Community Sense and Response Systems

Thesis by

Matthew N. Faulkner

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2014

(Defended May 1, 2014)

Acknowledgements

I am indebted to a number of people for their guidance and support throughout my time at Caltech. Foremost among these is my advisor, Andreas Krause, who shared his immense enthusiasm for research, taught me that effective communication is an important component of impact, and encouraged me through the twists and turns of my graduate career. I am also particularly grateful to my co-advisors Mani Chandy and Tom Heaton for their thoughtful and friendly advice, and to Joel Tropp and Yaser Abu-Mostafa for their insights towards improving this thesis.

Much of the work in this thesis was made possible by the efforts of the Community Seismic Network team. I would particularly like to thank Mani Chandy, Rob Clayton, Tom Heaton, Michael Aivazis, Julian Bunn, Ming-Hei Cheng, Richard Guy, Monica Kohler, Annie Liu, Michael Olson, and Leif Strand for several years of fruitful collaboration.

Contents

List of Figures

Abstract

The proliferation of smartphones and other internet-enabled, sensor-equipped consumer devices enables us to sense and act upon the physical environment in unprecedented ways. This thesis considers Community Sense-and-Response (CSR) systems, a new class of web application for acting on sensory data gathered from participants' personal smart devices. The thesis describes how rare events can be reliably detected using a decentralized anomaly detection architecture that performs client-side anomaly detection and server-side event detection. After analyzing this decentralized anomaly detection approach, the thesis describes how weak but spatially structured events can be detected, despite significant noise, when the events have a sparse representation in an alternative basis. Finally, the thesis describes how the statistical models needed for client-side anomaly detection may be learned efficiently, using limited space, via coresets.

The Caltech Community Seismic Network is a prototypical example of a CSR system that harnesses accelerometers in volunteers' smartphones and consumer electronics. Using CSN, this thesis presents the systems and algorithmic techniques to design, build and evaluate a scalable network for real-time awareness of spatial phenomena such as dangerous earthquakes.

Chapter 1

Introduction

Phones, wearable sensors, and home sensing are dramatically increasing the number of ways we measure the physical world. The quantity and variety of sensors in these devices promise to make applications more powerful and intuitive by allowing online services to observe the user's immediate physical context, and infer the broader environmental context from multiple users' data. Systems utilizing networked consumer-owned sensors - collectively referred to here as *sensor-aware apps* - can have sweeping impact on how we collect, access, and act on information relevant to our physical world. For example, commercial products like Google Glass, the Jawbone activity tracking wristband, the Nest home thermostat, and the FitBit are already quantifying sensory data about our environment, our health, and our homes. Inevitably, such devices bought for personal use will provide information that, when viewed in aggregate, provides insight into large-scale social and environmental phenomena.

This thesis focuses on a particular class of sensor-aware app, called *Community Sense-and-Response* (CSR) systems, and develops technology needed to make reliable, real-time decisions about spatial events such as earthquakes and natural disasters using large numbers of unreliable consumer sensors. CSR systems are distinguished from standard web applications by the use of individually owned or operated commercial sensor hardware as a means of observing, understanding, and ultimately acting on physical events. Just as social networks connect and share human-generated content, CSR systems work to gather, share, and act on sensory data from people's internet-enabled devices. I discuss the Caltech Community Seismic Network as a prototypical CSR system harnessing accelerometers in smartphones and consumer electronics, and describe the systems and algorithmic challenges of designing, building and evaluating a scalable system for real-time awareness of dangerous earthquakes using large numbers of unreliable consumer sensors.

1.1 From smart devices to sensitive devices

In the last several years, mobile computing has drastically reshaped the way that people produce and access information and is now the dominant means of internet access worldwide. 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. The smartphone and tablet market has certainly paved the road technologically for inexpensive internet-enabled, battery-powered sensor devices, but perhaps equally importantly, mobile computing has created the expectation that web services should be ubiquitously available during daily life. I conjecture that the omnipresence of mobile devices has also led many people to an expectation or desire that digital services be cognizant of their immediate physical context, providing contextually relevant information as needed and retreating unobtrusively at other times.

These trends partly explain the recent emergence of so many sensor-oriented consumer devices. Many products in this space are directly marketed towards personal sensing, environmental sensing, and providing contextually relevant information. Examples of personal sensing include health and wellness sensors such as the Basis smart watch, or the Nike FuelBand activity tracker and fitness monitor. Home-oriented products from Nest (such as thermostats and smoke alarms) technologically refresh familiar domestic products with the intent of optimizing their owner's immediate environment through unobtrusive measurements and mobile-friendly interfaces. Google's recent $3B acquisition of Nest and its foray into wearable technologies (Glass and Android Wear) demonstrate the web giant's increasing interest in sourcing physically relevant information to better provide its users with "The right information at just the right time".

1.2 The sensed world

> "The paramount value of the devices, in a sense, lies not in the hardware itself but the interconnectedness of that hardware. As the devices talk to each other, by building an aggregate picture of human behavior, they anticipate what we want before even we know." (Marcus Wohlsen, on Google's $3B acquisition of Nest)

Nearly 2 million Android and iOS devices are activated every day, each carrying numerous sensors and a high-speed internet connection. At present the core Android mobile OS recognizes 13 distinct sensor types, with additional "composite" sensors such as step counters. Apple's iOS offers similar sensors and a "motion co-processor". These sensors are largely focused on the phone and tablet market, but new product form factors like wristbands, glasses, watches, and home sensing promise to dramatically increase the repertoire of sensors available to the consumer. While the immediate

intent of these products is to sense the user's personal environment, the same technology is equally applicable to sensing city-wide or global environments.

Burke [16] first coined the term *participatory sensing* to describe potential applications that "task deployed mobile devices to form interactive, participatory sensor networks that enable public and professional users to gather, analyze and share local knowledge," and indeed, the last several years have witnessed the creation of several projects that seek to crowdsource sensor data from public volunteers or app users via consumer hardware and web technologies. For example, applications of participatory sensing include:

- Understanding traffic flows [63, 98, 68, 12]
- Identifying sources of pollution [6, 2]
- Monitoring public health [83]
- Responding to natural disasters like hurricanes, floods, and earthquakes [24, 35, 37, 66]
- Predicting adoption of trends and estimating app installation rates [90]

Additionally, [14] provides an excellent overview of how the tools of the social and mobile Web facilitate crowdsourcing data from individuals and their sensor devices, with an emphasis on human participation in sharing and understanding data.

These systems are made possible by volunteer sensors and low-cost web solutions for data collection and storage. However, as these systems mature, they will undoubtedly extend beyond data collection and begin to take real-time action on the community's behalf. For example, traffic networks may reroute traffic around an accident, or a seismic network may automatically slow trains to prevent derailing.

1.3 Reliable actions from unreliable data

In principle, sensing the world through consumer hardware is merely an HTTP PUT away. However, one of the main claims of this thesis is that acting on data from community sensors is fundamentally different from acting on data from scientific sensors, as well as fundamentally different from acting on data from most web applications. 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 data recorded by community sensors often include signals produced by the people who operate them. Additionally, many of the desired applications push the limits of our current understanding of physical phenomena.

Scale. Given that the most popular mobile apps can boast more than 100,000,000 downloads, it is exciting to imagine sensor-aware apps growing into vast global sensor applications. However, the

volume of raw data that could be produced by such a CSR network is astounding by any standard. Smartphones and other consumer devices often have multiple sensors, and can produce continuous streams of GPS position, acceleration, rotation, audio, and video data. While events of interest (e.g., traffic accidents, earthquakes, disease outbreaks) may be rare, devices must monitor continuously in order to detect them. Beyond obvious data heavyweights like video, rapidly monitoring even a *single* accelerometer or microphone produces hundreds of megabytes per day. Such data rates become challenging when we consider sensor-aware apps 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 [109]. 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.[1]

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

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

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

1.4 The Caltech Community Seismic Network

CSR systems have the potential to change how we understand and act on city-wide physical phenomena like natural disasters. One compelling example if this is the Caltech Community Seismic Network

[1] http://www.sec.gov/Archives/edgar/data/1418091/000119312513390321/d564001ds1.htm

(CSN) project. CSN is a system for real-time earthquake measurement and response. In contrast to traditional seismic networks that use a relatively small number of high quality seismometers, CSN emphasizes large numbers of consumer-grade sensors that are owned and operated by volunteers in the community. While this presents numerous challenges, it also promises a degree of spatial resoultion that would be infeasible for traditional networks. Indeed, given that smartphones, game consoles, and many wearable devices are equipped with MEMS accelerometers, it's fair to say that millions of sensors have already been deployed and are merely waiting to be harnessed. Using even a fraction of these sensors could provide a high spatial density that makes possible a range of applications in disaster response and earth science. The CSN system is used as a motivating example throughout much of this thesis; it is presented in detail in Chapter 3, and is outlined here.

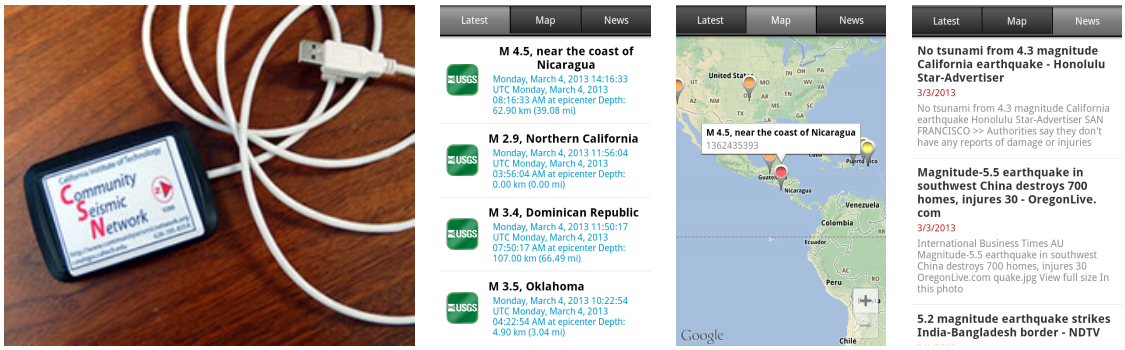


Figure 1.1: CSN volunteers contribute data from low-cost accelerometers (left) and from Android smartphones via the CSN app *CrowdShake* (right).

1.4.1 Community Sensors

The CSN project is designed to incorporate large numbers home and personal sensor devices that stream data to cloud applications. To better understand how these devices can be used for event detection and response, the CSN project has developed client applications for home (desktop) and personal (phone and tablet) platforms. Fig. 1.1 illustrates a USB accelerometer that volunteers may connect directly to their home laptop or desktop and an Android app for phones and tablets. These client types differ drastically in their ambient noise profiles, as well as in available computation and network resources. For example, continuously transmitting data may be acceptable for a desktop client, but would be an excessive drain on battery power and bandwidth for a mobile client.

1.4.2 Applications

Low cost, volunteer sensor devices make it possible to network larger numbers of sensors than has been done in traditional seismic networks. For example, the California Integrated Seismic Network operates several hundred sensors across California, while CSN seeks thousands of sensors

per city. Instrumenting a city block-by-block or building-by-building offers the opportunity for several valuable applications, including detailed disaster response planning, earth science, building health monitoring, and earthquake early warning.

Disaster response planning. Given that a large earthquake is likely to produce varying levels of damage throughout a city, emergency response teams would benefit from detailed maps of damage severity while planning emergency responses. At present, aerial photos and satellite images are used to assess damage after severe earthquakes, as shown in Fig. 1.2. This reconnaissance costs precious time. Instead, a city-wide network of sensors could provide fine-grained estimates of shaking intensity as a proxy for damage levels.

Earth science. The complex patterns of acceleration caused by earthquakes are due in part to variations in subsurface structure, and the particular details of the fault rupture process. High spatial sensor density provides observations of these effects, reducing the need for model interpolation between sensor locations.

Building health monitoring. Earthquakes may damage buildings in ways that are not readily apparent, but that could be inferred from sensors. For example, changes in a building's resonant frequency may indicate a loss of stiffness resulting from cracks or broken welds. Similarly, the occurrence of structural damage may manifest in acoustic or accelerometer data. Estimates of structural health could be used to prioritize post-quake inspections and help determine which buildings are safe to re-enter.

Earthquake Early Warning. Earthquake Early Warning (EEW) is another exciting application of community seismic networks. EEW operates on the principle that once an earthquake is detected, warning messages can be transmitted ahead of the relatively slow seismic waves. Warning times of tens of seconds to a minute are possible, though warning depends on the speed of detection and the distance from detectors to the recipient of the warning message. Consequently, large numbers of sensors may help in rapidly detecting an event. Community networks may also allow countries with limited seismic infrastructure - such as Haiti, Peru, or India - to establish early warning systems.

1.5 Decentralized Event Detection

Once sensor data is available, what is involved in using it? Using earthquakes as a concrete example, suppose that a strong earthquake begins near a metropolitan area, and that a 0.1% of the population contributes accelerometer data from a personally-owned internet-enabled device. In Los Angeles county, this means data from 10,000 noisy sensors located on a coastal basin of rock and sediment, striped with fault lines, and cross-hatched with vibration-producing freeways. A recent seismic study

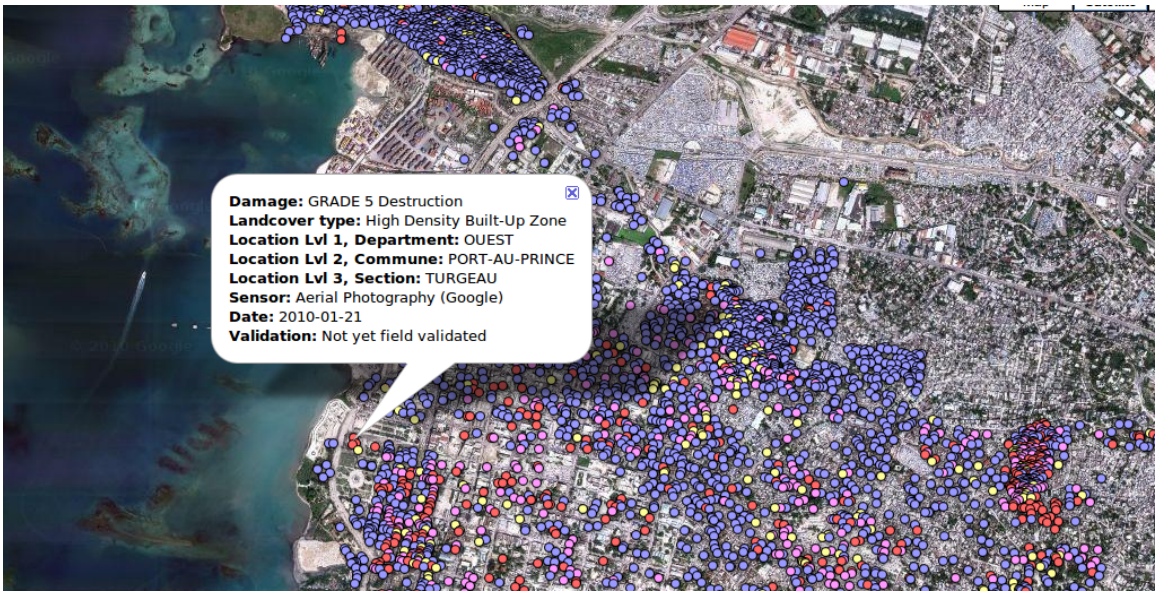


Figure 1.2: A satellite-based damage map from Haiti, following the January 12, 2010 earthquake, depicts localized variation in damage. Sensors could provide such information to emergency teams without waiting for aerial photos. Image provided by Google.

in Long Beach, CA demonstrates this in Fig. 1.3. How could we detect the quake, and estimate its location and magnitude as quickly as possible?

One direct approach from detection theory for determining the occurrence of an event (i.e., quake) from noisy observations is to collect all data centrally, and perform classification using a likelihood ratio test:

$$\frac{\mathbb{P}\left[\text{ all measurements } \middle|\text{ strong quake }\right]}{\mathbb{P}\left[\text{ all measurements } \middle|\text{ no quake }\right]} > \tau \tag{1.1}$$

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

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

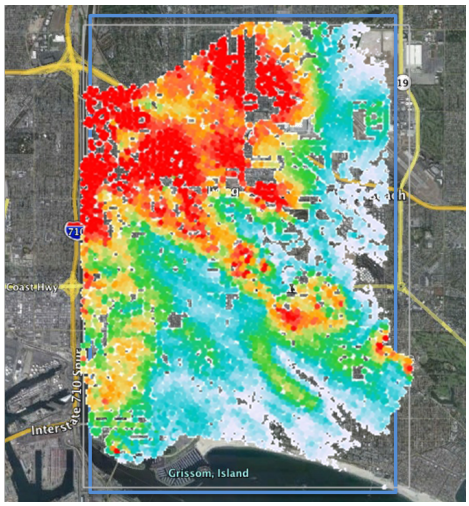


Figure 1.3: A map of peak acceleration amplitudes experienced in Long Beach, California during the Carson earthquake, as recorded by Signal Hill Petroleum. The map shows large variations in peak acceleration and the effects of a minor fault line.

[99]: each sensor individually performs a hypothesis test, for some threshold τ, and picks only when

$$\frac{\mathbb{P}\left[\text{ one sensor's measurements } \middle|\text{ strong quake }\right]}{\mathbb{P}\left[\text{ one sensor's measurements } \middle|\text{ no quake }\right]} > \tau. \tag{1.2}$$

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

$$\frac{\text{Bin}(S; r_T; N)}{\text{Bin}(S; r_F; N)} \geq \tau', \tag{1.3}$$

The parameters r_T and r_F are the true positive and false positive pick rates for a single sensor, and $\text{Bin}(\cdot, p, N)$ is the probability mass function of the Binomial distribution. Asymptotically optimal decision performance can be obtained by using the decision rules (1.2) and (4.2) with proper choice of the thresholds τ and τ' [99]. Additionally, collecting picks instead of raw data may help preserve user privacy by reducing the amount of raw sensor data that is centrally collected.

Challenges for the classical approach. While this classical, decentralized detection approach appears promising, detecting rare events from community sensors presents three main challenges:

1. How can we perform likelihood ratio tests on each sensor's data, when we do not have enough data (e.g., measurements of large, rare quakes) to accurately model sensor behavior during an event?

2. How can we model each sensor? Server-side modeling scales poorly, while on-device learning involves computational and storage limits.

3. How can we overcome the (strong) assumption of conditionally independent sensors, and incorporate spatial dependencies?

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

1.5.1 Leveraging "normal" data

The client-side hypothesis test in (1.2) requires two conditional probability distributions. The numerator models a particular device's acceleration during a strong quake, and due to the rarity of large quakes is impractical to obtain. In contrast, the denominator can be estimated from abundantly available "normal" data. Can we still hope to produce reliable picks?

It turns out that under mild conditions, a simple anomaly detection approach that uses only the probability of an acceleration time series in the absence of a quake can obtain the same asymptotically optimal performance. A given sensor now picks when

$$\mathbb{P}\left[\text{ one sensor's measurements } \middle|\text{ no quake }\right] < \tau. \tag{1.4}$$

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

1.5.2 Learning on smartphones with Coresets

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

In the CSN system, sensor data is modeled by a Gaussian Mixture Model (GMM) over a feature vector of acceleration statistics from short time windows, similar to how phonemes are used in speech recognition. GMMs are flexible, multi-modal distributions that can be practically estimated from data using the simple EM algorithm [11]. In contrast to estimating a single Gaussian, which can be fit knowing only the mean and variance of the data, estimating a GMM requires access to all the data; formally, GMMs do not admit finite sufficient statistics. This precludes, for example, our

ability to compress the 3GB of monthly acceleration data and still recover the same GMM that would have been learned from the full data. Fortunately, it turns out that the picture is drastically different for approximating a GMM: a GMM can be fit to an arbitrary amount of data, with an arbitrary approximation guarantee, using a finite amount of storage!

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

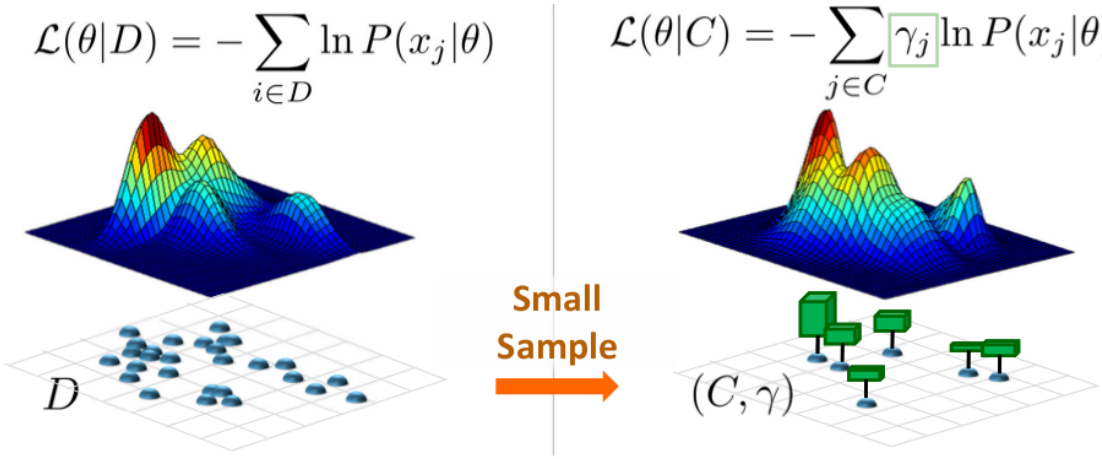


Figure 1.4: A coreset for Gaussian Mixture Models is a small, weighted subset of the data that allows us to fit a model that is "almost as good as" a model fit to the full dataset. The green bars denote the scalar weights γ_j associated with each selected point.

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

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

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

C can obtain approximately the same likelihood as learning the model from the entire, arbitrarily large D.

But how do we find C? [44] showed that efficient algorithms to compute coresets for projective clustering problems (e.g., k-means and generalizations) can provide coresets for GMMs. A key insight is that while uniformly subsampling the input may miss "important" regions of data, an adaptive sampling approach is likely to sample from "enough" regions to reliably estimate a mixture of k Gaussians; weighting the samples accounts for the sampling bias. Previous work [59] also identified that coresets for many optimization problems can be computed efficiently in the parallel or streaming model, and several of those results apply here. In particular, a stream of input data can be buffered to some constant size, and then compressed into a coreset. Careful merging and compressing of such coresets provides an approximation to the entire stream so far, while using space and update time polynomial in all the parameters, and logarithmic in n.

1.5.3 Understanding Spatial Phenomena

Quake detection in community networks requires finding a complex spatio-temporal pattern in a large set of noisy sensor measurements. The start of a quake may only affect a small fraction of the network, so the event can easily be concealed in both single-sensor measurements and network-wide statistics. Data from recent high-density seismic studies, shown in Fig. 1.3, demonstrate that localized variations in ground structure significantly impact the magnitude of shaking at locations only a kilometer apart. Consequently, effectively detecting earthquakes and estimating their impact requires algorithms that can learn subtle dependencies among sensor data, and detect changes within groups of dependent sensors.

The "classical" approach described in Sec. 1.5 assumes that the sensors provide independent, identically distributed measurements conditioned on the occurrence or non-occurrence of an event. In this case, the Cloud server would declare a detection if a sufficiently large number of sensors report picks. However, in many practical applications, the particular spatial configuration of the sensors matters, and the independence assumption is violated. Here, the natural question arises of how (qualitative) knowledge about the nature of the event can be exploited in order to improve detection performance.

Suppose that a small fraction of sensors in a large network observe the initial effects of a spatial phenomena. For example, a small number of smoke detectors may observe the start of a fire, or seismometers near a fault line may first observe shaking. If each sensor transmits a binary observation, the signal received by the Cloud server may be viewed as transmitting a vector $\mathbf{x} \in \mathbb{R}^p$ through a noisy channel where the signal is mostly zeros (sparse), but many bits in the received vector $\mathbf{y}$ are flipped due to noise. We should expect nearby sensors to be strongly correlated during a quake. If we knew groups of correlated sensors, detection could be improved by testing each group

separately.It turns out that this intuition (and some desirable analytic properties) can be captured by learning a orthonormal change-of-basis matrix that projects the binary messages received by the server onto a coordinate system that, roughly, aggregates groups of strongly correlated sensors. Given such a matrix $\mathbb{B}$ with columns $\mathbf{b}_i, \ldots, \mathbf{b}_p$, the server declares an event when

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

where τ again denotes some scalar threshold. To obtain reliable detection when the signal is weak (measured by the ℓ_0 pseudo-norm, $||\mathbf{x}||_0 < \sqrt{p}$), traditional hypothesis testing requires the error rate of each sensor (each element of $\mathbf{x}$) to *decrease* as the number of sensors p increases. This is in stark contrast to our intuition that more sensors should be better, and in contrast to the "numerous-but-noisy" approach of community sensing. However, Chapter 5 shows that if the matrix $\mathbf{B}$ is *sparsifying*, i.e., $||\mathbf{B}^T\mathbf{x}||_0 = p^\beta$, $||\mathbf{x}||_0 = p^\alpha$, $0 < \beta < \alpha < 1/2$, then the test $\max_i \mathbf{b}_i^T\mathbf{y} > \tau$ gives probability of miss and false alarm that decays to zero exponentially as a function of the "sparsification ratio" $||\mathbf{x}||_0/||\mathbf{B}^T\mathbf{x}||_0$, for any rate $r_F < 1/2$ of pick errors. Effectively, this allows large numbers of noisy sensors to contribute to reliable detection of signals that are observed only by a small fraction ($||\mathbf{x}||_0$) of sensors.

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

$$\min_{\mathbf{B}} ||\mathbf{B}^T\mathbf{X}||_0, \text{ subject to } \mathbf{B}\mathbf{B}^T = \mathbf{I} \tag{1.5}$$

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

(5.3) can be impractical to compute, and can be sensitive to noise or outliers in the data. Instead, we may wish to find a basis that sparsely represents "most of" the observations. More formally, we introduce a latent matrix $\mathbf{Z}$, which can be thought of as the "cause", in the transform domain, of the noise-free signals $\mathbf{X}$. In other words $\mathbf{X} = \mathbf{B}\mathbf{Z}$. We desire $\mathbf{Z}$ to be sparse, and $\mathbf{B}\mathbf{Z}$ to be close to the observed signal $\mathbf{Y}$. This suggests the next optimization, originally introduced for text modeling [20], as a heuristic for (5.3):

$$\min_{\mathbf{B},\mathbf{Z}} ||\mathbf{Y} - \mathbf{B}\mathbf{Z}||_F^2 + \lambda||\mathbf{Z}||_1, \text{ subject to } \mathbf{B}\mathbf{B}^T = \mathbf{I} \tag{1.6}$$

where $||\cdot||_F$ is the matrix Frobenius norm, and $\lambda > 0$ is a free parameter. (5.6) essentially balances the difference between $\mathbf{Y}$ and $\mathbf{X}$ with the sparsity of $\mathbf{Z}$: increasing λ more strongly penalizes choices of $\mathbf{Z}$ that are not sparse. For computational efficiency, the ℓ_0-norm is replaced by the convex and

heuristically "sparsity-promoting" ℓ_1-norm.

Although (5.6) is non-convex, fixing either $\mathbf{B}$ or $\mathbf{Z}$ makes the objective function with respect to the other convex. The objective can then be efficiently solved (to a local minima) via an iterative two-step convex optimization process.

Empirically, this approach gives strong performance in several event detection applications, including quake measurements of 1795 earthquakes following the Japanese Tohuku M9.0 quake, quake measurements from the Signal Hill dense seismic study, from the Community Seismic Network as well as simulated virus outbreaks in the Gnutella P2P network.

1.6 Contributions and Outline of the thesis

Figure 1.5: An architecture for decentralized event detection: each consumer device runs a client application for modeling its sensor data and transmits pick messages about potential events. A cloud server receives the stream of pick messages, and detects spatial-temporal events.

The remainder of this thesis presents systems and algorithms for large-scale event detection in CSR systems. For reference, it is helpful to refer to Fig. 1.5, which presents a cartoon overview of a system where individual sensor clients communicate directly with a cloud server. The sensor clients are responsible for local data modeling and processing, while the (cloud) server performs system-wide event detection.

Chapter 3 describes the CSN project in detail, which can be viewed as the overall structure of Fig. 1.5. CSN is a large, multi-year effort that reflects the work of many individuals. My primary contribution to the system, and the first contribution of this thesis, is the design and implementation of an Android sensor client that uses the accelerometer and GPS in smartphones for event detection and data collection. This chapter also discusses CSN components that operate alongside the Android client, including the cloud server, developed on Google App Engine by Michael Olson and Julian Bunn, and a desktop client (for Mac, Linux, and Windows) for CSN, developed by Lief Strand. The CSN system was first described in [37] and [21].

The second contribution of this thesis, presented in Chapter 4, is theory for decentralized detection of rare events, design choices for implementing the algorithm on Android devices, and experimental investigations of earthquake detection performance. This can be thought of as a particular implementation of the client-level processing and server-level event detection in Fig. 1.5. This work first appeared in [37].

The third contribution, presented in Chapter 5, is theory and experimental investigation of detecting spatially-structured signals. In [36], I formulate an event detection model of sparse binary signals observed through a noisy binary channel, and demonstrate that a sparsifying transform (designed from domain knowledge or learned from data) allows a network of numerous-but-noisy sensors to detect otherwise weak events. Chapter 5 also discusses preliminary investigations, first published as joint work in [88], into detecting spatially structured signals. Notably, this work incorporates sensor models from [37] into a simulation framework developed by Annie Liu. This chapter can be thought of as providing a more advanced design for the server-level event detection component of the above system cartoon.

The final contribution of the thesis is an application of coresets for Gaussian Mixture models to community sensing. Leveraging earlier results by Daniel Feldman and other on using coresets to approximate clustering tasks, [44] develops approximation results for Gaussian mixture models and related distributions. The ability to compute coresets in both parallel and streaming settings has large potential impact for learning from sensor data on mobile devices. I demonstrate that coresets provide practical approximations on a variety of real-world tasks, including data from smartphone accelerometers. Coresets provide a way to efficiently learn a probabilistic model of data at the sensor client level.

Chapter 2

Related Work

The work in this thesis centers on using mobile and cloud computing platforms to build sensor-aware applications capable of making reliable, real-time decisions, and as such is related to several bodies of research.

2.1 WSNs and Community sensing

Collecting and understanding sensor data is a key component of several disciplines, including wireless sensor networks (WSNs), community sensing, and participatory sensing.

Research in WSNs is similar in spirit to the concept of sensor-aware apps, but tends to differ in fundamental assumptions about target applications and resource availability. WSN research often assumes a network of limited-CPU devices with power-constrained short-range wireless radios, typically connected to each other and a centralized data sink via a multi-hop network. These assumptions have been partly motivated by a desire to perform long-term, unattended environmental monitoring. In such a setting, sensor devices may be required to operate on a strict power budget that precludes standard wifi or cellular communication.

Community / Participatory sensing has been used effectively in a variety of problem domains for environmental monitoring and data collection via community-owned or operated sensors. Several projects have used smartphones' position and motion sensing to

- monitor traffic flows and road conditions [63, 98, 68, 12, 81, 62]
- Community-owned sensors offer great potential in environmental monitoring [83, 103] by obtaining up-to-date measurements of the conditions participants are exposed to.
- Mobile phones and body sensors are used to encourage physical activity by categorizing body motion and comparing activities to exercise goals [25],
- Monitoring public health [83],

- Identifying sources of pollution [6, 2],
- Responding to natural disasters like hurricanes, floods, and earthquakes [24, 35, 37, 66]

Additionally, [14] provides an excellent overview of how the tools of the social and mobile Web facilitate crowdsourcing data from individuals and their sensor devices, with an emphasis on human participation in sharing and understanding data.

Commercial Sensor-aware apps. Technology from these fields has found new application in several commercial applications. Waze[1] crowd-sourced traffic and navigation, Nooly[2] and WeatherSignal[3] crowd-sourced weather apps, the Vaavud[4] wind meter for smartphones, and KitLocate's[5] tools for low-power position tracking and geofencing. A glance at TechCrunch will often provide a glimpse of the sensor app *du jour*. The Funf open sensing framework[6] is particularly relevant, as it allows the creation of Android apps for sensor data collection.

2.2 Seismic networks

CSN should be viewed in light of existing traditional seismic networks, and in light of recent projects that explore lower-cost consumer sensor technologies. While these projects differ in their sensors, algorithms and scope, they share a common intent of detecting earthquakes, mapping their intensity, and providing warnings and data products for emergency responders.

Among non-traditional seismic networks, perhaps the most closely related system is the QuakeCatcher network [24]. While QuakeCatcher shares the use of MEMS accelerometers in USB devices and laptops, the CSN 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. Mobile phones create far more complex statistical challenges, but also promise to facilitate much larger groups of volunteers. Kapoor et al. [66] analyzes the increase in call volume after or during an event to detect earthquakes. Another related effort is the NetQuakes project [101], which deploys more expensive stand-alone seismographs with the help of community participation. Our CSN Phidget sensors achieve different tradeoffs between cost and accuracy.

Japanese seismic network. Japan experiences a large number of seismic events due to earthquakes and volcanic activity, and contains several large seismic sensor networks. The Japan Metereological Agency (JMA) operates a seismic network containing roughly 1000 seismic sensors.

[1] www.waze.com
[2] www.nooly.com
[3] weathersignal.com
[4] vaavud.com
[5] www.kitlocate.com
[6] https://code.google.com/p/funf-open-sensing-framework/

Additionally, the National Research Institute for Earth Science and Disaster Prevention (NIED) currently operates 777 sensor stations, and Japanese local governments currently operate nearly 3000 sensor stations. The data from these sensors is used to detect earthquakes and produce estimates of hypocenter, magnitude and observed seismic intensity. The JMA uses information from these sensors to provide early warning alerts and issue reports to the public and disaster relief workers within minutes of a quake.

CISN. The California Integrated Seismic Network (CISN) is a collaboration among state, federal, and university organizations that perform earthquake monitoring. The network spans approximately 400 seismic stations located throughout California. The network is comprised of a variety of sensor types and technologies, including short-period, broadband, and strong-motion sensors. Fig. 2.1 depicts a "borehole" seismic station in the CISN network. Using these sensors, CISN determines earthquake warnings using the P-wave pulse width and peak initial 3-second displacement amplitude [107].

Figure 2.1: A CISN sensor station installed in a "borehole". Installing and maintaining such stations represents significant investment, but provides exceptional quality.

GPS displacement networks. Networks of high sample rate Global Positioning System (GPS) sensors seek to quickly and accurately observe surface displacements. Large earthquakes can produce centimeters to meters of ground surface displacement. Accurately estimating displacement in real-time and incorporating those estimates into early warning algorithms is a topic of ongoing research [74].

Earthquake Early Warning. 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 [26] 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 [5] 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. Recently, PreSEIS [13] uses a 2-layer feed-forward neural network trained on historic seismic data to estimate quake parameters including hypocenter location, moment magnitude, and earthquake rupture movement in real time.

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.

