Privacy-aware Regression Modeling of Participatory Sensing Data

Hossein Ahmadi[†], Nam Pham[†], Raghu Ganti[‡], Tarek Abdelzaher[†], Suman Nath[§], Jiawei Han[†]
hahmadi2@uiuc.edu, nampham2@uiuc.edu, rganti@us.ibm.com, zaher@uiuc.edu,
sumann@microsoft.com, hanj@uiuc.edu

Abstract

Many participatory sensing applications use data collected by participants to construct a public model of a system or phenomenon. For example, a health application might compute a model relating exercise and diet to amount of weight loss. While the ultimately computed model could be public, the individual input and output data traces used to construct it may be private data of participants (e.g., their individual food intake, lifestyle choices, and resulting weight). This paper proposes and experimentally studies a technique that attempts to keep such input and output data traces private, while allowing accurate model construction. This is significantly different from perturbation-based techniques in that no noise is added. The main contribution of the paper is to show a certain data transformation at the client side that helps keeping the client data private while not introducing any additional error to model construction. We particularly focus on linear regression models which are widely used in participatory sensing applications. We use the data set from a map-based participatory sensing service to evaluate our scheme. The service in question is a green navigation service that constructs regression models from participant data to predict the fuel consumption of vehicles on road segments. We evaluate our proposed mechanism by providing empirical evidence that: i) an individual data trace is generally hard to reconstruct with any reasonable accuracy, and ii) the regression model constructed using the transformed traces has a much smaller error than one based on additive data-perturbation schemes.

Categories and Subject Descriptors

K.4.1 [**Computing Milieux**]: Computers and Society–*Privacy*; G.0 [**Mathematics of Computing**]: General

General Terms

Algorithms, Design, Experimentation

Keywords

Privacy, Participatory sensing, Linear regression

1 Introduction

This paper develops a privacy-aware scheme for sharing participatory sensing data towards the computation of generalizable models from the data pool. We consider applications where construction of such models requires data that are private, whereas the models themselves are not. For example, individuals might not want to share their weight, food intake, and exercise habits on daily basis, yet a quantitative model on how food intake, weight loss, exercise, and other lifestyle choices are generally related could be of interest to many.

Participatory sensing applications have recently become a popular trend in providing community-wide services that "crowdsource" the task of data sensing [1, 7, 16, 23]. One can generally distinguish between at least two types of services from the perspective of the end-result computed by the service. The first type represents services where some statistic is computed from input data. For example, the service might compute the quality of air, speed, potholes, or pollen levels on city streets. These statistics pertain only to the neighborhoods where data was collected. It is not the purpose of the service to generalize; the existence of potholes or pollution in one neighborhood does not necessarily entail their existence in a different neighborhood.

In this paper, we consider a different type of participatory sensing applications; one where the collected data are multi-dimensional and sparse. Hence, the objective is to generalize from data we collected to extrapolate and predict data we do not have. An example might be to collect, in addition to pollution data, some context information as well, such as population density and eco-friendly behaviors of the population. This can then be used to build a model relating these factors to pollution levels and use it to predict pollution elsewhere, where we do not have measured pollution data. We particularly consider linear regression modeling since it has already been employed for similar purposes in participatory sensing

*This work was funded in part by NSF Grant 1040380.

†Department of Computer Science, University of Illinois at Urbana-Champaign.

‡IBM T. J. Watson Research Center.

§Networked Embedded Computing Group, Microsoft Research.

SenSys'10, November 3–5, 2010, Zurich, Switzerland.

applications (e.g., [19]). It is often the case, that the general models thus computed are themselves not private, but some or all inputs of those models are private data.

Many prospective applications fall into this category. For example, consider a participatory sensing service that models the energy consumption of a household based on the usage of various appliances and the time of year in order to help users save on the energy costs. Although the constructed model (for energy consumption) is public, the input and output values contain users' private information that cannot be easily shared; a user may not wish to reveal the usage of the appliances at specific times. This paper focuses on increasing the privacy of individual user data while enabling community-wide data *modeling* for prediction and extrapolation purposes.

