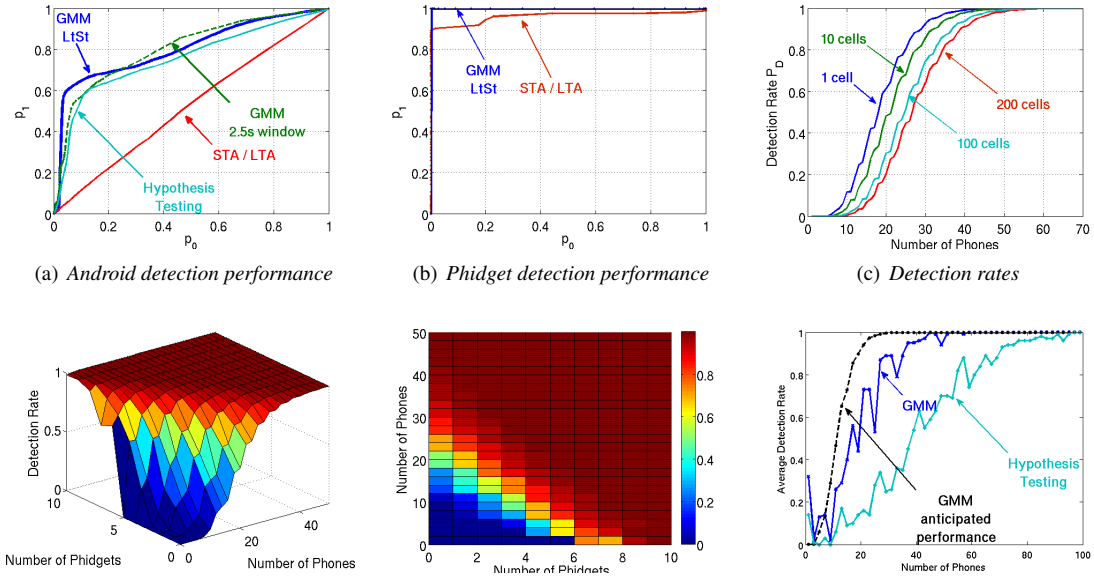
3.5 Experiments

Could a network of cheap community sensors detect the next large earthquake? We obtain accurate estimates of the distribution of normal sensor data by collecting records from volunteers' phones and and USB accelerometers. Using an earthquake shaketable and records of ground acceleration gathered by the Southern California Seismic Network (SCSN) during moderately large earthquakes, we obtain estimates of each sensor's ROC curves. These estimates of sensor performance allow us to evaluate the effect of network density and composition on the detection rate. Finally, we apply the learned detection models to data from the 2010 Baja California M7.2 quake.

Data Sets. While earthquakes are rare, data gathered from community sensors can be plentiful. To characterize "normal" (background) data, seven volunteers from our research group carried Android phones throughout their daily routines to gather over 7GB of phone accelerometer data. Similarly, an initial deployment of 20 USB accelerometers recorded 55GB of acceleration over a period of 4 months. However, due to the infrequent occurrence of large earthquakes, it could require many years of observation to obtain records from several dangerously large events. One approach to overcome this limitation is to simulate sensor observations from existing seismic records, and use these simulated observations for testing. The Southern California Seismic Network, a network of several hundred high-fidelity seismometers, operating since the 1920s, provides a database of such records. We extract a set of 32 records of moderately large (M5-5.5) events from stations at distances of under 40km from the event epicenter. Simulated sensor observations are produced by subsampling these records to 50 samples per second and superimposing them onto segments of Android or Phidget data. As we will see in our shaketable experiments, this method of obtaining simulated sensor data yields a reasonable estimate of detection performance when we reproduce quake records using a shaketable and directly sense the acceleration with both Androids and Phidgets.

Picking Algorithm Evaluation. In our first experiment, we evaluate the sensor-level effectiveness of our density-based anomaly detector. We compare four approaches: two baselines and two versions of our algorithm.

