*A Little Bit of Labeled Data Can Get You Started*

# Helping Mobile Apps Bootstrap with Fewer Users

**Xuan Bao**
Department of CS
Duke University
xuan.bao@duke.edu

**Aman Kansal**
Microsoft Research
Redmond
kansal@microsoft com

**Romit Roy Choudhury**
Department of ECE
Duke University
romit@ee.duke.edu

**Paramvir Bahl**
Microsoft Research
Redmond
bahl@microsoft.com

**David Chu**
Microsoft Research
Redmond
davidchu@microsoft com

**Alec Wolman**
Microsoft Research
Redmond
alecw@microsoft.com

## ABSTRACT

A growing number of mobile apps are exploiting smartphone sensors to infer user behavior, activity, or context. Inference requires training using labeled ground truth data. Obtaining labeled data for new apps is a "chicken-egg" problem. Without a reasonable amount of labeled data, apps cannot provide any service. But until an app provides useful service it is not worth installing and has no opportunity to collect user data. This paper aims to address this problem. Our intuition is that even though users are different, they exhibit similar patterns on certain sensing dimensions. For instance, different users may walk and drive at different speeds, but certain speeds will indicate driving for all users. These common patterns could be used as "seeds" to model new users through semi-supervised learning. We prototype a technique to automatically extract the commonalities to seed personalized inference models for new users. We evaluate the proposed technique through example apps and real world data.

## INTRODUCTION

The rich suite of sensors on mobile devices is enabling a new class of sensing applications that recognize user activities, their context, and environments, for several purposes [1]. Most of these applications rely on some form of inference over the raw sensor outputs. Inferencing entails (1) an initial *training phase*, where the raw data are labeled with the true value of the attribute being inferred and a model is established to relate the data to the inferred attribute, and (2) the *inference phase*, where the trained model is used to infer the attribute of interest from new unlabeled data.

For the inference to be reliable, the training phase needs to use representative data. For instance, if an app intends to infer user mood from phone usage [2], it needs labeled phone usage data from a large number of users with their specific behavior for training. Training on a few users, say the app developer or their friends, would result in poor inferencing accuracy for most other users. Models trained on the same user as the one for whom inference is to be performed are more accurate [2] since the raining data is highly representative for that user.

In practice, obtaining labeled data from a large number of users is hard. Obtaining labels from each user (to train a personalized model) is even harder. Users are typically not interested in providing labels for an app that they have not yet found to be useful. But without the training labels the app has poor inference performance, and is not very relevant to be used or even installed in the first place. We refer to this issue as the *bootstrapping problem*. It has been one of the main hurdles in bringing inference-based sensing apps to today's mobile app store.

In this paper we develop techniques to automatically adapt the inference models required by sensing applications to multiple users. The solution is based on an insight that applies well-studied semi-supervised learning methods to this problem. We believe that the training data will contain certain data points that occur with higher frequency and correspond to high confidence inference. Continuing with the simple example of detecting user mood, certain data points in the labeled data may reliably map to a happy or sad mood, while other data points may have lower reliability and may even correspond to a different mood for other users. Using only the high confidence points as seeds for semi-supervised learning, we incorporate the unlabeled data for each user to train a new model specific to that user. Since the unlabeled data is used from the user on which inference is performed, the inference model is expected to be accurate for the user. The unlabeled data is obtained from apps already in use that may not have assigned any ground truth labels of interest to the new applications.

Building up further on the above intuition we also extend our model adaptation strategy to exploit multiple sensors or data


*UbiComp '12*, Sep 5-Sep 8, 2012, Pittsburgh, USA.

dimensions to enhance accuracy. The idea is to find seeds on at least one of the sensing dimensions, and use them to infer on other dimensions. For instance, Alice's location may be a reliable feature to infer whether she is at home or office, but her location may not be a seed for labeling Bob's data, if Bob lives at a different location. Now, if accelerometer readings correlate well to location (say, distinct motion patterns emerge in home and office), then Alice's motion patterns may be matched to Bob's, and therefrom, Bob's office and home locations identified. We show how model adaptation is particularly effective because every labeled data, such as "I am at home", implicitly labels the location, accelerometer, and other sensor readings from that time instance. Thus, if any one dimension matches well between Alice and Bob, the rest of Bob's unlabeled data can potentially be labeled.

We evaluate our techniques in the context of multiple inference tasks using real world sensor data. We compare the accuracy of the adapted models generated by our techniques to the models trained using ground truth labels for each user. Results show that the proposed techniques are indeed effective for the applications studied. Of course, deploying our techniques in real world mobile app development frameworks, such as Android or iOS, will require extensive testing across a far greater number of applications and user-data. At this stage, we show a proof of concept that allows small scale app developers to bootstrap their apps with only a few users.

Finally, we note that since our system finds mechanisms to label unlabeled data, and gain from the process, significant amount of sensing data collected by different apps can be consolidated. The consolidated data can then be used to bootstrap new apps, particularly those that rely on collaborative sensing (e.g., GPS and accelerometer readings from many users to infer traffic congestion, or potholes [3–6]). Such apps are difficult to bootstrap on their own because they require a large number of users to begin with. Again, as a proof of concept, we prototype such a collaborative sensing app and bootstrap it from data provided by other apps. The prototype app partitions users current activity into into 2 classes – professional work and personal activity – and may be used to measure the work life balance within a population. The app is build using data that were collected entirely from other apps.

## DESIGN PHILOSOPHY

Our techniques are designed to reduce the need for labeled data that is often hard to collect. However, unlabeled data is still required to operate as may be expected for any data driven inference methods. To facilitate the collection and availability of such data, we make the following assumptions regarding the application development and usage scenario.

### Expansive User Coverage

We assume that applications that collect sensor data can share it across users and other applications. Such sharing is already available for location sensors through services such as Google Latitude[1]. This service collects location data from every application that senses location such as a weather app that senses location to display the weather relevant to the user's region, local search apps, and of course map or navigation apps. The collected data is made available to all applications, including apps that do not collect any data themselves but may just wish to get a historic trace of the data, again subject to user permissions. Similar sharing for other sensors on the phone such as accelerometers, light sensors, gyroscopes, magnetometers, proximity sensors, is not hard to envision. Even for sensors such as the microphone that are relatively more privacy intrusive, certain data features such as magnitude or variance, or features specifically designed to preserve privacy [7] may still be shared assuming appropriate access control is provided. Hence, not all sensor data (e.g., microphone, camera) collected by an app may be useful for other apps, but some generic sensors and features are likely to be relevant.