2.3 Decentralized detection

There has been a great deal of work in decentralized detection. The classical hierarchical hypothesis testing approach has been analyzed by Tsitsiklis [99]. Chamberland et al. [17] 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. [106] 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. [80] 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.

2.4 Anomaly Detection

There has also been significant amount of prior work on anomaly detection. Yamanishi et al. [108] 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. [31] 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. [97] 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. [92] 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. [110] demonstrate online SVMs to detect anomalies in process system calls in the context of intrusion detection. Onat et al. [89] 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.

2.5 Sparse detection

Detecting a sparse signal in the presence of strong noise is challenging without placing some assumptions on the class of signals. Signals that possess tree-structured dependencies naturally lend themselves to wavelet analysis where the basis is supported on subtrees. Singh et al.[96] proposed using a natural Haar-like wavelet basis constructed from agglomerative hierarchical clustering to encode the dependencies for improved detection under strong noise. Gavish [50] and Lee [73] also propose multi-scale bases for signals with tree structure. [96] and [73] construct a transform using agglomerative clustering to identify hierarchical tree structure. Krishnamrurthy et al.[69] further extends the analysis to graph structured network defined over spanning trees.

The work of Singh et al.[96] is particularly relevant, as they identify the asymptotic limits of detectability for the orthonormal basis and generative models used in Chapter 5. In contrast, we focus on the decentralized case with binary channel noise, and provide theoretical guarantees that hold even in the non-asymptotic regime. Several other possible forms of structured activation patterns have also been applied to sparse detection problems. [7] describes detecting sparse binary patterns with a variety of combinatoric structures under Gaussian noise. Lower bounds on minimax detection rates are given, and it is shown that forms of the scan statistic achieve within a log factor

of these rates. However, these results are asymptotic and appear computationally intractable for large sensor networks.

2.6 Scan statistics

Spatial and space-time scan statistics were first developed to monitor health data and disease outbreaks [70]. The main idea is to evaluate all subsets of the data for possible events. Of course, enumerating all possible $O(2^n)$ subsets is infeasible for even moderately sized problems, so later variations impose constraints on space and time to reduce the number of subsets, which is often done by scanning only subsets of distinct sizes and shapes [85, 86]. This approach reduces the complexity to $O(n^2)$ (and to $O(n)$ in [86]) but may also impair the detection performance if important signals are not well captured by tested subsets.

Linear Time Subset Scanning was recently developed to further reduce the complexity [86]. It states that for any scoring functions G, e.g., the ratio of count to baseline $G(s_i) = c_i/b_i$, that satisfies the LTSS property (that the subset S that maximizes sum of score $F(S)$ must only consist some of the top k scoring stations), then only $O(k)$ subsets should be evaluated to determine detection. However, this approach often gives undesirable spatially separated subsets and the spatial version that includes distance constraints suffers the same problem as predefined shape.

2.7 Basis Learning

Learning a sparsifying basis is intimately related to dictionary learning and topic models. Dictionary learning [4] attempts to find an overcomplete dictionary $\mathbf{D} \in \mathbb{R}^{n,K}$, $K > n$ that can sparsely encode signals in $\mathbb{R}^n$. Similarly, topic models [104] represent text data as a linear combination of a "topics", e.g., vectors of word frequencies. Topic models seek topics that sparsely approximate the documents, though the number of topics is significantly less than the number of words (i.e., the topic matrix is undercomplete).

ICA is a transformation method developed to recover nongaussian, statistically independent components $\mathbf{z}$ from their linear combination $\mathbf{x}$ [65], assuming that the linear transformation matrix $\mathbf{B}$ is orthonormal and $\mathbf{x} = \mathbf{B}\mathbf{z}$. The nongaussianity is required because the orthogonal transformation of any number of Gaussian distributions is inseparable. Strongly nongaussian data (i.e., having a very different distribution from Gaussian) is often sparse, and so nongaussianity is yet another measure of sparsity. ICA has enjoyed most success in signal separation and unsupervised feature learning. Recent work has extended it for overcomplete dictionary learning [72].

2.8 Theoretical results on mixtures of Gaussians

There has been a significant amount of work on learning and applying GMMs (and more general distributions). Perhaps the most commonly used technique in practice is the EM algorithm [33], which is however only guaranteed to converge to a local optimum of the likelihood. Dasgupta [28] is the first to show that parameters of an unknown GMM P can be estimated in polynomial time, with arbitrary accuracy ε, given i.i.d. samples from P. However, his algorithm assumes a common covariance, bounded excentricity, a (known) bound on the smallest component weight, as well as a separation (distance of the means), that scales as $\Omega(\sqrt{d})$. Subsequent works relax the assumption on separation to $d^{1/4}$ [29] and $k^{1/4}$ [102]. [8] is the first to learn general GMMs, with separation $d^{1/4}$. [45] provides the first result that does not require any separation, but assumes that the Gaussians are axis-aligned. Recently, [82] and [9] provide algorithms with polynomial running time (except exponential dependence on k) and sample complexity for arbitrary GMMs. However, in contrast to our results, all the results described above crucially rely on the fact that the data set D is actually generated by a mixture of Gaussians. The problem of fitting a mixture model with near-optimal log-likelihood for arbitrary data is studied by [8], who provides a PTAS for this problem. However, their result requires that the Gaussians are identical spheres, in which case the maximum likelihood problem is identical to the k-means problem. In contrast, our results make only mild assumptions about the Gaussian components. Furthermore, none of the algorithms described above applies to the streaming or parallel setting.

2.9 Coresets

Approximation algorithms in computational geometry often make use of random sampling, feature extraction, and ϵ-samples [60]. Coresets can be viewed as a general concept that includes all of the above, and more. For a comprehensive survey on this topic, see [42]. It is not clear that there is any commonly agreed-upon definition of a coreset, despite several inconsistent attempts to do so [57, 43].

Coresets have been the subject of many recent papers and several surveys [3, 27]. They have been used to great effect for a host of geometric and graph problems, including k-median [57], k-mean [43], k-center [58], k-line median [41] subspace approximation [41, 78], etc. Coresets also imply streaming algorithms for many of these problems [57, 3, 47, 43, 46, 71]. A framework that generalizes and improves several of these results has recently appeared in [42].

Chapter 3

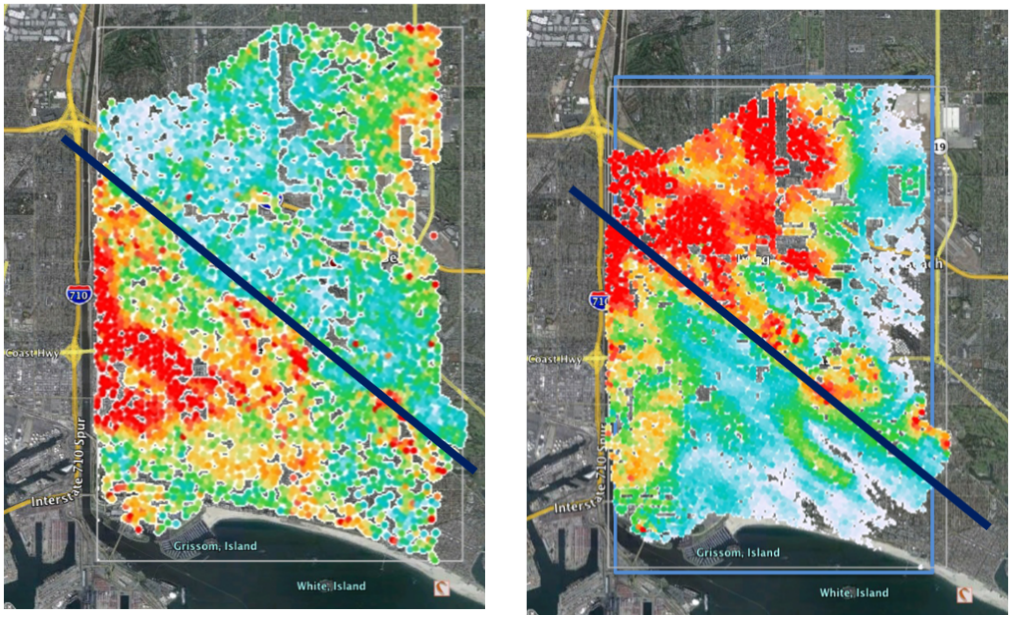
The Caltech Community Seismic Network

The *Community Seismic Network* project at Caltech is a collaboration between civil engineers, geophysicists, and computer scientists that seeks to rapidly detect earthquakes and provide real-time estimates of their impact using community-operated sensors. Given that large earthquakes are among the few scenarios that can threaten an entire city, the CSN project is built upon a vision of people sharing accelerometer data from their personal devices to collectively produce the information needed for effective real-time and post-event responses to dangerous earthquakes. To that end, CSN has partnered with more than a thousand volunteers in the Los Angeles area and cities around the world who contribute real-time acceleration data from their Android smartphones and low-cost USB-connected sensors. This chapter, based on [37], describes the project goals and challenges, as well as the client-server architecture implemented on Google App Engine, Android phones, and desktop computers. The main contribution is the design and implementation of the CSN-Droid and CrowdShake Android client.

3.1 Goals

Rapid Situational Awareness. After an earthquake, fire fighters, medical teams and other first-responders must build *situational awareness* before they can effectively deploy their resources. Due to variations in ground structure, two points that are only a kilometer apart can experience significantly different levels of shaking and damage, as illustrated in Fig. 3.1(a). Similarly, different buildings may receive differing amounts of damage due to the types of motion they experience. These differences in shaking can be important for allocating emergency relief resources. However, if communication has been lost in a city, it can take up to an hour for helicopter surveillance to provide the first complete picture of the damage a city has sustained. In contrast, a seismic network with fine spatial resolution could provide accurate measurements of shaking (and thus an estimate

of damage) *immediately*. Because sensors can detect the moderate P-wave shaking that precedes the damaging S-wave shaking, sensors are expected to report data before network and power are lost, and before cellular networks are overloaded by human communication.



(a) Carson (b) Compton

Figure 3.1: A map of peak acceleration amplitudes during the Carson and Compton earthquakes, as recorded by Signal Hill Petroleum in Long Beach California. The maps shows large variations in peak acceleration and the effects of a minor fault line, indicated by the overlaid line. Information on localized shaking intensity is valuable to emergency teams, and for mapping subterranean geological structures.

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

3.2 Crowdsourcing environmental sensing

The CSN project turns to volunteer participation from members of the community to obtain its sensor coverage. This community participation is ideal for seismic sensing for several reasons. First, community participation makes possible the densely distributed sensors needed for accurately measuring shaking throughout a city. For example, instrumenting the greater Los Angeles area at a spatial resolution of 1 sensor per square kilometer would require over 10,000 sensors. While traditional seismometer stations cost thousands of dollars *per sensor* to install and operate, the same number of sensors could be reached if 0.5% of the area's population volunteered data from their smartphones. The effects of finer spatial resolution are illustrated in Fig. 3.2. In this way, community sensors can provide fine spatial coverage, and complement existing networks of sparsely deployed, high quality sensors.

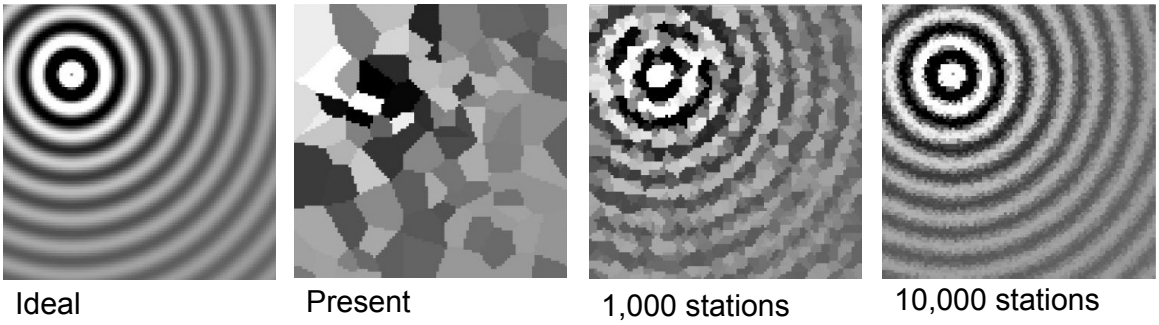


Figure 3.2: The low spatial density of many existing sensor networks makes it difficult to accurately reconstruct spatial phenomena from measurements. For example, the Southern California Seismic Network has roughly 100 sensors in the Los Angeles area. Using the present configuration of sensors, interpolation provides a crude estimate of an ideal radial pattern. Increasing the number of sensors allows even simple interpolation to adequately reconstruct the signal.

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

CSN makes it easy for the community to participate by using low-cost accelerometers and sensors already present in volunteers' Android phones. A free Android application on the Google Play app store called CSN-Droid makes volunteering data as easy as installing a new app. The CSN project also partners with LA-area schools and city infrastructure to freely distribute 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. Fig. 3.3 displays these sensors.

The recent trend towards smartphones and other Internet-enabled devices offers unique possi-

bilities for decentralized event detection. Smartphones contain a rich suite of sensors, such as GPS, accelerometers, and cameras, that can gather information about a variety of geospatial phenomena. smartphones, tablets and laptops have accelerometers that are being used by our project and others [23, 1] to obtain seismic measurements. It is exciting that worldwide cellphone coverage is dramatically increasing, and that recent projects such as Google's Project Loon ("Balloon powered internet for everyone") seek to provide internet to remote and developing areas. Global internet access and smartphone availability make it possible to extend community networks to developing regions such as Haiti, which suffers from devastating earthquakes. We speculate that as smartphones become less expensive, they will be adopted in greater numbers within developing economies and will provide a substitute for unavailable centralized infrastructure.

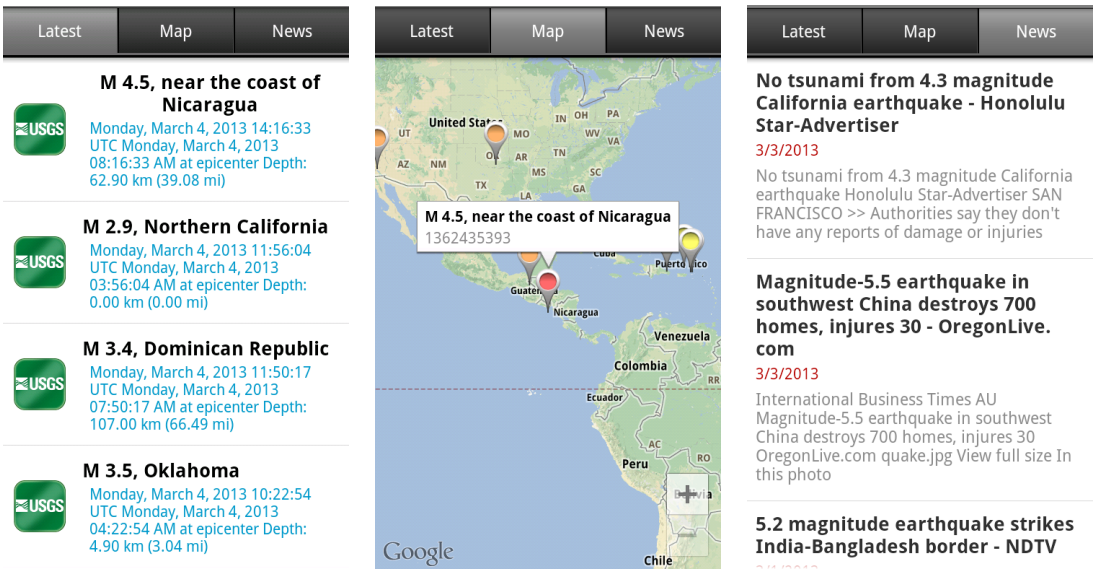


Figure 3.3: CrowdShake, a free Android app, allows volunteers to join the CSN network, contribute data, and access daily information about earthquakes worldwide.

3.3 Are consumer sensors adequate?

The low cost and ready availability of consumer sensors makes them an enticing foundation for sensing applications, but we must first ascertain that they are capable of measuring the phenomena of interest. For CSN, this means experimentally verifying that accelerometers in USB devices and mobile phones can sense typical acceleration patterns for moderately large and dangerously large earthquakes.

Android Phone Specifications. The motion sensors in smartphones have been designed foremost as an input modality for user interfaces and are often lacking in available technical specifications, and

	Mean	Std. Dev.	Min. Diff.
x (m/s^2)	-0.308843	0.0687218	0.0136203
y (m/s^2)	0.116929	0.0620888	0.0136203
z (m/s^2)	9.710770	0.0899053	0.0136200
t (s)	0.033375	0.0712159	0.0127570

Table 3.1: Mean, standard deviation, and minimum difference between measurements for the sampling rate (t) and accelerations along each of the three axes (x, y, z) recorded in one hour of stationary data on the Dev 1.

so it is necessary to empirically evaluate their sensitivity. As a simple baseline, Table 3.1 summarizes accelerometer recordings from an Android Dev Phone 1, which is functionally equivalent to the HTC Dream and T-Mobile G1, taken while the phone was stationary for one hour. Given that an M5 earthquake may produce accelerations of $0.5m/s^2$, and an M6 earthquake may produce accelerations of $1.5m/s^2$, these measurements suggest that an Android phone near the epicenter of a moderately large quake (M5 or greater) is likely to observe the shaking. Additionally, this data reflects that accelerometer samples in Android are produced at a variable rate.

Shake table validation.

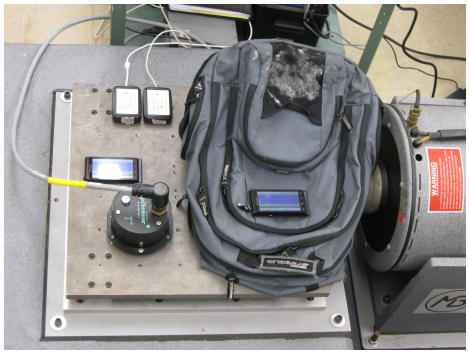
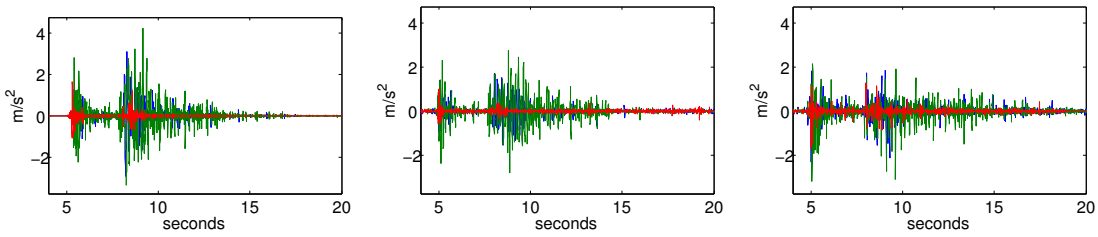


Figure 3.4: A shake table is used to reproduce the acceleration of historic quakes.

Experiments with a large actuator called a "shake table" allow us to expose sensors to accurate reproductions of historic, moderately large (M4.5-6.5) earthquakes. The shake table demonstrates that both USB sensors and the lower quality phone accelerometers can detect the smaller initial shaking (P-wave) and stronger secondary shaking (S-wave) that produce the characteristic signature of an earthquake, as shown in Fig. 3.5.

These results indicate that smartphone accelerometers are capable of observing acceleration produced by damaging earthquakes. Accelerometers in Android phones typically provide 12 to 14 bits of resolution; in contrast, 16 Phidget USB accelerometers provide significantly gerater sensitivity. Fig. 3.6 summarizes the capability of Android and Phidget sensors to detect earthquakes of different magnitudes.

A second experiment assesses whether community sensors can detect changes in the motion of



(a) 24-bit Episensor "ground truth" (b) Android phone on shake table (c) Android phone in backpack

Figure 3.5: Android accelerometers accurately record strong shaking during a shake table experiment. (a) Ground truth. (b) Android phone. (c) Android phone in backpack.

buildings caused by earthquakes.

Millikan Library on the Caltech campus is rather uniquely equipped with a large "oscillator": a rotating eccentric weight on the roof of the building. This oscillator can be used to excite the building at a controlled frequency, and can be used to study the resonant frequencies (modes) of the building. Data shown in Fig. 3.7 indicates that USB sensors can measure the low frequency resonance of large buildings. This is relevant for structural monitoring aplications, as changes in a building's resonant frequencies may indicate structural damage and a loss of stiffness.

3.4 Systems Challenges

The CSN project was designed in anticipation of several systems challenges that arise from networking vast numbers of community sensors into a reliable, real-time event detection system. These challenges include: large message volumes, rare events, heterogeneous sensors and client platforms, complex environmental noise, and user privacy. While some of these challenges are particularly relevant to earthquake monitoring, they demonstrate the breadth of issues which need to be considered for a successful CSR system.

Message volume. 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.

Rare events. The most important earthquakes are also the rarest. Empirically, the magnitude of earthquakes follows a power law distribution. The USGS reports that since 1900, on average

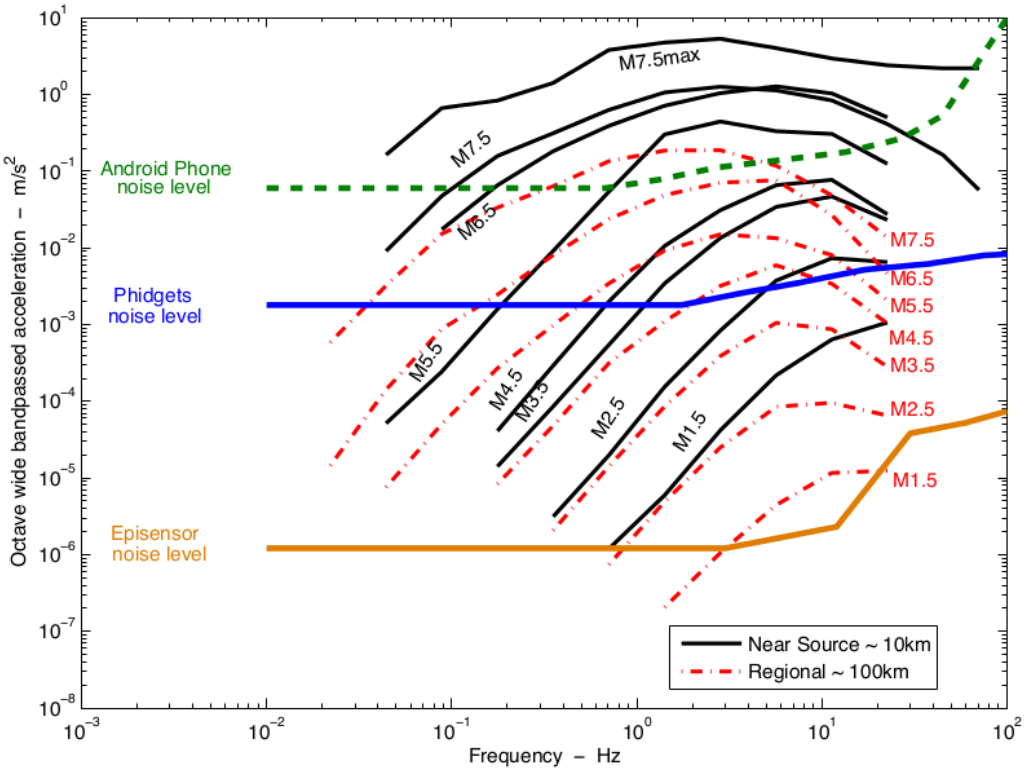


Figure 3.6: The accelerometers in smartphones (e.g., Android) and USB sensors (e.g., 16-bit Phidget accelerometers) are capable of sensing moderately large earthquakes. Scientific-quality sensors (e.g., the 24-bit Episensor) are capable of measuring quakes well below the level of human perception. Figure courtesy of Ming Hei Cheng.

only 1 earthquake of magnitude 8 or higher occurred each year.[1] In contrast, more than 100,000 moderately large (M3-M5) earthquakes occurred each year. This situation is challenging for two reasons. First, the rarity of high-magnitude earthquakes means that limited historical data is available for modeling and testing detection algorithms. It may also mean that large earthquakes have not yet been recorded in a given geographic region, and so models may need to be based on data from other, geologically different regions. Second, the relatively high rate of moderately large earthquakes compared to dangerously large earthquakes makes it important to both detect events and estimate their magnitude.

Heterogeneous sensors. Using community-based sensors for earthquake monitoring is particularly challenging due to the large variety of sensor hardware and client platforms. For example, more than 10,000 distinct Android device models have been sold, making it infeasible to perform a detailed characterization of each. Similarly, client software may need to operate across a range of platforms and operating systems, which inevitably introduce inconsistencies in reported data.

[1] http://earthquake.usgs.gov/earthquakes/eqarchives/year/eqstats.php

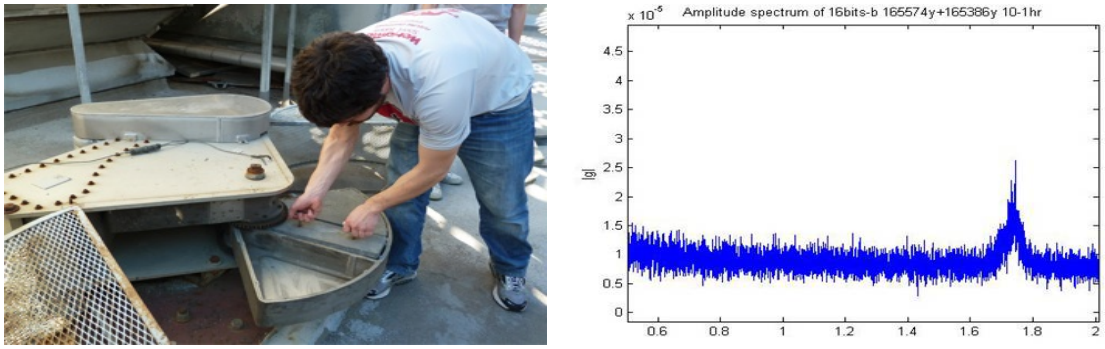


Figure 3.7: Eccentric weights oscillate Millikan Library, demonstrating that CSN hardware can observe resonant frequencies in buildings.

Further, these sensors will be mobile, intermittently available, and irregularly distributed spatially. Algorithms must account for these numerous sensor characteristics.

Complex environmental noise. Community sensors experience complex environmental noise. Community operated sensors are exposed to the unique environment of their owners: a home sensor is affected by activity within and around the house, while personal devices like phones and wearables are affected by their owner's behavior and periods of activity. While such noise reduces the reliability of each individual sensor, we may hope that much of this noise affects only one or a few sensors. Perhaps more problematic are noise sources that affect larger sets of sensors, such as construction, thunder, or the firing of the Caltech cannon. In this case, reliably detecting earthquakes means distinguishing other spatial vibrational events as non-seismic.

3.5 CSN Architecture

The CSN system involves three major components: a web server, stationary desktop clients with USB accelerometer, and an Android app for phones and tablets.

Managing a community sensor network and processing its data in real-time leads to challenges in scalability and data security. Cloud computing platforms, such as Amazon EC2, Heroku, or Google App Engine provide practical and cost-effective resources for reliably scaling web applications. The CSN network is built upon Google App Engine (GAE). Fig. 3.8 presents an overview of the CSN architecture. Heterogeneous sensors include cell phones, stand-alone sensors, and accelerometers connected via USB to host computers to the cloud. The cloud, in turn, performs event detection and issues notifications of potential seismic events. An advantage of the cloud computing system is that sensors anywhere in the world can connect merely by specifying a URL.

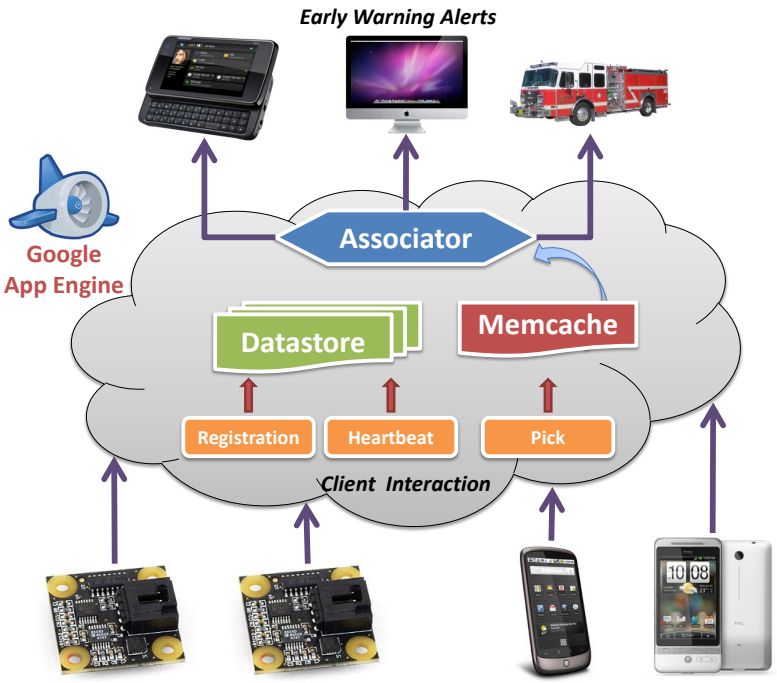


Figure 3.8: The CSN cloud maintains the persistent state of the network in Datastore, performs real-time processing of pick data via Memcache, and serves notifications and data products.

3.5.1 CSN messages

Fig. 3.8 depicts the main data flows through the cloud. Message types include *Client Registration*, *Client Update*, *Heartbeat*, *Pick*, and *CM Data Request*.

Client Registration. When a client first registers with the CSN network, the client provides a client ID and receives a secret that is used to hash the message ID in subsequent messages from the client to the Cloud server. These hashes are used to validate the origin of the message.

Client Update. Client Update messages allow the client to update metadata about the client, such as operating system version or location. This message type is not used by the Android client.

Heartbeat. Each client periodically sends a Heartbeat message to notify the server of its continuing operation. For Android clients, Heartbeat messages provide a means of maintaining relatively up-to-date position information. At present, Heartbeats are sent once every 10 minutes. Heartbeat messages may also be used to announce that they are going offline, allowing the server to perform more accurate computations involving the number of active clients in each region. The server's response to a heartbeat may contain a request for specific pieces of sensor data.

Pick. The raw stream of accelerometer data is continuously tested for anomalies, which are reported as pick messages. The Internet and cellphone networks are likely to be congested, if not damaged,

immediately after the occurrence of a disaster. Therefore, sensor data must be sent through the communication network to remote servers before the network gets congested or fails. Each message from a sensor should be short and self contained so that the server can process messages received out of order and after varying delays. A message cannot reference a previous message because the previous message may be lost.

GCM data request. First, client registration and heartbeat messages are persisted to the geographically-replicated Datastore. Next, incoming picks are spatially aggregated via geographic hashing into Memcache (a distributed in-memory data cache). While memcache is not persistent (objects can be ejected from the cache due to memory constraints), it is much faster than the datastore. Memcache is also ideal for computations that need to occur quickly, and, because memcache allows values to set an expiry time, it is also perfect for data whose usefulness expires after a period of time. Finally, an Associator performs the final event detection and issues notifications.

3.5.2 Advantages of Cloud Computing

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

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

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

The Situational Awareness Framework: a web backend for event detection. The emergence of affordable Platform-as-a-Service (PaaS) makes scaling (relatively) easy and affordable. The distributed nature of PaaS infrastructure allows data replication in geographically distinct locations, providing reliability and resiliency to localized emergencies. Structuring a sensor network as a web application makes it easy to extend the project's reach to any point on the globe.

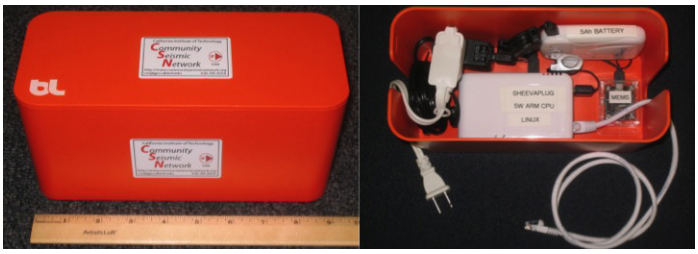
3.6 Desktop Client

Desktop clients with USB accelerometers provide low-to-moderate cost stationary sensors, with several advantages. Inexpensive ($< 100USD$) USB sensors are capable of observing earthquakes that are too small for most people to notice, as shown in Fig. 3.6. Desktop clients are subject to ambient vibrations, but can potentially be affixed to walls or other stable reference points. Similarly, desktop clients often have access to wired internet connections and power, allowing continuous operation and ample bandwidth for data collection.

The CSN desktop client software is designed with these favorable conditions in mind. Due to the availability of bandwidth, the client routinely transmits all raw acceleration measurements alongside its heartbeat messages. Pick messages are generated using a relatively simple pick algorithm which compares the low-frequency energy contained in a short period (several seconds) of acceleration time series data to the low-frequency energy contained in a longer period, and reports a pick if the short period contains significantly more energy. This algorithm, called the *Short-term average over Long-term average* (STA/LTA) is a simple ratio test that has proven to be reliable and effective. An NTP daemon is used to synchronize the desktop client's clock.

Event Detection using Averages: STA/LTA. STA/LTA (Short Term Average over Long Term Average) computes the ratio between the amplitude of a short time window (STA) and the amplitude of a long time window (LTA) and decides to "pick" when the ratio reaches above a threshold. In our analysis, we used a short term window size $ST = 2.5$ s and a long term window size $LT = 10$ s. This simple algorithm can detect sudden changes in transients that may indicate the occurrence of an event in a low-noise environment. In an ideal situation where the sensors have fixed orientation, the signal on each axis can be used to derive the direction of the incoming wave. We do not assume consistent orientation here, but instead simply take the L2 norm of all three axes before computing the STA/LTA.

Desktop clients also have their drawbacks for building a large sensor network for long-term operation. The cost of a USB accelerometer, while low, can be a hurdle to participation for a volunteer, and can limit the number of sensors that can be centrally provided. Volunteers may not immediately install their sensors, or may accidentally unplug them. Similarly, desktops are increasingly replaced by laptops or mobile phones, limiting the number of volunteers with continuously-operating desktops. To address these issues, CSN has distributed stand-alone "boxes" containing a sensor and a small SheevaPlug plug computer. These boxes require little to no configuration or maintenance from volunteers, and can simply be plugged into power and a network connection.



(a) "Orange Box"

(b) "Phidget USB accelerometer"

Figure 3.9: (a) An "Orange Box" sensor station contains a small SheevaPlug computer, a 16-bit Phidget USB sensor (b), and an optional backup power supply. It allows volunteers to deploy a sensor with low effort and maintenance.

3.7 Android apps for community sensing

This section discusses two versions of an Android app that volunteers may install on their Android phones or tablets in order to join the CSN network. After discussing the goals and challenges of the app, I discuss the design choices and architecture of the app, and how it interfaces with the CSN cloud server. The section ends with observations about using the Android platform for Community Sense and Response systems.

CSN-Droid is a mobile sensor client designed to abstract away the data complexities of an individually-owned sensor. CSN-Droid uses acceleration data from volunteers' phones or tablets to rapidly measure the intensity of earthquakes. When a device detects possible seismic activity, a pick message transmits the device's acceleration and location measurements. Location data is periodically transmitted to build an estimate of network coverage.