1. A hypothesis-testing based approach (as used by classical decentralized detection), which uses a GMM-based density estimate not just for $\mathbb{P}\left[X_{s,t} \mid E_t = 0\right]$, but also for $\mathbb{P}\left[X_{s,t} \mid E_t = 1\right]$. For training data, we use 80 historic earthquake examples of magnitude 4.5-5, superimposed on the sensor data.
2. A domain specific baseline algorithm, *STA/LTA*, which exploits the fact that the energy in earthquakes is broadband in 0-10Hz. It compares the energy in those frequencies in the last 2.5s to the energy at those frequencies in the previous 5s; a sharp rise in this ratio is interpreted as a quake.
3. A simplified GMM based approach, which uses features from a sliding window of 2.5 s length



(a) *Android detection performance*

(b) *Phidget detection performance*

(c) *Detection rates*

(d) *1 pick per minute, LtSt features*

(e) *1 pick per minute, alternate view*

(f) *Detection rate on Baja M7.2 (100 iterations averaged)*

Figure 3.8: (a,b) Sensor level ROC curves for magnitude 5-5.5 events, for Android (a) and Phidget (b) sensors. In (c-d), the system-level false positive rate is constrained to 1 per year, and the achievable detection performance is shown: (c) Detection rate as a function of the number of sensors in a 20 km $\times$ 20 km cell. We show the achievable performance guaranteeing one false positive per year, while varying the number of cells covered. (d,e) Detection performance for one cell, depending on the number of phones and Phidgets. (f) Actual detection performance for the Baja event. Note that our approach outperforms classical hypothesis testing, and closely matches the predicted performance.

4. Our full GMM approach, which combines combines features of the last 2.5s with features from the previous 5s (to better detect the onset of transient events).

Notice that implementing the hypothesis testing baseline in an actual system would require waiting until the sensors experienced such a number of earthquakes, carefully annotating the data, and then training a density estimator. On the other hand, our anomaly detection approach can be used as soon as the sensors have gathered enough data for an estimate of $\mathbb{P}\left[X_{s,t} \mid E_t = 0\right]$. We applied each of these four algorithms to test data based on historic earthquake recordings of magnitude 5-5.5 superimposed on held-out phone / Phidget data (i.e., data that was not used for training). The resulting estimated sensor ROC curves are shown in Fig. 3.8(a) and Fig. 3.8(b), respectively.

First note that in general the performance for the Phidgets is much better than for the phones. This is expected, as phones are subject to much more background noise, and the quality of the accelerometers in the Phidgets is better than those in the phones. For example, while the STA/LTA baseline provides good performance for the Phidgets (achieving up to 90% detection performance with minimal false positives), it performs extremely poorly for the phone client (where it barely outperforms random guessing). The other techniques achieve close to 100% true positive rate even for very small false positive rates. For the phone data,

both our anomaly detection approaches outperform the hypothesis testing baseline, even though they use less training data (no data about historic earthquakes). In particular for low false positive rates (less than 5%), the full GMM LtSt model outperforms the simpler model (that only considers 2.5s sliding windows). Overall, we find that both for the phones and the Phidgets, we can achieve detection performance far better than random guessing, even for very small false positive rates, and even for lower magnitude (M5-5.5) events. We expect even better detection performance for stronger events.

Sensor Fusion. Based on the estimated sensor-level ROC curves, we can now estimate the fusion-level detection performance. To avoid overestimating the detection performance, we reduce the estimated true positive rates, assuming that a certain fraction of the time (10% in our case) the sensors produce pure random noise. We now need to specify communication constraints $\bar{p}$ on how frequently messages can be sent from the sensors, as well as a bound $\bar{P}$ on the fusion-level false positive rate. We choose $\bar{p}$ to be at most one message per minute, and $\bar{P}$ to be at most one fusion-level false positive per year. This fusion-level false positive rate was chosen as representative of the time scale the CSN must operate on; in practice this would depend on the cost of deploying unnecessary response measures.