While individual user may only use applications of interest to them, collectively among all apps, coverage is likely to build up for a large number of users and sensor types. Ground truth labels will only be available for a very small fraction of the users. Often, the ground truth labels will be limited to the app developers who collect training data on their own devices or a small number of enthusiastic users. Some additional training data may exist through incentive based or fun and game driven labeling [8]. However, the bulk of data remains unlabeled. The key advantage of such sharing is that when a new user downloads an app, and the inference model needs to be adapted for this user, there is likely already unlabeled data available from this user. In some cases, the new app may be able to collect unlabeled data from this user during initial usage. We will use this unlabeled data for adapting the inference model for this user and provide more accurate inference.

### Critical Mass for Participatory Sensing

The sharing of data can enable new participatory sensing applications that might not reach critical mass on their own. Consider for example an application deployed by each user to rate driver performance [9] that collects accelerometer data and potentially uploads it to compare to other drivers in the relevant driving conditions. The mobile device may also have a navigation app that collects location data, and uploads this data to download relevant maps and traffic from the cloud. If data from these apps is shared, the accelerometer and GPS data collected from multiple drivers across a city could be used to enable an app that maps potholes on local roads [6]. The data sharing based design philosophy automatically enables participatory sensing apps.

### Design Overview

Figure 1 summarizes the two important aspects of this design – exploiting *model adaptation* and bootstrapping *participatory sensing*. Multiple apps from multiple users share their data in a common repository. Only a small number of users, recruited by the developers, have labeled data for each app and the *model adaptation* methods are applied in the shared repository to generate adapted inference models for all the other users. The adapted models may be downloaded to each

[1] http://www.google.com/latitude

user's device for accurate inference. Related techniques will be explained in detail in the next section.

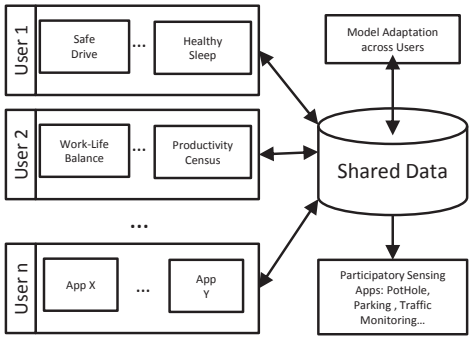



**Figure 1. Sharing data generated from individual applications bootstraps new participatory sensing applications.**

Moreover, the large-scale shared data/inferences can also be used to enable new *participatory sensing apps*. For example, a series of applications [3–6] can benefit from knowing whether the user is driving or not. Therefore, if any driving detection app (e.g., Safe Drive) is contributing this inference (driving/not driving) to the shared data repository, new applications can build their inference model based on known driving status without developing their own driving detection module. Similarly, applications monitoring users' context (e.g., sleep detection, home/work status detection) can also provide useful inferences to other applications. In the next subsection, we present four such prototype applications that exhibit this property.

### Prototype Applications

Sensing inference is useful for several applications. While we describe our methods generically, we use the following concrete inference tasks to illustrate and evaluate the concepts presented.

1. **Safe Drive.** People spend a significant amount of time driving [10] and the detection of driving can be used for preventing mobile device distractions [11], optimizing GPS use based on movement [12], and other context dependent tasks. We prototype the detection of driving using accelerometer and GPS sensors as one of our activity sensing examples. Similar inference has also been performed for inactivity detection [13].
2. **Healthy Sleep.** Detecting if the user is sleeping has multiple uses. The mobile device may change the sounds for alerts and notifications, or deactivate background tasks such as email synchronization if the user has forgotten to plug in their phone, so as to ensure that the battery can last until the time set for the morning alarm. Long term observations may be used to track health problems or sleep disorder [14]. We prototype sleep detection using several sensors including audio, GPS, light level, and time of day.
3. **Work-Life Balance.** Determining user context in terms of home or work can be useful for several ubiquitous computing applications [15], mobile applications, and operating system services [16]. The home may preheat itself, user mobile device home screen and security settings could be altered, alerts and ring-tones could be changed, and WiFi radio could be turned on or off, among other things. We prototype this detection using GPS and time.
4. **Productivity Census (Participatory sensing supported by existing apps).** As an example of a participatory sensing application, we consider the measurement of user activity from an economic perspective. The US Bureau of Labor Statistics conducts a survey known as the American Time Use Survey (ATUS) [17], that measures the amount of time people spend doing various activities, such as working (including educational activities), personal care activities (including sleep), and leisure activities. A summary view of the collected data from 2009 is shown if Figure 2.

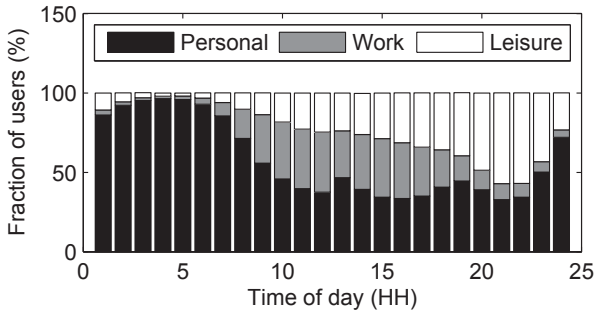



**Figure 2. Summary of American Time Use Survey Data showing user activities at different times.**

The data is used for economic research such as to quantify work activities, enhancing worker productivity, and tracking changes in work patterns with changing business cycles. The data is also useful for studying health and safety, as well as family-work-life balance. Currently the data is collected using manual surveys. This has a high cost and can only reach a small percentage of the population. An alternative is to use the sensors on mobile devices to automatically collect user activity data and continuously the time use characteristics for a large population. We use this as our example of a participatory sensing application that can be supported by existing applications.

Additionally, the techniques can be applied to any inference task where user specific training will help improve the accuracy compared to a generic model, such as cough detection from audio [7] or user inactivity [13].