Several options exist for sharing private data. The most popular include ensuring anonymity of the data source. To make service architecture simple and boost trust, we instead allow users to share their identities with the aggregation server openly but give them means to alter their data in a way that makes the original values hard to recover. It remains an open issue whether the average user will be more trusting of security mechanisms that promise to fully anonymise their identity or data alteration mechanisms that promise to irrecoverably alter their data. One advantage of the latter is that it can be entirely implemented on the client side, whereas some of the former need support from other nodes as well. Whether the perceived difference in the needed trust base is of consequence to the average user remains to be seen, which is an argument for exploring both types of approaches and letting deployment experience decide on their relative usability advantages. This paper investigates the data alteration approach.

Additive data perturbation (noise) is the most common way to alter data for privacy [4, 17, 20, 34]. The main idea behind the perturbation is to mask each individual user data with noise in such a way that the noise cancels out or can be decomposed in aggregate calculations. However, error is always introduced to the modeling process depending on the number of data streams and type of the application. Our approach is different in that it aims to introduce no additional error in modeling, regardless of the number of users or data streams in the system. We should note that the perturbation schemes apply to a more general class of models than linear regression models, as long as the noise distribution follows closely the user data distribution.

We transform the sensitive data at the client side to a set of aggregate features that usually do not reveal much information on original data. Community-wide models that are computed from these features are exactly the same as the models computed from the original community data. Users can control their degree of privacy by controlling the amount of aggregation involved in feature computation. To compute the features, time-series data are first segmented based on some prior knowledge about the system. The segmentation is on intervals long-enough to eliminate short-scale dynamics of the system. This way, the system can be represented as an approximately static (time-independent) relation between various attributes of a data segment. Next, specific feature

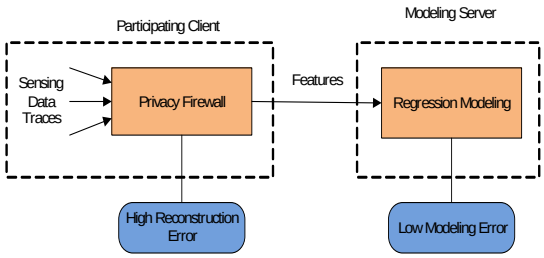


Figure 1. A privacy-aware participatory sensing model.

matrices are extracted from the data segments. They represent the information shared by each user with the community. Although our paper lacks formal proofs of its privacy properties, it presents intuitions and empirical evidence on why it is often hard to reverse the process and recover private user data based on the information shared.

We evaluate our technique using a sensory data set [21] collected for a green navigation service from current literature [19]. This map-based service enables finding the most fuel efficient path between two points for a specific car. The service relies on data collected from vehicles equipped with a diagnostic fuel-efficiency scanner device. Results suggest that in most cases user data cannot be recovered without significant error, while model reconstruction is always correct.

The rest of this paper is organized as follows: We present the motivation and the exact definition of the problem in Section 2. Section 3 describes the detailed steps involved in the privacy framework. In Section 4, we present the computation of a linear regression model using the data shared by users. We show that our proposed scheme does not change modeling error compared to model construction from the original raw private data. We discuss the privacy protection of such transformations in Section 5. In Section 6, we evaluate the privacy achieved by our technique using a case study based on the data set from a green navigation service. Section 7 discusses the potential cases leading to privacy breaches and open questions regarding the approach. We discuss related work on privacy in participatory sensing applications in Section 8. Finally, the paper concludes with Section 9.

2 Problem Formulation

Many participatory sensing applications calculate community statistics using data shared by the community [7]. Previous privacy-preserving approaches mostly focused on deriving community statistics while keeping individual data traces private [20, 34]. This paper considers a different category of participatory sensing applications where the application extrapolates and predicts unavailable data using models of the system.

Given the nature of applications we target, the following summarizes the assumptions and requirements of our scheme:

- The model relating user inputs to the outputs is public.
- Each data sample collected by an individual is private and may not be revealed.
- The models used in the service are linear in coefficients.
- The time-series data can be packed into uncorrelated

data samples by aggregation (over time for example).

2.1 Modeling in Participatory Sensing Applications

Consider an example participatory sensing service aimed to reduce household energy consumption. Each user collects the following data traces daily: i) the total time that various appliances are in use (e.g., TV), ii) the temperature inside the house, and iii) the outside temperature. By sharing the data, a regression model can be constructed and used to predict the energy consumption for a given usage pattern and season. Eventually, the service helps other residents to save on energy costs by adapting their appliance usage.