We consider sensors located in a geospatial areas of size 20 km $\times$ 20 km, called *cells*. The choice of this area is such that, due to the speed of seismic waves ($\approx 5 - 10km/s$), most sensors within one cell would likely detect the earthquake when computing features based on a sliding window of length 2.5s. However, in order to achieve larger spatial coverage we will need many spatial cells of 20 km $\times$ 20 km. For example, roughly 200 such cells would be needed to cover the Greater Los Angeles area. Increasing the number of cells additively increases the number of false positives due to the fact that multiple hypotheses (one per cell) are tested simultaneously. Consequently, to maintain our objective of one system-wide false positive per year, we must decrease the rate of annual false positives per cell. The effect on detection rates from this compensation as a function of the total number of cells is shown in Figure 3.8(c). Notice that even for 200 cells, approximately 60 phones per cell suffice to achieve close to 100% detection performance, as long as they are located close to the epicenter.

Sensor Type Tradeoffs. A natural question is what is the tradeoff between the different sensor types? Figures 3.8(d) and 3.8(e) shows the estimated detection performance as a function of the number of Phidgets and number of phones in the area, when constrained to one false alarm per year. Our results indicate that approximately 50 phones or 10 Phidgets should be enough to detect a magnitude 5 and above event with close to 100% success.

The results in Figures 3.8(d) and 3.8(e) also allow us to estimate how we could ensure sufficient detection performance if a given area contains only a limited number of active phone clients. For example, if only 25 phones are active in a cell, we could manually deploy 5 additional Phidgets to boost the detection performance from close to 70% to almost 100%.

Notice that all these results assume that the sensors are located close to the epicenter (as they assume the

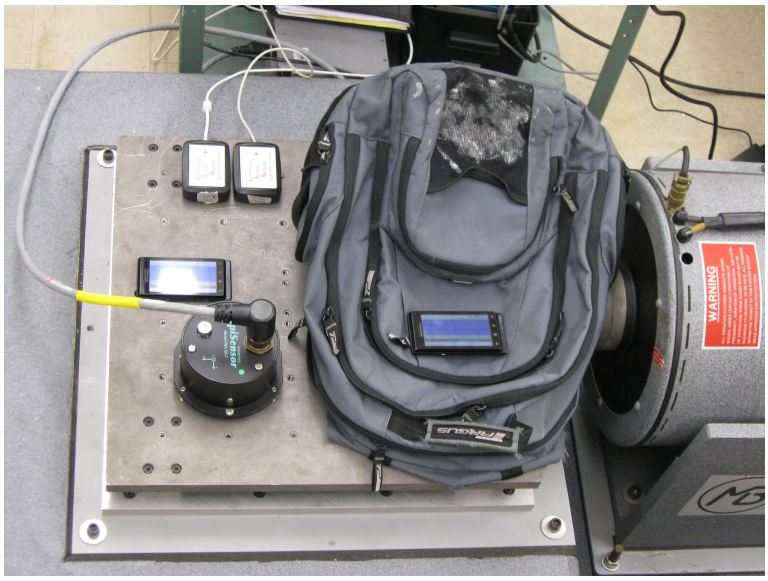
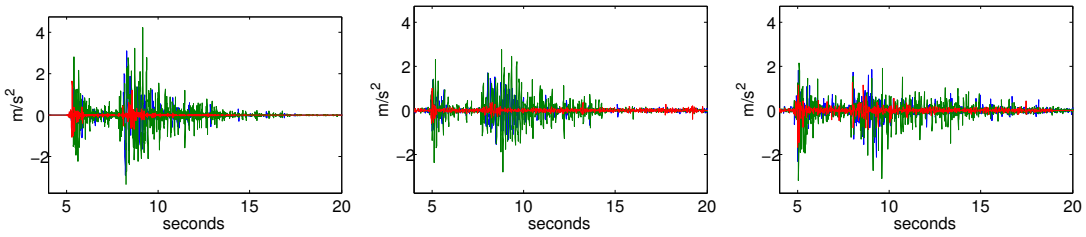


Figure 3.9: Experimental setup for playing back historic earthquakes on a shaketable, and testing their effect on the sensors of the CSN system.