## INFERENCE MODEL ADAPTATION

Our primary goal is to automatically adapt inference models for new users, solving the bootstrapping problem. To use a generic method for adaptation, independent of application semantics, the inference model must be represented in a common form. While many different types of algorithms and application-specific rules can be applied to perform inference on the sensor data, we focus on inference methods based on probabilistic machine learning techniques. This common structure is broad enough to cover a large class of inferences. Many sensing tasks for sensing both the users and their environments are based on probabilistic inference models [1,18]. Applications that fall outside of this structure can still take advantage of data sharing, but will not benefit from the automated adaptation of inference models.

Most probabilistic inference methods begin with a training data set that consists of the sensor data and labels that indicate the ground truth activity or phenomenon corresponding to each data sample or time window of samples. The labels are often application specific. Appropriate features that are likely to be beneficial for distinguishing between the inferred states are computed from the raw sensor data and an inference model is trained using statistical machine learning techniques such as Support Vector Machines, Bayesian Networks, Gaussian Process Regression, or various clustering methods. We follow this same general methodology, selecting specific methods for each step as conducive to model adaptation.

Figure 3 illustrates an overview of the process. The labeled data is used to establish inference models for training users. Then, the high confidence data points from the training user and unlabeled data from new users are combined to bootstrap model adaptation for these new users.

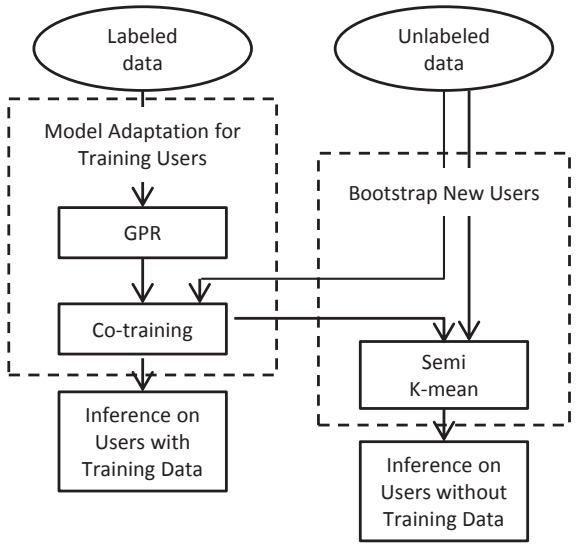



**Figure 3. Block diagram of proposed optimizations that exploit shared unlabeled data across multiple users and applications.**

Below, we describe the set of techniques (Gaussian process regression (GPR), co-training, semi K-mean) chosen for each stage of the inference tasks, starting with the training on a single user (the developer themselves) or a small set of volunteers, to the final stages of adapting the model to a large number of other users.

### Initialization

To build a sensing application, the developer begins with a training data set. The training set could be small and cover only a few users for a limited duration of time. These users are recruited by the developer to provide initial labeled data for a new application. Notice that, assuming the application uses probabilistic inference, such a training set would have been required even if not using the proposed adaptation techniques.

From a developer perspective, this simply means that they collect the training data as they would have. They may potentially collect on fewer users than would have been otherwise required. The training data consists of the raw sensor values, indexed by time stamps and a user identity (an anonymized, randomly generated, key may be used for the user to maintain their personalized model without revealing their personal information). Each sensor value is labeled with the ground truth value for the attribute to be inferred. For instance, for the detection of driving activity, a trace of sensor data may be collected over time and periods of driving labeled as "driving" and other periods labeled "not driving." The developer may optionally indicate which features they wish to compute from the raw sensor data (such as the first derivative, or speed, over GPS data may be identified to be useful. After uploading the training data, suppose the training set contains data from $k$ sensors and contains $N$ data points coming from one or a few users. Denoting the $i$-th data point as $\mathbf{S}^i = \{s_1^i, ..., s_k^i\}$, the data set becomes $\{\mathbf{S}^i\}_{i=1}^N$, with corresponding labels $\{l_i\}_{i=1}^N$.

The first step is to build an inference model from the training set. While several machine learning techniques are available for this, we wish to select one that quantitatively tracks uncertainty, since confidence level is used in adaptation across multiple users. Gaussian Process Regression (GPR) is a well known method that estimates uncertainty naturally during inference [19]. In GPR, one assumes there is a latent function $f_j$ that relates the sensor reading $s_j^i$ to the output $l_i$ for all data points $i$. The probability distribution $p(f_j)$ of the possible functions is assumed to be a Gaussian Process, characterized by its covariance matrix $\mathbf{K_j}$, where $j$ denotes the $j$-th sensor dimension. The covariance matrix captures the knowledge in the training data about the function $f$. $\mathbf{K_j}$ can be computed using standard methods from the training data once a kernel function, usually chosen from Radial Basis Function (RBF) based kernels, is assumed. Once $\mathbf{K_j}$ has been computed, GPR can be directly used to predict the labels for new observations, and inferences performed using GPR automatically include a measure of uncertainty.

At this stage, two types of model adaptations may be performed that exploit the unlabeled data available due to sharing across multiple applications. First, the additional unlabeled data available for the *training* user (such as data collected by other apps previously) can be used to enhance the model for the training user itself. Second, the model learned from the training user may be adapted to *additional* users.

### Adaptation for Training User

While the developer may only provide training data collected over a limited time duration, there may be other data along the same sensor dimensions available for other times, contributed from other apps previously, or collected within the app without labeling. Such data can be used to improve the accuracy of the inference model by adapting the model from that provided by the developer to one that incorporates additional data. A semi-supervised learning technique that enables this is co-training. Specifically, we use Bayesian co-training that can naturally incorporate the uncertainty measures and learned kernels from GPR. The uncertainty measures are extended to the unlabeled data as it is incorporated.

Also, if different data points have labels on different sensor dimensions, that data is easy to use.

Bayesian co-training methods are well-studied in literature and we refer the reader to prior work [20] for details. Briefly, the insight behind incorporating unlabeled data using co-training is that the unlabeled data is used to bias the distributions of the functions $f_j$ for each sensor dimension towards those function instances that maximize the consensus among the inferences made for unlabeled data points using each of the sensors. For example, in a driving detection app, for an unlabeled data point, if the accelerometer readings exhibit an ambiguous pattern, but the GPS based speed indicates driving with high confidence based on the initial model derived from labeled data, then the co-trained kernel will get biased towards the higher confidence inference (driving) for this data point. This will help infer the correct user state from similar accelerometer data observed later. Thus co-training allows the model to learn new accelerometer data patterns that were not available in the labeled data set.