Figure 1 illustrates our model of privacy-aware participatory sensing applications that construct a model of user data. In our architecture, data traces collected by participants are transformed on the client side to *neutral features*. The data can typically be divided into a variable we are interested in (e.g., household energy consumption) and other variables we believe are good predictors of the former. The objective is to compute a model relating the predictors (model input) to the variable of interest (model output). The neutral features computed from user data do not easily reveal user data and can thus be made available to any entity that needs them.

The process of attempting to compute the private data from the features is called *reconstruction*. An important question is: given the information that each user sends to the service, how accurately it is possible to reconstruct the values in the original data traces? To measure privacy, we calculate the sum of squared difference between each value in the original trace and the corresponding value in the reconstructed trace. A lower value means that the reconstructed trace is more similar to the original one and reveals more private information about the user.

The participatory sensing service includes a server that collects the neutral features of all participants to compute the regression model of the underlying measured phenomenon. The model has a predictive property and can thus be of interest to a broad population besides the participants who contributed data to its construction.

2.2 Design Goals and Metrics

There are two challenges in designing a system according to our model. First, we need a scheme that converts the data traces to features that reduces reconstruction accuracy of the data traces. Second, we need to perform the modeling using the shared features instead of the data traces themselves. This immediately leads to the two objectives for the system:

- **Minimize the modeling error**: It is desirable to reach the same level of modeling accuracy as one attained in a system not employing any data alterations.
- **Maximize the reconstruction (breach) error**: The higher the error in reconstruction of individual user data, the more it is ensured that the privacy of the user is not breached.

We define a *perfect* privacy-enabling data sharing scheme in participatory sensing applications to be a scheme that satisfies two conditions:

- *Perfect modeling:* Model construction from shared data produces exactly the same model as if the original private data traces were used.
- *Perfect neutrality:* Reconstruction of private user data from shared data yields the same error as if no additional information was available to the outside world (i.e., to an attacker) besides the computed public phenomenon model. Note, however, that the neutrality condition does not exclude information leaks that result from computing the public model itself. For example, if the model suggests that all adults in some population are between 5 and 6 feet high, then something is leaked about individual user data in that population. A perfectly neutral scheme should not introduce any *additional* leaks.

This paper presents and empirically evaluates a privacy-aware scheme for applications that aim to compute a linear regression model of some measured phenomenon. Our scheme satisfies the perfect modeling condition while empirical evidence suggests that it is also neutral in many cases.

3 Privacy Filter

A participant in a participatory-sensing application uses various sensing devices to collect data samples about a phenomenon (e.g., a thermometer to measure temperature inside a house, a GPS device to measure location, or an OBD-II port to measure vehicular fuel consumption). A privacy filter converts such data to features to be used for phenomenon modeling. In order to attain the goals of a perfect privacy-enabling scheme, our approach is first to convert the traces into a set of uncorrelated samples, we call segments. This is to foil correlation-based attacks. Clients then report neutral features computed over a set of segments. They are used later for model construction.

This section details the various algorithmic steps involved on the client side and presents the structure of the features which are sent to the server. Later in Section 4, we describe how our scheme achieves the perfect modeling property. In Section 5, we present empirical evidence to support some level of neutrality for our scheme and describe how it can be used to personalize the level of privacy provided.

The data collected by a user is usually in form of time-series. For example, a sensing device might record the value of temperature every second. However, the phenomenon model is typically concerned with predicting quantities at longer time-scales. For example, how much gas a vehicle will spend on a given route? How much energy a household will save if they installed motion-activated light controls? How much weight a 300lb person might lose if engaged in a particular diet and exercise routine? At those time-scales, fast system dynamics are averaged out. The aggregate output of a system in a sufficiently large time-interval is more correlated with the aggregate input in that interval and not what happens in other intervals. For example, one's weight loss in one month might be more correlated with food intake in that month, rather than the month before. This is not necessarily true of fluctuations at shorter timescales. This process of aggregation over appropriate timescales is the first step in our client-side data alteration. It is called *data segmentation*. The segment aggregate can be the sum (e.g. of consumed energy) or in some cases the average. The main reason to do

the segmentation is to eliminate correlations between samples taken at different times.