(a) *Episensor Recording* (b) *Android on Table* (c) *Android in Backpack*

Figure 3.10: Shake table comparison of 24-bit EpiSensor, Android, and Android in a backpack. Notice that the phone recordings closely match those of the affixed high-fidelity EpiSensor.

sensors experience maximum acceleration), and are thus to be taken with some care. Covering an area such as Greater Los Angeles likely requires tens of thousands of sensors.

Shaketable Validation.

Our previous experiments have used synthetically produced data (recorded seismic events superimposed on phone recordings) to simulate how different detection algorithms may respond to a moderately large earthquake. Is such a simulation-based approach valid? Would these sensors actually detect an earthquake from their own recordings? To answer these questions, we take recordings of three large historical earthquakes, and play them back on a *shaketable* (see Figure 3.5 for an illustration).

First, we test the ability of Android phones to accurately capture seismic events, relative to one of the sensors used in the SCSN. We reproduce records of three large M6-8 earthquakes on the shaketable, and record the motion using one Android placed on the table, and another in a backpack on the table. Ground truth acceleration is provided by a 24-bit EpiSensor accelerometer mounted to the table. A sample record from each

sensor is shown in Figure 3.10. Unlike the EpiSensor, the phones are not affixed and are free to slide. The backpack also introduces an unpredictable source of error. Despite these significant challenges, after properly resampling both signals and aligning them temporally, we obtain an average correlation coefficient of 0.745, with a standard deviation of 0.0168. This result suggests that the phones reproduce the waveforms rather faithfully.

A more important question than faithful reproduction of waveforms is whether the sensors can detect an earthquake played back on the shaketable. To assess this, we use the model trained on background noise data, as described above. We further use percentile estimation to choose the operating point which we experimentally determined to lead to high system-level detection performance above. All six of the recordings (three from the phone on the table and three from the phone in the backpack) were successfully detected.

The Previous Big One. To perform an end-to-end test of the entire system, we performed an experiment with the goal to find out whether our CSN would have been able to detect the last big event. A recent major earthquake in Southern California occurred on April 4, 2010. This M7.2 quake in Baja, California was recorded by SCSN, although the nearest station was more than 60km from the event epicenter. Using 8 recordings of this event, at distances of 63km to 162km, we produce simulated Android data and evaluate how many phones would have been needed to detect this event. Specifically, we constrain the system as before to one false alarm per year, and one message per minute in order to determine detection thresholds, sensor operating points and sensor thresholds for both the GMM anomaly and hypothesis testing detector, for each deployment size. We then simulate observations for each sensor in a deployment ranging from 1 sensor to 100 sensors. The models and thresholds are then applied to these observations to produce picks; the fusion center hypothesis test is then performed and the decision is made whether an event has occurred or not. The average detection rates for each deployment size (averaged over 100 iterations, using different Android data to simulate each observation) are shown in Figure 3.8(f) along with the estimated detection rates for the GMM-based anomaly detection. The latter estimate is based on the ROC that we estimated using a different collection of seismic events, as explained in our Picking Algorithm Evaluation section. Notice that the actual detection performance matches well the predicted detection performance. As baseline, we compare against the hypothesis testing based baseline (trained on 80 smaller-magnitude earthquakes). Anomaly detection significantly outperforms hypothesis testing, and suggests that a deployment of 60 phones in a cell 60 km of the epicenter would have been quite likely to detect the Baja, California M7.2 event.

Chapter 4

Conclusions

Large-scale community sensing offers the possibility of observing the physical world more rapidly and with finer resolution than has been previously possible. The growing number of sensor-equipped internet devices mean that people will be able to measure and act on more information about their environment, and that collectively they will be able to monitor and act upon large-scale phenomena like traffic, public health, pollution, and natural disasters.