Mathematically, according to co-training theory, the covariance matrix for the Gaussian Process characterizing the consensus function is given by:

$$\mathbf{K}_c = \left[ \sum_j \left( \mathbf{K}_j + \sigma_j^2 \mathbf{I} \right)^{-1} \right]^{-1}$$

where $\sigma_j$ represents the noise level associated with the $j$-th sensor, and $\mathbf{I}$ is the identity matrix. More reliable sensors can be assigned lower noise levels and can hence have a higher weight in the consensus. The co-variance matrix $\mathbf{K}_c$ yields an enhanced inference model that incorporates multiple sensors as well as any available unlabeled data for the most accurate inference possible using the available labeled and unlabeled data.

The actual application of the above methods for model adaptation is summarized in Algorithm 1. Briefly, GPR is first used to learn a model on each sensor dimension using only the labeled data and then co-training is applied to incorporate the unlabeled data and derive a joint inference model. In the specification, $\{\mathbf{S}^i\}_{i=N+1}^{M}$ denotes any available unlabeled data.

In practice, certain features computed over the raw sensor data are used rather than the raw sensor values since the inferred state label often has better dependence on the features. For instance, the driving activity state may be better inferred using the first derivative of GPS sensor values (speed). Hence the introduction of step 1 in the algorithm. In our prototype, we implement commonly used features, namely the magnitude, variance, first derivative, and most dominant frequency components of the data histogram. Other features can be considered as well.

Since the algorithm operates in an application agnostic manner, co-training weights must be determined automatically. The search space of weights in $k$ dimensions is rather large and for tractability, we use a common heuristic that compares the performance of various sensors pairwise (step 5), assigning relative weights in proportion to their accuracy. Sensors that yield accuracy below a threshold are dropped from $\mathbf{K}_c$. The variable $\alpha$ is used to denote a threshold of accuracy below which inference is not useful.

**Algorithm 1** ENHANCEMODEL($\{\mathbf{S}^i\}_{i=1}^{N}$, $\{l_i\}_{i=1}^{N}$, $\{\mathbf{S}^i\}_{i=N+1}^{M}$), $\alpha$

1: Compute features $g(\mathbf{S}^i)$ from raw data for each sensor.
2: Use labeled data $[\{g(\mathbf{S}^i)\}_{i=1}^{N}, \{l_i\}_{i=1}^{N}]$ to compute $\mathbf{K}_j$ for each $j$, using GPR techniques.
3: Drop sensors that yield accuracy below $\alpha$. Suppose $q$ sensors remain.
4: List $\binom{q}{2}$ pairs of sensors.
5: **for** each pair $r$ **do**
6: $w_r^*$ = 0.01
7: **for** weight $w_r$ in $\{0.01, 0.1, 1, 10, 100\}$ **do**
8: Incorporate $\{g(\mathbf{S}^i)\}_{i=N+1}^{M}$ using co-training with relative weight $w_r$ between sensors
9: If accuracy is better than with $w_r^*$, set $w_r^* = w_r$
10: Normalize relative weights across all pairs, obtaining $\{w_j\}_{j=1}^{q}$.
11: Incorporate $\{g(\mathbf{S}^i)\}_{i=N+1}^{M}$ using co-training with noise variances $\sigma_j = \{1/w_j\}_{j=1}^{q}$ to compute $\mathbf{K}_c$.
12: **return** $\mathbf{K}_c$

This algorithm can be applied to any application to derive an inference model. This inference model can be applied directly to data streams from the initial users for whom the training data was obtained. This inference model is also ready to be used for multi-user adaptation.

#### Bootstrapping Additional Users

Once the inference model is determined, and represented in a form that incorporates uncertainties, for the users with labeled data provided by the developer, we adapt this model to other users using semi-supervised learning.

The uncertainty estimate from the use of $\mathbf{K}_c$ on the labeled users gives an indication of which data points have highest inference confidence (estimated by separating out data points randomly into training and test sets). We now select top few percent highest confidence data points (20% in our prototype implementation) along with their labels as potentially useful labels for new users for whom no labeled data is available. Also, while each data point is multi-dimensional (has values for multiple sensor features) only those sensor dimensions that had the highest inference accuracy (those with the highest weights used in the co-training matrix $\mathbf{K}_c$) are selected.

##### *Personalization*

Since our goal is to adapt to the differences in behavior of the new user, we do not incorporate the unlabeled data from the new user into the training set from the labeled user and apply co-training to maximize consensus among the entire data set. That would hide the individual differences and converge towards an average behavior that may not work well for any

of the individual users. Instead we perform semi-supervised learning on each new user separately. Since there are no ground truth labels for the new users, there is no direct way to test the accuracy of different sensor dimensions. Hence we use a semi-supervised learning method that does not require the assignment of weights to different dimensions. A lightweight semi-supervised learning method that automatically uses all available sensor dimensions to the extent useful is *constrained k-means clustering* [21]. This method works under the assumption that the number of clusters, i.e., the inferred states, is the same across different users. Also, the high-confidence data points from the training user should be indicative of true states for at least the new user's data points close to them. For instance, a high confidence data point that indicates the inferred state 'driving' at a very fast speed for the trained user should also indicate the state 'driving' for the new user at similar speeds, even though the actual distribution of speeds (GPS data) and accelerometer data may be different across two users. One user may typically drive much slower, living in a busier city, or have smoother accelerometer outputs based on a smoother vehicle, compared to the other. These differences will result in different model parameters after semi-supervised learning is applied.

*Learning New User Models*

The actual procedure for applying semi-supervised learning to new users for model adaptation boils down to Algorithm 2. Here $\{\mathbf{S}^i\}_{i=1}^M$ represents the unlabeled data for the new user, and $\{\mathbf{S_0}^i\}_{i=1}^n$ represents the $n$ high confidence data points, sometimes referred to as *seeds*, selected for adaptation from the training users with labels.

The algorithm is essentially implementing standard constrained k-means clustering but with one modification specific to our scenario: *to accept reduced dimensionality seed points*.