3.7.1 Goals and Challenges

Engaging a volunteer user base. The app needs to provide value to the end user. While many volunteers may be sufficiently enthusiastic about science, or earthquakes in particular, attracting a large user base requires interesting, relevant content. To this end, the Android client provides a

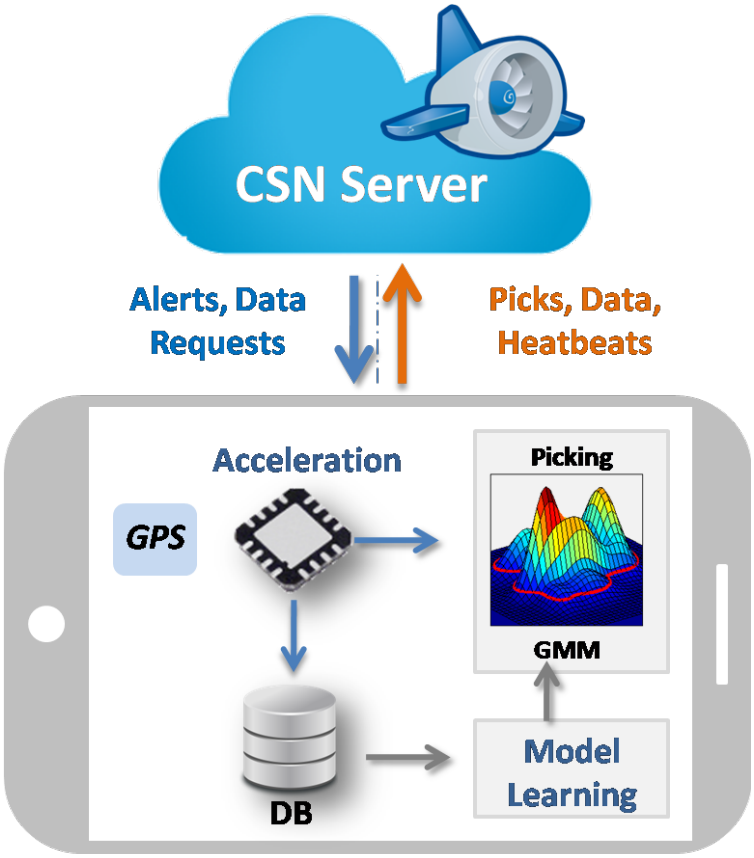
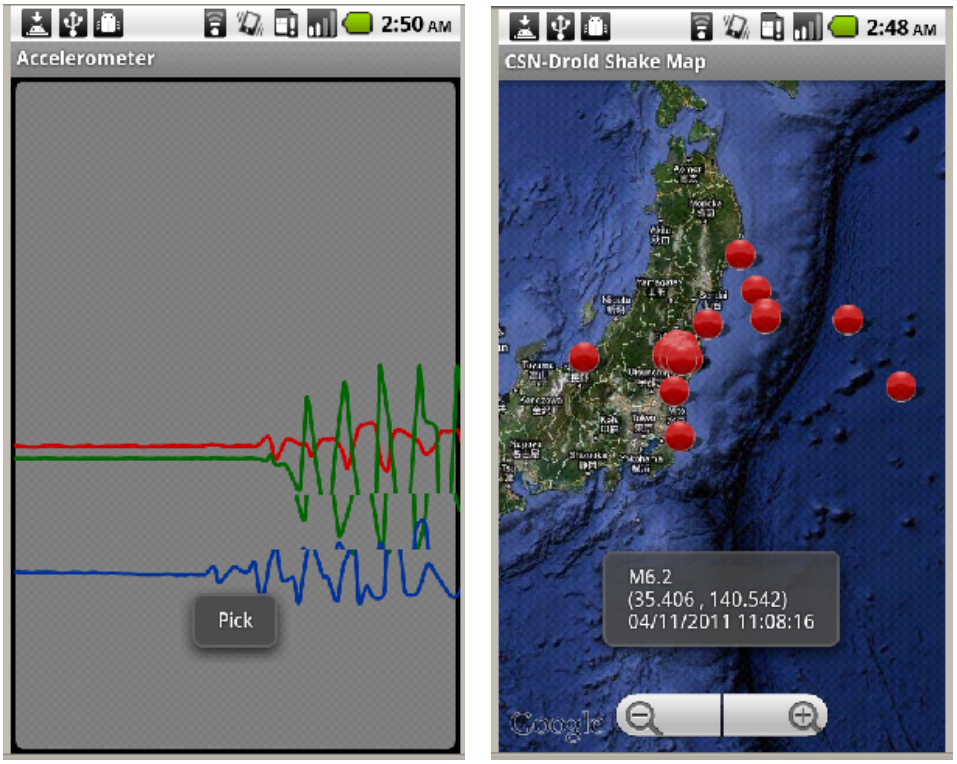


Figure 3.10: CSN-Droid stores and processes sensor data locally on an Android phone or tablet; sends pick messages during potential quakes; receives alerts; and responds to data requests.

simple, clean visualization of earthquakes worldwide, and a list of recent news articles related to earthquakes.

CSN-Droid served as a viable proof-of-concept for crowd-sourcing earthquake data from smartphones. More than 500 people worldwide installed the app. However, users commonly reported that the app was not very interesting or engaging. Similarly, the title CSN-Droid was ambiguous, and was not enticing to users who had searched in the app market for "quake" or "earthquake".

Partly to address these challenges, I developed CrowdShake with the intent of providing a user experience that provides fresh information about earthquakes on a daily basis. To accomplish this, the app displays a set of earthquakes, drawn from a USGS data feed, in map or list format. The app only displays earthquakes that occurred within 24 hours, ensuring that content is fresh. Additionally, the app has a "News" screen that provides a list of daily news items related to large earthquakes. This set of news items is drawn from a custom Google news feed.



(a) CSN-Droid accelerometer view (b) CSN-Droid map view

Figure 3.11: The CSN-Droid user interface provides a simple Map view of worldwide earthquakes, and visualized the device accelerometer time series.

3.7.2 Design choices

Fig. 3.10 shows the internal data flow and the messaging between client and cloud. At the core of the application is a suite of sensors, including the 3-axis accelerometer and GPS. The raw stream of accelerometer data is continuously tested for anomalies, which are reported as pick messages. The raw data is also stored temporarily in a local database. This both allows the server to issue *data requests* for specific intervals of data, and allows updates to the anomaly detection model. The client also listens for push notifications from the server, implemented via Google's Cloud Messaging services.

3.7.3 An interface of Fragments

To better address a variety of device form factors, the CrowdShake interface is composed of a number of Fragments. Fragments, introduced in Android 3.0, provide a means of encapsulating parts of an interface. These parts can then be combined to produce an overall interface in a way that is suitable to the particular device. For example, a low-priority interface component might be visible on a large tablet, but require additional navigation to access on a small phone screen.

3.7.4 Event Detection

The CSN-Droid and CrowdShake apps detect and report significant changes in accelerometer behavior. The apps use a GMM-based anomaly detection scheme, described in detail in Sec. 4.1, which requires that sensor data be represented as (small) vectors. Android provides raw accelerometer data as a stream of timestamped triples of X,Y,Z axis acceleration. In order to use this data for anomaly detection, it undergoes preprocessing and segmentation and then is converted into feature vectors.

Given that Android devices are subject to motion and may be in any orientation, the first preprocessing step is to identify the downward direction and remove the contribution due to gravity. A 10 second low-pass filter is used to determine the direction of gravity; the acceleration time series is rotated so that this direction is the negative Z axis, and then one unit of gravity is subtracted along this direction.

Once the acceleration time series has been oriented with respect to gravity, the time series is cut into short segments that will ultimately be used to compute feature vectors. In much the same way that short segments of audio data are used as phonemes in audio processing, short windows of accelerometer are used to capture sudden changes in motion. A sliding window of 8.5 seconds' worth of acceleration is created, and split into three segments: the oldest 5 seconds of measurements provide a *long-term window*; the next second is discarded; the most recent 2.5 seconds comprise the *short-term window*. This choice is motivated by a desire to detect sudden changes in acceleration characteristics by comparing baseline motion (the long-term window) with the immediate data (the short-term window). One second of dead time between the windows avoids ambiguous segmentation that partially capture a transition in behavior in both time windows.

The resulting acceleration time series segments only represent a few seconds of motion, but may contain up to 1000 accelerometer samples. Motivated by the observation that most seismic motion occurs at low frequency, a discrete Fourier transform is used to identify the energy across a range of frequencies in each time series window. In addition to these Fourier coefficients, the variance and maximum absolute acceleration for each window are computed. The features for each axis of acceleration are then concatenated into a total feature vector. Empirically, overall performance can be improved by "compressing" this total feature vector via principle component analysis (PCA). The final feature vector is produced by retaining a fixed number (e.g., 16) of the top PCA dimensions, and concatenating the magnitude of signal present in the discarded PCA dimensions. This last quantity was added to represent the amount of signal not well modeled by the top PCA dimensions, and has been empirically found to be useful.

Finally, the probability assigned to each feature vector by a Gaussian Mixture Model is computed, and compared to a likelihood threshold to identify sufficiently low-probability feature vectors. These anomalies are identified as picks.

3.7.5 Data Logging

The app has a SQLite database that contains a fixed number of "chunks" of accelerometer data. Chunks are filled with a fixed number of accelerometer samples (e.g., 10000 samples) and then persisted to the database. If the database already contains the maximum number of "chunks", then the oldest chunk is deleted. Chunks are indexed by the timestamp of their first and last sample.

3.7.6 Observations about Android for CSR systems

Fragmentation and Obsolescence.

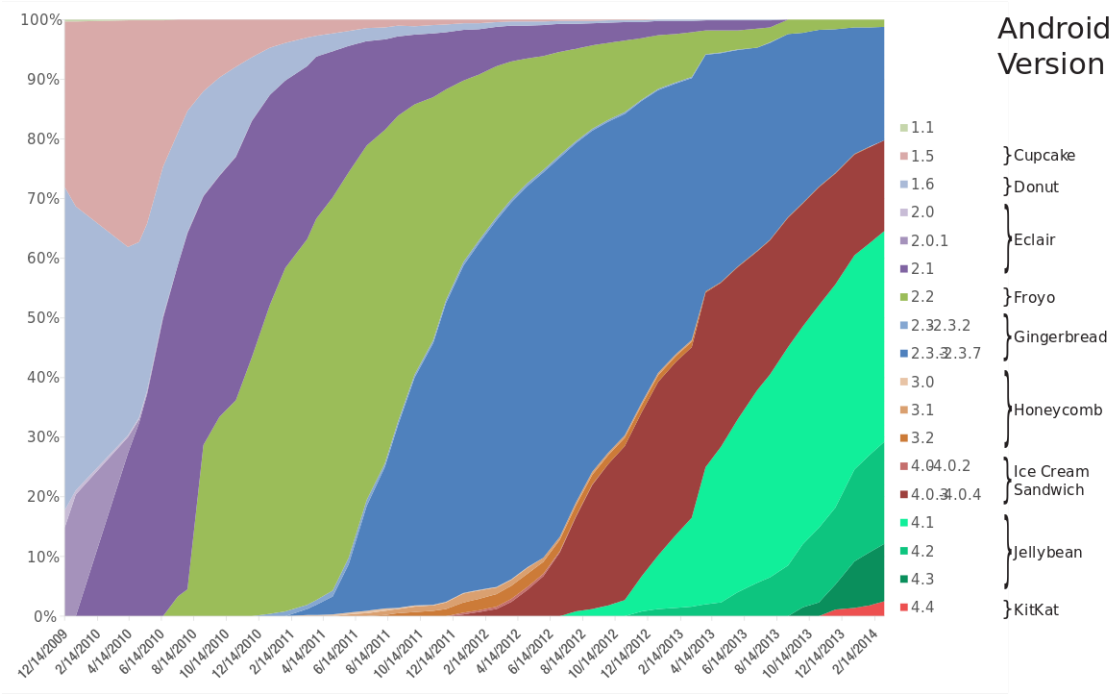


Figure 3.12: The distribution of Android OS versions over time shows substantial software fragmentation. Notably, it is rare for the majority of devices to run any single OS version. Image courtesy of Wikimedia Commons.

Android is an open platform, giving great freedom to device manufacturers to create devices and modify the operating system. While this openness fosters innovation, and has certainly encouraged the market growth of Android, it also presents many challenges to the app developer. Chief among these are fragmentation and obsolescence.

Android is notable for its hardware fragmentation. In contrast to Apple's iPhone, which has only undergone a handful of hardware updates, there are more than 10,000 distinct Android device models, with no single device manufacturer responsible for more than half of those models. For the developer, this means that there is no "canonical" device for testing with, and that instead the

developer must be prepared for a large number of screen resolutions, sensor behaviors, available memory, and network speeds.

Software fragmentation is also a concern. Since the first release of CSN-Droid, the Android platform has gone through five major operating system versions (Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, and KitKat), spanning 11 different API levels. Fig. 3.12 displays the distribution of Android OS versions over time. The source for the Android operating system is licensed under the Apache License, which allows device manufacturers to install modified versions of the OS on their devices. It has been observed that devices sold from Google Play (which ship with the standard OS releases) may differ in behavior from the same devices sold directly by their manufacturer.

The mobile market is fast-paced, and in the quest for ever-greater libraries and services, many things quickly become obsolete. The first CSN app, CSN-Droid, made use of the Google Maps API v1 map library and the Google Cloud To Device Message (C2DM) library, both of which were deprecated within two years of the app's release. In this case, deprecation means that API keys are no longer available for those services, preventing subsequent release of apps using them. Both libraries were replaced by superior services (Maps API v2, and Google Cloud Messaging), but such upgrades mean mandatory additional app development just to maintain existing apps.

Possible Improvements. In principle, Android devices maintain accurate system clocks by receiving GPS timestamps. Unfortunately, some Android devices reportedly[2] do not compensate for differences between GPS time and UTC time, resulting in clock errors of up to 15 seconds.

Finally, compelling apps are interactive and personal. It should be easier for users to see their own contribution to CSN. Social media is effective for publicity. The app should incorporate social media channels, possibly by allowing users to share data products (maps, videos, etc.) from the app. This could provide greater exposure for the project and the app.

[2] https://code.google.com/p/android/issues/detail?id=5485

Chapter 4

Decentralized Anomaly Detection

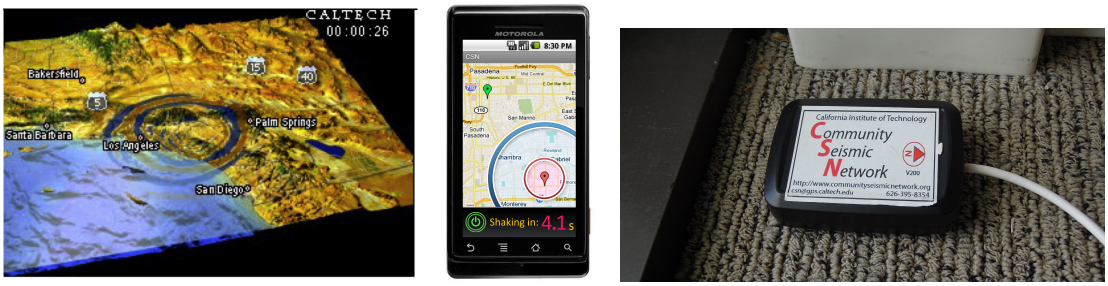
This chapter outlines a methodology developed to rapidly detect quakes from thousands of community sensors. As we will see, the computational power of community devices can be harnessed to overcome the cacophony of noise in community-operated hardware, and that on-device learning yields a decentralized architecture that is scalable and heterogeneous while still providing rigorous performance guarantees. Much of this work first appeared in [37].

Event detection is a primary goal of CSR systems. Community sensing has recently been used to detect rare and unpredictable events, such as traffic accidents [63, 81, 67] and earthquakes [24]. Rare events are often difficult or impossible to model and characterize a priori, yet we wish to maximize detection performance. Further, heterogeneous, community-operated sensors may differ widely in quality and communication constraints, due to variability in hardware and software platforms, as well as differing in environmental conditions.

This chapter presents a principled approach towards detecting rare events from community-based sensors. Due to the unavailability of data characterizing the rare events, our approach is based on anomaly detection; sensors learn models 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 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.

4.1 The Decentralized Detection Problem

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



(a) *Earthquake (Chino Hills, Magn. 5.4, July 2008)*

(b) *Early warning*

(c) *Phidget sensor with housing*

Figure 4.1: (a) Seismic waves (P- and S-waves) during an earthquake. (b) Anticipated user interface for early warning using our Google Android application. (c) 16-bit USB MEMS accelerometers with housing that we used in our initial deployment.

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 3.7.4 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 (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* [99]: we define two thresholds τ, τ', and let $M_{s,t} = 1$ if and only if

$$\frac{\mathbb{P}[X_{s,t} \mid E_t = 1]}{\mathbb{P}[X_{s,t} \mid E_t = 0]} \geq \tau. \tag{4.1}$$

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

$$\frac{\mathrm{Bin}(S_t; p_1; N)}{\mathrm{Bin}(S_t; p_0; N)} \geq \tau', \tag{4.2}$$

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 (4.1) and (4.2) with proper choice of the thresholds τ and τ' [99].

There has also been work in *quickest detection* or *change detection* (cf., [91] 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:

1. Sensors are highly heterogeneous (i.e., the distributions $\mathbb{P}[X_{s,t} \mid E_t]$ are different for each sensor s)
2. Since events are rare, we do not have sufficient data to obtain good models for $\mathbb{P}[X_{s,t} \mid E_t = 1]$
3. 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 [87]. 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 (e.g., of the form (4.1)) that minimize the number of messages sent.

4.2 Online Decentralized Anomaly Detection

This section describes an 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 ($M_{s,t} = 1$) if and only if, for its current readings $X_{s,t} = x$ it holds that

$$L_0(x) < \tau_s \tag{4.3}$$

for some sensor specific threshold τ_s. Note that using this decision rule, for a pick it holds that $\mathbb{P}[M_{s,t} = 1 \mid E_t = e] = \mathbb{P}[L_0(X_{s,t}) < \tau_s \mid E_t = e] = p_e$. This anomaly detection approach hinges on

the following fundamental anti-monotonicity assumption: that

$$L_0(x) < L_0(x') \Leftrightarrow \frac{\mathbb{P}\left[x \,|\, E_t = 1\right]}{\mathbb{P}\left[x \,|\, E_t = 0\right]} > \frac{\mathbb{P}\left[x' \,|\, E_t = 1\right]}{\mathbb{P}\left[x' \,|\, E_t = 0\right]}, \tag{4.4}$$

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 (4.3) and (4.1) are equivalent, for an appropriate choice of thresholds.

Proposition 4.2.1. *Suppose Condition* (4.4) *holds. Then for any threshold* τ *for rule* (4.1)*, there exists a threshold* τ_s *such that rule* (4.3) *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:

1. 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)?
2. How should the sensors choose the thresholds τ_s?
3. Which true positive and false positive rates p_1, p_0 and threshold τ', cf., (4.2), 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 [93] and particle filtering [40]. 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 [52].

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 re-estimate τ_s at every time step. Unfortunately, percentiles do not have sufficient statistics as the mean and other moments do. Moreover, Munro and Paterson [84] 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 [55]. We summarize our analysis in the following proposition:

Proposition 4.2.2. *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 4.2.2 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.

Uniform convergence rates for density estimation have been established as well [51], 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

Algorithm 1: 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 $N(p_0^i) = \min\{N' : \sum_{S>N'} \mathrm{Bin}(S; p_0^i; N) \leq \bar{P}\}$;

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

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

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

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 (4.2), 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 (4.2) is equivalent to setting $\widehat{E}_t = 1$ if and only if $S_t \geq N(p_0)$, 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:

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

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 (4.4) holds, the following result, which is a consequence of [99], holds:

Theorem 4.2.3. *Suppose condition* (4.4) *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 5.8, 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 $\widehat{p}_1$?

One way to quantify detection performance is with a Receiver Operator Characteristic (ROC) curve. The curve plots true positive rate against false positive rate for each possible decision threshold. ROC curves allow us to estimate the obtainable TPR of a sensor, given a constraint on its FPR, such as a limit on the average number of pick messages per day.

Suppose we have access to an estimate of the sensors' 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 4.3(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 1 formalizes this procedure. We summarize our analysis in the following theorem:

Theorem 4.2.4. *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 1 chooses an operating point p_0 to maximize a lower bound on the true detection performance, i.e., $\widehat{P}_D(p_0) \leq P_D(p_0)$.*

4.3 Experimental Evaluation

Could a network of cheap community sensors detect the next large earthquake? This section explores whether consumer sensors are adequate for sensing important earthquakes, and analyzes the effectiveness of distributed anomaly detection for quake detection.

- We measure the sensitivity of commercial hardware by reproducing historic moderately large earthquakes gathered by the Southern California Seismic Network (SCSN) via a shake table.

We find that mobile phone accelerometers and consumer USB accelerometers are capable of sensing moderately large earthquakes.

- In the absence of recordings of large earthquakes by mobile phones, we simulate observations of quakes by combining accelerometer time series collected from volunteers' smartphones with time series data of historic Los Angeles area earthquakes. This data set allows us to investigate the average performance of single-sensor detection (pick algorithms).

- Earthquakes are spatial-temporal events, and so it is natural to ask how the distribution of population (and hence, sensors) impacts the detection of different earthquakes as they traverse a city. Combining a simulator for earthquake propagation with sensor-level detection simulators allows us to investigate how quake of a given magnitude and origin would be observed by a particular configuration of community sensors, and assess the time needed to reliably detect the event.

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, 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 40 km 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.

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. 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.

4.3.1 Decentralized detection: device-level performance

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 both for $\mathbb{P}[X_{s,t} \mid E_t = 0]$, as well as $\mathbb{P}[X_{s,t} \mid E_t = 1]$. For training data, we use 80 historic earthquake examples of magnitude M4.5-5, 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 2.5s length

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 $\mathbb{P}[X_{s,t} \mid E_t = 0]$. We applied each of these four algorithms to test data created by superimposing historic earthquake recordings of magnitude M5-5.5 on phone and Phidget data that was not used for training. The resulting estimated sensor ROC curves are shown in Fig. 4.3(a) and Fig. 4.3(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.

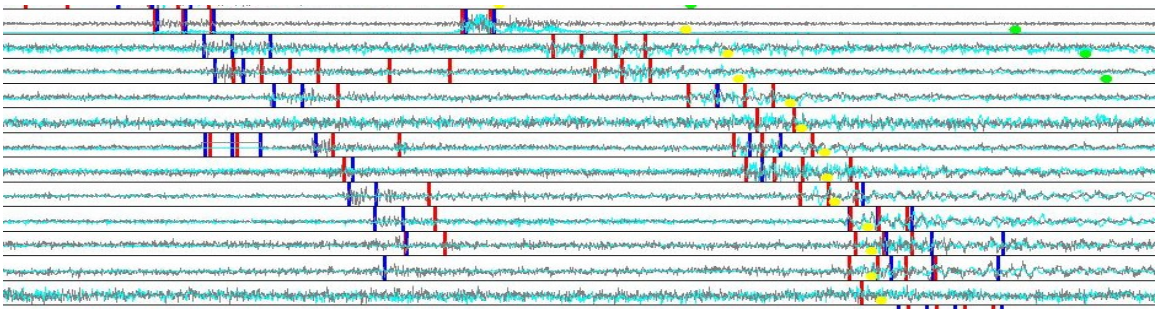
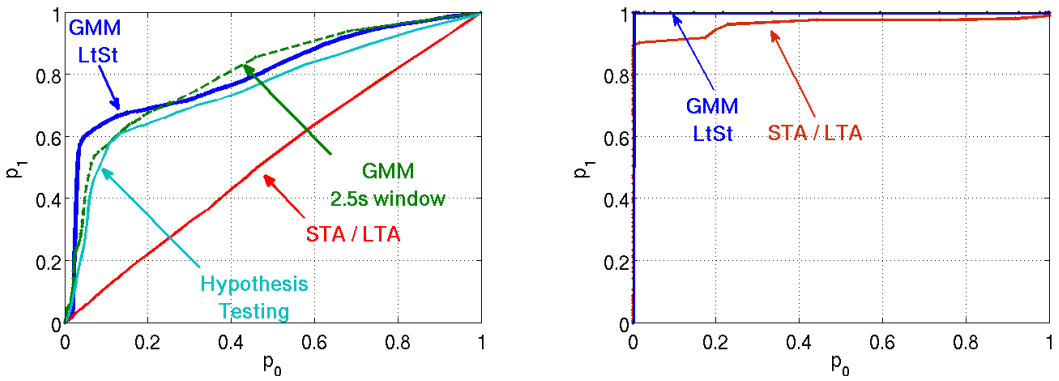


Figure 4.2: CSN sensors produced picks (blue and red bars) for both P-wave and S-wave of the Anza M3.6 earthquake. Time series plots are arranged by distance to quake epicenter.

4.3.2 From sensor performance to network performance

Based on the estimated sensor-level ROC curves, we can now estimate the network-wide 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 a message can be sent from each sensor, 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



(a) *Android detection performance* (b) *Phidget detection performance*

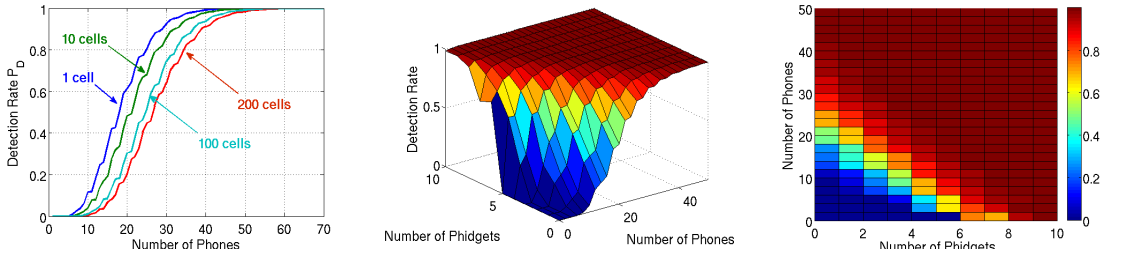
Figure 4.3: In all plots, the system-level false positive rate is constrained to 1 per year and the achievable detection performance is shown. (a,b) Sensor level ROC curves on magnitude M5-5.5 events, for Android (a) and Phidget (b) sensors.

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 taking unnecessary response measures.

We consider sensors located in 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 ($\approx$ 5-10 km/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 4.4(a). 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 4.4(b) and 4.4(c) 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 4.4(b) and 4.4(c) 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



(a) *Detection rates* (b) *1 pick per minute, LtSt features* (c) *1 pick per minute, alternate view*

Figure 4.4: (a) Detection rate as a function of the number of sensors in a 20 km × 20 km cell. We show the achievable performance guaranteeing one false positive per year, while varying the number of cells covered. (b,c) Detection performance for one cell, depending on the number of phones and Phidgets.

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 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.

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 ($32.25°N, -115.28°E$). This M7.2 quake in Baja, California was recorded by SCSN, although the nearest station was more than 60 km from the event epicenter. Using 8 recordings of this event, at distances of 63 km to 162 km, 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 4.5 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 from the epicenter would have been quite likely to detect the Baja, California M7.2 event.

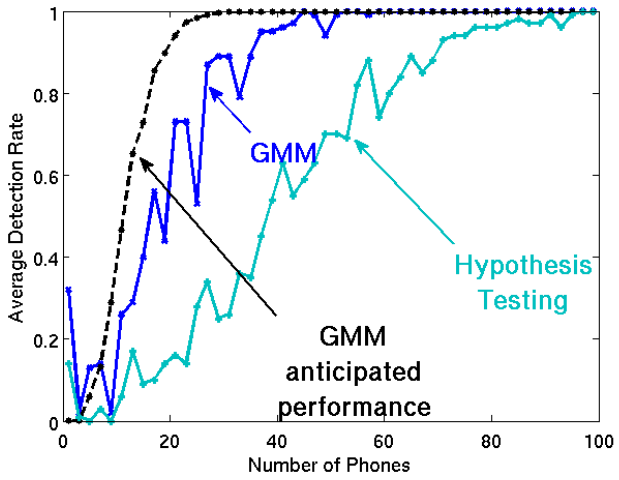


Figure 4.5: Actual detection performance for the Baja event (100 iterations averaged). Note that our approach outperforms classical hypothesis testing, and closely matches the predicted performance.

Next, we evaluate the ability of community sensors to detect future quakes for which no training data is available. While earthquakes are rare, data gathered from community sensors can be plentiful. To characterize "normal" (background) data, seven volunteers carried Android phones throughout their daily routines to gather over 7GB of phone accelerometer data, and 20 USB accelerometers recorded 55GB of acceleration. From this data, we estimated models for each sensor type's normal operating behavior. We evaluated anomaly detection performance on 32 historic records of moderately large (M5-5.5) events (as recorded by the Southern California Seismic Network). Fig. 4.3 summarizes the ability of individual sensors to transmit "event" or "no event" to the cloud server, in the form of Receiver Operating Characteristic curves, and shows anomaly detection outperforming several standard baselines: the vertical axis is the attainable detection (pick) rate of a single sensor, against the horizontal axis of allowable false detection (pick) rate. Combining the accuracy results for USB and Android sensors, Fig. 6.5 shows the tradeoff of detecting with a mix of sensor types, while constraining to one false alarm per year. Our results indicate that approximately 50 phones or 10 Phidgets should be enough to detect a nearby magnitude 5 or larger event with close to 100% success.

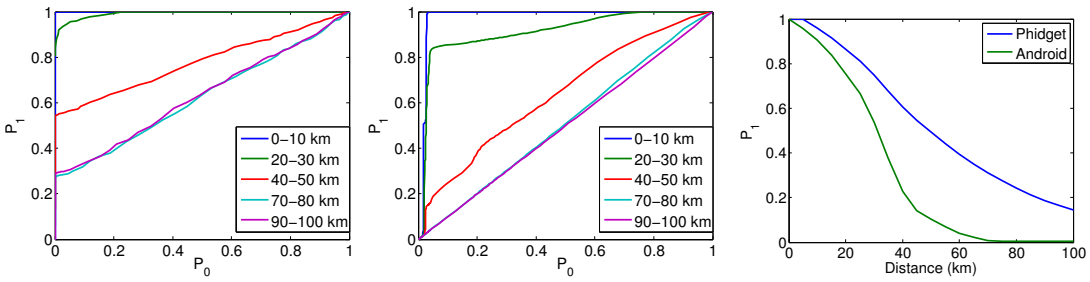
Chapter 5

Rapid detection of structured spatial patterns

Quake detection in community networks requires finding a complex spatio-temporal pattern in a large set of noisy sensor measurements. The start of a quake may only affect a small fraction of the network, so the event can easily be concealed in both single-sensor measurements and network-wide statistics. Data from recent high-density seismic studies, Fig. 1.3, show that localized variations in ground structure significantly impact the magnitude of shaking at locations only a few kilometers apart. Consequently, effective quake detection requires algorithms that can learn subtle dependencies among sensor data, and detect changes within groups of dependent sensors. In this sense, quake detection is prototypical of many challenging real-time detection problems, including detecting epidemic outbreaks [95], intrusions in networks [111], and sudden changes in traffic patterns [48].

The previous chapters have introduced a decentralized system architecture using cloud computing with desktop and mobile clients, and explained how a decentralized anomaly detection approach can allow practical scalability by dividing the detection task into per-client anomaly detection tasks and a network-wide event detection task. This chapter, based on [36] and [88], explores a different aspect of event detection with large numbers of community sensors: how can knowledge of the spatial dependencies among sensors be used – or learned – to improve detection?

To help answer this question, two approaches for incorporating spatial structure are presented. The first approach, described in Sec. 5.3, uses the concept of a *geocell* to subdivide the network into disjoint regions. These regions are tested individually for the occurrence of an event. While conceptually simple, testing by geocell has several practical advantages and allows easy evaluation of event and sensor density scenarios. The second approach, discussed in Sec. 5.4, considers learning network-wide dependencies from historical data or, alternatively, using qualitative knowledge of sensor similarity.



(a) Phidget (b) Android (c) Detection vs. Distance

Figure 5.1: ROC curves for (a) Phidget and (b) Android. (c) Detection performance vs. distance from epicenter under the guarantee of at most 1 false message per hour for the Phidget and 1 false message every 5 minutes for the Androids.

5.1 Preliminaries

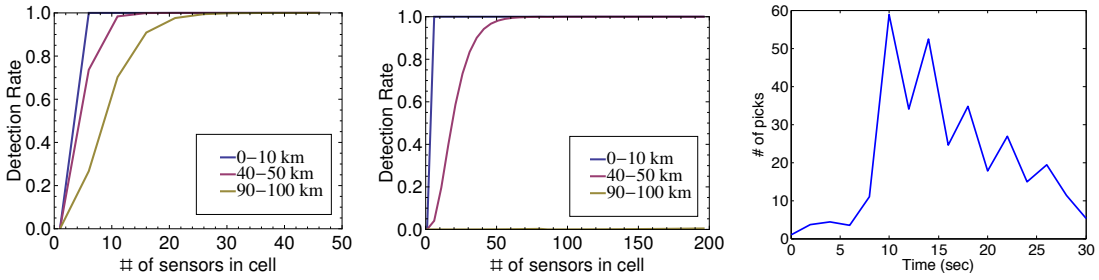
As preliminaries, this chapter adopts the pick model from Sec. 1.5. To help study spatiotemporal events, this chapter also introduces a simulator for the CSN network that generates pick messages as a function of event parameters and (virtual) sensor location. An estimate of sensor performance is obtained by using historic quake measurements, partitioned according to distance from sensor to epicenter.

5.1.1 Lower Bounds for Sensor Performance

As a first step towards simulating the behavior of the CSN network, Receiver Operating Characteristic (ROC) curves are used to gain insight into the detection performance of each client type as a function of event magnitude and distance from client to event epicenter.

We obtain simulated acceleration time series recordings for both Phidget and Android clients by combining historical earthquake recordings from the USGS Southern California Seismic Network (SCSN) with noise recordings from volunteers' Phidget and Android sensors. We collected a set of 54 SCSN records of magnitude M5-5.5 earthquakes from seismic stations between 0-100 km. The SCSN recordings are down-sampled to 50 samples per second to be comparable with low-end consumer accelerometers, and are then overlaid with Phidget or Android recordings from the volunteer data set in order to obtain a realistic noise profile.

Using these simulated acceleration time series, we can compute ROC curves for each sensor type under a variety of seismic scenarios. From a data set of magnitude $M = 5 - 5.5$ earthquakes, we extract 5 sets of records, containing data from stations at varying distances away from the epicenter. The data sets correspond to distance ranges d in kilometers, $d = \{0-10, 20-30, 40-50, 70-80, 90-100\}$. Fig. 5.1 illustrates the performance of the the STA/LTA algorithm (evaluated on synthetic records made with volunteers' Phidget data), and the anomaly detection algorithm (evaluated on



(a) Phidget (b) Android (c) Expected Traffic

Figure 5.2: (a) Single cell of varying number of Phidgets observing 3 levels of seismic events of M5.5 and lower. (b) Single cell of varying number of Androids observing 3 levels of seismic events of M5.5 and lower. (c) Number of pick messages received by the system as a function of time since the event started.

synthetic records made with volunteers' Android data). These ROC curves demonstrate that the Phidgets - higher resolution sensors that are typically not subjected to user motion - obtain superior performance to the Android sensors at all distance ranges. The curves also reflect the $1/r^2$ decay rate of shaking intensity, where r is distance from the quake epicenter.