Very large numbers of sensors will be required to compensate for per-device noise, and to attain a high spatial density. Fortunately, the community already possesses the necessary hardware. However, utilizing the data produced by a large (e.g. city-wide or global) network of sensors requires techniques for reducing the amount of data transmitted from each device, and the amount of data processed centrally.

This thesis presents two approaches for selective data gathering in community sensor networks. The first approach considers the case of learning about the state of the environment over time, and leverages powerful no-regret algorithms to learn to select a fixed set of sensors, in order to maximize a utility function. The DOG algorithm was shown to provide strong guarantees for a useful class of sensing objective functions – those which are submodular. The algorithm has extremely low communication requirements, and scales well to large sensor deployments. The DOG algorithm can be extended to address several concerns in wireless sensor networks: the LAZYDOG variant is suited to networks of internet-connected devices which communicate with a cloud server, local rules or heuristics for detecting and transmitting potentially valuable observations. We empirically demonstrate the effectiveness of our algorithm on several real-world sensing tasks.

As the second contribution, we present a principled approach towards detecting rare events based on learning sensor-specific decision thresholds online, in a distributed way. It maximizes anomaly detection performance at a fusion center, under constraints on the false alarm rate and number of messages per sensor. We then present an implementation of our approach in the Community Seismic Network (CSN), a community sensing system with the goal of rapidly detecting earthquakes using cell phone accelerometers, consumer USB devices and cloud-computing based sensor fusion. We experimentally evaluate our approach based on a pilot deployment of the CSN system. Our results, including data from shake table experiments, indicate the effectiveness of our approach in distinguishing seismic motion from accelerations due to normal daily

manipulation. They also provide evidence of the feasibility of earthquake early warning using a dense network of cell phones.

We studied the problem of detecting rare, disruptive events using community-held sensors. Our approach learns local statistical models characterizing normal data (e.g., acceleration due to normal manipulation of a cellphone), in an online manner. Using local online percentile estimation, it can choose operating points that guarantee bounds on the sensor-level false positive frequency, as well as the number of messages sent per sensor. We then showed how a conservative estimate of the sensors' ROC curves can be used to make detection decisions at a fusion center which guarantee bounds on the false positive rates for the entire system, as well as maximize a lower bound on the detection performance. The pessimistic predicted true positive rates allow us to assess whether a given density of sensors is sufficient for the intended detection task. This online decentralized anomaly detection approach allows us to cope with the fundamental challenge that rare events are very difficult or impossible to model and characterize a priori. It also allows the use of heterogeneous, community-operated sensors that may differ widely in quality and communication constraints.

We then presented an implementation of our approach in the Community Seismic Network (CSN), a novel community sensing project with the goal of rapidly detecting earthquakes using cell phone accelerometers and consumer USB devices. We presented empirical evidence suggesting how cloud-computing is an appropriate platform for real-time detection of rare, disruptive events, as it naturally copes with peaked load, and is designed for redundancy and replication. We furthermore experimentally assessed the sensitivity of our sensors, estimating and evaluating ROC curves using experiments involving data obtained through the playback of historical earthquakes on shaketables. These assessments provide evidence of the likely detection performance of the CSN as a function of the sensor density. For example, we found that approximately 100 Android clients, or 20 Phidgets per 20 km $\times$ 20 km area may be sufficient to achieve close to 100% detection probability for events of magnitude 5 and above, while bounding the false positive rate by 1 per year. While these results are very promising, they have to be taken with some care. In particular, the results are based on the assumption that the phones are located very close to the epicenter of the quake (so they experience maximum acceleration). To enable coverage of the Greater Los Angeles area, this would require a uniformly high density of sensors (tens of thousands of sensors) across the entire domain. We will defer the detailed study of spatial effects, and numbers of sensors needed to achieve spatial coverage, to future work.