This section first describes how the segmentation is done and why the segmented data cannot be shared without possibly violating privacy of the individual. The next step in our scheme is to convert the segmented data into neutral features that are less of a threat to users' privacy.

3.1 Data Segmentation

In order to perform segmentation on raw data traces, we first need to decide on the time interval to use. This often requires application specific knowledge. There are three criteria involved: i) a large enough time interval ensures that the system can be described using a static model and hence remove correlations among samples, ii) the time interval should result in an accurate prediction, and iii) the time interval should be usable by the participatory-sensing service.

In our household energy consumption example, various time intervals such as a day, a month, or a year ensure a static description of the system and so the first property is achieved. Very small time intervals like daily energy consumption usually highlight the effect of modeling noise and therefore fail to achieve the second property. On the other hand, the energy consumption over a year may not be as useful since laws of large numbers may make such averages converge, resulting in ill-condition matrices. Hence, considering the data collected by each user in a given application, as illustrated in the previous example, the data might be segmented into one-month intervals before being shared with the community. Table 1 presents sample values for such segmentation. The above discussion is to present general guidelines. Actual segmentation period will vary substantially from one application to the next.

The result of the segmentation is a set of n data points. Each data point consists of d input values corresponding to model inputs (input dimensions) and a single output value. Note that, by doing the segmentation, we actually remove the time attribute from all of the data and make each data point time-independent. In particular, there is no particular order maintained for the segmented data. We use y_i to denote the value of the output attribute in the ith segment (e.g., energy consumption) and x_{ij} to denote the value of j-th input of segment i. Formally, an appropriate time interval for the segmentation ensures that y_i can be estimated accurately using:

$$\hat{y}_i = f(x_{i1}, \ldots, x_{id})$$

Although the data segmentation performs aggregation on the original data trace, sharing raw segments can result in a breach of privacy. For example, Table 1 shows appliance usage and temperature inside a house each month. Now, these values can easily show whether a residence is occupied or not in a particular month. Therefore, we need to take another step and only share some features of the segmented data that are less likely to threaten privacy. This is presented next.

3.2 Neutral Features

A multi-dimensional linear regression model relates an output variable to several predictor variables or simply inputs. Consider the segmented data from above and the notation used to represent input and output values in each data point. Let $Y = \{y_i : 1 \leq i \leq n\}$ and $X = \{x_{ij} | 1 \leq i \leq n, 1 \leq j \leq d\}$. We should note that a linear regression model is only linear with respect to the regression coefficients. The predictor variables can be any arbitrary function of the input attributes. Several general purpose regression techniques explore a range of feature spaces (e.g., quadratic features where the predictors are product of two input attributes). To generalize, we use $w_{ij} = g_j(x_{i1}, \ldots, x_{id})$ and denote $W = \{w_{ij} | 1 \leq i \leq n, 1 \leq j \leq k\}$. A linear regression model of this system describes the system using:

$$Y = W\eta + \varepsilon$$

where ε is assumed to be a zero-mean error term with a constant variance σ^2. η is the model coefficient matrix and can be estimated by various regression techniques [29]. Consequently, the output estimate, $\hat{Y}$, is given using:

$$\hat{Y} = W\eta$$

In our household energy consumption example, g_j is simply an identity function (i.e. $W = X$). In many cases, sharing W instead of X does not resolve the issue of privacy since g_j's are simple reversible functions making X discoverable from any given W.

Since the goal of our system is to calculate η for the whole community, a simple idea is that each user computes the η using its local information and only shares the regression coefficients. There is no way of reconstructing the values of X and Y only by knowing the function relating them. However, it is also impossible to combine the partially-calculated models (η's) and obtain the global model without extra information.

What is needed are the features that represent correlations between different attributes. Our idea is that a correlation matrix reveals very limited information about the data trace but has enough information to be used for regression modeling. Let X_u and Y_u represent the data corresponding to each user u. Let W_u be the predictor matrix corresponding to user u. We define the *neutral feature matrices* of the data collected by user u as follows:

- $\rho_u = Y_u^T Y_u$
- $\nu_u = W_u^T Y_u$
- $\Theta_u = W_u^T W_u$

We first observe that none of the matrices used in the above definition depends on the number of samples collected by the user. In other words, *regardless of the number of samples the user has collected, the same amount of data is sent to the server for modeling*. As we see later in Section 5, this property helps enabling users' privacy. Moreover, this simple property enables users to achieve a personalized level of privacy. A user who shares more data samples in a single transaction achieves a higher level of privacy.