5.1.2 Analyzing groups of sensors by Geocell

One practical way of spatially aggregating data is to partition the data according to rectilinear regions of latitude and longitude. *Geocells* [88] provide an efficient way to query data according to a multi-resolution grid of latitude and longitude. With a slight abuse of notation, "geocell" shall refer to one member of such a rectilinear partitioning.

Consider a number of sensors occupying a relatively small geocell (e.g., several street blocks). Inside this geocell, each sensor experiences similar seismic shaking during an event, and independent noise (such as motions caused by a cell phone's user) in the absence of an event. We can roughly say that all sensors within a geocell have the same signal to noise ratio (SNR) and that their picks can be well approximated as independent, identically distributed binary random variables when conditioned on whether an event has occurred or not. Thus, from the ROC of a single sensor, we can analyze the collective behavior of a group of sensors. By fixing the decision rule for each sensor, and a decision rule for geocell-wide event detection, we can evaluate the event detection performance as a function of the number of sensors in the geocell.

The sensor decision rules can be specified by constraining the maximum allowable rate of false positive picks. Here, we constrain the Phidget USB sensors to produce at most 1 false pick per hour, and constrain the Android sensors to at most 1 false pick per 5 minute interval. The cell-wide false positive rate is constrained to no more than 1 per year, on average. Fig. 5.2(a) and Fig. 5.2(b) show cell detection performance as a function of sensor density, generated from synthetic M5-5.5 records. These results indicate that a cell containing 30 Phidgets or 100 Androids could reliably detect a

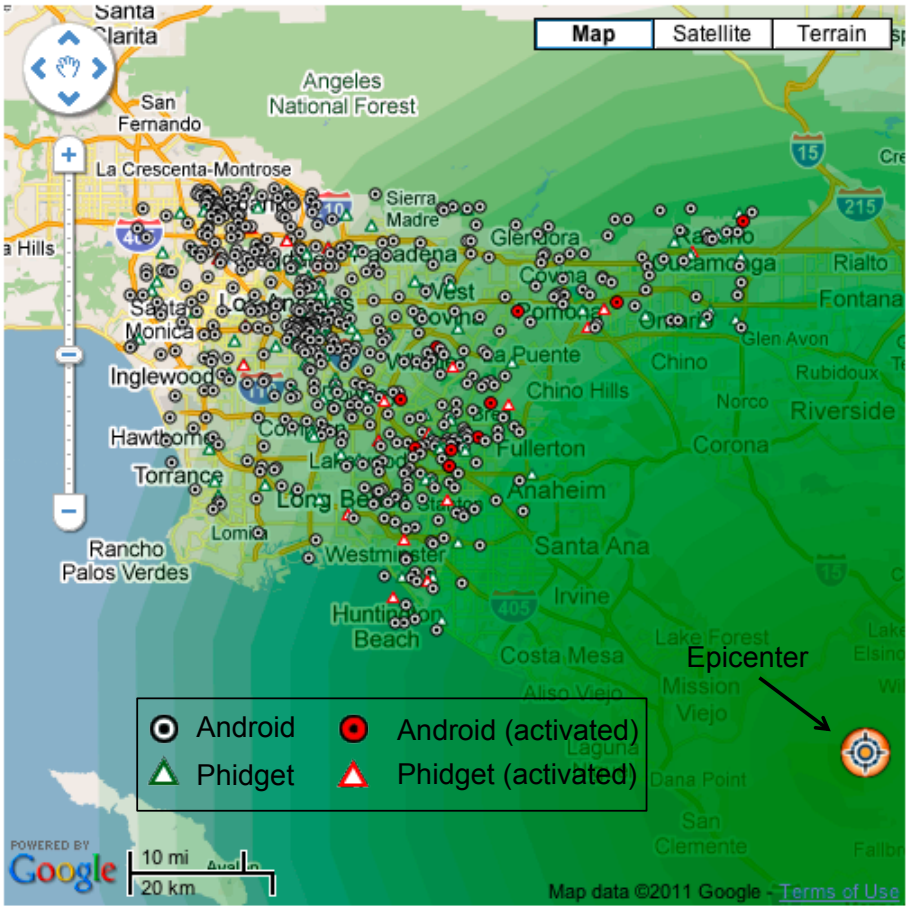


Figure 5.3: Snapshot of simulated detection of a M5.5 event 80 km outside the great Los Angeles area. There are 100 desktop clients and 1000 Android clients distributed according to population density. The snapshot is taken 20 seconds after the event occurred.

moderately large earthquake at a distance of 50km from the epicenter, and that the higher-quality Phidget sensors are capable of detecting the signal from up to 100km.

5.1.3 Simulation Platform

Earthquakes are complex phenomena whose effects differ according to a variety of geological factors. Seismologists have developed detailed simulations that capture these subtle relationships, but as this work is interested in the dominant behavior of a network comprised of a large numbers of relatively noisy sensors, we use a relatively simple model. The simulator, developed by Liu [88], generates time series of pick messages as a function of each client's sensor type and location, as well as event origin and magnitude using a seismic model based on a point source (i.e., the epicenter is a single point) and isotropic wave propagation (i.e., the wave travels in all directions with equal speed). As discussed in Sec. 4.2, each client's false positive rate may be chosen to maximize detection performance while

satisfying the per-sensor constraints on average message rates.

Given the location of each sensor in the network and the origin of a seismic event, the simulator computes the probability that each sensor picks during each time instance, and generates picks with these probabilities to produce a time series of pick messages. Fig. 5.3 shows a snapshot of simulated detection of a M5.5 event. The snapshot is taken 20 seconds after the event occurs. For this simulation, the Phidget FPR is set at 1 pick per hour and the Android FPR is set to 1 pick every 5 minutes. The pick messages are timestamped after factoring in network delays.

5.2 Naive Event Association

As a baseline, consider the *naive* aggregation policy described in Sec. 4.1. At each time step, the system decides whether an event has occurred based on the pick messages it has received so far. The naive decision rule is to perform hypothesis testing on the aggregated pick counts in the past few seconds, that is, to compute the ratio of likelihood for the two hypothesis: 1) that there is an event ($p = p_1$), and 2) that there is no event ($p = p_0$). In other words, the naive decision rule performs the test

$$\frac{Binomial(k; n, p_1)}{Binomial(k; n, p_0)} \geq r$$

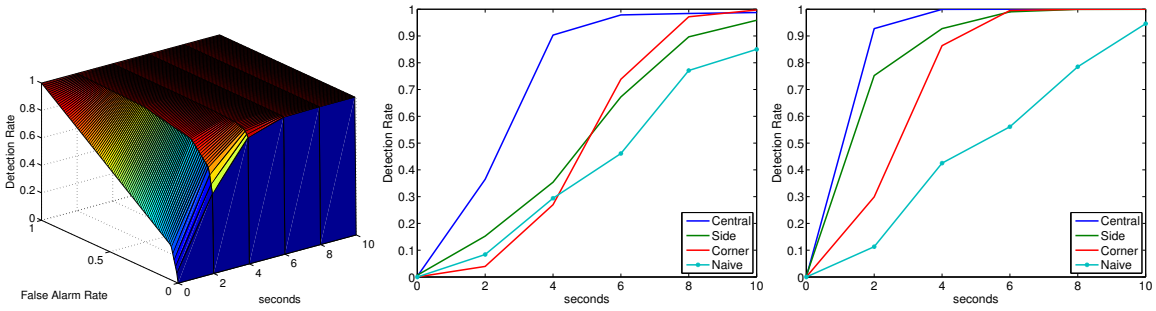
where n is the total number of sensors in the system, k is the number of picks observed in the past few seconds and r is the decision threshold chosen to satisfy a constraint on the system false positive rate. If the inequality holds, the system declares detection.

This policy is referred to as performing *naive* aggregation because it disregards the varying strengths of geospatial correlation between sensors as well as the temporal pattern in which seismic waves cross the network. Depending on the distance and direction of the wave relative to the region of sensors, different number of sensors are affected at a given time. Therefore it may not be reasonable to consider measurements from all sensors equally at all times.

We collected 1000 sets of measurements from 2000 Androids and 20 Phidgets separately during a simulated M5.5 event within 60 km of downtown Los Angeles. During a period of $T = 0 - 10$ seconds after the event occurs, we perform the naive association algorithm on each 2-second interval and compute the system level detection rate while maintaining the guarantee of at most 1 false alarm per year. The results are shown in Fig. 5.4 as the lower bounds for the two types of sensors.

5.3 Spatial Aggregation by Geocell

Even crude knowledge of the spatio-temporal structure of an event should allow for improved detection performance, as it makes possible tests that aggregate data from groups of clients that tend

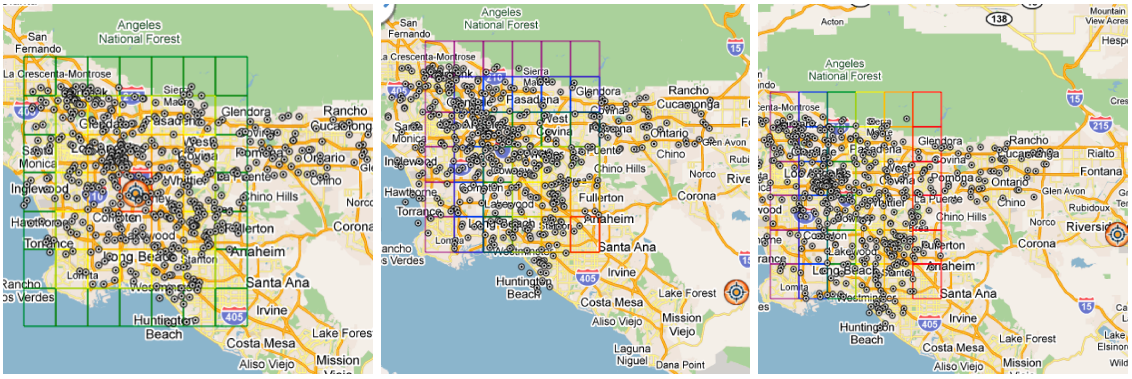


(a) ROC transition in the corner scenario with 2000 Androids.

(b) 2000 Androids

(c) 20 Phidgets

Figure 5.4: Detection of a M5.5 event with (b) 2000 Androids and (c) 20 Phidgets in the three scenarios described in Figure 5.5. This result guarantee at most 1 false alarm per year at the system-wide level. Results computed using the geocell-based association algorithm are compared to those using the naive algorithm.



(a) Central Model

(b) Corner Model

(c) Side Model

Figure 5.5: Regions of different sizes and shapes are activated in different sequence for each of the three scenarios. The rainbow-colored rings indicate the order of activation. Red: first. Purple: last

to co-activate. For the CSN network, events can be very coarsely modeled as originating at a point source and traveling outward isotropically at a fixed speed. With this assumption in mind, we can break down the detection problem into a few case scenarios in terms of how the incoming waves come to contact with the sensors. By exploiting sensor co-activation patterns in these scenarios, one can design a more logical on-line event association algorithm.

Figure 5.5 shows three such possible cases after pre-gridded the area into geocells — (a) the epicenter is inside the cluster, (b) the epicenter is diagonally away from the cluster, and (c) the epicenter is on the side and away from the cluster. In each of these cases, regions of different sizes and shapes will be activated in different sequences during an event. While it is computationally nontrivial to partition a 2-dimensional space into arbitrary regions, the geocell library provides the tools to compute these regions efficiently. We can perform hypothesis testing in parallel for each possible regions to improve the system-wide detection performance.

We computed the system-wide detection performance for each of the scenarios illustrated in Figure 5.5 with either 20 Phidgets or 2000 Androids distributed according to the population density in the area. The regions in terms of activation sequence are identified a priori using the geocell library. A region consists of multiple nearby geocells. Each geocell is $\approx 10 \times 10$ km in size, which is approximately how far the shock wave travels in 2 seconds. Two seconds is also roughly the short-term integration window used in both STA/LTA and anomaly detection algorithm. We can thus safely assume that all sensors in the same region have the same SNR and model them as independently identically distributed random variables, following the analysis in Section 5.1.2. In each time step of 2 seconds, we perform hypothesis testing on each of the regions and compute the system-wide ROC curves.

Figure 5.4(a) shows an example of how ROC curves of 2000 Androids evolve in time for the corner case illustrated in Figure 5.5(b). We slice the surface of this figure at the false alarm rate of 1 per year and retrieve the detection rate as a function of time in Figure 5.4(b) and 5.4(c). The results are compared to the baseline results computed using the naive total association algorithm discussed in Sec. 5.2. These results clearly highlight the benefit of intelligent event association by locality. They also touch on the tradeoffs between delayed decision making and gain in detection confidence. In the case with 2000 Androids (Fig. 5.4(b)), we can fire off alarm at T=2 second that allows us to give 10s of seconds of early warning to surrounding cities such as Santa Barbara or San Diego but with only 20% confidence. Or we can wait till T=10 second or after to fire the alarm with $\approx 100\%$ confidence but give slower warnings.

5.4 Beyond Geocells: learning spatial dependencies

The previous results demonstrate the utility of aggregating data via regions based on event spatio-temporal dynamics, but made coarse assumptions in order to implement the idea. This section formalizes the idea that small groups of sensor clients may observe an event, and that recognizing these co-activations may provide improved detection. Further, it should be possible to *learn* an effective aggregation scheme that makes use of the large number of noisy devices that characterize CSR systems.

Standard approaches in decentralized detection [100] assume that the sensors provide i.i.d. measurements conditioned on the occurrence or non-occurrence of an event. In this case, the fusion center would declare a detection if a sufficiently large number of sensors report picks, corresponding to the *naive* aggregation approach . However, in many practical applications, the particular spatial configuration of the sensors matters, and the i.i.d. assumption is violated. Here, the natural question arises of how (qualitative) knowledge about the nature of the event can be exploited in order to improve detection performance. In this section, we propose to use *sparsification* to optimize detection.

In particular, we linearly represent the network-wide noisy, binary activation patterns in a suitable basis, which is carefully chosen so that "typical" activations (associated with the events of interest) are sparsely represented in the basis. This effectively concentrates the signal energy along a small number of basis coordinates. Natural questions, addressed in this work, are thus: When can we expect sparse representations to aid detection? And, which bases are appropriate for this purpose?

As a starting point, consider a wavelet basis that emerges naturally when sensors are clustered hierarchically [79, 50, 96]. [96] recently proposed a generative model for network data that produce strong localized dependencies and weaker long-range dependencies. They show that in the limit as the number of sensors $p \to \infty$, signals drawn from the model are concentrated in a small number of coordinates of the wavelet basis. As a result, the wavelet basis can detect sparse patterns in strong Gaussian noise that cannot be detected by single-sensor readings or the network-wide average.

However, these existing results are *centralized* in nature: they require aggregation of measurements from all sensors in the network. As our first major contribution, we analyze the wavelet basis and prove theoretically that similar improvements in detection performance can be achieved using *only the binary pick messages of the decentralized setting.*

The first major contribution of this chapter uses a wavelet basis that emerges naturally when sensors are clustered hierarchically to provide improved event detection. We prove theoretically that when the wavelet basis sparsifies the received picks, decentralized detection becomes possible in a noise regime that cannot be handled by a simple network-wide average. We derive strong bounds on the detection rate when events are drawn from a recently proposed *latent tree model* that produce strong localized dependencies and weaker long-range dependencies.

One of the strengths of the wavelet basis is that it can be constructed using as little information as a matrix of pairwise similarity between sensors, e.g., a covariance matrix or a qualitative measure of similarity. However, additional information such as event simulations or measurements of events in the network are often available. Incorporating this information should improve detection. As our second major contribution, we show how modern results from dictionary learning can be used to directly learn sparsifying bases from simulated or measured training data.

As third main contribution, we perform extensive empirical studies of detection using measurements of 1795 earthquakes following the Japanese Tohuku M9.0 quake, quake measurements from the Signal Hill dense seismic study, from the Community Seismic Network as well as simulated virus outbreaks in the Gnutella P2P network.

5.4.1 The weak spatial detection problem

We are interested in the problem of detecting whether or not some phenomenon (say an earthquake with magnitude above some threshold, or an epidemic) is present at any locations monitored by a massive network of noisy sensors. We model the presence of the phenomenon at locations $1, \ldots, p$

as a binary vector $\mathbf{x} = [x_1, \ldots, x_p] \in \{0,1\}^p$ that is observed by noisy sensors. The Gaussian noise model is a natural choice for sensor observations, where sensor i observes

$$y_i = x_i + \epsilon_i,$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. In our seismic detection application, the variables y_i may refer to accelerometer readings of a sensor deployed at location i. This continuous noise model captures how a subset of sensors in areas experiencing shaking observe a shift in the mean of their accelerometer measurements, while the rest of the network observes i.i.d. noise.

The decentralized setting. In many domains, collecting the raw sensor measurements of all sensors would require prohibitive bandwidth to transmit (e.g., the accelerometers in one million smartphones produce ≈ 30 Terabytes of data each day). A natural way to circumvent this bottleneck is to use decentralized detection [100] where sensors individually test their measurements and report the occurrence of a possible event. As an example, the CSN system employs a *hierarchical anomaly detection* approach [38] that allows each sensor to transmit only the results of a local anomaly detection computation (known as a *picking algorithm*) to the fusion center. We can model the resulting picks using a *binary symmetric channel* noise model, where

$$y_i = \begin{cases} x_i & \text{with prob. } 1-\pi \\ 1-x_i & \text{with prob. } \pi, \end{cases}$$

for some error rate $0 < \pi \leq \frac{1}{2}$. The goal of the detection problem is to distinguish the null hypothesis $\mathcal{H}_0$, $x_i = 0$ for all i (i.e., no earthquake present) from the alternate hypothesis $\mathcal{H}_1$, where $x_i = 1$ for one or more i (i.e., the earth is shaking at least at one location i).

Decentralized linear detection. While (decentralized) hypothesis testing in general has been studied extensively, here we focus on the particularly challenging, and not well understood, setting where the patterns $\mathbf{x}$ are *sparse* and have strong noise. This is exactly the case for our motivating example of community seismic networks, where we wish to detect the event as early as possible (i.e., few sensors have been reached yet), and each sensor is very noisy. Formally, we quantify the sparsity of a vector $\mathbf{x}$ as the number of non-zero elements $x_i \neq 0$, denoted by the ℓ_0-norm $||\mathbf{x}||_0$. Generally, we will be interested in quantifying the detection performance as the network grows. We say $\mathbf{x}$ is *sparse* if $||\mathbf{x}||_0$ grows as $p^{1-\alpha}$ for some $1/2 < \alpha < 1$, where a larger α means a sparser signal. Thus, as the number p of sensors grows, the ratio of sensors reached by the event $||\mathbf{x}||_0/p = p^{-\alpha}$ vanishes as $p \to \infty$.

We focus on hypothesis tests of *linear functions of the observations*, i.e., for some matrix $\mathbf{B}$ with

columns $\mathbf{b}_1, \ldots, \mathbf{b}_n$, we consider hypothesis tests of the form

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

for some threshold τ. Proper choice of the basis $\mathbf{B}$ can lead to dramatically improved detection performance, i.e., with the same false positive rate much sparser signals (or much higher noise) can be tolerated.

5.4.2 Detecting sparsifiable events

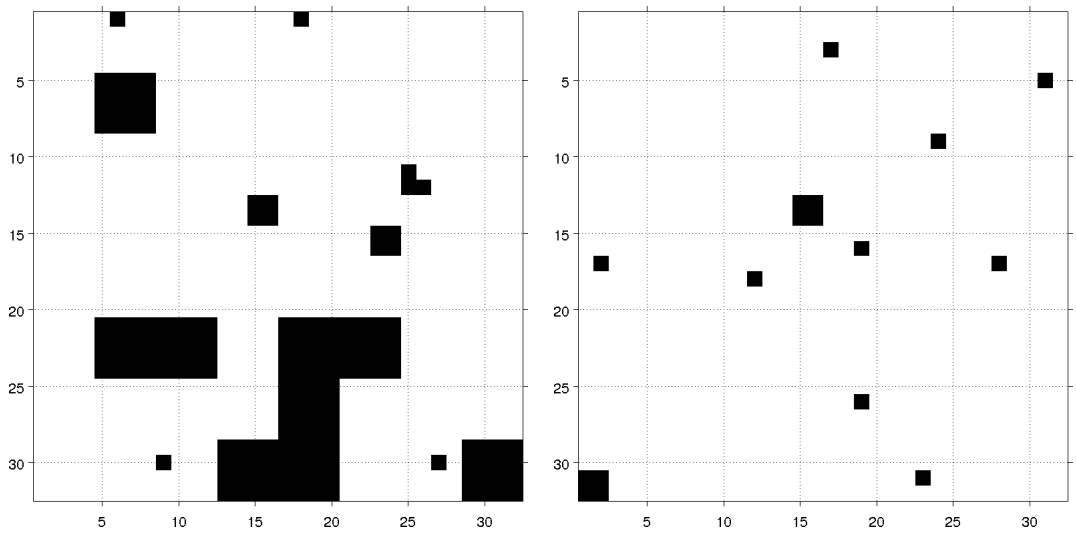
Detecting sparse signals in the decentralized setting is fundamentally challenging. Suppose the expected number of errors in the network is p^γ for some $0 < \gamma < 1$, and the per-sensor error rate $\pi = p^\gamma/p$. Could we use the observed number of picks $||\mathbf{y}||_0$ to detect a pattern with $||\mathbf{x}||_0 = p^{1-\alpha} < p^{0.5}$ non-zero entries?

Under both $\mathcal{H}_0$ and $\mathcal{H}_1$, the variance of $||\mathbf{y}||_0$ grows as p^γ. Consider the variable $||\mathbf{y}||_0/\sqrt{p^\gamma}$: it has variance converging to 1 under both $\mathcal{H}_0$ and $\mathcal{H}_1$. Under $\mathcal{H}_0$, its mean is $p^{0.5\gamma}$, and under $\mathcal{H}_1$ its mean is $p^{1-\alpha-0.5\gamma}(1-2\pi) + p^{0.5\gamma}$. For $\gamma > 2(1-\alpha)$, the distributions of $||\mathbf{y}||_0$ under $\mathcal{H}_0$ and $\mathcal{H}_1$ converge, while for $\gamma < 2(1-\alpha) < 1$ the distributions are asymptotically separable. The statistic $||\mathbf{y}||_0$ (classically used in decentralized detection) can only provide reliable detection if the *per-sensor* error rate *decreases* ($\pi = p^\gamma/p \to 0$) as the network size p grows. That is, as the network grows, the sensors must have vanishing error rate for $\mathcal{H}_0$ and $\mathcal{H}_1$ to be separable.

Fortunately, data is rarely unstructured. Even when the network-wide activation pattern is sparse, the activation pattern *within some groups* may be dense and thus more easily detectable. As a starting point, we might partition the network into disjoint groups, or *clusters*, that capture strong dependencies among sensors in the hope that these clusters will individually have a high signal to noise ratio. Ideally, the number and size of the clusters would be adapted to the structure of the activation pattern, but the appropriate *scale* of the clustering is not clear in advance. Hierarchical clustering is useful for finding meaningful clusters at a range of scales, and is compatible with efficient data aggregation systems [77] for sensor networks. Recently, hierarchical clustering has been used to define wavelets bases for trees, graphs, and high-dimensional data [50, 96]. For example, a Haar wavelet basis is defined by a hierarchical clustering: whenever two clusters c_l and c_r are merged into a cluster of coarser scale, a unit vector is created,

$$\mathbf{b}_i \propto \left(\frac{1}{|c_l|} \mathbb{1}_{c_l} - \frac{1}{|c_r|} \mathbb{1}_{c_r} \right) \tag{5.1}$$

where $\mathbb{1}_c$ indicates the support of cluster c. The clustering algorithm performs $p-1$ merges; the $p-1$ vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{p-1}$ along with the constant vector $\frac{1}{\sqrt{p}}\mathbb{1}_p$ form the columns of an orthonormal



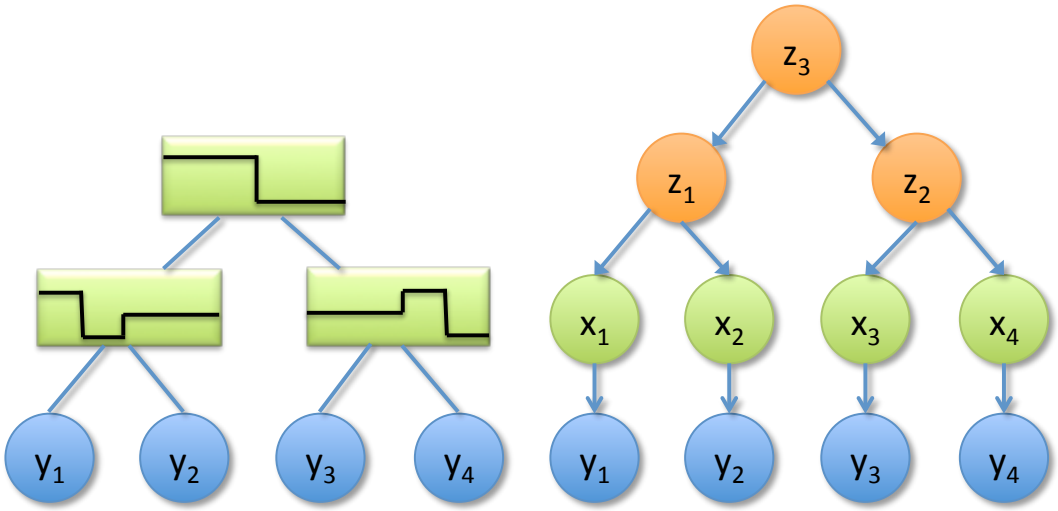
(a) $||\mathbf{x}||_0 = p^{.9}, ||\mathbf{B}^T\mathbf{x}||_0 = p^{.25}$ (b) $||\mathbf{x}||_0 = p^{.7}, ||\mathbf{B}^T\mathbf{x}||_0 = p^{.25}$

Figure 5.6: Sparsification $||\mathbf{B}^T\mathbf{x}||_0 \ll ||\mathbf{x}||_0$ exploits spatially coherent activation patterns, while small $||\mathbf{x}||_0$ produces fewer activations. The data are drawn from a quad-tree of height 5.

matrix $\mathbf{B}$. Multiplying the network observations $\mathbf{y}$ by $\mathbf{B}$ is a projection onto a new basis, where each coordinate $\mathbf{b}_i$ corresponds to the difference between the relative number of activations in a pair of merged clusters. Fig. 5.7(a) illustrates the basis functions of the transformation. The transform $\mathbf{B}$ has the property that each element $\mathbf{b}_i$ corresponds to *local* averages over sets of related nodes c_l and c_r. Under the assumption that many sets usually activate (or do not activate) jointly, events may be clearly apparent as strong signal along a small number of basis elements. More formally, patterns in $\mathbf{x}$ supported on the clusters used to define $\mathbf{B}$ will tend to be concentrated in a few elements $\mathbf{b}_i$, and so $||\mathbf{B}^T\mathbf{x}||_0 \ll ||\mathbf{x}||_0$. Fig. 5.6 shows that sparsifiable data is inherently structured.

A basis for detection. Just as a Fourier transform maps an acoustic signal into a coordinate frame that yields insight about the frequency content of the signal, multiplying network activations $\mathbf{x}$ by the basis $\mathbf{B}$ maps the sensor data onto a new coordinate system defined by the hierarchical clustering, and can expose correlated activations. In the following, we prove that with the Haar wavelet basis, the "sparsification ratio" $\frac{||\mathbf{x}||_0}{||\mathbf{B}^T\mathbf{x}||_0}$ plays a central role in achievable error rates. In particular, the following new theorem (with proof outline in the appendix) shows the power of a sparsifying transform to concentrate the signal along at least one new coordinate without concentrating the random noise.

Theorem 5.4.1. *Let* $\mathbf{B}$ *be a Haar basis that sparsifies a signal* $\mathbf{x}$*, i.e.* $||\mathbf{B}^T\mathbf{x}||_0 = p^{1-\beta}$*,* $||\mathbf{x}||_0 = p^{1-\alpha}$*,* $0 < \alpha < \beta < 1$*. Let* $\mathbf{y}$ *be the signal observed through a binary symmetric channel, with error rate* π *bounded away from* $1/2$ *(i.e., for some* $\epsilon > 0$*,* $\pi < 1/2 - \epsilon$*). Then applying the test* $|\mathbf{b}^T\mathbf{y}| > \frac{(1-2\pi)}{2}\sqrt{\frac{||\mathbf{x}||_0}{||\mathbf{B}^T\mathbf{x}||_0}}$ *to each of the* $p-1$ *non-constant basis elements* $\mathbf{b} \in \mathbf{B}$ *gives false negative*



(a) Transform (b) Latent Tree Model

Figure 5.7: Illustration of the transform (a), constant $\mathbf{b}_0$ not shown. (b) Latent Tree Model for $d = 2$ and $p = 4$. The sensors $\mathbf{y}$ measure the pattern $\mathbf{x} = [x_1, \ldots, x_4]$ up to some noise. The pattern is structured hierarchically; variables z_i represent the variables of the latent tree.

rate (FNR) and false positive rate (FPR) bounded as

$$FNR \leq 2p \exp\left(-\frac{(1-2\pi)^2}{2}\frac{||\mathbf{x}||_0}{||\mathbf{B}^T\mathbf{x}||_0}\right) \to 0 \text{ as } p \to \infty,$$
$$FPR \leq 2p \exp\left(-\frac{(1-2\pi)^2}{2}\frac{||\mathbf{x}||_0}{||\mathbf{B}^T\mathbf{x}||_0}\right) \to 0 \text{ as } p \to \infty$$

Proof of Theorem 5.4.1. Let k be the size of clusters merged by a non-constant basis element $\mathbf{b}$. If no signal is present, the number of picks is the support of each cluster is binomially distributed: $\text{picks}(c_i) \sim \text{Bin}(\cdot; k, \pi)$, and so the difference $\text{picks}(c_i) - \text{picks}(c_2)$ is zero-mean, with variance $2k\pi(1-\pi)$ The projection $b^{(k)}y$ of observations y onto the basis element scales each pick by $\frac{1}{\sqrt{2k}}$ (recall that each basis element has unit L2 length), so if y is only i.i.d noise, $b^{(k)}y$ is zero-mean, with variance $\pi(1-\pi)$. Let $\tau = \frac{(1-2\pi)}{2}\sqrt{\frac{||\mathbf{x}||_0}{||\mathbf{B}^T\mathbf{x}||_0}}$. Under $\mathcal{H}_0$, $\mathbb{E}\left[|\mathbf{b}^T\mathbf{y}|\right] = 0$, and is the sum of $2k$ terms (k from each cluster) taking values $\{-\frac{1}{\sqrt{2k}}, 0, \frac{1}{\sqrt{2k}}\}$. Hoeffding's inequality gives $\mathbb{P}\left[|\mathbf{b}^T\mathbf{y}| \geq \tau\right] \leq 2\exp\left(-\tau^2\right) \to 0$.

$$\mathbb{P}\left[|b^{(k)}y| \geq \epsilon\right] \leq 2\exp\frac{-2\epsilon^2}{\sum_{i=1}^{2k}(1/\sqrt{2k})^2} = 2\exp\left(-2\epsilon^2\right)$$

Plugging in the threshold $\sqrt{\frac{||\mathbf{x}||_0}{||\mathbf{B}^T\mathbf{x}||_0}}$ gives Substituting τ for ϵ gives

$$\mathbb{P}\left[|b^{(k)}y| \geq \tau\right] \leq 2\exp\left(-\tau^2\right) \to 0$$

Taking the union bound, FPR $\leq 2p\exp\left(-\tau^2\right) \to 0$. Under $\mathcal{H}_1$, for some $\mathbf{b}$, $\mathbb{E}\left[|\mathbf{b}^T\mathbf{y}|\right] \geq \sqrt{\frac{||\mathbf{x}||_0}{||\mathbf{B}^T\mathbf{x}||_0}}(1-2\pi)$. As under $\mathcal{H}_0$, Hoeffding's inequality bounds the probability of deviation by τ from the mean (conveniently, $\mathbb{E}\left[|\mathbf{b}^T\mathbf{y}|\right] - \tau > \tau$): $\mathbb{P}\left[|\mathbf{b}^{(k)}\mathbf{y}| \leq \tau\right] \leq 2\exp\left(\tau^2\right) \to 0$. Taking the union bound over p basis elements, FNR $\leq 2p\exp\left(-\tau^2\right) \to 0$.

This theorem states that *for any* constant error rate π, as the network size p grows, the probability of miss (FNR) and the probability of false alarm (FPR) are driven to 0 by the decision rule that declares "event" when $|\mathbf{b}^T\mathbf{y}|$ exceeds the specified threshold, for any of the $p-1$ non-constant basis elements $\mathbf{b}$. For comparison, recall that reliable detection using the network-wide pick count $||\mathbf{y}||_0$ requires the error rate π to rapidly decay to zero as p grows.

The sparsifying basis $\mathbf{B}$ thus enables reliable detection in a broad noise regime that cannot be detected by the network-wide average. This insight shows that indeed quality of sensors can be traded against quantity. Of course, this strong result assumes that the event signal $\mathbf{x}$ is sufficiently sparsifiable by the basis $\mathbf{B}$. This assumption holds both in a natural theoretical model, discussed next, and appears to be reasonable for many types of real world data.

Modeling sparsifiable events. When is sensor data sparsifiable? Let us consider again the natural hierarchical basis $\mathbf{B}$, defined according to Eq. 5.1 as introduced at the beginning of this section. Singh [96] shows that for this particular basis, the assumption $||\mathbf{B}^T\mathbf{x}||_0 \ll ||\mathbf{x}||_0 \leq \sqrt{p}$ is fulfilled when the pattern $\mathbf{x}$ is drawn from an intuitive class of generative models. For completeness, the model is presented here. In this model, dependencies among sensors are modeled via a *tree* of regular degree d: the leaves correspond to the event occurrence x_i at each sensor, and internal nodes correspond to the occurrence or non-occurrence of an event at a particular region and scale. Let $\ell = 0, 1, \ldots L$ denote the level in the tree, where the activations $\{x_i\}$, $i = 1, \ldots, p$ are leaves at level $L = \log_d p$, and the root is at $\ell = 0$. The internal (non-leaf) nodes in the tree capture multi-scale dependencies among the leaves. Let $\mathbf{z}$ denote all nodes in the tree. The joint distribution of $\mathbf{z}$ factorizes as

$$p(\mathbf{z}) = p(z_0)\prod_{\ell=1}^{L}\prod_{i\in V_\ell} p(z_i|z_{\text{parent}(i)}) \tag{5.2}$$

where V_l denotes the vertices at layer l. The probability that a node equals its parent is specified by $\gamma_\ell = \ell\beta\log d$. This coupling is weaker near the root and stronger near the leaves, producing multi-scale dependencies. Sufficiently weak dependencies are considered negligible, and so the latent variables $z_i \in \ell_0$ at some initial level ℓ_0 are drawn independently from their parents: $p(z_i = 1|z_{\text{parent}(i)}) = p(z_i = 1) \propto e^{\gamma_{\ell_0}}$. This approximates distant regions of the network as inde-

pendent. The conditional probability of a node z_i at $\ell > \ell_o$ is

$$p(z_i|z_{\text{parent}(i)}) \propto \begin{cases} e^{\gamma_\ell} & \text{, if } z_i = z_{\text{parent}(i)} \\ 1 & \text{, if } z_i \neq z_{\text{parent}(i)} \end{cases}$$

Patterns drawn from this model are localized and multi-scale, as illustrated with a quad-tree in Fig. 5.6.