These two approaches - online distributed sensor selection, and distributed anomaly detection for rare events - address alternate ends of the sensing spectrum. The first approach is best suited to sensing an environment when training data is available, and when the environment is fixed or more slowly changing; the network repeatedly samples a small set of sensors learns over time to maximize the value of its selections. The latter approach is suited to a detecting rare events characterized by a rapid departure from typical network behavior. The observation-depended activation heuristic (OD-DOG) can be viewed as one means to bridge the divide between the two cases.

Bibliography

[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] Z. Abrams, A. Goel, and S. Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In *IPSN*, pages 424–432, 2004.

[3] R.M. Allen, H. Brown, M. Hellweg, O. Khainovski, P. Lombard, and D. Neuhauser. Real-time earthquake detection and hazard assessment by ElarmS across California. *Geophysical Research Letters*, 36(5), 2009.

[4] 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.

[5] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[6] X. Bai, S. Kumar, Z. Yun, D. Xuan, and T. H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *ACM MobiHoc*, 2006.

[7] A. Blum, M. Hajiaghayi, K. Ligett, and A. Roth. Regret minimization and the price of total anarchy. In *STOC*, pages 373–382, 2008.

[8] 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.

[9] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *World Sensor Web Workshop, ACM Sensys*, 2006.

[10] 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.

[11] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Stat. Sci.*, 10(3):273–304, Aug. 1995.

[12] J.F. Chamberland and V.V. Veeravalli. Decentralized detection in sensor networks. *Signal Processing, IEEE Transactions on*, 51(2):407–416, 2003.

[13] cnet news. Android market share to surge over next four years. http://news.cnet.com/8301-1035_3-20015799-94.html, September 2010.

[14] 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.

[15] N. A. C. Cressie. *Statistics for Spatial Data*. Wiley, 1991.

[16] G. Cua, M. Fischer, T. Heaton, and S. Wiemer. Real-time performance of the Virtual Seismologist earthquake early warning algorithm in southern California. *Seismological Research Letters*, 80(5):740, 2009.

[17] A. Das and D. Kempe. Algorithms for subset selection in linear regression. In *STOC*, pages 45–54, 2008.

[18] M. Davy, F. Desobry, A. Gretton, and C. Doncarli. An online support vector machine for abnormal events detection. *Signal processing*, 86(8):2009–2025, 2006.

[19] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, pages 588–599, 2004.

[20] 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.

[21] A. Ostfeld et al. The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, 134(6):556–568, 2008.

[22] 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.

[23] P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14(1):11–21, 2004.

[24] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[25] D. P. Foster and R. Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29(1-2):7–35, October 1999.

[26] Evarist Giné-Masdéu and A. Guillou. Rates of strong uniform consistency for multivariate kernel density estimators. *Ann Inst. H. Poincaré*, 38:907–921, 2002.

[27] D. Golovin, M. Faulkner, and A. Krause. Distributed online sensor selection. *arXiv.org*, arXiv:1002.1782 [cs.LG], Feb. 2010.

[28] R. Gomes, M. Welling, and P. Perona. Incremental learning of nonparametric Bayesian mixture models. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[29] H. H. Gonzalez-Banos and J. Latombe. A randomized art-gallery algorithm for sensor placement. In *Proc. 17th ACM Symp. Comp. Geom.*, pages 232–240, 2001.

[30] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 58–66. ACM, 2001.

[31] R. Herring, A. Hofleitner, S. Amin, T.A. Nasr, A.A. Khalek, P. Abbeel, and A. Bayen. Using mobile phones to forecast arterial traffic through statistical learning. *Submitted to Transportation Research Board*, 2009.

[32] D. S. Hochbaum and W. Maas. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32:130–136, 1985.

[33] 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.

[34] 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.

[35] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proc. of Uncertainty in Artificial Intelligence (UAI)*, 2005.

[36] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *AAAI Nectar track*, pages 1650–1654, 2007.

[37] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *J. Wat. Res. Plan. Mgmt.*, 136(6), 2008.

[38] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. In *JMLR*, volume 9, pages 235–284, 2008.

[39] 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.