While $\mathbf{S}^i$ for the new user is $k$-dimensional, the seed points may have less than $k$ dimensions since only the high confidence sensor dimensions are selected in the seeds. Suppose $\mathbf{S_0}^i$ is $p$ dimensional. Suppose $k'$ and $p'$ represent the number of features computed on $k$ and $p$ dimensions respectively. Also, suppose there are $L$ states to infer, implying that $l_i$ takes values in $\{1, ..., L\}$. Then, $g(\mathbf{S_0}^i)|l_i = x$ represents the labeled points with label $l_i$, sometimes referred to as cluster $l_i$. The difference in dimensions of $\mathbf{S}$ and $\mathbf{S_0}$ causes us to introduce step 3 in the algorithm before the standard constrained k-means clustering algorithm can be applied. This step is computing, for each cluster in the labeled data, the unlabeled point that is closest to that cluster. The remaining $k' - p'$ dimensions of the selected point are used to expand the centroid of the labeled cluster to $k'$ dimensions from $p'$ dimensions. The value of $n$ is taken as 20% of $N$, the number of labeled training points, in our implementation. The variable $\theta$ denotes the threshold distance that is used to declare that the constrained k-means algorithm has converged when the centroids move less than $\theta$. Step 7 simply implements the standard constrained k-means algorithm. The algorithm returns the adapted centroids after constrained k-means has been applied. These centroids can now be used for inference using a clustering algorithm for future sensor observations.

**Algorithm 2** ADAPTMODEL($\{\mathbf{S}^i\}_{i=1}^M$, $\{\mathbf{S_0}^i\}_{i=1}^n$, $\{l_i\}_{i=1}^n$), $\theta$

1: Compute features $g(\mathbf{S}^i)$, $g(\mathbf{S_0}^i)$ from raw data for each sensor.
2: Compute the $p'$-dimensional Euclidean distance between $g(\mathbf{S}^i)$ and $g(\mathbf{S_0}^i)$.
3: **for** each label $x \in \{1, ..., L\}$ **do**
4: Find the point $g_x$ in $g(\mathbf{S}^i)$ that is closest to points $g(\mathbf{S_0}^i)|l_i = x$, using $p'$ dimensional distances.
5: To the $p'$ dimensional centroid of cluster $x$, add the remaining $k' - p'$ values from the corresponding dimensions of $g_x$. Denote the expanded centroid as $c_x$.
6: Initialize constrained k-means algorithms with centroids $c_x$ with weights corresponding to number of points in $\{\mathbf{S_0}^i\}_{i=1}^n$ with label $x$.
7: **repeat**
8: **for** each unlabeled point $g(\mathbf{S}^i)$ **do**
9: Find nearest centroid $c_x$ and add point to cluster $x$
10: Update cluster centroid $c_x$ with new point
11: **until** centroids move less than $\theta$
12: **return** Return $\{c_x\}_{x=1}^L$

As an illustration of this algorithm, we consider the driving detection activity, using the data set described in the evaluation section. Figure 4 shows the original cluster centroids using the seed points from the trained user, as extended to $k' = 3$ dimensions after step 3. The figure also shows the evolved centroids after the model adaptation has been applied using the unlabeled data from the new user. Three features coming from two sensors are shown. Accelerometer feature 1 refers to the magnitude of the mean of the 3D accelerometer vector computed over a 10 sample window. Accelerometer feature 2 refers to the magnitude of variance over a similar data window. The sensor dimensions are normalized to be between 0 and 1. Cluster 2 represents driving activity while cluster one represents not driving.

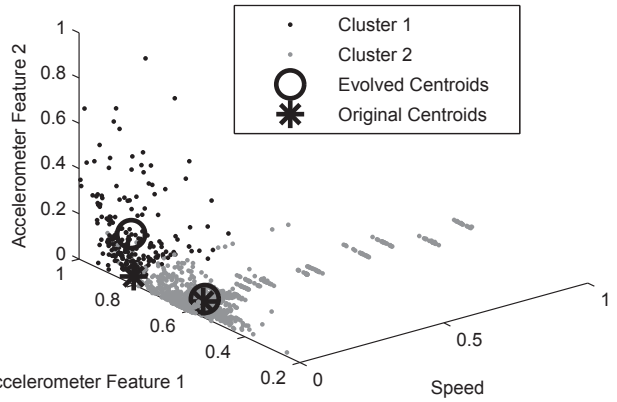



**Figure 4. Evolution of cluster centers after model adaptation.**

### Summary

The combination of methods discussed above inference adaptation is shown in Figure 3. The labeled data is used for learning GPR models for the training users which may optionally be enhanced using unlabeled data for the training users themselves through co-training. This model suffices to perform inference for the training users.

The high confidence data points from the training user and unlabeled data from new users are used for model adaptation to enhance the operation of inference on those new users. The adapted model for each user is used for inference for that specific user.

### Limitations

The proposed method still has its limitations. For example, like any learning method, if a user behaves completely differently from any known user (e.g., constantly drive slower than running), the model adaptation may fail to generate an accurate inference model for her. Also, current implementation does not accommodate for device heterogeneity. Accommodating devices with different sensing accuracy and sampling rate is one of our future work.

## EVALUATION

We evaluate the performance of the proposed adaptation methods on real world data sets to study the inference accuracy achieved. The evaluations are performed in the context of the applications mentioned in the prototype application section, namely healthy sleep (sleep activity detection), work-life balance (inferring at home or work), safe drive (driving activity detection), and productivity census (the participatory sensing app).

### Datasets

While the proposed techniques help alleviate the need for labeled datasets, we still need labeled ground truth data for testing the accuracy of the proposed techniques. This requires us to collect labeled data for each user to be included in our evaluations. We collected the following two datasets for this purpose.

- **Sleeping Activity Dataset (used for work-life balance app and healthy sleep app).** We collected ground truth labeled data recording the sleep activity for 5 volunteers (recruited from among graduate students at our lab and their families). Each user carried a mobile device that collected light, proximity, GPS, accelerometer, and audio magnitude data for 3 to 7 days. The actual sleeping times were recorded manually. This dataset contains two sets of labels, home/work and sleep/awake, associated with each 1 minute interval.