Bounds for finite networks. When the event $\mathbf{x}$ is drawn via Eq. (5.2), Singh [96] showed that as the number of sensors goes to infinity, the assumption $||\mathbf{B}^T\mathbf{x}||_0 \ll ||\mathbf{x}||_0 \leq \sqrt{p}$ holds with high probability. However, these results do not clearly indicate whether the bounds are effective for large (e.g., hundreds to tens of thousands of sensors) but finite networks. Next, we provide a stronger bound on the sparsification ratio obtained by the wavelet transform, and explain how Theorem 5.4.1 can be strengthened to provide bounds on FNR and FPR for fixed network size.

Theorem 5.4.2. *Let* $\mathbf{x}$ *be a pattern drawn at random from the latent tree model with uniform degree* d *and depth* $L = \log_d p$. *Let* $\ell_0 = \frac{\alpha}{\beta}$ *and* $\gamma_\ell = \ell\beta \log d$ *for* $\ell \geq \ell_0$, *where* $0 \leq \alpha \leq \beta \leq 1$. *Then for* $0 < \epsilon < 1$,

$$\mathbb{P}\left[||\mathbf{x}||_0 > (1+\epsilon)(Lp^{1-\alpha})\right] \leq \exp\left(-\frac{\epsilon^2}{3}Lp^{\alpha(\frac{1}{\beta}-1)}\right)$$
$$\mathbb{P}\left[||\mathbf{x}||_0 < (1-\epsilon)cp^{1-\alpha}\right] \leq \exp\left(-\frac{c\epsilon^2}{2}p^{\alpha(\frac{1}{\beta}-1)}\right)$$

where $c = \left(\frac{1}{4}\right)^{(\frac{1}{\alpha}-\frac{1}{\beta}+0.5)}$ *is constant with respect to* p.

$$\mathbb{P}\left[||\mathbf{B}^T\mathbf{x}||_0 > (1+\epsilon)d^2\log_d p \cdot p^{(1-\beta)}\right] \leq \exp\left(-\frac{\epsilon^2}{3}d \cdot p^{(1-\beta)}\right)$$

$$\mathbb{P}\left[\frac{||\mathbf{x}||_0}{||\mathbf{B}^T\mathbf{x}||_0} > \frac{\kappa(\epsilon)}{\log_d p} \cdot \frac{p^{1-\alpha}}{p^{1-\beta}}\right] \geq 1 - 2\exp\left(-\frac{c\epsilon^2}{2}p^{\alpha(\frac{1}{\beta}-1)}\right)$$

where $c = \left(\frac{1}{4}\right)^{(\frac{1}{\alpha}-\frac{1}{\beta}+0.5)}$ *and* $\kappa(\epsilon) = \frac{(1-\epsilon)}{(1+\epsilon)}\frac{c}{d^2}$ *are constant with respect to* p.

Proof of Theorem 5.4.2. Let $X_T^{(i)}$ denote the leaves in the i^{th} subtree rooted at level ℓ_0. In the latent tree model, the nodes at level ℓ_0 are independent, and so the numbers of active leaves in each subtree $\|X_T^{(i)}\|_0$, $i = 1, \ldots, d^{\ell_0}$ are i.i.d. We now obtain upper and lower bounds on $\mathbb{E}\left[\|X_T^{(i)}\|_0\right]$.

Lemma 1. $\mathbb{E}\left[\|X_T^{(i)}\|\right] \geq c \cdot p^{1-\alpha}p^{-\frac{\alpha}{\beta}}$ *where*
$c = \left(\frac{1}{4}\right)^{(\frac{1}{\alpha}-\frac{1}{\beta}+0.5)}$ *is constant with respect to* p.

Proof. Observe that $\mathbb{P}[X_i = 1] \geq q_{\ell_0} \prod_{\ell > \ell_0}^{L} (1 - q_\ell) > q_{\ell_0}(1 - q_{\ell_0})^{L - \ell_0}$, which is the probability that X_i and all its ancestors to level ℓ_0 are active. Linearity of expectation gives

$$\mathbb{E}\left[\|X_T^{(i)}\|_0\right] \geq \frac{1}{2} p^{1-\alpha} p^{-\frac{\alpha}{\beta}} \left[(1 - p^{-\alpha})^{(\log_d p^\alpha)}\right]^{(\frac{1}{\alpha} - \frac{1}{\beta})}$$

Line 2 uses the fact that $\frac{d^{-\beta\ell}}{2} \leq q_\ell \leq d^{-\beta\ell}$.
Line 3 uses $L - \ell_0 = L(1 - \frac{\alpha}{\beta}) = (\log_d p^\alpha)(\frac{1}{\alpha} - \frac{1}{\beta})$
Line 4 uses $d^L = p$. The term $\left[(1 - p^{-\alpha})^{(\log_d p^\alpha)}\right]^{(\frac{1}{\alpha} - \frac{1}{\beta})}$ can be bounded as follows. Let $w = p^\alpha$, and consider $(1 - p^{-\alpha})^{(\log_d p^\alpha)} = (1 - \frac{1}{w})^{(\log_d w)}$. Now, $w > 1$, and so $(1 - \frac{1}{w})^{(\log_d w)} > (1 - \frac{1}{w})^w$, which is a monotone increasing function that converges to e^{-1} as $w \to \infty$. Thus $(1 - \frac{1}{w})^w > \frac{1}{4}$ for $w \geq 2$, or equivalently, for $p \geq 2^{1/\alpha}$. Then,

$$\left[(1 - p^{-\alpha})^{(\log_d p^\alpha)}\right]^{(\frac{1}{\alpha} - \frac{1}{\beta})} \geq \left(\frac{1}{4}\right)^{(\frac{1}{\alpha} - \frac{1}{\beta})}$$

and

$$\mathbb{E}\left[\|X_T^{(i)}\|_0\right] \geq \frac{1}{2} p^{1-\alpha} p^{-\frac{\alpha}{\beta}} \left[\frac{1}{4}\right]^{(\frac{1}{\alpha} - \frac{1}{\beta})}$$

for $p \geq 2^{1/\alpha}$. □

Lemma 2. *$(1 - \frac{1}{x})^x$ is monotonic increasing for $x > 1$.*

Proof.

$$\frac{d}{dx}\left[(1 - \frac{1}{x})^x\right] = \left(\frac{x-1}{x}\right)^x \left(\frac{1}{x-1} - \log\left(\frac{x}{x-1}\right)\right)$$

Using $\ln \frac{x}{x-1} = \sum_{n=1}^{\infty} \frac{1}{nx^n} < \sum_{n=0}^{\infty} \frac{1}{x^n} - 1 = \frac{1}{x-1}$, the rightmost term is shown to be positive. Then, $\frac{d}{dx}\left[(1 - \frac{1}{x})^x\right] > 0$ for $x > 1$. □

Let $W_T^{(i)} = \frac{\|X_T^{(i)}\|_0}{p^{1-\frac{\alpha}{\beta}}}$, and $W = \sum_i W_T^{(i)}$. There are $p^{1-\frac{\alpha}{\beta}}$ leaves in each $X_T^{(i)}$, so $W_T^{(i)} \in [0, 1]$. There are $p^{\frac{\alpha}{\beta}}$ subtrees, so by Lemma 1, $cp^{-\alpha} p^{\frac{\alpha}{\beta}} < \mathbb{E}[W]$. Hoeffding's inequality gives, for $0 < \epsilon < 1$,

$$\mathbb{P}\left[\|X\|_0 > (1 + \epsilon)(Lp^{-\alpha} p^{\frac{\alpha}{\beta}}) p^{1-\frac{\alpha}{\beta}}\right] \leq \exp\left(-\frac{\epsilon^2}{3} Lp^{-\alpha} p^{\frac{\alpha}{\beta}}\right)$$

$$\mathbb{P}\left[\|X\|_0 > (1 + \epsilon)(Lp^{1-\alpha})\right] \leq \exp\left(-\frac{\epsilon^2}{3} Lp^{\alpha(\frac{1}{\beta} - 1)}\right)$$

$$\mathbb{P}\left[W < (1 - \epsilon) cp^{-\alpha} p^{\frac{\alpha}{\beta}}\right] \leq \exp\left(-\frac{\epsilon^2}{2} cp^{-\alpha} p^{\frac{\alpha}{\beta}}\right)$$

$$\mathbb{P}\left[\|X\|_0 < (1 - \epsilon) cp^{1-\alpha}\right] \leq \exp\left(\frac{c\epsilon^2}{2} p^{\alpha(\frac{1}{\beta} - 1)}\right)$$

Next, we will say an *edge flip* occurs at level ℓ when a node at level ℓ does not equal its parent. The number of non-zero coefficients is bounded as $||\mathbf{B}^T\mathbf{x}||_0 \leq dL \cdot F$, where F is the number of edge flips in the tree. An edge at level ℓ flips with probability $q_\ell = 1/(1+d^{\beta\ell}) < d^{-\beta\ell}$, so we find that

$$\mathbb{E}[F] = \frac{d^{(1-\beta)(L+1)} - d^{1-\beta}}{d^{(1-\beta)} - 1} < d \cdot d^{L(1-\beta)}$$

Let $\bar{\mu} = d \cdot d^{L(1-\beta)}$ For $0 < \epsilon < 1$, the Hoeffding inequality gives $\mathbb{P}[F > (1+\epsilon)\bar{\mu}] \leq \exp\left(-\frac{\epsilon^2}{3}\bar{\mu}\right)$ from which follow

$$\mathbb{P}\left[\left\|\mathbf{B}^T x\right\|_0 > (1+\epsilon)d^2 L \cdot d^{L(1-\beta)}\right] \leq \exp\left(-\frac{\epsilon^2}{3} d \cdot d^{L(1-\beta)}\right)$$

$$\mathbb{P}\left[\left\|\mathbf{B}^T x\right\|_0 > (1+\epsilon)d^2 \log_d p \cdot p^{(1-\beta)}\right] \leq \exp\left(-\frac{\epsilon^2}{3} d \cdot p^{(1-\beta)}\right)$$

This result shows that the crucial sparsification ratio $\frac{||\mathbf{x}||_0}{||\mathbf{B}^T\mathbf{x}||_0}$ in Theorem 5.4.1 grows at (within a log factor of) the desired rate $p^{1-\alpha}/p^{1-\beta}$, with probability that increases exponentially with network size p. This theorem can be used to derive bounds on FNR and FPR for a specified network size p and model parameters α, β, degree d: the bound is substituted for $\frac{||\mathbf{x}||_0}{||\mathbf{B}^T\mathbf{x}||_0}$ in Theorem 5.4.1, and the probability that the above bound does not hold can be added to the resulting FNR and FPR.

In summary, in this section, we have shown that under a natural model of sensor activations, there indeed exists a sparsifying transform which leads to improved detection performance. In the following, we show how one can adapt a basis to data in order to achieve maximal sparsification.

5.5 Sparsifying Basis Learning

Sec. 5.4.2 shows that if an event is "sparsifiable" we can better separate $\mathcal{H}_0$ and $\mathcal{H}_1$ by projecting (multiplying by a basis $\mathbf{B}$) the observations $\mathbf{y}$ onto a different coordinate system where the signal is concentrated into fewer components (a "sparser representation" of the signal). The Haar wavelet basis is an example of a basis that improves detection of signals with certain structured (hierarchical) dependencies. In general, can we construct or learn a sparsifying basis without assuming such dependencies?

Let $\mathbf{B}$ be an orthonormal matrix and $\mathbf{x}$ a vector of uncorrupted binary activations, Theorem 5.4.1 states that the sparsification ratio $\frac{||\mathbf{x}||_0}{||\mathbf{B}^T\mathbf{x}||_0}$ directly impacts the amount of separation between $\mathcal{H}_0$ and $\mathcal{H}_1$. In fact, given that $||\mathbf{x}||_0$ is fixed, the two hypotheses are maximally separated when $||\mathbf{B}^T\mathbf{x}||_0$ is minimized. In other words, we can construct the optimal basis by solving the following optimization

problem:

$$\arg\min_{\mathbf{B}} ||\mathbf{B}^T\mathbf{X}||_0, \text{ subject to } \mathbf{B}\mathbf{B}^T = \mathbf{I} \tag{5.3}$$

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

However, direct minimization of $||\mathbf{B}^T\mathbf{X}||_0$ is NP-hard in general [30]. In practice, the ℓ_0-norm is often replaced by the convex and "sparsity-promoting" ℓ_1-norm [18]. This suggests the following relaxation heuristic for (5.3).

$$\arg\min_{\mathbf{B}} ||\mathbf{B}^T\mathbf{X}||_1, \text{ subject to } \mathbf{B}\mathbf{B}^T = \mathbf{I} \tag{5.4}$$

where $||\cdot||_1$ is the maximum absolute column sum of the matrix.

Direct approximation. For large problems, we are interested in efficiently computable heuristics for Eq. (5.4). *Independent Component Analysis* (`ICA`) is one such approximation, and solves the following optimization problem

$$\arg\min_{\mathbf{B}} G(\mathbf{B}^T\mathbf{X}), \quad \text{subject to} \quad \mathbf{B}\mathbf{B}^T = \mathbf{I} \tag{5.5}$$

where G is a nonlinear convex smooth approximation to the ℓ_1 penalty function, e.g., $\log\cosh(x)$, $-\exp(-x^2/2)$, and x^4 [65]. Fig. 5.5 illustrates these functions in relation to the linear penalty function.

Eq. (5.5) can be solved with stochastic gradient algorithm by taking the derivative of G. However this approach is often slow and requires fine tuning; this leads to the development of "FastICA", an efficient fixed-point algorithm. Implementation details of FastICA and in-depth analysis can be found in [65].

Let $g = G'$, the one unit algorithm for fastICA is given below for completeness.

Algorithm 2: `ICA` one-unit solution

$\mathbf{b} \leftarrow$ random unit vector
while $\mathbf{b}$ *not converged* **do**
  $\mathbf{b} \leftarrow \mathbb{E}\left[\mathbf{x}g(\mathbf{b}^T\mathbf{x})\right] - \mathbb{E}\left[g'(\mathbf{b}^T\mathbf{x})\right]\mathbf{x}$;
  $\mathbf{b} \leftarrow \mathbf{b}/\left\|\mathbf{b}\right\|$;

There exists two variations of FastICA, both of which use Alg. 2 as subroutine. The *deflationary orthogonalization* greedily finds and fixes one component $\mathbf{b}_i$ at a time. The *symmetric orthogonalization* finds all components $\mathbf{b}_i, \forall i = 1, \ldots, p$ and orthogonalizes all of them at the end of each iteration [64]. While the *symmetric* approach utilizes parallelization and is multitudes faster than *deflationary* approach, the latter gives much better results empirically.

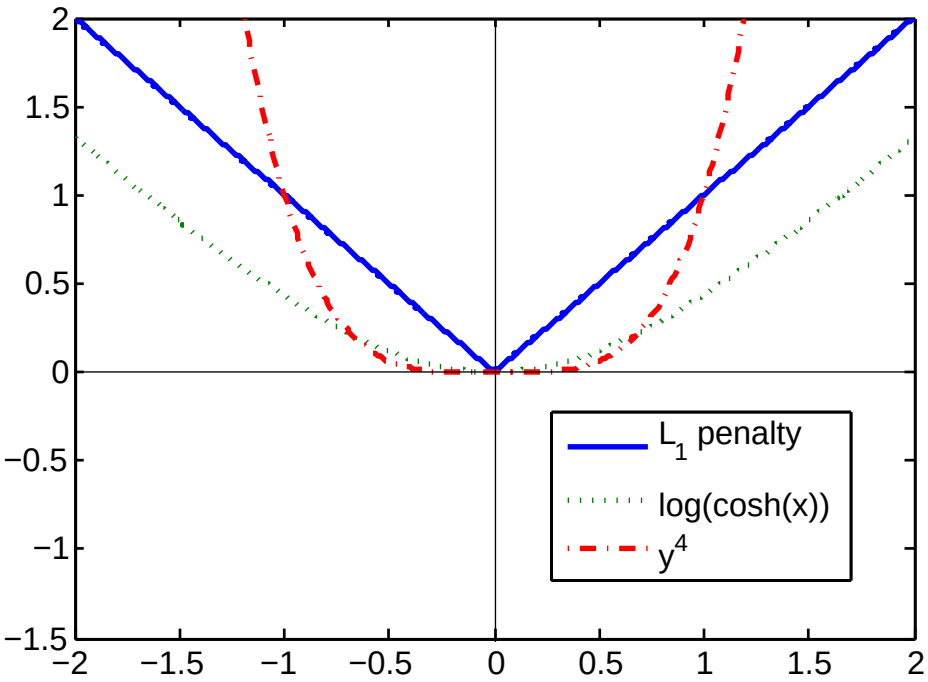


Figure 5.8: Smooth ℓ_1 approximation functions used in ICA with the linear ℓ_1 penalty function plotted in blue solid line.

ICA is a mature technique that finds a linear transformation for non-Gaussian data so that the components are as independent as possible in the transform domain. For unlabeled training data $x \in \mathbb{R}^{p \times n}$ where p is the dimension of the data and n the number of samples, ICA is traditionally defined as the following optimization problem [65]:

Noise-tolerant relaxed approximation. Ideally we want to learn from noise-free observations $\mathbf{X}$. However, training data constructed from real-world measurements will contain noise or outliers, and instead we are forced to train with $\mathbf{Y}$, which is the observation matrix $\mathbf{X}$ corrupted with noise.

Consequently, we may not be able to obtain the "best" basis by optimizing $\mathbf{B}^T\mathbf{Y}$ as in `ICA`. Instead, we may wish to find a basis that sparsely represents "most of" the observations. More formally, we introduce a latent matrix $\mathbf{Z}$, which can be thought of as the "cause", in the transform domain, of the noise-free signal $\mathbf{X}$. In other words $\mathbf{X} = \mathbf{BZ}$. We desire $\mathbf{Z}$ to be sparse, and $\mathbf{BZ}$ to be close to the observed signal $\mathbf{Y}$. This motivates the next optimization:

$$\arg\min_{\mathbf{B},\mathbf{Z}} \|\mathbf{Y} - \mathbf{BZ}\|_F^2 + \lambda\|\mathbf{Z}\|_1, \text{ subject to } \mathbf{BB}^T = \mathbf{I} \tag{5.6}$$

where $||\cdot||_F$ is the matrix Frobenius norm, and $\lambda > 0$ is a free parameter. Eq. (5.6) essentially balances the difference between $\mathbf{Y}$ and $\mathbf{X}$ with the sparsity of $\mathbf{Z}$: increasing λ more strongly penalizes choices of $\mathbf{Z}$ that are not sparse.

Although Eq. (5.6) is non-convex, fixing either $\mathbf{B}$ or $\mathbf{Z}$ makes the objective function with respect

to the other convex. The objective can then be solved in an iterative two-step convex optimization process — *Orthogonal Procrustes* [54] and *LASSO with orthonormal design* [15]. The two-step procedure is given below.

Algorithm 3: SLSA two-step convex optimization procedure

Step 1: Orthogonal Procrustes

Fix $\mathbf{Z}$, solve $\min_{\mathbf{B}} ||\mathbf{Y} - \mathbf{BZ}||_F^2, : \mathbf{BB}^T = \mathbf{I}$

$M \leftarrow \mathbf{YZ}^T$;

$M = U\Sigma V^T$;

$\mathbf{B} \leftarrow UV$;

Step 2: LASSO with orthonormal design

Fix $\mathbf{B}$, solve $\min_{\mathbf{Z}} \|\mathbf{Y} - \mathbf{BZ}\|_F^2 + \lambda\|Z\|_1$

$K \leftarrow \mathbf{Z}^T\mathbf{Y}$;

$\mathbf{Z} \leftarrow \text{sign}(K) \times \max(|K| - \lambda)$;

The formulation of Eq. (5.6) and solution in Alg. 3 is equivalent to *Sparse Latent Semantic Analysis* (`SLSA`) [19], which was introduced for applications involving topic models for text data. Here we adopt the name for consistency.

We note that both Eq. (5.5) and Eq. (5.6) should be viewed as efficiently computable heuristics for Eq. (5.3), which is a non-convex optimization over the Stiefel manifold of all size-p orthonormal matrices. As such, they are practical expedients towards the goal of obtaining a sparsifying basis.

5.6 Implementation in Wireless Sensor Networks

In this section, we describe practical issues necessary for using a sparsifying basis for event detection in real-world sensor networks. We highlight how the previous problem formulation can be separated into two computational steps:

- *Offline training* of basis learning and detection threshold selection;
- *Online detection* via decentralized detection or in-network data aggregation.

5.6.1 Offline Training

The three sparsifying bases (`haar` wavelet, `ICA`, `SLSA`) considered here can be easily implemented and are available in many off-the-shelf optimization packages. Basis learning in small networks ($p < 100$) is very fast in general (within seconds) and can be done online. For larger networks ($p > 500$), offline training may be more suitable.

Basis learning.

Learning a basis for p sensors requires at least p measurements of the network. If this is not available (e.g., a seismic network with 1000 sensors may not yet have observed 1000 earthquakes,

or a network of health sensors may not have observed an epidemic), then simulations provide a practical way to supplement real data. One advantage of using simulations in this way is that while simulations may be slow and compute-intensive, the learned basis produces a fast and efficient detection rule. In Sec. 5.8, we empirically assess the amount of data required to train a good basis and present two case studies using only data generated from simulations.

Supplementing a training set of real data with data from simulations requires domain knowledge of the applications, e.g., simulation of seismic waves or disease infection models. When training with such a combined data set, it may be appropriate to place higher weights on the real data than the simulated data. This can be easily achieved with a modified version of `SLSA`.

Selecting the detection threshold. An event is reported whenever $|\mathbf{b}_i^T\mathbf{y}| \geq \tau$ for any non-constant $\mathbf{b}_i \in \mathbf{B}$. The threshold τ is typically chosen as a value that satisfies constraints on the false positive rate during cross validation with historical data of event observations. This approach does not rely on positive training examples, and so a threshold τ can be learned using only the noise profile of each sensor. Suppose sensors $i = 1, 2, \ldots, p$ have binary error rates $\pi_1, \ldots, \pi_p$, we have $\mathbb{E}\left[|\mathbf{b}^T\mathbf{y}|\right] = \sum_i^p b_i\pi_i$. Given that the basis is orthonormal, under $\mathcal{H}_0$, Hoeffding's Inequality states that

$$\mathbb{P}\left[|\mathbf{b}^T\mathbf{y}| > \tau\right] \leq \exp\left(-2\left(\tau - \mathbb{E}\left[|\mathbf{b}^T\mathbf{y}|\right]\right)^2\right)$$

By setting the right hand side to a false positive rate constraint, we can easily derive a threshold that satisfies the system requirement. In particular, in order to ensure that $|\mathbf{b}^T\mathbf{y}| \leq \tau$ for all $\mathbf{b} \in \mathbf{B}$ (i.e., no false alarm happens) with probability at least $1 - \delta$, it suffices to choose

$$\tau = \max_{\mathbf{b}\in\mathbf{B}} \mathbb{E}\left[|\mathbf{b}^T\mathbf{y}|\right] + \sqrt{\frac{1}{2}\log\frac{p}{\delta}}.$$

This approach is similar in flavor to the threshold selection method in [38].

5.6.2 Online Detection

At runtime, the fusion center collects information from the sensors and applies the threshold τ to the statistics $|\mathbf{B}^T\mathbf{y}|$. Depending on the network structure, this aggregation can be done in-network.

Decentralized Detection. The proposed sparsifying bases are suitable for both measurements from binary or other real-valued sensors. However, in large sensor networks, it is infeasible to constantly stream raw measurements to the fusion center. Instead, it may be desirable to offload the computation from the fusion center to each sensor locally so that only a small amount of information (e.g., a single message) is communicated infrequently when a significant signal is detected. For example, sensors in the Community Seismic Network perform *local anomaly detection* and communicate

"abnormal" accelerations (using hypothesis testing) as a binary signal [38].

In-network Aggregation.

In a multi-hop WSN where individual sensors have limited transmission range, it is likely that the communication topology will be hierarchically organized in a way that closely resembles the spatial configuration of the sensors. Consequently, strong dependencies are likely to exist among groups of sensors that are near in the communication network. This suggests that the test statistics $\mathbf{b}_i\mathbf{y}$ can also be computed *in-network* by hierarchically aggregating sensor data into clusters. For example, by using the number of hops needed to communicate between a pair of nodes as a measure of the dissimilarity of two sensors, hierarchical clustering produces the transforms $\mathbf{B}$ supported over groups of communication-efficient clusters. These clusters may compute the transform $\mathbf{By}$ in a bottom-up fashion while simultaneously testing for detection.

For bases that lack obvious spatial hierarchy, it is possible to adaptively build a routing tree to minimize the communication distance between groups of sensors that tend to co-activate in a sparse sensor setting [49].

5.7 Preserving user privacy

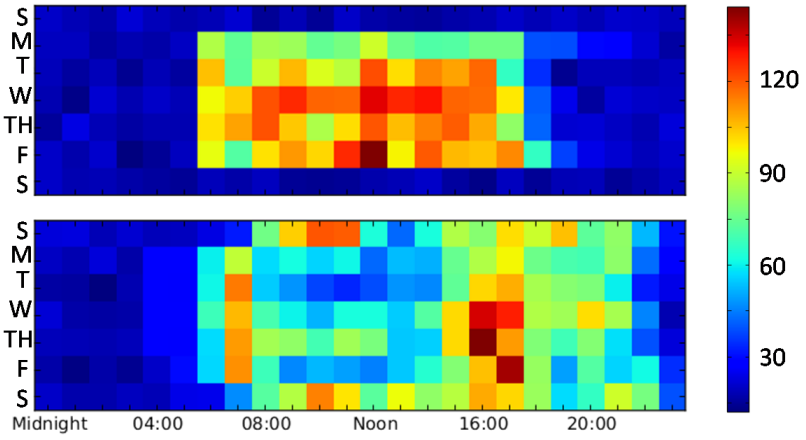


Figure 5.9: Even simple statistics such as pick rates can reveal patterns of activity and occupancy in the area around a sensor. A Phidget sensor in an office setting (above) shows high ambient noise (human activity) during weekday business hours, while a Phidget in a home (below) shows typical evening and weekend activity.

Personal sensor devices make it easy to collect immense amounts of personal information, and so it is important for systems designers and members of the public to understand the potential tradeoff between privacy and the performance of digital services. Fig. 5.9 demonstrates how even simple data like false pick rates can inadvertently convey information about patterns of human behavior. While revealing by itself, such sensor data is likely to be combined with other information to create

an increasingly detailed picture of each user's daily activities. Indeed, it is standard practice for internet services to aggregate multiple sources of information about their users: "When you upload, submit, store, send or receive content to or through our Services, you give Google (and those we work with) a worldwide license to use, host, store, reproduce, modify, create derivative works (such as those resulting from translations, adaptations or other changes we make so that your content works better with our Services), communicate, publish, publicly perform, publicly display and distribute such content."[1] Given the open-ended implications of sensor data collection, it is natural to ask, how little data can be collected while still adequately performing a task?

Theorem 5.4.1 and Theorem 5.4.2 provide a way to quantify the relationship between the number of sensors, the error rates of each sensor, and the performance of the network to detect sparsifiable events. This relationship makes it possible to determine the maximum allowable error rate for each sensor when all other parameters are held fixed, with the implication that sensors with lower error rates may deliberately corrupt some of their data while maintaining network-wide performance criteria. Fig. 5.10 demonstrates how the maximum allowable error rate increases as a function of network size.

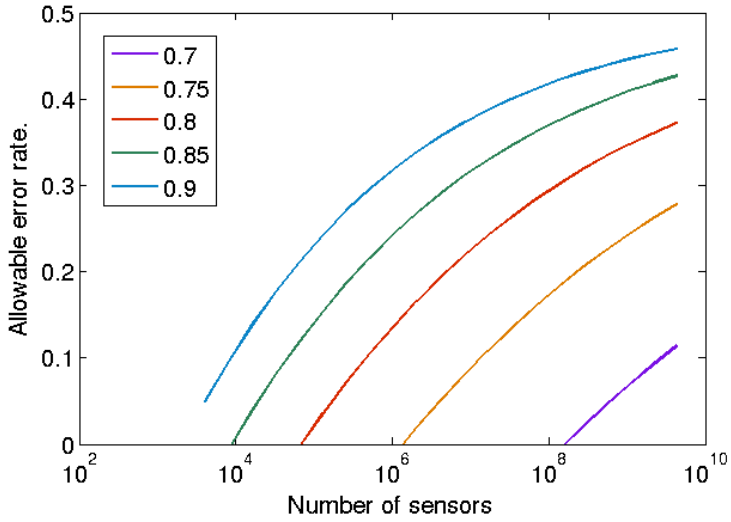
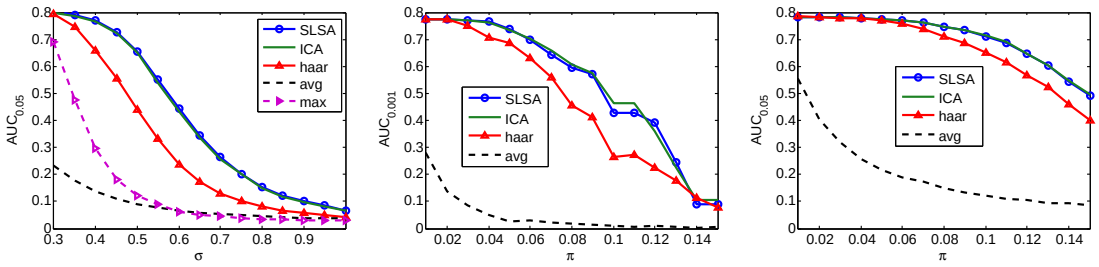


Figure 5.10: Thm. 5.4.2 implies a relationship between the number of sensors and the maximum per-sensor error rate π required to achieve a given system-wide false positive rate. Here, the system-wide false positive rate is fixed at 0.1, and the maximum allowable error rate is plotted as a function of the number of sensors, across a range of values for β. $\alpha = 0.5$.

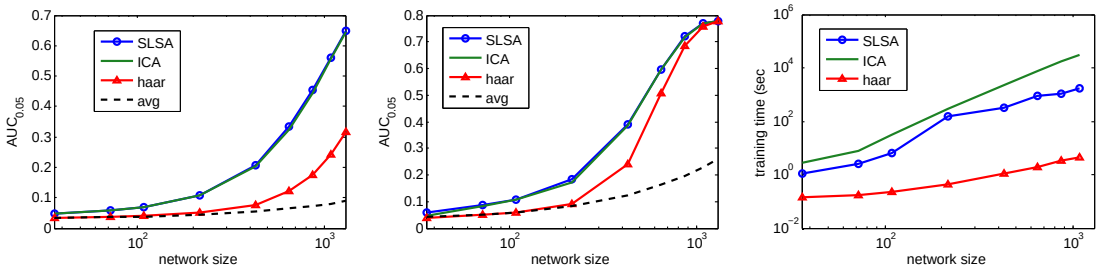
Deliberately increasing the false pick rate of each sensor serves to limit the rate of information collected about a given user. This is in contrast to other approaches to privacy, such as differential privacy [34] which typically adds noise to the result of a query function operating on a database of sensitive information. Volunteers may be uncomfortable about providing information to the

[1] Google Terms of Service, http://www.google.com/intl/en/policies/terms/



(a) Gaussian noise (b) Binary noise, $AUC_{0.001}$ (c) Binary noise, $AUC_{0.05}$

Figure 5.11: Comparing the three bases — `SLSA`, `ICA`, `haar` to baselines — global average (and single max in (a)) on a synthetic data set generated from the latent tree model. Figures (b) and (c) evaluate two different false positive constraints. The learned bases significantly outperform the baselines under strong noise.



(a) Gaussian noise, $\sigma = 0.5$ (b) Binary noise, $\pi = 0.04$ (c) Training time

Figure 5.12: Detection performance as a function of network size $p = [36, 72, 108, 216, 432, 648, 864, 1080, 1296]$ using all 20,000 training samples. The learned bases show more than 5x performance improvement compared to the baselines in (a) and (b).

"trusted" database assumed by differential privacy schemes, and so may wish to reduce the accuracy of information that their sensors report.

5.8 Experiments

We empirically evaluate the detection performance of the three sparsifying bases: `SLSA`, `ICA`, and hierarchical wavelets (`haar`) trained and tested on both simulated and real measurements in different domains. The experimental setup is summarized here.

Baseline algorithms. In keeping with our focus of very large community sensor networks, we compare against baselines that could potentially be computed for real-time detection on tens of thousands of sensors, and that are naturally suited to the client-server communication model of internet-enabled sensors.

- `avg`: network-wide average, $1/p \sum_i^p y_i$;
- `max`: single sensor maximum, $\max_i y_i$;

- SS-k: scan statistics that aggregates the k-nearest neighbors for each sensor [86];
- SS-r: scan statistics that aggregates all sensors within a radius r for each sensor [86].

Evaluation data sets. The data sets include

- **Synthetic** data from latent tree model, 1296 nodes;
- **Gnutella P2P network**: 1769 nodes;
- **Japan seismic network**: 721 nodes;
- **CSN seismic network**: 128 nodes;
- **Long Beach** seismic network: 1,000 nodes.

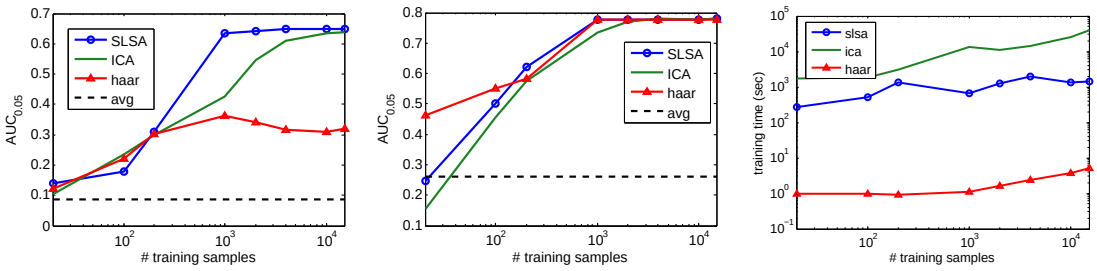
Evaluation metrics and goals. We adopt two metrics in the evaluation of detection performance:

- AUC_f: measures the area-under-curve (AUC) in the Receiver Operating Characteristic (ROC) curve only for false positive rates between 0 and $f, f \leq 1$. The integral AUC_f takes values in $[0, f]$ and is normalized to 1 for simplicity. E.g., $AUC_{0.05} = 0.8$ indicates that the detection performance reaches 80% of the optimal performance under the false positive constraint of 5 false alarms every 100 tests.
- *Detection time*: the time it takes for the test statistics to exceed a threshold that is selected to satisfy a certain system false positive requirement. Rapid and reliable detection is a key requirement for many time sensitive applications. For example, in earthquake response sub-seconds improvement in detection time can allow utility companies to shut down large transformers that are responsible for long and costly recovering period post a major earthquake.