[40] F. Kuhn, T. Locher, and R. Wattenhofer. Distributed selection: a missing piece of data aggregation. *Commun. ACM*, 51(9):93–99, 2008.

[41] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[42] Sickweather LLC. Sickweather. http://www.sickweather.com.

[43] F. Martincic and L. Schwiebert. Distributed event detection in sensor networks. In *Systems and Networks Communications, 2006. ICSNC'06. International Conference on*, page 43. IEEE, 2006.

[44] P. Mohan, V.N. Padmanabhan, and R. Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM, 2008.

[45] 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.

[46] J. I. Munro and M. S. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12(3):315–323, 1980.

[47] K. Nakano and S. Olariu. A survey on leader election protocols for radio networks. In *Parallel Architectures, Algorithms and Networks, 2002. I-SPAN '02.*, pages 63–68, 2002.

[48] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14(1):265–294, 1978.

[49] X.L. Nguyen, M.J. Wainwright, and M.I. Jordan. Nonparametric decentralized detection using kernel methods. *Signal Processing, IEEE Transactions on*, 53(11):4053–4066, 2005.

[50] I. Onat and A. Miri. An intrusion detection system for wireless sensor networks. In *Wireless And Mobile Computing, Networking And Communications, 2005.(WiMob'2005), IEEE International Conference on*, volume 3, pages 253–259. IEEE, 2005.

[51] PollMap. Pollmap lunch break edition. http://nutrition.esri.com/lunchbreak/.

[52] H. Vincent Poor and Olympia Hadjiliadis. *Quickest Detection.* Cambridge University Press., 2009.

[53] S. Rajasegarar, C. Leckie, JC Bezdek, and M. Palaniswami. Centered hyperspherical and hyperellipsoidal one-class support vector machines for anomaly detection in sensor networks. *Information Forensics and Security, IEEE Transactions on*, 5(3):518–533, 2010.

[54] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58:527–535, 1952.

[55] M.A. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001.

[56] V. Singhvi, A. Krause, C. Guestrin, J. Garrett, and H.S. Matthews. Intelligent light control using sensor networks. In *SenSys*, pages 218–229, 2005.

[57] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *ICC*, pages 472–476, 2001.

[58] M. Streeter and D. Golovin. An online algorithm for maximizing submodular functions. Technical Report CMU-CS-07-171, Carnegie Mellon University, 2007.

[59] M. Streeter and D. Golovin. An online algorithm for maximizing submodular functions. In *NIPS*, pages 1577–1584, 2008.

[60] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *Proceedings of the 32nd international conference on Very large data bases*, pages 187–198. VLDB Endowment, 2006.

[61] 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.

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

[63] USGS. Netquakes. http://earthquake.usgs.gov/monitoring/netquakes, 2010.

[64] P. Völgyesi, A. Nádas, X. Koutsoukos, and Á. Lédeczi. Air quality monitoring with sensormap. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 529–530. IEEE Computer Society, 2008.

[65] J. Vondrák. *Submodularity in Combinatorial Optimization*. PhD thesis, Charles University, Prague, Czech Republic, 2007.

[66] Manfred K. Warmuth and Dima Kuzmin. Randomized pca algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 9:2287–2320, 2008.

[67] M. Widmann and C. S. Bretherton. 50 km resolution daily precipitation for the pacific northwest. http://www.jisao.washington.edu/data_sets/widmann/, May 1999.

[68] J.L. Williams, J.W. Fisher III, and A.S. Willsky. Performance guarantees for information theoretic active inference. In *AISTATS*, 2007.

[69] G. Wittenburg, N. Dziengel, C. Wartenburger, and J. Schiller. A system for distributed event detection in wireless sensor networks. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 94–104. ACM, 2010.

[70] K. Yamanishi, J.I. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8(3):275–300, 2004.

[71] 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.

[72] Y. Zhang, W. Lee, and Y.A. Huang. Intrusion detection techniques for mobile wireless networks. *Wireless Networks*, 9(5):545–556, 2003.

[73] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing*, 19(2):61–72, 2002.