- **Driving Activity Dataset (used for safe drive app).** We also collected labeled data for driving activity detection. This set comprises sensor data including GPS, accelerometer and other mobile device sensors from 14 mobile devices. The users carried the devices in their pocket, hand, or backpack, and engaged in driving as well as other activities including walking and not moving. Each user was followed by another person who labeled the ground truth activity for the entire user trace. Each trace was collected for 1 hour, representing a total of 14 hours of ground truth labeled data. This data set is labeled with transportation modes, walking/driving, associated with the mode switching time (e.g., the user starts driving at 7 AM).

- **Productivity Census** Moreover, we show that a productivity census app can be built using the inferences provided by existing apps.

### Evaluation Metric

The performance of sensing applications can be evaluated by comparing inference results with labeled ground truth. The comparison result can be measured by various information retrieval metrics such as precision/recall/fallout or a ROC curve. In our evaluation, we choose the metric of accuracy defined as:

$$\text{Accuracy} = (\text{True Positives} + \text{True Negatives}) / \text{All Samples}$$

### Intra-User Adaptation

After initialization to establish an inference model for the training data provided by the developer, we use Bayesian co-training to incorporate any unlabeled data available for those users, to derive an enhanced inference model. To evaluate the advantage of model enhancement for the training users using co-training, we first initialize the inference model using Gaussian Process Regression (GPR) on each sensor dimension using labeled data alone, and compare the accuracy with the co-trained model. The available data was partitioned into a training set taking 50% of the data and the remaining 50% was used for testing.

Figure 5 shows the accuracy achieved by GPR and co-training for various sensors for sleeping activity detection. Similar results are shown for driving activity detection in Figure 6. Co-training shows a clear advantage for almost all users.

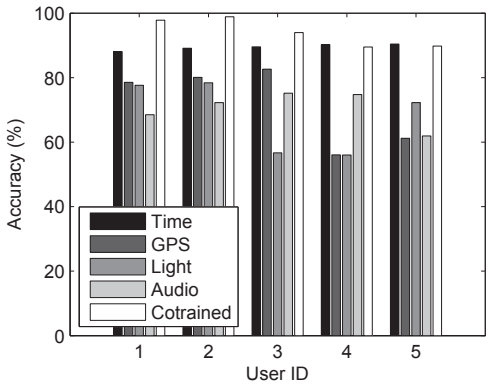



**Figure 5. Co-training advantage for sleeping activity detection.**

The co-training results for detecting if the user is at home or work are similar, except that the co-training advantage over using GPS is very small for a given user with labeled data. For this sensing task, the evaluation for model adaptation across users is more important and is presented in the following subsection.

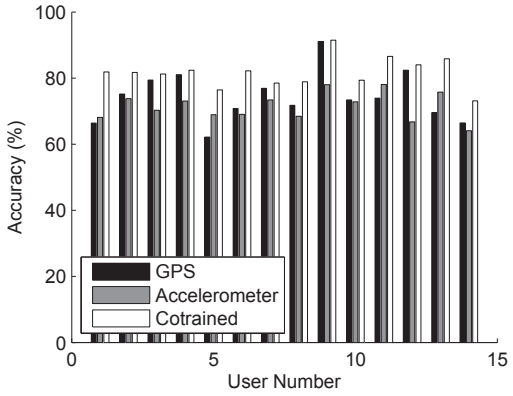



**Figure 6. Co-training advantage for driving activity detection.**

One of the reasons for using GPR and co-training was that this automatically maintains a quantitative measure of uncertainty. As an illustration, the uncertainty, measured as the standard deviation, $\sigma$, is plotted in Figure 7 for sleeping activity detection. The data points are sorted by the uncertainty value. Low values of $\sigma$ indicate low uncertainty and high confidence. We use the 20% of the data points from the training data for model adaptation to other users.

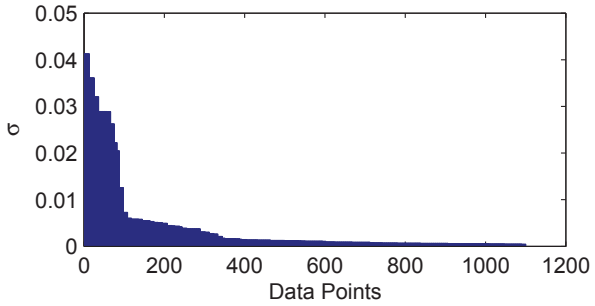



**Figure 7. Uncertainty for one labeled user's data points after GPR based learning.**

### Inter-User Model Adaptation

The bootstrapping performance is studied as follows. We pretend that the app developer collected only one user's data as training data and provided it for the GPR model initialization. We then learned an inference model from this one user, referred to as the training user, and adapted the model individually to the remaining users.

For the driving detection dataset with 14 users, using the first user for training, Figure 8 shows the accuracy achieved by each of the other 13 users through the adapted models. This accuracy is compared to the accuracy that could have been achieved using the ground truth labels for that user. The figure shows that the adapted model performs pretty close to the model that could be learned using ground truth labels. The adapted model is much easier to obtain since it can be learned without any labeled data for the remaining users. The bars at the bottom show the difference in accuracy achieved by the adapted model compared to the model learned using ground truth data.

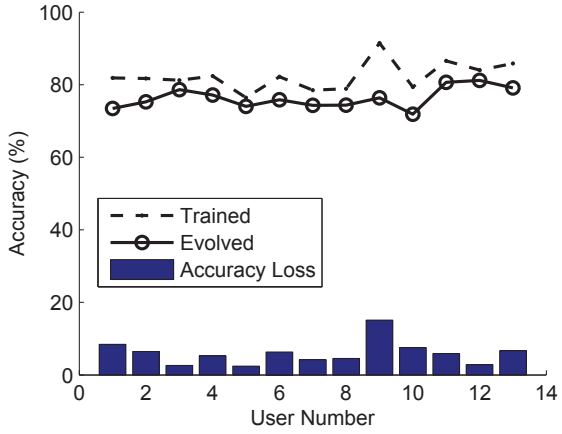



**Figure 8. Accuracy after evolving the model from one user to the 13 others, for driving activity detection.**

The above test was also performed using each of the other users as the training user and adapting the model to the remaining 13 users in each case. These cross-validation results are summarized in Figure 9. The average difference in accuracy is plotted for each of the users used as training users along with the standard deviation across all 13 remaining users. We see that the accuracy loss is less than 10% in all cases.