The computational cost associated with extracting the feature matrices is proportional to the square of the number of samples and number of variables, n^2 and k^2. Note that the number of samples shared by a user can be as large as n in the

Table 1. A sample segmented data set.

Month	Elec. Cons. (MWh)	Avg. Appliance Usage (hours)	Avg. Inside Temp.	Avg. Outside Temp.
Jul.	1.230	2.5	74	79
Aug.	0.870	3.9	72	73
Sept.	1.00	1.5	72	70
Oct.	1.45	1.2	71	56
Nov.	2.1	3.4	70	44
Dec.	2.75	2.3	70	26

worst-case computation analysis. Calculating Θ_u is the most computationally expensive transformation requiring $O(k^2n^2)$ operations. Calculating ρ_u and n_u requires $O(n^2)$ and $O(kn^2)$ operations respectively. The data segmentation is linear in terms of the number of segments (n) and therefore is dominated by feature extraction process.

The following example illustrates how the feature matrices are obtained from the data and shared.

EXAMPLE 1 (FEATURES). *Given the segmented data in Table 1, the client calculates the following feature matrices:*

$$\rho_u = [1.23 \quad 0.87 \quad 1.00 \quad 1.45 \quad 2.10 \quad 2.75] \\ [1.23 \quad 0.87 \quad 1.00 \quad 1.45 \quad 2.10 \quad 2.75]^T = 17.3448$$

$$\nu_u = \begin{bmatrix} 2.5 & 74.0 & 79.0 \\ 3.9 & 72.0 & 73.0 \\ 1.5 & 72.0 & 70.0 \\ 1.2 & 71.0 & 56.0 \\ 3.4 & 70.0 & 44.0 \\ 2.3 & 70.0 & 26.0 \end{bmatrix}^T \begin{bmatrix} 1.23 \\ 0.87 \\ 1.00 \\ 1.45 \\ 2.10 \\ 2.75 \end{bmatrix} = \begin{bmatrix} 23.173 \\ 668.11 \\ 475.78 \end{bmatrix}$$

$$\Theta_u = \begin{bmatrix} 2.5 & 74.0 & 79.0 \\ 3.9 & 72.0 & 73.0 \\ 1.5 & 72.0 & 70.0 \\ 1.2 & 71.0 & 56.0 \\ 3.4 & 70.0 & 44.0 \\ 2.3 & 70.0 & 26.0 \end{bmatrix}^T \begin{bmatrix} 2.5 & 74.0 & 79.0 \\ 3.9 & 72.0 & 73.0 \\ 1.5 & 72.0 & 70.0 \\ 1.2 & 71.0 & 56.0 \\ 3.4 & 70.0 & 44.0 \\ 2.3 & 70.0 & 26.0 \end{bmatrix} = \\ \begin{bmatrix} 42 & 1058 & 863.8 \\ 1058 & 30685 & 25018 \\ 863.8 & 25018 & 22218 \end{bmatrix}$$

4 The Application Server

Given the feature matrices sent by the client, this section describes how the application server constructs a regression model. Also, we discuss how the first objective in designing a privacy-enabling scheme is achieved. Assuming that all the segmented data, (Y and W) are available to the server, several regression techniques can be used to obtain the regression coefficients [29]. Again, consider the regression model $Y = W\eta + \varepsilon$. Assuming that ε follows an unknown distribution, a Least Squared Estimator (LSE) is the best linear estimator of Y. LSE is an unbiased estimator [29] meaning that it has the same mean as the true regression coefficients. In this setting, the LSE is obtained using:

$$\eta = (W^T W)^{-1} W^T Y \tag{1}$$

Since the segmented data are not available to the server, we need an alternative approach to derive η only using the shared features. An important characteristic of the feature matrices is that they are *distributive*. One can obtain the feature matrices of a community simply by adding the feature matrices of sub-communities. Let $u_1, \ldots, u_m$ be the m users of the participatory sensing application. Each user contributes matrices ρ_{u_i}, ν_{u_i}, and Θ_{u_i} to the community based on its private data W_{u_i}, and Y_{u_i}. We can write $W = [W_{u_1} \mid \cdots \mid W_{u_m}]^T$ and $Y = [Y_{u_1} \mid \cdots \mid Y_{u_m}]^T$. These are the complete set of sensing values collected by the users, which indeed are not available to the server. We define ρ, ν, and Θ as follows:

$$\rho = Y^T Y = [Y_{u_1}^T \mid \cdots \mid Y_{u_m}^T] \begin{bmatrix} Y_{u_1} \\ \vdots \\ Y_{u_m} \end{bmatrix} = \sum_{i=1}^{m} Y_{u_i}^T Y_{u_i} = \sum_{i=1}^{m} \rho_{u_i}$$

$$\nu = W^T Y = [W_{u_1}^T \mid \cdots \mid W_{u_m}^T] \begin{bmatrix} Y_{u_1} \\ \vdots \\ Y_{u_m} \end{bmatrix} = \sum_{i=1}^{m} W_{u_i}^T Y_{u_i} = \sum_{i=1}^{m} \nu_{u_i}$$

$$\Theta = W^T W = [W_{u_1}^T \mid \cdots \mid W_{u_m}^T] \begin{bmatrix} W_{u_1} \\ \vdots \\ W_{u_m} \end{bmatrix} = \sum_{i=1}^{m} W_{u_i}^T W_{u_i} = \sum_{i=1}^{m} \Theta_{u_i}$$

Here, ρ, ν, and Θ are calculated from the shared features. These are the only information that the server use to construct a model. To do that, we can rewrite (1) and make the coefficients only depending on the feature matrices:

$$\eta = (W^T W)^{-1} W^T Y = \Theta^{-1} \nu \tag{2}$$

The above equation enables the server to calculate the regression coefficient only using the shared features by $\eta = \Theta^{-1}\nu$. In many cases, the server also needs to calculate the regression error. Again, assuming to have access to the whole data set, the error can be derived as follows:

$$Err = (Y - W\hat{\eta})^T (Y - W\hat{\eta})$$

Similar to the derivation of η, we need to rewrite this equation so that Err can be calculated only using ρ, ν, and Θ:

$$Err = Y^TY - (X\hat{\eta})^TY - Y^TX\hat{\eta} + (X\hat{\eta})^TX\hat{\eta} = \rho - \hat{\eta}^T\nu - \nu^T\hat{\eta} + \hat{\eta}^T\Theta\hat{\eta} \quad (3)$$

The above process derives the values of η and Err without requiring access to the users' data and by only using the shared features. Meanwhile, the derivation produces exactly the same results as if having access to the raw data. Therefore, our scheme successfully achieves the first design objective that is not to impose any additional modeling error.

We can observe that the modeling process employed at the server is computationally efficient. Specifically, computing ρ, ν, and Θ need $O(nk^2)$ operations. The reason being that each matrix addition takes at most $O(k^2)$ time (size of the largest matrix, Θ) and we can have at most n users in the system hence n additions. Calculating the regression coefficients needs matrix inversion of a $k \times k$ matrix leading to an $O(k^3)$ computation time. Error calculation takes $O(k^3)$ to complete as well. Since n is much larger than k in most cases, the total computation time is dominated by $O(nk^2)$. The server-side computation is significantly more efficient than modeling the whole raw data set where users do not use a privacy firewall (which is $O(n^3)$).

5 Privacy Analysis

Our approach to study how the method presented in Section 3 helps enabling privacy is to show that is not likely to reconstruct user data accurately and efficiently. We first formulate our measure of privacy as the amount of error incurred while the best reconstruction of the data is done. In particular, let y_i and $w_{i1}, \ldots, w_{ik}$ be the segmented data that are transformed into the feature matrices. Let $\tilde{y}_i$ and $\tilde{w}_{i1}, \ldots, \tilde{w}_{ik}$ be the values in the reconstructed set. We define the reconstruction error for each attribute j to be

DEFINITION 1 (RECONSTRUCTION ERROR). *The reconstruction error of $\mathbf{w_j}$ is the normalized sum of squared difference between the data set and the reconstructed trace:*

$$RE(\mathbf{w_j}) = \frac{\sum_{i=1}^{n}(\tilde{w}_{ij} - w_{ij})^2}{\sigma_j^2}$$

where σ_j^2 is the variance of the variable $\mathbf{w_j}$.