5.8.1 Synthetic Data

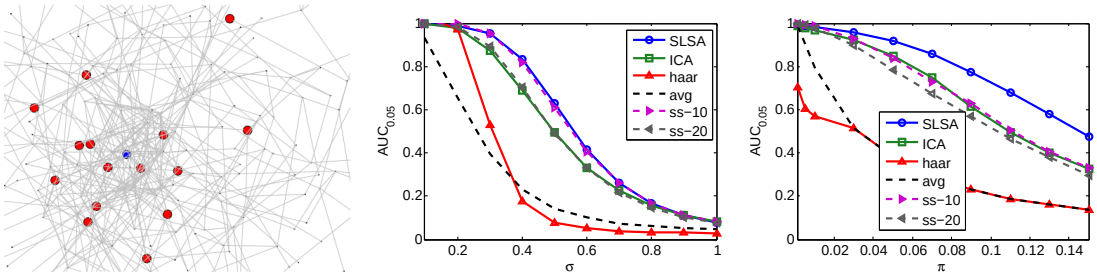
We generate samples from the latent tree model for network activation as described in Sec. 5.4.2. The tree contains $p = 1296$ leaf nodes with degree $d = 6$ and depth $L = 4$. We choose the sparsifying parameters $\alpha = 0.5$ and $\beta = 0.95$ so that that the expected number of total activations $||\mathbf{x}||_0 < \sqrt{p}$ is sparse. Of the three bases, `haar` is constructed from the known tree model whereas `ICA` and `SLSA` are trained with 20,000 samples drawn from the model. The bases are tested on 20,000 separate samples corrupted with Gaussian or binary channel noise. For the Gaussian noise case, the range of σ is chosen to satisfy the weak signal constraint, i.e., $\sigma > \frac{1}{\sqrt{2 \log p}} = 0.2641$.

Fig. 5.11 shows that all three bases outperform the naive baselines under both Gaussian and binary noise. Note that, perhaps surprisingly, both the learned `ICA` and `SLSA` outperform `haar` even though the latter is constructed from the known latent tree model.



(a) Gaussian noise, $\sigma = 0.5$ (b) Binary noise, $\pi = 0.04$ (c) Training time

Figure 5.13: Detection performance as a function of of training data size. (a)(b) shows it only takes approximately 2,000 samples for both `ICA` and `SLSA` to achieve the same performance as using all 20,000 samples. `SLSA` is 10 times faster to train than `ICA` as shown in (c).



(a) Sample network activation (b) Gaussian noise (c) Binary noise

Figure 5.14: Experiment with Gnutella-P2P network. (a) visualizes $\sim 1/10$ of the total network with a sample activation pattern colored. Blue: first infected node, Red: nodes subsequently infected through the cascade. (b)(c) shows that the learned bases achieve and exceed the state of the art algorithms that use additional prior knowledge of the network.

Next we study how the network size and the number of training samples affect the quality of learned basis and detection performance.

Increasing network size. We perform basis learning with subsets of the network, using $p = [36, 72, 108, 216,$
$432, 648, 864, 1080, 1296]$ sensors and $n = 20,000$ training samples. Fig. 5.12 shows that the detection performance of the learned bases grows more than 5x faster than the baseline. Note that `haar` is now learned from data; this accounts for the slight inferior performance compared to that in Fig. 5.11.

Increasing number of training samples. With the network size fixed, we evaluate bases learned from increasing numbers of training samples $n = [20, 100, 200, 1000,$
$2000, 4000, 10000, 15000]$. Fig. 5.13 shows that `haar` outperforms at smaller training size since it assumes a simple hierarchical structure. It also shows that it takes only 2,000 samples for `ICA` and `SLSA` to achieve the same detection performance as using all 20,000 samples.

5.8.2 Gnutella P2P network data

Our next set of experiments simulate virus outbreaks on a peer-to-peer network. We obtain a snapshot of the Gnutella P2P file sharing network[2] through the Stanford Network Analysis Project (SNAP). 1,769 nodes of the highest degree of connectivity were selected from this network for the experiment. Fig. 5.14(a) visualizes part of this sub network. We simulate 40,000 outbreak events – "*cascades*" – that mimic virus outbreaks on this directed network. We adopt the independent cascade model, where a starting node is picked at random, and whenever a node r is infected, a connected node w is infected with decreasing probability as a function of distance to r,

$$\mathbb{P}\left[x_i = 1\right] = \max\{0, a - dist(i, r) \times b\}$$

where r is the first infected node randomly chosen among all nodes $i = 1, \ldots, p$ and $dist(\dot{})$ is the number of hops between node i and r.

Here, `haar` is constructed as spanning tree wavelet basis, using the known network structure [69] and Wilson's uniform spanning tree (UST) sampling method on a directed graph via random walk [105]. We also apply the subset scan baseline `SS`-k [86] for reference. The parameter k is "optimally" selected based on the prior knowledge that on average between 10 and 30 nodes are activated in each event in the cascade model.

For comparison with *scan statistics*, we implement the subset scan with spatial "k-fixed neighbors" constraints as described in [86]. Each subset i consists of node V_i and its k nearest neighbors. The detection statistics takes the subset with the highest aggregated value and compares that to a threshold. In general, k depends on event structure and intensity that can't be determined a priori. However in this experiment k is selected based on the prior knowledge of the cascade model that on average there are between 10 and 30 nodes activated; in other words, this is the optimal subset scan results.

Fig. 5.14(b) and Fig. 5.14(c) compare the detection performance evaluated on 40,000 testing samples. Both `SLSA` and `ICA` demonstrate superior detection performance compared to the state of the art algorithms that use additional prior knowledge of the network.

5.8.3 Japan seismic network data

Next we turn to perhaps one of the most robust and long-running sensor networks in the world – the Japan seismic network. We obtain 48-hour, 150 GB of recordings from 721 *Hi-net* NIED seismometers for the dates March 18 and 19, 2011, just one week after the Tohoku M9.0 earthquake on March 11, 2011. On both days, 1,000+ events ranging from M1.0 - M6.0 were recorded in the the Japan Meteorology Agency catalog.[3]Many events triggered clustered activations as observed in

[2] http://snap.stanford.edu/data/p2p-Gnutella05.html

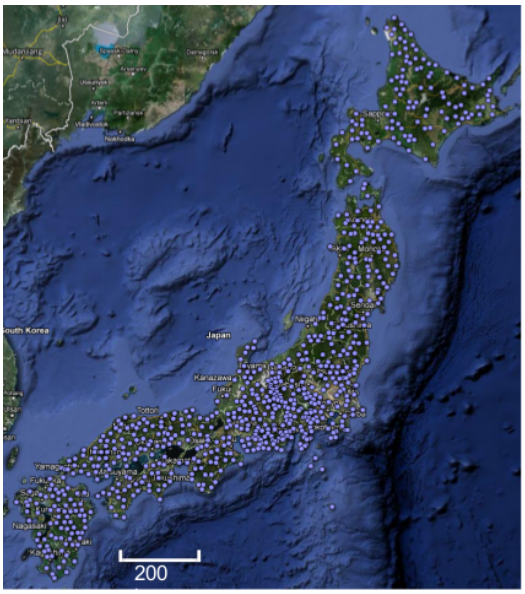


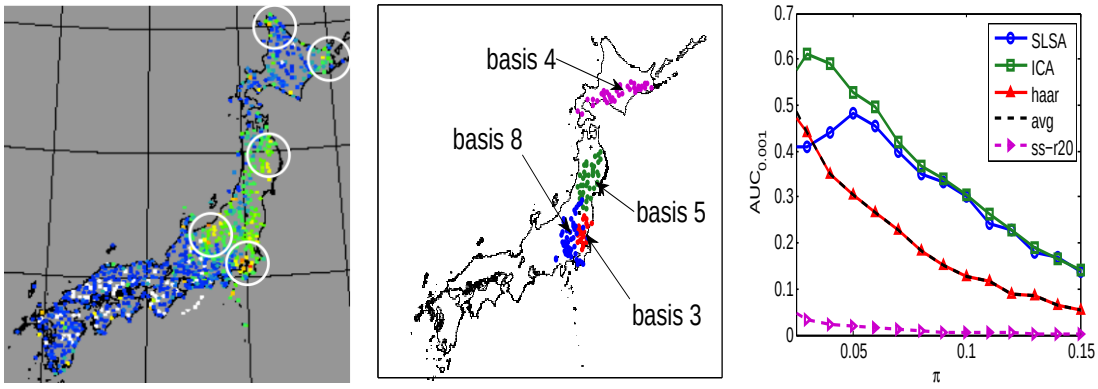
Figure 5.15: There are approximately 1,000 ($\approx$ 1 per 20 km^2) seismic stations that observe and record on average 500+ seismic events that occur in Japan each day. Among those 1,000 stations, $\approx$ 800 are *Hi-net* stations, pictured here. These research-grade sensors are professionally installed and maintained and therefore experience very little instrument or ambient noise.

Fig. 5.16(a).

For all 1795 events recorded on March 18, 2011, 10 snapshots of network activations at a two-second period were taken after the first detection at each event to construct the training data set of [p x n] = [721 x 17950]. The learned bases are tested on the first one-second data of the 1324 events recorded on March 19, 2011. We added binary noise of different error rate to control the problem complexity.

For the comparison with the `SS`-r baseline, the aggregation distance r is selected to be 20km which is roughly the distance covered by the seismic waves in a 2-second period. Fig. 5.16(c) presents the performance in detecting within two seconds of event arrival under a very small false positive constraint of 0.001. Of the three learned bases, both `ICA` and `SLSA` show significant gain in detection power, whereas `haar` has no improvement over the `avg` baseline. Perhaps surprisingly `SS`-r20 performs very poorly in comparison. An explanation is that most of the events during this period originated from the ocean and affects an array of stations along the coast. However, this pattern is not captured by the fixed radius subset scan construction. This explanation is supported by the plot of four prominent basis elements from `ICA` in Fig. 5.16(b). This example demonstrates the limited detection capability of subset scan for unknown patterns and the power of learning-based detection algorithms such as `ICA` and `SLSA`.

[3] http://www.hinet.bosai.go.jp/REGS/JMA/?LANG=en



(a) Localized activation patterns (b) 4 prominent ICA basis (c) Detection performance

Figure 5.16: (a) Japan's seismic network frequently exhibit localized activation patterns (circled). Colors indicate raw accelerations (red: large shaking, blue: no shaking). The learned bases are able to capture these nonuniform patterns with basis elements such as the ones in (b) and show 2x better detection performance compared to the baselines (c), while algorithms with hard-coded patterns such as SS-r20 fail to perform well in this scenario.

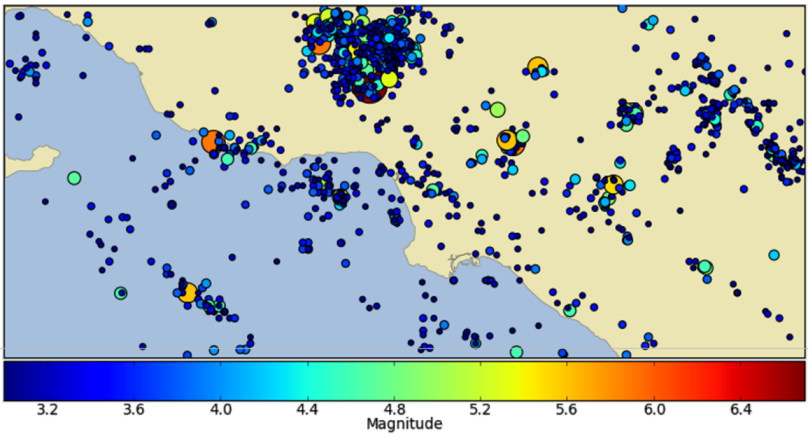
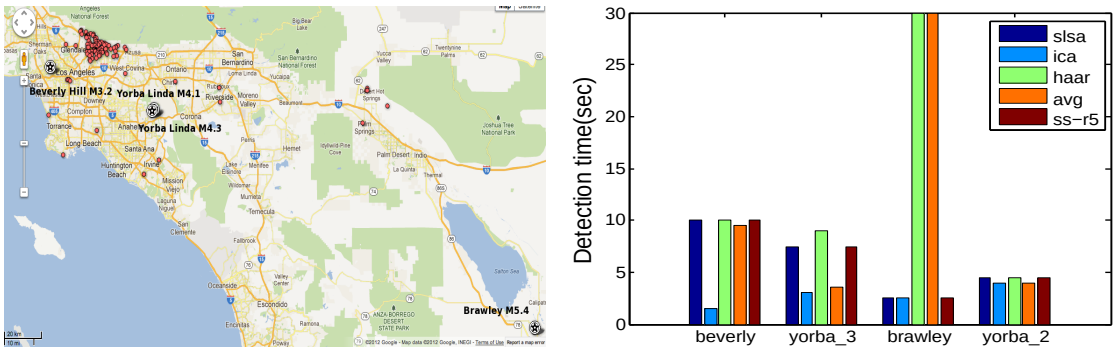


Figure 5.17: Locations and magnitudes of Southern California quakes since 1973.

5.8.4 Dense and participatory seismic networks

Lastly, we consider two dense, real-world seismic networks in Southern California. We show that good bases can be learned without historical sensor data: instead, we simply use basic earthquake simulators to generate the binary activation patterns for training, as discussed in Sec. 5.6.1. In the shortage of testing data – only a small number of events have been recorded by these networks, not enough to reliably compute AUC scores – the detection performance is evaluated in terms of detection time with detection thresholds computed as described in Sec. 5.6.1. This measure of time is critical in many applications; seconds or sub-second savings may enable automated responses that prevent huge loss of capital and lives.

[3] http://earthquake.usgs.gov/earthquakes/eqarchives/epic/



(a) Sensors (red) and events (starred)

(b) Detection time comparison

Figure 5.18: CSN network. (a) plots the layout of 128 sensors and epicenter of 4 recorded events. (b) The learned bases detect on average several seconds faster than the baselines under the constraint of at most one false alarm a year.

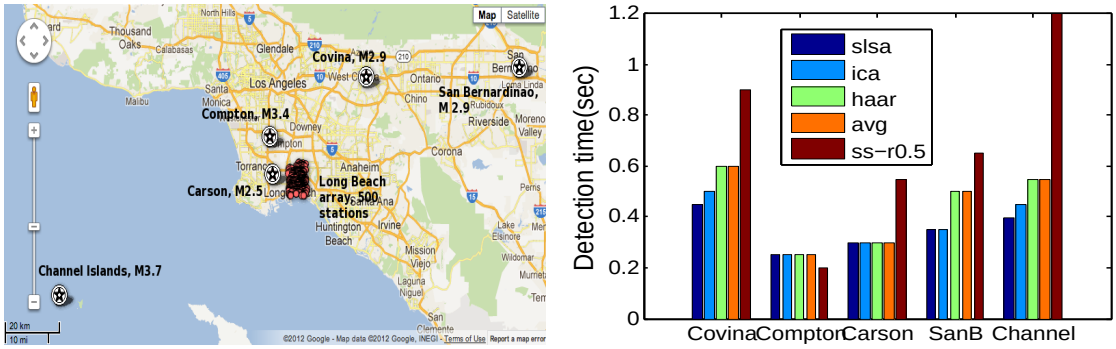
Generating training data. We generate training data using a basic earthquake simulator in the following two steps. First we randomly sample an earthquake from a prior distribution of seismic events in Southern California that is constructed from a list of historic earthquakes (Fig. 5.17) available in the USGS database.[4]Then time sequences of sensor activations are generated from an earthquake model that computes the expected wave arrival time with the encoded speed of seismic waves and distance to the hypocenter. This model is simplistic compared to many state-of-the-art earthquake simulators, yet captures qualitative spatio-temporal dependencies. An activation probability similar to that in [76] is used to simulate signal attenuation for unreliable noisy sensors.

Community Seismic Network. We simulate 1,000 network activation snapshots for 128 Community Seismic Network [22] sensors as described above. After training, each algorithm is then evaluated on its ability to detect four recent events using real measurements recorded by the network. Fig. 5.18(a) shows the spatial layout of the network and the hypocenters of the four events. Fig. 5.18(b) summarizes detection performance: the bases learned from simple simulations in general achieve faster detection than other algorithms, e.g., 8 seconds faster in detecting the Beverly Hills event. Note that `ICA` performs better than `SLSA`, as simulations are noise-free.

Long Beach Array.

The Long Beach network consists of approximately 5,000 sensors covering an area of 5 x 7 km. The network was deployed for 6 months during the first half of 2011 to provide detailed images of the Signal Hill Oil Field in Long Beach, California. These sensors are in the same performance regime as the USB accelerometers adopted in the CSN project and are constantly exposed to high level, time-varying noise. During the deployment period, a total number of 5 detectable earthquakes were recorded by the network (Fig. 5.19(a)). Fig. 1.3 is a visualization of one of the events.

We take a subset of 1,000 sensors and train the sparsifying bases with 2,000 simulated events.



(a) Sensors and events layout

(b) Detection time comparison

Figure 5.19: Long Beach array. (a) shows the layout of 1,000 stations and 5 recorded events. (b) Under the constraint of at most one false alarm a year, the learned bases detect on average 0.1 seconds faster than the baselines, which is significant considering it only takes 1 second for the seismic wave to travel through the network and only 0.5 seconds for the network to be saturated with signals.

The results in Fig. 5.19(b) show that the learned bases detect on average 0.1 second faster, especially for the more difficult events that are smaller and further away. This improvement in detection time is significant considering that it only takes about one second for the quake to travel through the entire network.

Chapter 6

Coresets for scalable and resource-constrained GMM learning

Motivated by a desired to probabilistically model the streams of data produced by sensor-equipped devices, this chapter presents algorithms and results for training mixtures of Gaussians or related distributions on massive data sets using the concept of a *coreset*. Previously, coresets have been designed to provide approximation guarantees for geometric problems such as K-means clustering. It turns out that much of that geometric machinery can be lifted to the statistical realm to provide approximation guarantees for statistical problems, as well. Despite the fact that Gaussian mixture models do not admit finite sufficient statistics, this chapter explains how learning a "good enough" mixture model only requires retaining a finite (weighted) subset of the input data. Leveraging earlier coreset results, this allows estimating GMMs in both the streaming and parallel setting, which are both highly relevant for learning about sensor data stored across a network of client devices. The main contribution of this chapter is experimental evaluation of coresets on several applications, including accelerometer data from CSN. The results here first appeared in [44].

An introduction to Coresets. The concept of a coreset first developed for approximating geometric problems, and while there are several competing definitions of a coreset, they are typically a small set of points (possibly weighted, usually a subset of the input) with the property that models fitting the coreset will also provide a good fit for the original data set.

It is helpful to think about coresets with respect to a specific task and a specific approximation guarantee. For example, consider the *k-means clustering problem*: given a set P of n points in $\mathbb{R}^d$, choose a set of k points in $\mathbb{R}^d$, called *centers*, such that the sum of squared distances from the points in P to their nearest center is minimized. Finding an optimal set of centers is NP-hard, and so we may instead seek a set of centers that provide a $(1+\epsilon)$ approximation to the sum of squared distances between P and the optimal centers. We say that a (weighted) set of points is a (k, ϵ)-coreset for the

k-means problem if the clustering cost of the coreset (a weighted sum of squared distances) given an arbitrary set of k centers is $(1 \pm \epsilon)$ of the clustering cost of the full input. This notation makes clear that the coreset depends on both the task (here, the number of centers k), and the desired approximation factor ϵ.

Coresets have been used for more problems than k-means, and have been developed for a wide class of *projective clustering* problems. In a projective clustering problem, we have an input set P and we want to approximate it using a collection of subspaces, called *centers*. We fix the number of subspaces to be k, and require each to be of dimension j. Let $x = (x_1, \ldots, x_k)$ be the collection of centers. Our goal is to choose the centers to minimize the *projective clustering error*, which is the distance from each point $p \in P$ to its nearest center:

$$\text{cost}(P, x) = \sum_{p \in P} dist(p, x) \tag{6.1}$$

In the simplest case, the centers are just points (affine subspaces with dimension $j = 0$), and we recover the k-means or k-medians problem by choice of distance function

$$dist(p, x) = \min_{x_i \in x} ||p - x_i||_1^2$$

In the k-means problem, ℓ_2 distance is used instead:

$$dist(p, x) = \min_{x_i \in x} ||p - x_i||_2^2$$

TODO: transition from these projective clustering objectives to the GMM objective

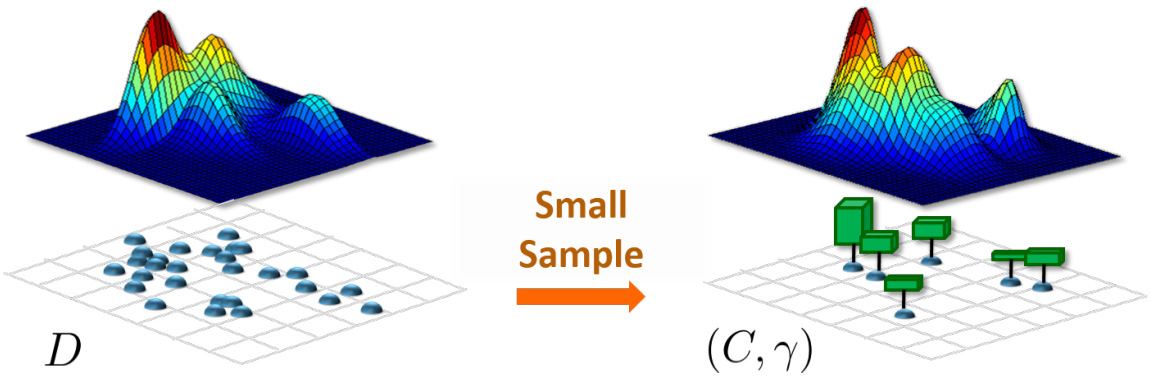


Figure 6.1: A coreset for mixture models is a small, weighted subset of the data that provides approximation guarantees on the "goodness" of models learned from it.

Results. We show that, perhaps surprisingly, Gaussian mixtures admit coresets of size *independent* of the size of the data set. More precisely, we prove that a weighted set of $O(dk^3/\varepsilon^2)$ data points

suffices for computing a $(1+\varepsilon)$-approximation for the optimal model on the original n data points. Moreover, such coresets can be efficiently constructed in a map-reduce style computation, as well as in a streaming setting. Our results rely on a novel reduction of statistical estimation to problems in computational geometry, as well as new complexity results about mixtures of Gaussians. We empirically evaluate our algorithms on several real data sets, including a density estimation problem in the context of earthquake detection using accelerometers in mobile phones.

We consider the problem of training statistical mixture models, in particular mixtures of Gaussians and some natural generalizations, on massive data sets. Such data sets may be distributed across a cluster, or arrive in a data stream, and have to be processed with limited memory. In contrast to parameter estimation for models with compact sufficient statistics, mixture models generally require inference over latent variables, which in turn depends on the full data set. We show that Gaussian mixture models (GMMs), and some generalizations, admit small *coresets*: A coreset is a weighted subset of the data which guarantees that models fitting the coreset will also provide a good fit for the original data set. Perhaps surprisingly, we show that Gaussian mixtures admit coresets of size *independent* of the size of the data set.

We focus on ε-semi-spherical Gaussians, where the covariance matrix Σ_i of each component i has eigenvalues bounded in $[\varepsilon, 1/\varepsilon]$, but some of our results generalize even to the semi-definite case. In particular, we show that given a data set D of n points in $\mathbb{R}^d$, $\varepsilon > 0$ and $k \in \mathbb{N}$, how one can efficiently construct a weighted set C of $\mathcal{O}(dk^3/\varepsilon^2)$ points, such that for any mixture of k ε-semi-spherical Gaussians $\theta = [(w_1, \mu_1, \Sigma_1), \ldots, (w_k, \mu_k, \Sigma_k)]$ it holds that the log-likelihood $\ln P(D \mid \theta)$ of D under θ is approximated by the (properly weighted) log-likelihood $\ln P(C \mid \theta)$ of C under θ to arbitrary accuracy as $\varepsilon \to 0$. Thus solving the estimation problem on the coreset C (e.g., using weighted variants of the EM algorithm, see Section 6.2.3) is almost as good as solving the estimation problem on large data set D. Our algorithm for constructing C is based on adaptively sampling points from D and is simple to implement. Moreover, coresets can be efficiently constructed in a map-reduce style computation, as well as in a streaming setting (using space and update time per point of $\operatorname{poly}(dk\varepsilon^{-1} \log n \log(1/\delta))$).

Existence and construction of coresets have been investigated for a number of problems in computational geometry (such as k-means and k-median) in many recent papers (*cf.*, surveys in [3, 27]). Here, we demonstrate how these techniques from computational geometry can be lifted to the realm of statistical estimation. As a by-product of our analysis, we also close an open question on the VC dimension of arbitrary mixtures of Gaussians.

We evaluate our algorithms on several synthetic and real data sets. In particular, we use our approach for density estimation for acceleration data, motivated by an application in earthquake detection using mobile phones.

6.1 Background and Problem Statement

Fitting mixture models by MLE. Suppose we are given a data set $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq \mathbb{R}^d$. We consider fitting a mixture of Gaussians $\theta = [(w_1, \mu_1, \Sigma_1), \ldots, (w_k, \mu_k, \Sigma_k)]$, i.e., the distribution $P(\mathbf{x} \mid \theta) = \sum_{i=1}^{k} w_i \mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i)$, where $w_1, \ldots, w_k \geq 0$ are the mixture weights, $\sum_i w_i = 1$, and μ_i and Σ_i are mean and covariance of the i-th mixture component, which is modeled as a multivariate normal distribution $\mathcal{N}(\mathbf{x}, \mu_i, \Sigma_i) = \frac{1}{\sqrt{|2\pi\Sigma_i|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i)\right)$. In Section 6.3, we will discuss extensions to more general mixture models. Assuming the data was generated i.i.d., the negative log likelihood of the data is $\mathcal{L}(D \mid \theta) = -\sum_j \ln P(\mathbf{x}_j \mid \theta)$, and we wish to obtain the maximum likelihood estimate (MLE) of the parameters $\theta^* = \arg\min_{\theta \in \mathfrak{C}} \mathcal{L}(D \mid \theta)$, where $\mathfrak{C}$ is a set of constraints ensuring that degenerate solutions are avoided[1]. Hereby, for a symmetric matrix $\mathbf{A}$, $\operatorname{spec} \mathbf{A}$ is the set of all eigenvalues of $\mathbf{A}$. We define

$$\mathfrak{C} = \mathfrak{C}_\varepsilon = \{\theta = [(w_1, \mu_1, \Sigma_1), \ldots, (w_k, \mu_k, \Sigma_k)] \mid \forall_i : \operatorname{spec}(\Sigma_i) \subseteq [\varepsilon, 1/\varepsilon]\}$$

to be the set of all mixtures of k Gaussians θ, such that all the eigenvalues of the covariance matrices of θ are bounded between ε and $1/\varepsilon$ for some small $\varepsilon > 0$.

Approximating the log-likelihood. Our goal is to approximate the data set D by a weighted set $C = \{(\gamma_1, \mathbf{x}'_1), \ldots, (\gamma_m, \mathbf{x}'_m)\} \subseteq \mathbb{R} \times \mathbb{R}^d$, such that $\mathcal{L}(D \mid \theta) \approx \mathcal{L}(C \mid \theta)$ for all θ, where we define $\mathcal{L}(C \mid \theta) = -\sum_i \gamma_i \ln P(\mathbf{x}'_i \mid \theta)$.

What kind of approximation accuracy may we hope to expect? Notice that there is a nontrivial issue of scale: Suppose we have a MLE θ^* for D, and let $\alpha > 0$. Then straightforward linear algebra shows that we can obtain an MLE θ^*_α for a scaled data set $\alpha D = \{\alpha \mathbf{x} : \mathbf{x} \in D\}$ by simply scaling all means by α, and covariance matrices by α^2. For the log-likelihood, however, it holds that $\mathcal{L}(\alpha D \mid \theta^*_\alpha) = d \ln \alpha + \mathcal{L}(D \mid \theta^*)$. Therefore, optimal solutions on one scale can be efficiently transformed to optimal solutions at a different scale, while maintaining the same *additive error*. This means, that any algorithm which achieves absolute error ε at any scale could be used to achieve parameter estimates (for means, covariances) with *arbitrarily* small error, simply by applying the algorithm to a scaled data set and transforming back the obtained solution. Thus, we cannot expect to approximate $\mathcal{L}(D \mid \theta)$ to additive error ε, since by rescaling, the error in the parameters (means, covariances) could be made arbitrarily small in relation to $\mathcal{L}(D \mid \theta)$.

An alternative, scale-invariant approach may be to strive towards approximating $\mathcal{L}(D \mid \theta)$ up to *multiplicative error* $(1 + \varepsilon)$. Unfortunately, this goal is also hard to achieve: Choosing a scaling parameter α such that $d \ln \alpha + \mathcal{L}(D \mid \theta^*) = 0$ would require any algorithm that achieves any bounded multiplicative error to essentially incur *no error at all* when evaluating $\mathcal{L}(\alpha D \mid \theta^*)$. The

[1] equivalently, $\mathfrak{C}$ can be interpreted as prior thresholding.

above observations hold even for the case $k = 1$ and $\Sigma = I$, where the mixture θ consists of a single Gaussian, and the log-likelihood is the sum of squared distances to a point μ and an additive term.

Motivated by the scaling issues discussed above, we use the following error bound that was suggested in [8] (who studied the case where all Gaussians are identical spheres). We decompose the negative log-likelihood $\mathcal{L}(D \mid \theta)$ of a data set D as

$$\mathcal{L}(D \mid \theta) = -\sum_{j=1}^{n} \ln \sum_{i=1}^{k} \frac{w_i}{\sqrt{|2\pi\Sigma_i|}} \exp\left(-\frac{1}{2}(\mathbf{x}_j - \mu_i)^T \Sigma_i^{-1} (\mathbf{x}_j - \mu_i)\right) = -n \ln Z(\theta) + \phi(D \mid \theta)$$

where $Z(\theta) = \sum_i \frac{w_i}{\sqrt{|2\pi\Sigma_i|}}$ is a normalizer, and the function ϕ is defined as

$$\phi(D \mid \theta) = -\sum_{j=1}^{n} \ln \sum_{i=1}^{k} \frac{w_i}{Z(\theta)\sqrt{|2\pi\Sigma_i|}} \exp\left(-\frac{1}{2}(\mathbf{x}_j - \mu_i)^T \Sigma_i^{-1} (\mathbf{x}_j - \mu_i)\right).$$

Hereby, $Z(\theta)$ plays the role of a normalizer, which can be computed *exactly*, independently of the set D. $\phi(D \mid \theta)$ captures all dependencies of $\mathcal{L}(D \mid \theta)$ on D, and via Jensen's inequality, it can be seen that $\phi(D \mid \theta)$ is always nonnegative.

We can now use this term $\phi(D \mid \theta)$ as a reference for our error bounds. In particular, we call $\tilde{\theta}$ a $(1+\varepsilon)$-approximation for θ if $(1-\varepsilon)\phi(D \mid \theta) \leq \phi(D \mid \tilde{\theta}) \leq \phi(D \mid \theta)(1+\varepsilon)$.

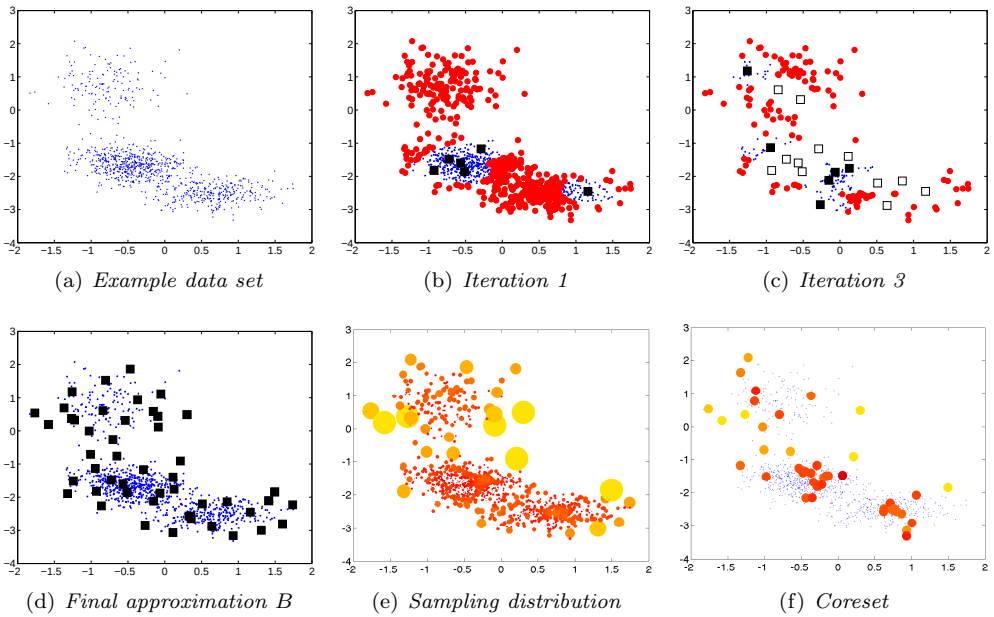
Coresets. We call a weighted data set C a (k, ε)-*coreset* for another (possibly weighted) set $D \subseteq \mathbb{R}^d$, if for all mixtures $\theta \in \mathfrak{C}$ of k Gaussians it holds that

$$(1-\varepsilon)\phi(D \mid \theta) \leq \phi(C \mid \theta) \leq \phi(D \mid \theta)(1+\varepsilon).$$

Hereby $\phi(C \mid \theta)$ is generalized to weighted data sets C in the natural way (weighing the contribution of each summand $\mathbf{x}'_j \in C$ by γ_j). Thus, as $\varepsilon \to 0$, for a sequence of (k, ε)-coresets C_ε we have that $\sup_{\theta \in \mathfrak{C}} |\mathcal{L}(C_\varepsilon \mid \theta) - \mathcal{L}(D \mid \theta)| \to 0$, i.e., $\mathcal{L}(C_\varepsilon \mid \theta)$ uniformly (over $\theta \in \mathfrak{C}$) approximates $\mathcal{L}(D \mid \theta)$. Further, under the additional condition that all variances are sufficiently large (formally $\prod_{\lambda \in \mathrm{spec}(\Sigma_i)} \lambda \geq \frac{1}{(2\pi)^d}$ for all components i), the log-normalizer $\ln Z(\theta)$ is negative, and consequently the coreset in fact provides a multiplicative $(1+\varepsilon)$ approximation to the log-likelihood, i.e.,

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

Note that if we had access to a (k, ε)-coreset C, then we could reduce the problem of fitting a mixture model on D to one of fitting a model on C, since the optimal solution θ_C is a good approximation (in terms of log-likelihood) of θ^*. While finding the optimal θ_C is a difficult problem, one can use a (weighted) variant of the EM algorithm to find a good solution. Moreover, if $|C| \ll |D|$,



(a) *Example data set* (b) *Iteration 1* (c) *Iteration 3*

(d) *Final approximation B* (e) *Sampling distribution* (f) *Coreset*

Figure 6.2: Illustration of the coreset construction for example data set (a). (b,c) show two iterations of constructing the set B. Solid squares are points sampled uniformly from remaining points, hollow squares are points selected in previous iterations. Red color indicates half the points furthest away from B, which are kept for next iteration. (d) final approximate clustering B on top of original data set. (e) Induced non-uniform sampling distribution: radius of circles indicates probability; color indicates weight, ranging from red (high weight) to yellow (low weight). (f) Coreset sampled from distribution in (e).

running EM on C may be orders of magnitude faster than solving it on D. In Section 6.2.3, we give more details about solving the density estimation problem on the coreset.