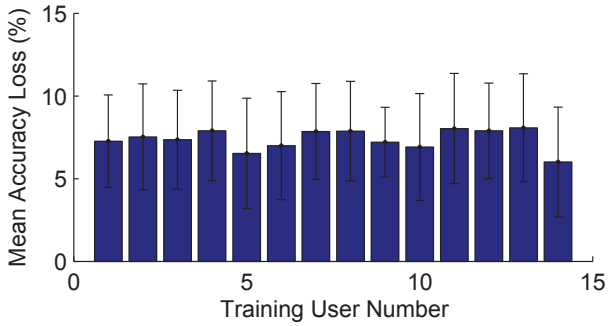
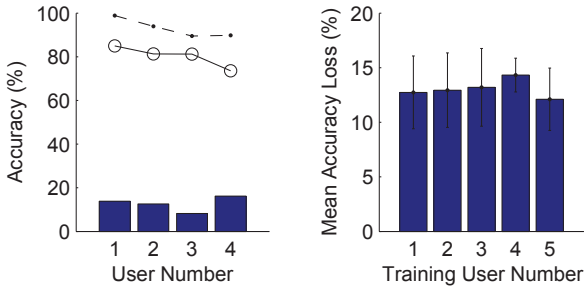



**Figure 9. Cross validation of model adaptation: mean accuracy after adapting model to other users, starting from different users as the training user.**

Performing the same experiments for the detecting sleeping activity, Figure 10(a) shows the performance of the adapted model trained from the first user's data to other users, as well as the the models trained on each user. We see that the adapted model performs close to the model that uses per user labels. Again, performing cross validation across the users in the dataset, Figure 10(b) shows that the the accuracy loss due to adaptation is small.

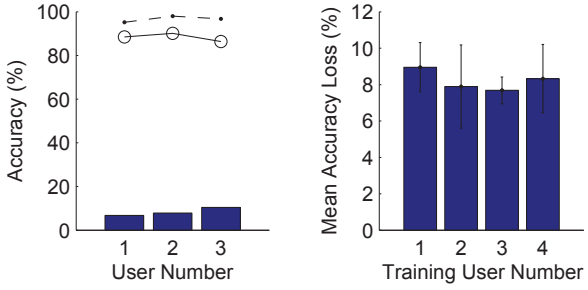
Similarly, for the inference tasks of detecting if the user is at home or work, we perform the same experiments. Figure 11(a) shows an illustrative case for adapting the model trained on the first user to remaining three and Figure 11(b)




(a) From user 1 (b) Cross Validation

**Figure 10. Model adaptation for sleeping activity. The legend for Fig. 10(a) is same as in Fig. 8.**

shows the cross validation result across users. Again the loss in accuracy compared to acquiring ground truth labels for all users is small. One of the users in the dataset always stayed home and is omitted as they had an inference accuracy of 100%.




(a) From user 1 (b) Cross Validation

**Figure 11. Model adaptation for detecting if the user is at home or work. The legend is same as in Fig. 8.**

### Participatory Sensing Application

We now illustrate a participatory sensing application, time use survey, that is enabled using two of the individual activity detection applications on multiple users' phones, namely the sleeping detection and inferring if the user is at home or work. The users initially just deploy the individual applications since they provide useful service to those users such as for heating control [15]. However as the amount of data collected by these individual apps grows, additional apps become feasible.

This participatory app can automate the Time Use Survey [17] conducted by US Department of Labor, previously described with the prototype applications. The participatory app can eliminate the need for manual surveys. Figure 12 shows the statistics generated using our prototype that is bootstrapped off of the data coming from individual activity sensing tasks. Of course, our prototype detects a much smaller number of user states compared to the detailed survey conducted by the US Department of Labor that includes over two dozen different activities in its raw data. However, this prototype already produces important summary statistics that are beneficial for some of the use cases of the time use survey data and illustrates the feasibility of an automated survey with much larger coverage. Interestingly, our data was collected by volunteering graduate students and shows a different work-life pattern than the US average data shown in Figure 2.

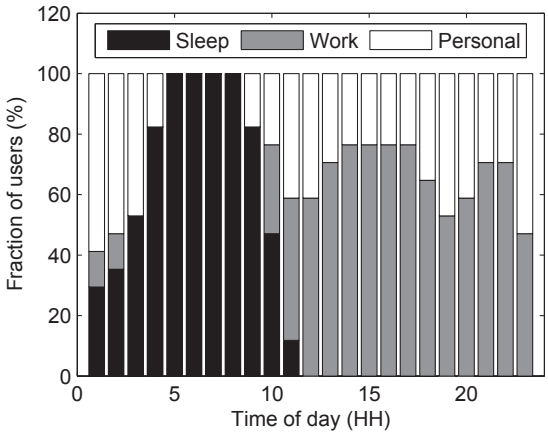



**Figure 12. Participatory sensing: Productivity census.**

## RELATED WORK

The use of sensor data from multiple mobile devices to improve the performance of sensing inference has been considered before in [18]. However, the focus of the work was on improving the robustness of a common model shared among all devices. Semi-supervised learning techniques were employed to recruit additional sensor data from multiple operating environments and multiple devices. The specific semi-supervised learning methods employed very different due to the differences in the nature of the problem constraints and solution requirements. Rather than learning a common model, we are building personalized models for all users. While prior work exploited similarities among users, we adapt to differences. Our focus is on bootstrapping inference trained on a small set of users to multiple other users to reduce the burden on app developers.

Collection of sensor data from multiple mobile devices has been proposed in various platforms and applications [3–6, 18,22–29]. User carried devices have been used for mapping road traffic or potholes to city-wide user movement patterns. Programming infrastructures to support such participatory applications have also been developed [30]. Our goal in this paper is to provide techniques that make it easier to develop. We assist the development of such applications in two ways. First, we make it easier for participatory sensing applications to achieve critical mass since unlabeled data from other applications already installed by users can be used. Second, we enable the participatory app to correctly infer the metric of interest on multiple users even when each user has some

variations in their behavior.

## CONCLUSIONS

We presented methods that enable multiple applications and multiple devices to share their unlabeled sensor data to benefit each other. Inference tasks for existing applications that have been trained on a small number of users can be adapted to work for new users using unlabeled data from those users supplied by other applications running on those new user devices. The developer is relieved of the effort to gather training data on a large number of users and can bootstrap their app with a small initial user base. The key intuition behind adaptation of the inference models is based on semi-supervised learning methods. We selectively apply the highest confidence data points in the highest confidence sensing dimensions to seed the clustering methods for the new users.