In order to measure the privacy, consider the case that no information is shared by the user. The mean of the variables (i.e. $\tilde{y}_i = E[\mathbf{y}], \tilde{w}_{i1} = E[\mathbf{w_1}], \ldots, \tilde{w}_{ik} = E[\mathbf{w_k}]$) are the best estimators when no observations are available. The resulting reconstruction error is simply:

$$RE(\mathbf{w_j}) = \frac{\sum_{i=1}^{n}(E[\mathbf{w_j}] - w_{ij})^2}{\sigma_j^2} = \frac{\sigma_j^2}{\sigma_j^2} = 1 \quad (4)$$

A reconstruction is effective if the reconstruction error for some variable is less than the error resulted from using the mean value. In other words, if the reconstruction error is greater than or equal to 1, it means that the transformed matrices have almost no useful information:

DEFINITION 2 (PRIVACY-ENABLING TRANSFORMATION). *A transformation is called privacy-enabling if the reconstruction error is always greater than or equal to* 1.

In this section, we study the privacy-enabling properties of the scheme proposed in Section 3. Our approach is to first discover how an attacker can infer information about the private data traces based on the features. We formalize this problem and call it optimal reconstruction. We discuss the accuracy and complexity of an optimal reconstruction scheme. Next, we use empirical evidence to support our heuristics about the conditions under which the privacy is most likely preserved.

5.1 Privacy-Enabling Properties

In order to study how our approach tries to preserve users' privacy, we show that it is hard to accurately reconstruct the private data for user u (i.e., Y_u and W_u) from ρ_u, ν_u, and θ_u. Given the transformed matrices and their relation to the user data, there are a set of values (or a subspace) that satisfies the relation and produces the same matrices. Since the only information available from the user are the transformed matrices, an optimal reconstruction should pick the most likely values among those which satisfy the transformation. For example, it is much more probable for a user to drive a $3000lbs$ vehicle rather than a $500lbs$ vehicle. Therefore, when both values satisfy the transformation, $3000lbs$ is more likely to be the accurate reconstruction. We formally define the optimal reconstruction as follows:

DEFINITION 3 (OPTIMAL RECONSTRUCTION). *An* Optimal Reconstruction *is to find the values $\tilde{Y}_u$ and $\tilde{W}_u$ that produce the given transformed matrices ρ_u, ν_u, Θ_u while maximizing the joint probability of observing such values.*

We assume that distribution information is available to the attacker. This usually comes from publicly available knowledge about the measured variable. For example, the weight distribution of vehicles sold in the US may be a public piece of information available to the attacker. The optimal reconstruction can be formulated as the solution to the following optimization problem:

$$\begin{aligned} \max \quad & \Pi_{i=1}^{n} p(y_i, w_{i1}, \ldots, w_{ik}) \quad (5) \\ \sum_{i=1}^{n} y_i^2 &= \rho_u \\ \sum_{i=1}^{n} y_i w_{ji} &= \nu_u(j) \quad : \quad \forall 1 \le j \le k \\ \sum_{i=1}^{n} w_{j_1 i} w_{j_2 i} &= \Theta_u(j_1, j_2) \quad : \quad \forall 1 \le j_1, j_2 \le k \end{aligned}$$

where $p(y, w_1, \ldots, w_k)$ is the probability of observing values $y, w_1, \ldots, w_k$ which may be known to the attacker as prior knowledge. We emphasize that likelihood maximization is a key to the reconstruction. Here, we assume that data from any two segments shared by the user are independent. This is the case because each segment can belong to an arbitrary

time and location and is uncorrelated to others due to segmentation. Note that the optimal reconstruction does not directly calculate values of model inputs, X, but rather the values of matrix W. However, in many applications the exact values of X can be derived from W. In Section 6, we discuss the details of such derivation when doing a case study.

Another important point here is that the value of n and consequently the number of variables to be reconstructed is not available to the attacker in most of the cases. To overcome this, one may try several guesses of n and choose the reconstruction with the maximum likelihood. Perhaps, starting with a value of n that makes the number of constraints and variables equal (i.e. $n = \frac{k}{2} + 1$) and increasing n until the value of the objective function is minimized.