The key question is whether small (k, ε)-coresets exist, and whether they can be efficiently constructed. In the following, we answer this question affirmatively. We show that, perhaps surprisingly, one can efficiently find coresets C of size *independent* of the size n of D, and with polynomial dependence on $\frac{1}{\varepsilon}$, d and k.

6.2 Efficient Coreset Construction via Adaptive Sampling

Naive approach: uniform sampling. A naive approach towards approximating D would be to just pick a subset C uniformly at random. In particular, suppose the data set is generated from a mixture of two spherical Gaussians ($\Sigma_i = \mathbf{I}$) with weights $w_1 = \frac{1}{\sqrt{n}}$ and $w_2 = 1 - \frac{1}{\sqrt{n}}$. Unless $m = \Omega(\sqrt{n})$ points are sampled, with constant probability no data point generated from Gaussian 2 is selected. By moving the means of the Gaussians arbitrarily far apart, $\mathcal{L}(D \mid \theta_C)$ can be made arbitrarily worse than $\mathcal{L}(D \mid \theta_D)$, where θ_C and θ_D are MLEs on C and D respectively. Thus, even for two well-separated Gaussians, uniform sampling can perform arbitrarily poorly. This example

already suggests that, intuitively, in order to achieve small multiplicative error, we must devise a sampling scheme that adaptively selects representative points from all "clusters" present in the data set. However, this suggests that obtaining a coreset requires solving a chicken-and-egg problem, where we need to understand the density of the data to obtain the coreset, but simultaneously would like to use the coreset for density estimation.

Better approximation via adaptive sampling. The key idea behind the coreset construction is that we can break the chicken-and-egg problem by first obtaining a rough approximation B of the clustering solution (using more than k components, but far fewer than n), and then to use this solution to bias the random sampling. Surprisingly, a simple procedure which iteratively samples a small number β of points, and removes half of the data set closest to the sampled points, provides a sufficiently accurate first approximation B for this purpose. This initial clustering is then used to sample the data points comprising coreset C according to probabilities which are roughly proportional to the squared distance to the set B. This non-uniform random sampling can be understood as an importance-weighted estimate of the log-likelihood $\mathcal{L}(D \mid \theta)$, where the weights are optimized in order to reduce the variance. The same general idea has been found successful in constructing coresets for geometric clustering problems such as k-means and k-median [42]. The pseudocode for obtaining the approximation B, and for using it to obtain coreset C is given in Algorithm 4.

Algorithm 4: Coreset construction

Input: Data set D, ε, δ, k
Output: Coreset $C = \{(\gamma(\mathbf{x}_1), \mathbf{x}_1), \ldots, (\gamma(\mathbf{x}_{|C|}), \mathbf{x}_{|C|})\}$
$D' \leftarrow D$; $B \leftarrow \emptyset$;
while $|D'| > 10dk \ln(1/\delta)$ **do**
  Sample set S of $\beta = 10dk \ln(1/\delta)$ points uniformly at random from D';
  Remove $\lceil |D'|/2 \rceil$ points $\mathbf{x} \in D'$ closest to S (i.e., minimizing $\mathrm{dist}(\mathbf{x}, S)$) from D';
  Set $B \leftarrow B \cup S$;
Set $B \leftarrow B \cup D'$;
for *each* $b \in B$ **do** $D_b \leftarrow$ the points in D whose closest point in B is b. Ties broken arbitrarily;
;
for *each* $b \in B$ *and* $\mathbf{x} \in D_b$ **do**
  $m(\mathbf{x}) \leftarrow \left\lceil \frac{5}{|D_b|} + \frac{\mathrm{dist}(\mathbf{x}, B)^2}{\sum_{\mathbf{x}' \in D} \mathrm{dist}(\mathbf{x}', B)^2} \right\rceil$;
Pick a non-uniform random sample C of $10\lceil dk|B|^2 \ln(1/\delta)/\varepsilon^2 \rceil$ points from D, where for every $\mathbf{x}' \in C$ and $\mathbf{x} \in D$, we have $\mathbf{x}' = \mathbf{x}$ with probability $m(\mathbf{x}) / \sum_{\mathbf{x}' \in D} m(\mathbf{x}')$;
for *each* $\mathbf{x}' \in C$ **do** $\gamma(\mathbf{x}') \leftarrow \frac{\sum_{\mathbf{x} \in D} m(\mathbf{x})}{|C| \cdot m(\mathbf{x}')}$;

We have the following result, proved in the supplemental material:

Theorem 6.2.1. *Suppose C is sampled from D using Algorithm 4 for parameters ε, δ and k. Then,*

with probability at least $1-\delta$ *it holds that for all* $\theta \in \mathfrak{C}_\varepsilon$,

$$\phi(D \mid \theta)(1-\varepsilon) \leq \phi(C \mid \theta) \leq \phi(D \mid \theta)(1+\varepsilon).$$

In our experiments, we compare the performance of clustering on coresets constructed via adaptive sampling, vs. clustering on a uniform sample. The size of C in Algorithm 4 depends on $|B|^2 = \log^2 n$. By replacing B in the algorithm with a constant factor approximation B', $|B'| = l$ for the k-means problem, we can get a coreset C of size *independent* of n. Such a set B' can be computed in $O(ndk)$ time either by applying exhaustive search on the output C of the original Algorithm 4 or by using one of the existing constant-factor approximation algorithms for k-means (say, [56]).

6.2.1 Sketch of Analysis: Reduction to Euclidean Spaces

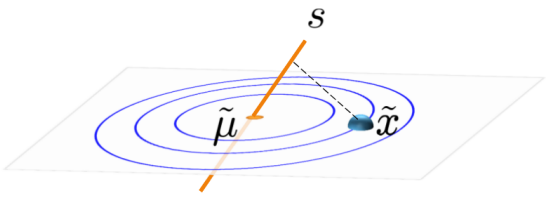


Figure 6.3: Level sets of the distances between points on a plane (green) and (disjoint) k-dimensional subspaces are ellipses, and thus can represent contour lines of the multivariate Gaussian.

The key insight in the proof is that the contribution $\log P(\mathbf{x} \mid \theta)$ to the likelihood $\mathcal{L}(D \mid \theta)$ can be expressed in the following way:

Lemma 3. *There exist functions* ϕ, ψ, *and* f *such that, for any point* $\mathbf{x} \in \mathbb{R}^d$ *and mixture model* θ, $\ln P(\mathbf{x} \mid \theta) = -f_{\phi(\mathbf{x})}(\psi(\theta)) + Z(\theta)$, *where*

$$f_{\tilde{\mathbf{x}}}(y) = -\ln \sum_i \tilde{w}_i \exp\left(-W_i \mathrm{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, \mathbf{s}_i)^2\right).$$

Hereby, ϕ *is a function that maps a point* $\mathbf{x} \in \mathbb{R}^d$ *into* $\tilde{\mathbf{x}} = \phi(\mathbf{x}) \in \mathbb{R}^{2d}$, *and* ψ *is a function that maps a mixture model* θ *into a tuple* $y = (s, w, \tilde{\mu}, W)$ *where* w *is a* k*-tuple of nonnegative weights* $\tilde{w}_1, \ldots, \tilde{w}_k$ *summing to 1,* $s = \mathbf{s}_1, \ldots, \mathbf{s}_k \subseteq \mathbb{R}^{2d}$ *is a set of* k d*-dimensional subspaces that are weighted by weights* $W_1, \cdots, W_k > 0$, *and* $\tilde{\mu} = \tilde{\mu}_1, \cdots, \tilde{\mu}_k \in \mathbb{R}^{2d}$ *is a set of* k *means.*

The main idea behind Lemma 3 is that level sets of distances between points and subspaces are quadratic forms, and can thus represent level sets of the Gaussian probability density function (see Figure 6.3 for an illustration). We recognize the "soft-min" function $\wedge_{\mathbf{w}'}(\eta) \equiv -\ln \sum_i w'_i \exp(-\eta_i)$

as an approximation upper-bounding the minimum $\min(\eta) = \min_i \eta_i$ for $\eta_i = W_i \text{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, \mathbf{s}_i)^2$ and $\eta = [\eta_1, \ldots, \eta_k]$. The motivation behind this transformation is that it allows expressing the likelihood $P(\mathbf{x} \mid \theta)$ of a data point $\mathbf{x}$ given a model θ in a purely geometric manner as soft-min over distances between points and subspaces in a transformed space. Notice that if we use the minimum $\min()$ instead of the soft-min $\wedge_{\tilde{w}}()$, we recover the problem of approximating the data set D (transformed via ϕ) by k-subspaces. For semi-spherical Gaussians, it can be shown that the subspaces can be chosen as points while incurring a multiplicative error of at most $1/\varepsilon$, and thus we recover the well-known k-means problem in the transformed space. This insight suggests using a known coreset construction for k-means, adapted to the transformation employed. The remaining challenge in the proof is to bound the additional error incurred by using the soft-min function $\wedge_{\tilde{w}}(\cdot)$ instead of the minimum $\min(\cdot)$. We tackle this challenge by proving a generalized triangle inequality adapted to the exponential transformation, and employing the framework described in [42], which provides a general method for constructing coresets for clustering problems of the form $\min_{\mathbf{s}} \sum_i f_{\tilde{\mathbf{x}}}(\mathbf{s})$.

As proved in [42], the key quantity that controls the size of a coreset is the pseudo-dimension of the functions $F_d = \{f_{\tilde{\mathbf{x}}} \text{ for } \tilde{\mathbf{x}} \in \mathbb{R}^{2d}\}$. This notion of dimension is closely related to the VC dimension of the (sub-level sets of the) functions F_d and therefore represents the complexity of this set of functions. The final ingredient in the proof of Theorem 6.2.1 is a new bound on the complexity of mixtures of k Gaussians in d dimensions.

6.2.2 Streaming and Parallel Computation

One major advantage of coresets is that they can be constructed in parallel, as well as in a streaming setting where data points arrive one by one, and it is impossible to remember the entire data set due to memory constraints. The key insight is that coresets satisfy certain composition properties, which have previously been used by [57] for streaming and parallel construction of coresets for geometric clustering problems such as k-median and k-means.

1. Suppose C_1 is a (k, ε)-coreset for D_1, and C_2 is a (k, ε)-coreset for D_2. Then $C_1 \cup C_2$ is a (k, ε)-coreset for $D_1 \cup D_2$.

2. Suppose C is a (k, ε)-coreset for D, and C' is a (k, δ)-coreset for C. Then C' is a $(k, (1+\varepsilon)(1+\delta) - 1)$-coreset for D.

In the following, we review how to exploit these properties for parallel and streaming computation.

Streaming. In the streaming setting, we assume that points arrive one-by-one, but we do not have enough memory to remember the entire data set. Thus, we wish to maintain a coreset over time, while keeping only a small subset of $\mathcal{O}(\log n)$ coresets in memory. There is a general reduction that shows that a small coreset scheme to a given problem suffices to solve the corresponding problem

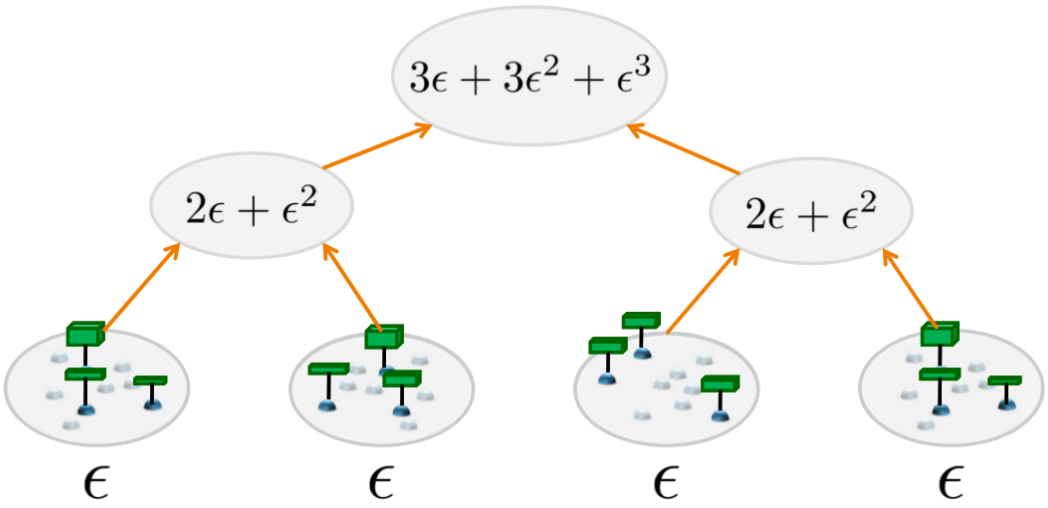


Figure 6.4: Tree construction for generating coresets in parallel or from data streams. The bottom ovals denote (k, ϵ)-coresets computed for disjoint sets of data. Each higher-level oval represents the result of a "merge-and-compress" operation, where a (k, ϵ)-coreset is computed from the union of the coresets beneath it. When performed in this way, the approximation error grows logarithmically with the number of "leaf" coresets.

on a streaming input [10, 57]. The idea is to construct and save in memory a coreset for every block of $\text{poly}(dk/\varepsilon)$ consecutive points arriving in a stream. When we have two coresets in memory, we can merge them (resulting in a (k, ε)-coreset via property (1)), and compress by computing a single coreset from the merged coresets (via property (2)) to avoid increase in the coreset size. An important subtlety arises: While merging two coresets (via property (1)) does not increase the approximation error, compressing a coreset (via property (2)) *does* increase the error. A naive approach that merges and compresses immediately as soon as two coresets have been constructed, can incur an exponential increase in approximation error. Fortunately, it is possible to organize the merge-and-compress operations in a binary tree of height $O(\log n)$, where we need to store in memory a single coreset for each level on the tree (thus requiring only $\text{poly}(dk\varepsilon^{-1} \log n)$ memory). Figure 5.7 illustrates this tree computation. In order to construct a coreset for the union of two (weighted) coresets, we use a weighted version of Algorithm 4, where we consider a weighted point as duplicate copies of a non-weighted point (possibly with fractional weight). A more formal description can be found in [43]. We summarize our streaming result in the following theorem.

Theorem 6.2.2. *A (k, ε)-coreset for a stream of n points in $\mathbb{R}^d$ can be computed for the ε-semi-spherical GMM problem with probability at least $1-\delta$ using space and update time* $\text{poly}(dk\varepsilon^{-1} \log n \log(1/\delta))$.

Parallel and distributed computations. Using the same ideas from the streaming model, a (non-parallel) coreset construction can be transformed into a parallel one. We partition the data

into sets, and compute coresets for each set, independently, on different computers in a cluster. We then (in parallel) merge (via property (1)) two coresets, and compute a single coreset for every pair of such coresets (via property (2)). Continuing in this manner yields a process that takes $\mathcal{O}(\log n)$ iterations of parallel computation. This computation is also naturally suited for map-reduce [32] style computations, where the map tasks compute coresets for disjoint parts of D, and the reduce tasks perform the merge-and-compress operations. Figure 5.7 illustrates this parallel construction.

We summarize our distributed computation result in the following theorem.

Theorem 6.2.3. *A (k, ε)-coreset for a set of n points in $\mathbb{R}^d$ can be computed for the ε-semi-spherical GMM problem with probability at least $1-\delta$ using m machines in time $(n/m)\cdot\text{poly}(dk\varepsilon^{-1}\log(1/\delta)\log n)$.*

6.2.3 Fitting a GMM on the Coreset using Weighted EM

One approach, which we employ in our experiments, is to use a natural generalization of the EM algorithm, which takes the coreset weights into account. We here describe the algorithm for the case of GMMs. For other mixture distributions, the E and M steps are modified appropriately.

Algorithm 5: Weighted EM for Gaussian mixtures

Input: Coreset C, k, TOL
Output: Mixture model θ_C
$\mathcal{L}_{old} = \infty$; Initialize means $\mu_1, \ldots, \mu_k$ by sampling k points from C with probability proportional to their weight. Initialize $\Sigma_i = I$ and $w_i = \frac{1}{k}$ for all i;
repeat
  $\mathcal{L}_{old} = \mathcal{L}(C \mid \theta)$;**for** $j = 1$ **to** n **do** **for** $i = 1$ **to** k **do** Compute $\eta_{i,j} = \gamma_i \frac{w_i \mathcal{N}(\mathbf{x}'_j; \mu_i, \Sigma_i)}{\sum_\ell w_\ell \mathcal{N}(\mathbf{x}'_j; \mu_\ell, \Sigma_\ell)}$;
  ;
  ;
  for $i = 1$ **to** k **do**
    $w_i \leftarrow w_i / \sum_\ell w_i$; $\mu_i \leftarrow \sum_j \eta_{i,j}\mathbf{x}'_j / \sum_j \eta_{i,j}$; $\Sigma_i \leftarrow \sum_j \eta_{i,j}\big(\mathbf{x}'_j - \mu_i\big)\big(\mathbf{x}'_j - \mu_i\big)^T / \sum_j \eta_{i,j}$;
until $\mathcal{L}(C \mid \theta) \geq \mathcal{L}_{old} - TOL$;

Using a similar analysis as for the standard EM algorithm, Algorithm 5 is guaranteed to converge, but only to a local optimum. However, since it is applied on a much smaller set, it can be initialized using multiple random restarts. In our experiments, we show that running weighted EM on the coreset typically leads to comparable performance as running EM on the full data set.

6.3 Extensions and Generalizations

We now show how the connection between estimating the parameters for mixture models and problems in computational geometry can be leveraged further. Our observations are based on the link between mixture of Gaussians and projective clustering (multiple subspace approximation) as shown in Lemma 3.

Generalizations to non-semi-spherical GMMs. For simplicity, we generalized the coreset construction for the k-means problem, which required assumptions that the Gaussians are ε-semi-spherical. However, several more complex coresets for projective clustering were suggested recently (*cf.*, [42]). Using the tools developed in this article, each such coreset implies a corresponding coreset for GMMs and generalizations. As an example, the coresets for approximating points by lines [41] implies that we can construct small coresets for GMMs even if the smallest singular value of one of the corresponding covariance matrices is zero.

Generalizations to ℓ_q distances and other norms. Our analysis is based on combinatorics (such as the complexity of sub-levelsets of GMMs) and probabilistic methods (non-uniform random sampling). Therefore, generalizations to other non-Euclidean distance functions, or error functions such as (non-squared) distances (mixture of Laplace distributions) is straightforward. The main property that we need is a generalization of the triangle inequality, as proved in the supplemental material. For example, replacing the squared distances by non-squared distances yields a coreset for mixture of Laplace distributions. The double triangle inequality $\|a-c\|^2 \leq 2(\|a-b\|+\|b-c\|^2)$ is replaced by Hölder's inequality, $\|a-c\|^2 \leq 2^{O(q)}\|a-b\|+2\|b-c\|^2$. Such a result is straightforward from our analysis, and we summarize it in the following theorem.

Theorem 6.3.1. *Let $q \geq 1$ be an integer. Consider Algorithm 4, where $\operatorname{dist}(\cdot,\cdot)^2$ is replaced by $\operatorname{dist}(\cdot,\cdot)^q$ and ε^2 is replaced by $\varepsilon^{O(q)}$. Suppose C is sampled from D using this updated version of Algorithm 4 for parameters ε, δ and k. Then, with prob. at least $1-\delta$ it holds that for all $\theta \in \mathfrak{C}_\varepsilon$,*

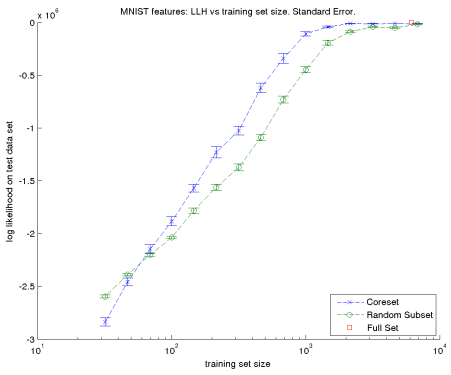
$$\phi(D \mid \theta)(1-\varepsilon) \leq \phi(C \mid \theta) \leq \phi(D \mid \theta)(1+\varepsilon),$$

where $Z(\theta) = \sum_i \frac{w_i}{g(\theta_i)}$ and $\phi(D \mid \theta) = -\sum_{\mathbf{x}\in D} \ln \sum_{i=1}^{k} \frac{w_i}{Z(\theta)g(\theta_i)} \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(\mathbf{x}-\mu_i)\right\|^q\right)$ using the normalizer $g(\theta_i) = \int \exp\left(-\frac{1}{2}\left\|\Sigma_i^{-1/2}(\mathbf{x}-\mu_i)\right\|^q\right) d\mathbf{x}$.

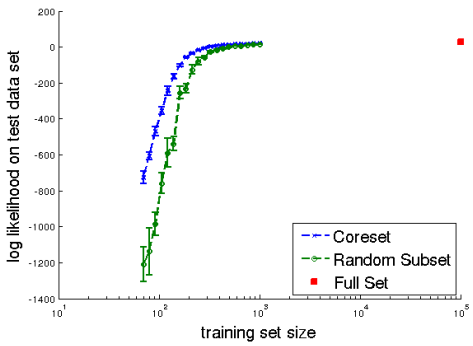
6.4 Experiments

We experimentally evaluate the effectiveness of using coresets of different sizes for training mixture models. We compare against running EM on the full set, as well as on an unweighted, uniform sample from D. Results are presented for three real datasets.

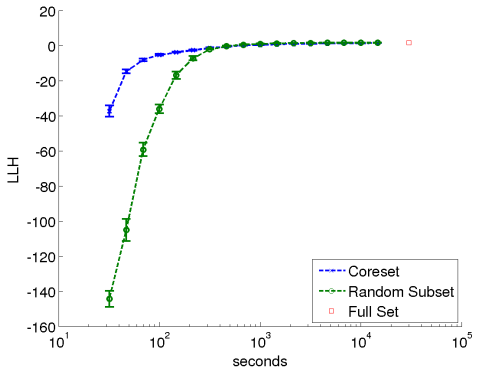
MNIST handwritten digits. The MNIST dataset contains 60,000 training and 10,000 testing grayscale images of handwritten digits. As in [53], we normalize each component of the data to have zero mean and unit variance, and then reduce each 784-pixel (28x28) image using PCA, retaining only the top $d = 100$ principal components as a feature vector. Using the digit labels, we construct training and testing sets with a skewed distribution over the digits, whereby digit i occurs in pro-



(a) *MNIST Skewed*



(b) *Tetrode recordings*



(c) *CSN data*

Figure 6.5: Experimental results for three real data sets. We compare likelihood of the best model obtained on subsets C constructed by uniform sampling, and by the adaptive coreset sampling procedure.

portion to 0.5^i. From the training set, we produce coresets and uniformly sampled subsets of sizes between 30 and 5000, using the parameters $k = 10$ (a cluster for each digit), $\beta = 20$ and $\delta = 0.1$ (see Algorithm 4), and fit GMMs using EM with 3 random restarts. The log likelihood (LLH) of each model on the testing data is shown in Figure 6.5(a).

Neural tetrode recordings. We also compare coresets and uniform sampling on a large dataset containing 319,209 records of rat hippocampal action potentials, measured by four co-located electrodes. As done by [53], we concatenate the 38-sample waveforms produced by each electrode to obtain a 152-dimensional vector. The vectors are normalized so each component has zero mean and unit variance. The 319,209 records are divided in half to obtain training and testing sets. From the training set, we produce coresets and uniformly sampled subsets of sizes between 70 and 1000, using the parameters $k = 33$ (as in [53]), $\beta = 66$, and $\delta = 0.1$, and fit GMMs. The log likelihood of each model on the held-out testing data is shown in Figure 6.5(b). Coreset GMMs obtain consistently higher LLH than uniform sample GMMs for sets of the same size.

CSN cell phone accelerometer data.

smartphones with accelerometers are being used by the Community Seismic Network (CSN) as inexpensive seismometers for earthquake detection. In [39], 7 GB of acceleration data were recorded from volunteers while carrying and operating their phone in normal conditions (walking, talking, on desk, etc.). From this data, 17-dimensional feature vectors were computed (containing frequency information, moments, etc.). The goal is to train, in an online fashion, GMMs based on normal data, which then can be used to perform anomaly detection to detect possible seismic activity. Motivated by the limited storage on smartphones, we evaluate coresets on a data set of 40,000 accelerometer feature vectors, using the parameters $k = 6$, $\beta = 12$, and $\delta = 0.1$. Figure 6.5(c) presents the results of this experiment. Notice that on this data set, coresets show an even larger improvement over uniform sampling. We hypothesize that this is due to the fact that the recorded accelerometer data is imbalanced, and contains clusters of vastly varying size, so uniform sampling does not represent smaller clusters well.

Chapter 7

Conclusion

This thesis has described theory and methodology for building CSR systems that provide real-time detection of spatial events using large numbers of community-owned sensor devices.

It presented the Caltech Community Seismic Network as an exemplar CSR system, and discussed how volunteer-operated sensor clients networked to cloud servers offers a practical and scalable architecture. The CSN project demonstrates how CSR systems can act on large-scale events by improving disaster response. CSN is also representative of the challenges faced by CSR systems, including the need to detect events in real-time, provide performance guarantees in the form of constrained false detections and estimates on detection performance, as well as the challenges inherent in using diverse, community hardware.

Interestingly, the CSN project was formed with the anticipation that sensor-equipped smart devices would play an increasing role in everyday life. This optimism has been confirmed, as mobile internet access outstripped traditional laptops and desktops. CSN-Droid and CrowdShake are two iterations of a free Android app that volunteers may use to join the CSN network. These apps share a foundation of anomaly detection and data collection, but differ in how they approach user engagement. The growing number of wearable sensor devices and other sensor hardware further confirms that trends are underway to extensively instrument our world with internet-connected consumer devices.

Decentralized Anomaly Detection. As the second main contribution, this thesis presents a decentralized anomaly detection approach for detecting rare, disruptive events using community-held sensors. Detecting rare events like dangerous quakes is made possible by learning 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.

Motivated by quake detection in large community seismic networks, the third contribution of the thesis is evidence that large numbers of noisy sensors can be leveraged to detect weak-but-structured events: constructing or learning a *sparsifying basis* enables detection of sparse event patterns in the decentralized setting. I provide theoretical bounds on the power of sparsification using a `haar` wavelet basis that emerges naturally from a hierarchical clustering of sensors, and obtained strong bounds on error rates for events produced by the latent tree model that can be evaluated for any network size. These results strengthen and complement previous work on the limits of detectability of sparse patterns in Gaussian noise.

We extend the intuition for the wavelet transform's success - its ability to concentrate signals with a small number of basis elements - and obtained a general framework to learn sparsifying bases for detection. We considered two optimizations, `ICA` and `SLSA`, for learning a basis that approximately maximizes sparsification, and explain how it can be implemented in sensor networks using real or simulated data in the absence of sufficient training data.

Finally, we thoroughly evaluate the detection performance of the sparsifying bases on several problem domains: simulated virus outbreaks on the Gnutella P2P network; detecting quakes following the Tohuku M9.0 event in the Japan seismic network with bases learned from network measurements; and detecting quakes recorded by the dense Long Beach and Community Seismic Network sensors using simulated measurements for training. In all domains, learned bases outperform previous state-of-the-art algorithms. We believe that our insights are an important step towards solving challenging detection problems using large-scale, noisy, participatory sensor networks.

The final contribution of the thesis is an evaluation of coresets for efficient learning of Gaussian Mixture Models. We have shown how to construct coresets for estimating parameters of GMMs and natural generalizations. Our construction hinges on a natural connection between statistical estimation and clustering problems in computational geometry. To our knowledge, our results provide the first rigorous guarantees for obtaining compressed ε-approximations of the log-likelihood of mixture models for large data sets. The coreset construction relies on an intuitive adaptive sampling scheme, and can be easily implemented. By exploiting certain closure properties of coresets, it is possible to efficiently construct them in both the parallel and streaming setting, which promises to allow client-level modeling of large sensor data sets on resource constrained devices, as demonstrated on CSN sensor data and other data sets.

Final thoughts.

This thesis outlined several algorithmic and systems principles that facilitate detecting rare and complex spatial signals using large numbers of low-cost community sensors. We have found that employing machine learning at each stage of a decentralized architecture allows efficient use of sensor-level and cloud-level resources, and is essential to providing performance guarantees when little can be said about a particular community sensor, or when little is known about the events we seek to detect.

Community sensing is applicable to a variety of application domains, including disasters like fires, floods, radiation, epidemics, and traffic accidents, as well as monitoring the pollution, pedestrian traffic, and acoustic noise levels in urban environments. In all of these cases "responding" may range from taking physical action to merely devoting additional resources to an event of interest. While the CSN project is motivated by detecting and reacting to strong earthquakes, we believe that community sense and response systems for these domains and others will require a similar blueprint of machine learning and scalable systems.

Bibliography

[1] Measuring shaking intensity with mobile phones, 3 2011.

[2] 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.

[3] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximations via coresets. *Combinatorial and Computational Geometry - MSRI Publications*, 52:1–30, 2005.

[4] M. Aharon, M. Elad, and A. Bruckstein. K-svd: Design of dictionaries for sparse representation. *Proc. of SPARS*, 5:9–12, 2005.

[5] 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.

[6] 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.

[7] Ery Arias-Castro, Emmanuel J Candes, Arnaud Durand, et al. Detection of an anomalous cluster in a network. *The Annals of Statistics*, 39(1):278–304, 2011.

[8] Sanjeev Arora and Ravi Kannan. Learning mixtures of separated nonspherical gaussians. *Annals of Applied Probability*, 15(1A):69–92, 2005.

[9] M. Belkin and K. Sinha. Polynomial learning of distribution families. In *In Proc. Foundations of Computer Science (FOCS)*, 2010.

[10] Jon Louis Bentley and James B. Saxe. Decomposable searching problems i: Static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980.

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

[12] 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.

[13] Maren Böse, Friedemann Wenzel, and Mustafa Erdik. Preseis: A neural network-based approach to earthquake early warning for finite faults. *Bulletin of the Seismological Society of America*, 98(1):366–382, 2008.

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

[15] Peter Bühlmann and Sara van de Geer. *Statistics for High-Dimensional Data*. Methods, Theory and Applications. Springer, June 2011.

[16] 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.

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

[18] S S Chen, D L Donoho, and M A Saunders. Atomic Decomposition by Basis Pursuit. *SIAM review*, 2001.

[19] X Chen, Y Qi, B Bai, Q Lin, and J G Carbonell. Sparse latent semantic analysis. *NIPS Workshop*, 2010.

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

[21] Robert W Clayton, Thomas Heaton, Mani Chandy, Andreas Krause, Monica Kohler, Julian Bunn, Richard Guy, Michael Olson, Matthew Faulkner, MingHei Cheng, et al. Community seismic network. *Annals of Geophysics*, 54(6), 2012.

[22] R.W. Clayton, T. Heaton, et al. Community seismic network. *Annals of Geophysics*, 54(6), 2012.

[23] E Cochran and J Lawrence. The quake-catcher network: Citizen science expanding seismic horizons. *Seismological Research Letters*, 80:26, Jan 2009.

[24] 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.

[25] S. Consolvo, D.W. McDonald, T. Toscos, M.Y. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, et al. Activity sensing in the wild: a field trial of ubifit garden. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1797–1806. ACM, 2008.

[26] 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.

[27] A. Czumaj and C. Sohler. Sublinear-time approximation algorithms for clustering via random sampling. *Random Struct. Algorithms (RSA)*, 30(1-2):226–256, 2007.

[28] S. Dasgupta. Learning mixtures of gaussians. In *Fortieth Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 1999.

[29] S. Dasgupta and L.J. Schulman. A two-round variant of em for gaussian mixtures. In *Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2000.

[30] G Davis, S Mallat, and M Avellaneda. Adaptive greedy approximations. *Constructive Approximation*, 13(1):57–98, March 1997.

[31] 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.

[32] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, 2004.

[33] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Statist. Soc. Ser. B*, 39:1–38, 1977.

[34] Cynthia Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[35] 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.

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

[37] 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.

[38] 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 *Information Processing in Sensor Networks (IPSN)*, 2011.

[39] Matthew Faulkner, Michael Olson, Rishi Chandy, Jonathan Krause, K. Mani Chandy, and Andreas Krause. The next big one: Detecting earthquakes and other rare events from community-based sensors. In *In Proc. ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2011.

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

[41] D. Feldman, A. Fiat, and M. Sharir. Coresets for weighted facilities and their applications. In *Proc. 47th IEEE Annu. Symp. on Foundations of Computer Science (FOCS)*, pages 315–324, 2006.

[42] D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *Proc. 41th Annu. ACM Symp. on Theory of Computing (STOC)*, 2011.

[43] D. Feldman, M. Monemizadeh, and C. Sohler. A PTAS for k-means clustering based on weak coresets. In *Proc. 23rd ACM Symp. on Computational Geometry (SoCG)*, pages 11–18, 2007.

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

[45] J. Feldman, R. A. Servedio, and R. O'Donnell. Pac learning axis-aligned mixtures of gaussians with no separation assumption. In *COLT*, 2006.

[46] G. Frahling, P. Indyk, and C. Sohler. Sampling in dynamic data streams and applications. *Int. J. Comput. Geometry Appl.*, 18(1/2):3–28, 2008.

[47] G. Frahling and C. Sohler. Coresets in dynamic geometric data streams. In *Proc. 37th Annu. ACM Symp. on Theory of Computing (STOC)*, pages 209–217, 2005.

[48] R. Ganti, I. Mohomed, R. Raghavendra, and A. Ranganathan. Analysis of data from a taxi cab participatory sensor network. *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 197–208, 2012.