The sharing of data for bootstrapping the inference tasks for multiple applications has the additional benefit of consolidating data in a shared repository. This can automatically enable many participatory sensing applications that require sensor data collected by a large number of users through other applications previously installed by the users.

While the presented techniques are prototyped and evaluated on multiple inference tasks using real world data, we realize that their widespread usage requires testing across a much larger set of applications and inferences. The current evaluation results indicate that the direction of research and the initial methods designed to solve the bootstrapping problem are promising and worth developing further.

## REFERENCES


1. Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T. Campbell, "A survey of mobile phone sensing," *Comm. Mag.*, 2010.
2. Robert LiKamWa, Yunxin Liu, Nicholas Lane, and Lin Zhong, "Can your smartphone infer your mood?," in *PhoneSense*, 2011.
3. Baik Hoh, Marco Gruteser, Ryan Herring, Jeff Ban, Daniel Work, Juan-Carlos Herrera, Alexandre M. Bayen, Murali Annavaram, and Quinn Jacobson, "Virtual trip lines for distributed privacy-preserving traffic monitoring," in *ACM MobiSys*, 2008.
4. Andreas Krause, Eric Horvitz, Aman Kansal, and Feng Zhao, "Toward community sensing," in *IPSN*, 2008.
5. Suhas Mathur, Tong Jin, Nikhil Kasturirangan, Janani Chandrasekaran, Wenzhi Xue, Marco Gruteser, and Wade Trappe, "Parknet: drive-by sensing of road-side parking statistics," in *ACM MobiSys*, 2010.
6. Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan, "The pothole patrol: using a mobile sensor network for road surface monitoring," in *ACM MobiSys*, 2008.
7. Eric C. Larson, TienJui Lee, Sean Liu, Margaret Rosenfeld, and Shwetak N. Patel, "Accurate and privacy preserving cough sensing using a low-cost microphone," in *ACM UbiComp*, 2011.
8. D. Rudchenko, T. Paek, and E. Badger, "Text text revolution: A game that improves text entry on mobile touchscreen keyboards," in *Proceedings of PERVASIVE*, 2011.
9. State Farm Insurance, "Driver feedback iPhone app," `http://www.statefarm.com/mobile/driverfeedback/driverfeedback.asp`.
10. Florian Alt, Dagmar Kern, Fabian Schulte, Bastian Pfleging, Alireza Sahami, and Albrecht Schmidt, "Enabling micro-entertainment in vehicles based on context information," in *AutomotiveUI*, 2010.
11. Jie Yang, Simon Sidhom, Gayathri Chandrasekaran, Tam Vu, Hongbo Liu, Nicolae Cecan, Yingying Chen, Marco Gruteser, and Richard P. Martin, "Detecting driver phone use leveraging car speakers," in *ACM MobiCom*, 2011.
12. Zhenyun Zhuang, Kyu-Han Kim, and Jatinder Pal Singh, "Improving energy efficiency of location sensing on smartphones," in *ACM MobiSys*, 2010.
13. James Reinebold, Harshvardhan Vathsangam, and Gaurav Sukhatme, "Inactivity recognition: Separating moving phones from stationary users," in *PhoneSense*, 2011.
14. W. Karlen, C. Mattiussi, and D. Floreano, "Adaptive sleep/wake classification based on cardiorespiratory signals for wearable devices," in *IEEE BIOCAS*, 2007.
15. J. Scott, A.J.B. Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges, and N. Villar, "Preheat: controlling home heating using occupancy prediction," in *ACM Ubicomp*, 2011.
16. D. Chu, A. Kansal, J. Liu, and F. Zhao, "Mobile apps: Its time to move up to condos," in *HotOS*. USENIX Association, 2011.
17. United States Department of Labor, "American time use survey," `http://www.bls.gov/tus/overview.htm`.
18. Emiliano Miluzzo, Cory T. Cornelius, Ashwin Ramaswamy, Tanzeem Choudhury, Zhigang Liu, and Andrew T. Campbell, "Darwin phones: the evolution of sensing and inference on mobile phones," in *ACM MobiSys*, 2010.
19. Carl Edward Rasmussen and Christopher K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.
20. Shipeng Yu, Balaji Krishnapuram, Rmer Rosales, Harald Steck, and R. Bharat Rao, "Bayesian co-training," *Journal of Machine Learning Research*, , no. 12, pp. 2649–2680, 2011.
21. Sugato Basu, Arindam Banerjee, and R. Mooney, "Semi-supervised clustering by seeding," in *ICML*, 2002.
22. Tarek Abdelzaher, Yaw Anokwa, Peter Boda, Jeff Burke, Deborah Estrin, Leonidas Guibas, Aman Kansal, Samuel Madden, and Jim Reich, "Mobiscopes for human spaces," *IEEE Pervasive Computing*, April 2007.
23. Deborah L. Estrin, "Participatory sensing: applications and architecture," in *ACM MobiSys*, 2010.
24. Shravan Gaonkar, Jack Li, Romit Roy Choudhury, Landon Cox, and Al Schmidt, "Micro-blog: sharing and querying content through mobile phones and social participation," in *ACM MobiSys*, 2008.
25. Nokia Research Center, "Sensing the world with mobile devices," Tech. Rep., Nokia Research Center, December 2008.
26. Tomas Gerlich, James Biagioni, Timothy Merrifield, and Jakob Eriksson, "Tracking transit with easytracker," in *SenSys*, 2011.
27. C. Ratti, A. Sevtsuk, and S. Huang ans R. Pailer, "Mobile landscapes: Graz in real time," in *Proceedings of the 3rd Symposium on LBS and TeleCartography*, November 2005.
28. Xuan Bao and Romit Roy Choudhury, "Movi: mobile phone based video highlights via collaborative sensing," in *ACM MobiSys*, 2010.
29. Aman Kansal, Michel Goraczko, and Feng Zhao, "Building a sensor network of mobile phones," in *IPSN*, 2007.
30. Moo-Ryong Ra, Bin Liu, Tom La Porta, and Ramesh Govindan, "Medusa: A programming framework for crowd-sensing applications," in *ACM MobiSys*, 2012.