[49] Jie Gao, Leonidas Guibas, Nikola Milosavljevic, and John Hershberger. Sparse data aggregation in sensor networks. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pages 430–439. ACM, 2007.

[50] M. Gavish, B. Nadler, and R.R. Coifman. Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning. In *Proc. International Conf. on Machine Learning, Haifa, Israel*, 2010.

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

[52] 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.

[53] Ryan Gomes, Andreas Krause, and Pietro Perona. Discriminative clustering by regularized information maximization. In *Proc. Neural Information Processing Systems (NIPS)*, 2010.

[54] J.C. Gower and G.B. Dijksterhuis. *Procrustes Problems*. Oxford Statistical Science Series. OUP Oxford, 2004.

[55] 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.

[56] S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.

[57] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *Proc. 36th Annu. ACM Symp. on Theory of Computing (STOC)*, pages 291–300, 2004.

[58] S. Har-Peled and K. R. Varadarajan. High-dimensional shape fitting in linear time. *Discrete & Computational Geometry*, 32(2):269–288, 2004.

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

[60] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inf. Comput.*, 100(1):78–150, 1992.

[61] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. In *Annu. ACM Symp. on Computational Geometry (SoCG)*, 1986.

[62] 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.

[63] 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.

[64] A Hyvärinen. Independent component analysis: algorithms and applications. *Neural networks*, 2000.

[65] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. Wiley-Interscience, June 2001.

[66] 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.

[67] A. Krause, E. Horvitz, A. Kansal, and F. Zhao. Toward community sensing. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 481–492. IEEE Computer Society, 2008.

[68] 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.

[69] Akshay Krishnamurthy, James Sharpnack, and Aarti Singh. Detecting Activations over Graphs using Spanning Tree Wavelet Bases. *arXiv.org*, stat.ML, June 2012.

[70] Martin Kulldorff. A spatial scan statistic. *Communications in Statistics - Theory and Methods*, 26(6):1481–1496, January 1997.

[71] C. Lammersen and C. Sohler. Facility location in dynamic geometric data streams. In *Proc. 16th Annu. European Symp. on Algorithms (ESA)*, pages 660–671, 2008.

[72] QV Le, A Karpenko, and J Ngiam. ICA with Reconstruction Cost for Efficient Overcomplete Feature Learning. *Neural Information Processing Systems*, 2011.

[73] Ann B Lee, Boaz Nadler, and Larry Wasserman. Treelets: an adaptive multi-scale basis for sparse unordered data. *The Annals of Applied Statistics*, pages 435–471, 2008.

[74] Xingxing Li, Maorong Ge, Yong Zhang, Rongjiang Wang, Peiliang Xu, Jens Wickert, and Harald Schuh. New approach for earthquake/tsunami monitoring using dense gps networks. *Scientific reports*, 3, 2013.

[75] Yi Li, Philip M. Long, and Aravind Srinivasan. Improved bounds on the sample complexity of learning. In *Symp. on Discrete Algorithms*, pages 309–318, 2000.

[76] Annie Liu, Michael Olson, Julian Bunn, and K. Mani Chandy. Towards a discipline of geospatial distributed event based systems. In *DEBS '12: Proc. of the 6th ACM International Conf. on Distributed Event-Based Systems*, July 2012.

[77] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.

[78] M.W. Mahoney and P. Drineas. CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697, 2009.

[79] Stéphane Mallat. *A wavelet tour of signal processing*. Academic press, 1999.

[80] 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.

[81] 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.

[82] A. Moitra and G. Valiant. Settling the polynomial learnability of mixtures of gaussians. In *In Proc. Foundations of Computer Science (FOCS)*, 2010.

[83] 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.

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

[85] D B Neill and A W Moore. Rapid detection of significant spatial clusters. In *Proc. of the tenth ACM SIGKDD . . .*, 2004.

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

[87] 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.

[88] Michael Olson, Annie Liu, Matthew Faulkner, and K Mani Chandy. Rapid detection of rare geospatial events: earthquake warning applications. In *Proceedings of the 5th ACM international conference on Distributed event-based system*, pages 89–100. ACM, 2011.

[89] 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.

[90] Wei Pan, Yaniv Altshuler, Alex Paul Pentland, and Nadav Aharony. Methods and apparatus for prediction and modification of behavior in networks, May 29 2012. US Patent App. 13/482,925.

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

[92] 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.

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

[94] M. Schmitt. On the complexity of computing and learning with multiplicative neural networks. *Neural Computation*, 14(2):241–301, 2002.

[95] G. Shmueli and H. Burkom. Statistical challenges facing early outbreak detection in biosurveillance. *Technometrics*, 52(1):39–51, 2010.

[96] Aarti Singh, Robert Nowak, and Robert Calderbank. Detecting Weak but Hierarchically-Structured Patterns in Networks. In *The 13th International Conf. on Artificial Intelligence and Statistics (AISTATS)*, September 2010.

[97] 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.

[98] 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.

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

[100] John N Tsitsiklis et al. Decentralized detection. *Advances in Statistical Signal Processing*, 2(2):297–344, 1993.

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

[102] S. Vempala and G. Wang. A spectral algorithm for learning mixture models. In *In Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

[103] 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.

[104] Q. Wang, J. Xu, H. Li, and N. Craswell. Regularized latent semantic indexing. In *Proc. 34th Internat. ACM SIGIR Conf. on Research and Development in Information, SIGIR*, volume 11, pages 685–694, 2011.

[105] D B Wilson. Generating random spanning trees more quickly than the cover time. *Proc. of the twenty-eighth annual ACM symposium on Theory of computing*, pages 296–303, 1996.

[106] 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.

[107] Yih-Min Wu and Hiroo Kanamori. Development of an earthquake early warning system using real-time strong motion signals. *Sensors*, 8(1):1–9, 2008.

[108] 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.

[109] 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.

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

[111] C.V. Zhou, C. Leckie, and S. Karunasekera. A survey of coordinated attacks and collaborative intrusion detection. *Computers & Security*, 29(1):124–140, 2010.

Appendix A

Supplemental Material: Chapter 6

A.1 A Geometric Approach for Mixture of Gaussians

Let $\tilde{D} = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n\} \subseteq \mathbb{R}^{2d}$ and S be a set of subspaces of $\mathbb{R}^{2d}$. Let Y be the set of all 4-tuples $(s, w, \tilde{\mu}, W)$, where $s = (s_1, \ldots, s_k) \in S^k$, $\tilde{\mu} = (\tilde{\mu}_1, \cdots, \tilde{\mu}_k)$, $W = (W_1, \cdots, W_k) \in \mathbb{R}^k$, $w = (w_1, \cdots, w_k) \in \mathbb{R}^k$, $\sum_{i=1}^k w_i = 1$ and $w_i, W_i \geq 0$, and $\tilde{\mu}_i \in \mathbb{R}^{2d}$ for every i, $1 \leq i \leq k$.

For every $\tilde{\mathbf{x}} \in \mathbb{R}^{2d}$ and a set $X \subseteq \mathbb{R}^{2d}$, we define $\text{dist}(\tilde{\mathbf{x}}, X) = \min_{q \in X} \text{dist}(\tilde{\mathbf{x}}, q)$. For a tuple $y = (s, w, \tilde{\mu}, W) \in Y$, we define

$$\text{dist}(\tilde{\mathbf{x}}, y) = \min_{1 \leq i \leq k} W_i \text{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, s_i).$$

For every $\tilde{\mathbf{x}} \in \mathbb{R}^{2d}$ and $y = (s, w, \tilde{\mu}, W) \in Y$ we define

$$f_{\tilde{\mathbf{x}}}(y) = -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-W_i \text{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, s_i)^2\right)\right).$$

Note that a flat (affine subspace) can be represented by a subspace s_1 and a translation vector $\tilde{\mu}$. The distance from a point $\mathbf{x}$ to the flat is then the distance between $(\mathbf{x} - \tilde{\mu})$ to the subspace s_1.

Lemma 4. *Let $\Sigma \in \mathbb{R}^{d \times d}$ be a covariance matrix whose smallest singular value is $W_{\min}^2 > 0$. Let $\mu \in \mathbb{R}^d$. Then there is a vector $\tilde{\mu} \in \mathbb{R}^{2d}$ and a d-dimensional subspace s of $\mathbb{R}^{2d}$ such that for every $\mathbf{x} \in \mathbb{R}^d$ and a corresponding $\tilde{\mathbf{x}} = (\mathbf{x}, 0, \ldots, 0) \in \mathbb{R}^{2d}$ we have*

$$(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) = W_{\min}^{-1/2} \text{dist}(\tilde{\mathbf{x}} - \tilde{\mu}, s)^2.$$

Proof. Let $\Sigma^{-1} = LL^T$ be the Cholesky decomposition of Σ^{-1}, $L = UDV^T$ denote the SVD of L, and $D_{i,i}$ denote the ith entry of the diagonal of D, $1 \leq i \leq d$. Let $A^T = W_{\min} D^T U^T$ and

$\tilde{\mu} = (\mu, 0, \ldots, 0) \in \mathbb{R}^{2d}$. Since $\Sigma = (LL^T)^{-1} = UD^{-2}U^T$, we have

$$\frac{1}{\max\left\{D_{1,1}^2, \ldots, D_{d,d}^2\right\}} = \min\left\{D_{1,1}^{-2}, \ldots, D_{d,d}^{-2}\right\} = W_{\min}^2.$$

Hence,

$$\begin{aligned} W_{\min}^{-1/2} \left\|A^T(\mathbf{x}-\mu)\right\|^2 &= \left\|W_{\min}^{-1} A^T(\mathbf{x}-\mu)\right\|^2 \\ &= \left\|D^T U^T(\mathbf{x}-\mu)\right\|^2 = \left\|V D^T U^T(\mathbf{x}-\mu)\right\|^2 \\ &= \left\|L^T(\mathbf{x}-\mu)\right\|^2 = (\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu). \end{aligned} \tag{A.1}$$

We denote by I the identity matrix. We have that $I - A^T A = I - W_{\min}^2 D^2$ is a semi-positive definite matrix. Hence, there is a Cholesky decomposition $M^T M = I - A^T A$ of $I - A^T A$. Let $S^T = [A^T \mid M^T] \in \mathbb{R}^{d \times 2d}$, and let s denote a d-dimensional subspace of $\mathbb{R}^{2d}$ that is orthogonal to every column of S. Since $S^T S = A^T A + M^T M = I$, the columns of S are mutually orthogonal unit vectors. Hence,

$$\left\|A^T(\mathbf{x}-\mu)\right\| = \left\|S^T(\tilde{\mathbf{x}}-\tilde{\mu})\right\| = \left\|SS^T(\tilde{\mathbf{x}}-\tilde{\mu})\right\| = \operatorname{dist}(\tilde{\mathbf{x}}-\tilde{\mu}, s).$$

Together with (A.1) we obtain

$$W_{\min}^{-1/2} \operatorname{dist}(\tilde{\mathbf{x}}-\tilde{\mu}, s)^2 = W_{\min}^{-1/2} \left\|A^T(\mathbf{x}-\mu)\right\|^2 = (\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu).$$

□

For the case that the smallest singular value $W_{\min}$ of L is at least 1, we have

$$(\mathbf{x}-\mu)^T L^T L(\mathbf{x}-\mu) = \operatorname{dist}(\tilde{\mathbf{x}}-\tilde{\mu}, s)^2,$$

by replacing $W_{\min}$ with 1 in the above proof.

Theorem A.1.1 (Jensen's inequality)**.** *For every convex function $g : \mathbb{R} \to \mathbb{R}$ and a random variable X, we have $g(E[X]) \geq E[g(X)]$.*

Lemma 5. *For every $y = (s, w, \tilde{\mu}, W) \in Y$, $0 < \varepsilon < 1/2$, and $\tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \in \mathbb{R}^{2d}$, we have*

$$f_{\tilde{\mathbf{x}}}(y) \leq (1+2\varepsilon) f_{\tilde{\mathbf{x}}'}(y) + \frac{3 W_{\max} \operatorname{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')^2}{\varepsilon},$$

where $W_{\max} = \max_{1 \leq i \leq k} W_i$.

Proof. Put $y = (s, w, \tilde{\mu}, W) \in Y$, $\tilde{\mathbf{x}}, \tilde{\mathbf{x}}' \in \mathbb{R}^{2d}$ and $1 \leq i \leq k$. By the triangle inequality,

$$\begin{aligned}|\mathrm{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, s_i)^2 - \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2| &= |\mathrm{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, s_i) - \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)|(\mathrm{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, s_i) + \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)) \\ &\leq \mathrm{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, \tilde{\mathbf{x}}' - \tilde{\mu}_i)(2\mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, x_i) + \mathrm{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, \tilde{\mathbf{x}}' - \tilde{\mu}_i)) \\ &= \mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')^2 + 2\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')\mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i).\end{aligned} \tag{A.2}$$

If $\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') \leq \varepsilon \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)$ we have $\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')\mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i) \leq \varepsilon \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2$. Otherwise, $\mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i) < \mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')/\varepsilon$ so $\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')\mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i) < \mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')^2/\varepsilon$. In both cases we thus obtain

$$\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')\mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i) \leq \mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')^2/\varepsilon + \varepsilon \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2.$$

Together with (A.2) we have

$$\begin{aligned}|\mathrm{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, s_i)^2 - \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2| &\leq \mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')^2 + 2\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')^2/\varepsilon + 2\varepsilon \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2 \\ &= 3\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')^2/\varepsilon + 2\varepsilon \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2.\end{aligned} \tag{A.3}$$

Let $W_{\max} = \max_{1 \leq i \leq k} W_i$. Using (A.3),

$$\begin{aligned}f_{\tilde{\mathbf{x}}}(y) &= -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-W_i \mathrm{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, s_i)^2\right)\right) \\ &\leq -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-W_i\left(\mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2 + 3\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')/\varepsilon + 2\varepsilon \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2\right)\right)\right) \\ &= -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-W_i(1+2\varepsilon)\mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2 - 3W_i \mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')/\varepsilon\right)\right) \\ &\leq -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-W_i(1+2\varepsilon)\mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2\right) + 3W_{\max}\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')/\varepsilon\right) \\ &= -\ln\left(\sum_{i=1}^{k} w_i \left(\exp\left(-W_i \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2\right)\right)^{1+2\varepsilon}\right) + 3W_{\max}\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')/\varepsilon.\end{aligned} \tag{A.4}$$

Let X be a random variable such that $X = \exp\left(-W_i \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2\right)$ with probability w_i, $1 \leq i \leq k$. Since $z^{1+2\varepsilon}$ is a convex function over $z \in [0, 1]$, applying Theorem A.1.1 with $g(z) = z^{1+2\varepsilon}$ yields $\mathbb{E}\left[\left[\right] g(X)\right] \geq g(\mathbb{E}\left[\left[\right] X\right])$. Hence,

$$\begin{aligned}\sum_{i=1}^{k} w_i \left(\exp\left(-W_i \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2\right)\right)^{1+2\varepsilon} &= \mathbb{E}\left[\left[\right] X^{1+2\varepsilon}\right] = E[g(X)] \geq g(E[X]) = (E[X])^{1+2\varepsilon} \\ &= \left(\sum_{i=1}^{k} w_i \exp\left(-W_i \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2\right)\right)^{1+2\varepsilon}.\end{aligned}$$

Combining the last inequality with (A.4) yields

$$\begin{aligned}
f_{\tilde{\mathbf{x}}}(y) &\leq -\ln\left(\sum_{i=1}^{k} w_i \left(\exp\left(-W_i \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2\right)\right)^{1+2\varepsilon}\right) + 3W\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')/\varepsilon \\
&\leq -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-W_i \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2\right)\right)^{1+2\varepsilon} + 3W_{\max}\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')/\varepsilon \\
&= -(1+2\varepsilon)\ln\left(\sum_{i=1}^{k} w_i \exp\left(-W_i \mathrm{dist}(\tilde{\mathbf{x}}' - \tilde{\mu}_i, s_i)^2\right)\right) + 3W_{\max}\mathrm{dist}(p, p')/\varepsilon \\
&= (1+2\varepsilon) f_{\tilde{\mathbf{x}}'}(y) + 3W_{\max}\mathrm{dist}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')/\varepsilon.
\end{aligned}$$

□

Lemma 6. *Let y^* be a tuple that minimizes $\sum_{\tilde{\mathbf{x}}\in P} \mathrm{dist}(\tilde{\mathbf{x}}, y)^2$ over every $y \in Y$. Then for every $y \in Y$*

$$\sum_{\tilde{\mathbf{x}}\in\tilde{D}} \mathrm{dist}(\tilde{\mathbf{x}}, y^*)^2 \leq \sum_{\tilde{\mathbf{x}}\in\tilde{D}} f_{\tilde{\mathbf{x}}}(y).$$

Proof. Fix $y = (s, w, \tilde{\mu}, W) \in Y$ and $\tilde{\mathbf{x}} \in \tilde{D}$. We have

$$\begin{aligned}
f_{\tilde{\mathbf{x}}}(y) &= -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-W_i \mathrm{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, s_i)^2\right)\right) \\
&\geq -\ln\left(\sum_{i=1}^{k} w_i \cdot \exp\left(\mathrm{dist}(\tilde{\mathbf{x}}, y)^2\right)\right) = \mathrm{dist}(\tilde{\mathbf{x}}, y)^2.
\end{aligned}$$

Hence,

$$\sum_{\tilde{\mathbf{x}}\in\tilde{D}} f_{\tilde{\mathbf{x}}}(y) \geq \sum_{\tilde{\mathbf{x}}\in\tilde{D}} \mathrm{dist}(\tilde{\mathbf{x}}, s_m)^2 \geq \sum_{\tilde{\mathbf{x}}\in\tilde{D}} \mathrm{dist}(\tilde{\mathbf{x}}, y^*)^2$$

□

A.2 Coresets For Semi-Spherical Mixtures of Gaussians

Let $y^* \in Y$ be a tuple that minimizes $\sum_{\tilde{\mathbf{x}}\in\tilde{D}} \mathrm{dist}(\tilde{\mathbf{x}}, y)$ over every $y \in Y$. Suppose that $B = \left\{b_1, b_2, \cdots, b_{|B|}\right\} \subseteq \mathbb{R}^{2d}$ and $\alpha > 0$ such that

$$\sum_{\tilde{\mathbf{x}}\in\tilde{D}} \mathrm{dist}(\tilde{\mathbf{x}}, B)^2 \leq \alpha \sum_{\tilde{\mathbf{x}}\in\tilde{D}} \mathrm{dist}(\tilde{\mathbf{x}}, y^*). \tag{A.5}$$

Let $\left\{\tilde{D}_1, \cdots, \tilde{D}_{|B|}\right\}$ be a partition of $\tilde{D}$, where $\tilde{D}_i = \left\{\tilde{\mathbf{x}} \in \tilde{D} \mid \mathrm{dist}(\tilde{\mathbf{x}}, b_i) = \mathrm{dist}(\tilde{\mathbf{x}}, B)\right\}$ are the points that are served by b_i, $1 \leq i \leq |B|$. Ties are broken arbitrarily.

Lemma 7. *For every $1 \le i \le |B|$, $\tilde{\mathbf{x}} \in \tilde{D}_i$ and $y = (s, w, \tilde{\mu}, W) \in Y$, we have*

$$\frac{f_{\tilde{\mathbf{x}}}(y)}{\sum_{\tilde{\mathbf{x}} \in \tilde{D}} f_{\tilde{\mathbf{x}}}(y)} \le \frac{O(1)(\varepsilon + \alpha W_{\max})}{\varepsilon |\tilde{\mathbf{x}}_i|} + \frac{O(1)\mathrm{dist}(\tilde{\mathbf{x}}, B)^2}{\varepsilon \sum_{\tilde{\mathbf{x}} \in \tilde{\mathbf{x}}} \mathrm{dist}(\tilde{\mathbf{x}}, B)^2}.$$

Proof. Put $y = (s, w, \tilde{\mu}, W) \in Y$, $1 \le i \le |B|$, $\tilde{\mathbf{x}} \in \tilde{D}_i$, $n_i = |\tilde{D}_i|$, and $b = b_i$. By Lemma 5,

$$f_{\tilde{\mathbf{x}}}(y) \le (1 + 2\varepsilon) f_b(y) + \frac{O(1) W_{\max} \mathrm{dist}(\tilde{\mathbf{x}}, b)^2}{\varepsilon} \le 2 f_b(y) + \frac{O(1) W_{\max} \mathrm{dist}(\tilde{\mathbf{x}}, B)^2}{\varepsilon}. \tag{A.6}$$

We now bound $f_b(y)$. Applying Lemma 5 with $\tilde{\mathbf{x}} \in \tilde{D}_i$ and b yields

$$f_b(y) \le (1 + 2\varepsilon) f_{\tilde{\mathbf{x}}}(y) + \frac{O(1) W_{\max} \mathrm{dist}(\tilde{\mathbf{x}}, b)^2}{\varepsilon}.$$

Summing over every $\tilde{\mathbf{x}} \in \tilde{D}_i$ yields

$$\begin{aligned} n_i \cdot f_b(y) &\le (1 + 2\varepsilon) \sum_{\tilde{\mathbf{x}} \in \tilde{D}_i} f_{\tilde{\mathbf{x}}}(y) + \frac{O(1) W_{\max} \sum_{\tilde{\mathbf{x}} \in \tilde{D}_i} \mathrm{dist}(\tilde{\mathbf{x}}, b)^2}{\varepsilon} \\ &\le 2 \sum_{\tilde{\mathbf{x}} \in \tilde{\mathbf{x}}} f_{\tilde{\mathbf{x}}}(y) + \frac{O(1) W_{\max} \sum_{\tilde{\mathbf{x}} \in \tilde{D}} \mathrm{dist}(\tilde{\mathbf{x}}, B)^2}{\varepsilon} \end{aligned}$$

Plugging the last inequality in (A.6) yields

$$f_{\tilde{\mathbf{x}}}(y) \le \frac{2}{n_i} \sum_{\tilde{\mathbf{x}} \in \tilde{D}} f_{\tilde{\mathbf{x}}}(y) + \frac{O(1) W_{\max} \sum_{\tilde{\mathbf{x}} \in \tilde{D}} \mathrm{dist}(\tilde{\mathbf{x}}, B)^2}{\varepsilon n_i} + \frac{O(1) W_{\max} \mathrm{dist}(\tilde{\mathbf{x}}, B)^2}{\varepsilon}.$$

Hence,

$$\frac{f_{\tilde{\mathbf{x}}}(y)}{\sum_{\tilde{\mathbf{x}} \in \tilde{D}} f_{\tilde{\mathbf{x}}}(y)} \le \frac{2}{n_i} + \frac{O(1) W_{\max} \sum_{\tilde{\mathbf{x}} \in \tilde{D}} \mathrm{dist}(\tilde{\mathbf{x}}, B)^2}{\varepsilon n_i \sum_{\tilde{\mathbf{x}} \in \tilde{\mathbf{x}}} f_{\tilde{\mathbf{x}}}(y)} + \frac{O(1) W_{\max} \mathrm{dist}(\tilde{\mathbf{x}}, B)^2}{\varepsilon \sum_{\tilde{\mathbf{x}} \in \tilde{D}} f_{\tilde{\mathbf{x}}}(y)}. \tag{A.7}$$

By (A.5) and Lemma 6,

$$\sum_{\tilde{\mathbf{x}} \in \tilde{D}} \mathrm{dist}(\tilde{\mathbf{x}}, B)^2 \le \alpha \sum_{\tilde{\mathbf{x}} \in \tilde{D}} \mathrm{dist}(\tilde{\mathbf{x}}, y^*)^2 \le \alpha \sum_{\tilde{\mathbf{x}} \in \tilde{D}} f_{\tilde{\mathbf{x}}}(y).$$

Using the last inequality with (A.7) yields

$$\frac{f_{\tilde{\mathbf{x}}}(y)}{\sum_{\tilde{\mathbf{x}} \in \tilde{D}} f_{\tilde{\mathbf{x}}}(y)} \le \frac{2\varepsilon + O(1) \cdot W_{\max} \alpha}{\varepsilon n_i} + \frac{O(1) W_{\max} \mathrm{dist}(\tilde{\mathbf{x}}, B)^2}{\varepsilon \sum_{\tilde{\mathbf{x}} \in \tilde{D}} \mathrm{dist}(\tilde{\mathbf{x}}, B)^2}.$$

□

Theorem A.2.1 ([42]). *Let F be a set of n functions from Y to $[0, \infty)$ and $0 < \varepsilon < 1/4$. Let*

$m : F \to \mathcal{N} \setminus \{0\}$ be a function on F such that

$$m(f) \geq n \cdot \max_{x \in X} \frac{f(x)}{\text{cost}(F, x)}. \tag{A.8}$$

For each $f \in F$, let $g_f : Y \to [0, \infty)$ be defined as $g_f(y) = f(y)/m(f)$. Let G_f consists of m_f copies of g_f, and let S be a random sample of

$$t = \frac{\dim(F) \left(\sum_{f \in F} m(f)\right)^2}{(\varepsilon n)^2}$$

functions from the set $G = \bigcup_{f \in F} G_f$. Then for every $y \in Y$,

$$|\sum_{f \in F} f(y) - \sum_{f \in S} f(y)| \leq \varepsilon \sum_{f \in F} f(y).$$

Theorem A.2.2 ([42])**.** *Let $B \subseteq \mathbb{R}^d$ be the set that is computed during the execution of the algorithm 4. Then*

$$\sum_{\mathbf{x} \in D} \text{dist}(\mathbf{x}, B) \leq O(1) \min_{C^* \subseteq \mathbb{R}^d, |C^*| = k} \sum_{\mathbf{x} \in D} \text{dist}(\mathbf{x}, C^*)$$

A.3 Complexity of Mixture of Gaussians

Definition A.3.1 (range space [61])**.** *A* range space *is a pair $(F, \mathbf{ranges})$ where F is a set, and* **ranges** *is a set of subsets of F. The* dimension *of the range space $(F, \mathbf{ranges})$ is the smallest integer d, such that for every $G \subseteq F$ we have*

$$\Big| \{G \cap \mathbf{range} \mid \mathbf{range} \in \mathbf{ranges}\} \Big| \leq |G|^d \ .$$

The dimension of a range space relates (but is not equivalent) to a term known as the VC-dimension of a range space.

Definition A.3.2 (range space and dimension of F [75])**.** *Let F be a finite set of functions from a set X to $[0, \infty)$. The* dimension $\dim(F)$ *of F is the dimension of the range space $\big(F, \mathbf{ranges}(F)\big)$, where $\mathbf{ranges}(F)$ is the* range space of F*, that is defined as follows. For every $x \in X$ and $r \geq 0$, let $\mathbf{range}(F, x, r) = \{f \in F \mid f(x) \leq r\}$. Let $\mathbf{ranges}(F) = \{\mathbf{range}(F, x, r) \mid x \in X, r \geq 0\}$.*

Theorem A.3.3 ([94])**.** *Let q be a natural number and suppose that G is the class of real-values functions in the variables $y_1, \cdots, y_d$ satisfying the following conditions: For every $f \in G$ there is affine functions $g_1, \cdots, g_r$ where $r \leq q$, in the variables $y_1, \cdot, y_d$ such that f is an affine combination*

of $y_1, \cdots, y_d$ and $e^{g_1}, \cdots, e^{g_r}$. Then G has solution set components bound

$$|B| = 2^{O(d^2q^2)}.$$

Theorem A.3.4.

$$\dim(Y) = O(kd^2).$$

Proof. Let $y = (s, \tilde{\mu}, w) \in Y$. Since s is a hyperplane, there is a tuple of $d+1$ vectors $h_0, \cdots, h_{d+1} \in \mathbb{R}^d$ such that $s = \left\{h_0 + \sum_{i=1}^{d} a_i h_i \mid a_1, \ldots, a_d \in \mathbb{R}\right\}$, and

$$\begin{aligned}\text{dist}^2(\mathbf{x}, s) &= \|\mathbf{x} - h_0\|_2^2 - \sum_{i=1}^{d} \left((\mathbf{x} - h_0)^T h_i\right)^2 \\ &= \|\mathbf{x} - h_0\|_2^2 - \sum_{i=1}^{d} (\mathbf{x}^T h_i - h_0^T h_i)^2.\end{aligned}$$

For two vectors $(m_1, \cdots, m_s) \in \mathbb{R}^s$ and $(y_1, \cdots, y_t) \in \mathbb{R}^t$, we denote by my the tuple $m_1, \cdots, m_s, y_1, \cdots, y_t$. Let $h' = (1, h_0 \cdots h_d) \in \mathbb{R}^{d(d+1)+1}$, and $\mathbf{x}' = (1, \mathbf{x}) \in \mathbb{R}^{d+1}$. Hence, we have

$$\text{dist}^2(\mathbf{x}, s) = \sum_{i_0, i_1 \in [d+1], i_2, i_3 \in [d(d+1)+1]} c_{i_0,i_1,i_2,i_3} p'_{i_0} \mathbf{x}'_{i_1} h'_{i_2} h'_{i_3}, \tag{A.9}$$

where c_{i_0,i_1,i_2,i_3} is a constant that depends only on $i_0, \ldots, i_3$, and equals to zero for all except $d_1 = O(d^2)$ terms of the summation. That is, $\text{dist}^2(\mathbf{x}, \cdot)$ is an affine function . □

Theorem A.3.5. *Let $D \subseteq \mathbb{R}^d$, $\delta, \varepsilon > 0$, $k \geq 1$ and $W_{\min}, W_{\max} > 0$. Let $\mathfrak{C}$ be the collection of all mixtures of Gaussians $\theta = [(w_1, \mu_1, \Sigma_1), \ldots, (w_k, \mu_k, \Sigma_k)]$ such that Σ_i is a $d \times d$ covariance matrix whose singular values are between $W_{\min}^2$ and $W_{\max}^2$, for every $1 \leq i \leq k$. Let C be the output of Algorithm 4.*

Then, with probability at least $1 - \delta$, for every mixture $\theta \in \mathfrak{C}$ we have

$$\left|\sum_{\mathbf{x} \in D} \ln \sum_{i=1}^{k} w_i \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i)\right) - \sum_{\mathbf{x} \in C} \gamma(\mathbf{x}) \ln \sum_{i=1}^{k} w_i \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i)\right)\right| \leq \varepsilon \frac{W_{\max}^2}{W_{\min}^2} \mathcal{L}(D \mid \theta)$$

Proof. For every $\mathbf{x} \in D$ define $\tilde{\mathbf{x}} = \tilde{\mathbf{x}}(\mathbf{x}) = (\mathbf{x}, 0, \ldots, 0) \in \mathbb{R}^{2d}$ and let $\tilde{D} = \{\tilde{\mathbf{x}}(\mathbf{x}) \mid \mathbf{x} \in D\}$. Fix $\theta = [(w_1, \mu_1, \Sigma_1), \ldots, (w_k, \mu_k, \Sigma_k)] \in \mathfrak{C}$. For every i, $1 \leq i \leq k$, let W_i be the inverse of the smallest singular value of Σ_i. By Lemma 4 there is a vector $\tilde{\mu}_i$, and a subspace s_i such that

$$-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i) = W_i \text{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, s_i)^2.$$

Let $y = y(\theta) = (s, w, \tilde{\mu}, W)$ where $\tilde{\mu} = (\tilde{\mu}_1, \cdots, \tilde{\mu}_k)$, $w = (w_1, \cdots, w_k)$, $W = (W_1, \cdots, W_k)$ and $s = (s_1, \cdots, s_k)$, $Y = \{y(\theta) \mid y \in Y\}$. For every $\tilde{\mathbf{x}} \in \tilde{D}$ and $y = (s, w, \tilde{\mu}, W) \in Y$ we define

$$f_{\tilde{\mathbf{x}}}(y) = -\ln\left(\sum_{i=1}^{k} w_i \exp\left(-W_i \mathrm{dist}(\tilde{\mathbf{x}} - \tilde{\mu}_i, s_i)^2\right)\right),$$

and $F = \left\{f_{\tilde{\mathbf{x}}} \mid \tilde{\mathbf{x}} \in \tilde{D}\right\}$. By Theorem A.3.4, we have $\dim(F) = O(d^2 k)$. For every $f \in F$, let

$$m(f) = \frac{O(1)(\varepsilon + \alpha W_{\max})}{\varepsilon |\tilde{\mathbf{x}}_i|} + \frac{O(1)\mathrm{dist}(\tilde{\mathbf{x}}, B)^2}{\varepsilon \sum_{\tilde{\mathbf{x}} \in \tilde{\mathbf{x}}} \mathrm{dist}(\tilde{\mathbf{x}}, B)^2},$$

and note that $\sum_{f \in F} m(f) = O(|B|)$. By Lemma 7, for every $f \in F$ we have

$$m(f) \geq n \cdot \max_{y \in Y} \frac{f(y)}{\sum_{f \in F} f(y)}.$$

Let G_f consists of m_f copies of g_f, and note that C in Algorithm 4 is a uniform random sample from G_F. By Theorem A.2.1 we thus have

$$\left|\sum_{f \in F} f(y) - \sum_{f \in C} f(y)\right| \leq \varepsilon \sum_{f \in F} f(y).$$

□

Observation A.3.6. *Let $\theta = [(w_1, \mu_1, \Sigma_1), \ldots, (w_k, \mu_k, \Sigma_k)] \in \mathfrak{C}$ such that*

$$\forall_i : \left(\prod_{\lambda \in \mathrm{spec}(\Sigma_i)} \lambda\right) \geq \frac{1}{(2\pi)^d}.$$

If C is a (k, ε)-coreset for $D \subseteq \mathbb{R}^d$, then $\mathcal{L}(C \mid \theta)$ is a $(1+\varepsilon)$-approximation for $\mathcal{L}(D \mid \theta)$. That is,

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

Proof. Since $\sum_i w_i = 1$, we have by the assumption on θ

$$Z(\theta) = \sum_i \frac{w_i}{\sqrt{|2\pi\Sigma_i|}} \leq \max_i \frac{1}{\sqrt{|2\pi\Sigma_i|}} = \max_i \frac{1}{\sqrt{(2\pi)^d \prod_{\lambda \in \mathrm{spec}(\Sigma_i)} \lambda}} \leq 1.$$

Hence, $-n \ln Z(\theta) \geq 0$. By this and the assumption on θ,

$$\begin{aligned}
\mathcal{L}(C \mid \theta) &= -n \ln Z(\theta) + \phi(C \mid \theta) \\
&\leq -n \ln Z(\theta) + (1+\varepsilon)\phi(D \mid \theta) \\
&\leq (1+\varepsilon)\big(-n \ln Z(\theta) + \phi(D \mid \theta)\big) \\
&= (1+\varepsilon)\mathcal{L}(D \mid \theta).
\end{aligned}$$

Similarly,

$$\begin{aligned}
\mathcal{L}(C \mid \theta) &= -n \ln Z(\theta) + \phi(C \mid \theta) \\
&\geq -n \ln Z(\theta) + (1-\varepsilon)\phi(D \mid \theta) \\
&\geq (1-\varepsilon)\big(-n \ln Z(\theta) + \phi(D \mid \theta)\big) \\
&= (1-\varepsilon)\mathcal{L}(D \mid \theta).
\end{aligned}$$

□