Next in this section, we discuss why one cannot easily reconstruct the original values from the transformed matrices. We first describe why given a large enough n, the reconstruction accuracy is expected to be low. Then, we argue that finding an accurate reconstruction becomes computationally expensive for large values of n.

5.1.1 *Inaccuracy of Reconstruction*

We first discuss the following question: If someone succeeds at finding an optimal reconstruction, how close are the reconstructed data points to the actual data trace? Intuitions to answer this question are presented below. Later, we demonstrate an example answer for a specific empirical study.

First observe that the number of constraints in the optimal reconstruction problem is $\frac{k^2+3k}{2} + 1$. When the number of data points $n(k+1)$ is less than the number of constraints, only a single feasible assignment exits. Hence, the reconstruction may be performed with 100% accuracy and no privacy is preserved. On the other hand, when n tends to infinity, the feasible solution space becomes more and more relaxed as the number of constraints remain constant. In this case, the optimal solution converges to $(E[\mathbf{y}], E[\mathbf{w_1}], \ldots, E[\mathbf{w_k}])$. The reconstruction error tends to 1, using a derivation similar to (4). This means that for large enough n, it becomes unlikely to reconstruct the private data.

5.1.2 *Inefficiency of Reconstruction*

Next, we give intuitions on why optimally reconstructing data as in (5) becomes computationally expensive for large values of n and k. We emphasize that our discussion is the worst-case complexity analysis for finding the optimal solution and merely gives a hint regarding the complexity of reconstruction in general. The analysis does not hold for the cases where a non-optimal solution can still be a threat to privacy, where the particular form of features allows for fast computation, or where the values of n and k are always small.

It is obvious that the objective probability distribution should be approximated by a well defined smooth function. Otherwise, the optimal solution cannot be obtained in less than an exponential time even without any constraints. Let us assume that the objective function has a very simple form (e.g., linear or convex quadratic). Still, the constraint space is not convex since the equality constraints are not affine [5]. This means that there is no hope to solve such a problem through convex optimization methods. On the other hand,

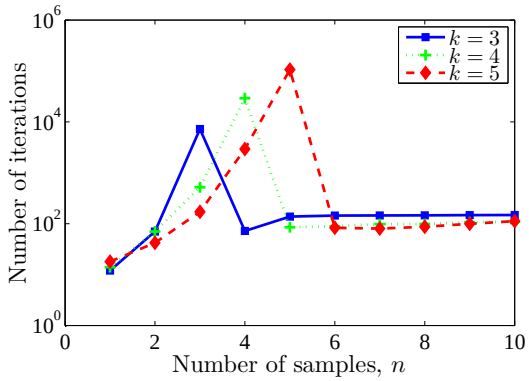


Figure 2. The number of iterations needed for the reconstruction.

any general non-convex quadratically constrained program is shown to be NP-Hard [37]. In fact, any integer linear program can be written as a quadratically constrained optimization problem. Therefore, an optimal reconstruction of the user data trace takes an exponential amount of time with respect to the number of variables, $n(k+1)$, unless $P = NP$.

5.2 Conditions to Protect Privacy

It is important for the privacy firewall to identify the conditions where the privacy of the user data is not protected at all. In particular, this happens when the number of segments used to create features is small. Here, we experimentally study the privacy-enabling properties of feature matrices to derive a heuristic lower bound on the number of segments to be used in the client. The client application simply warns the user if the number of samples used for the transformation is less than the threshold.

Our first step in order to evaluate privacy is to approximate the reconstruction process by using a heuristic model of the objective function. Our heuristic is to assume that the maximum likelihood is obtained when the solution is close to the expected value of each parameter's marginal distribution. Therefore, we minimize the distance of each variable from the expected value of its corresponding distribution. Let $\mathbf{y}$ and $\mathbf{w_i}$ be the random variables corresponding to the output attribute and the predictor variables respectively.

$$\min \sum_{i=1}^{n} [(y_i - E[\mathbf{y}])^2 + (w_{i1} - E[\mathbf{w_1}])^2 + \ldots + (w_{ik} - E[\mathbf{w_k}])^2]$$

$$\sum_{i=1}^{n} y_i^2 = \rho_u$$

$$\sum_{i=1}^{n} y_i w_{ji} = \nu_u(j) \quad : \quad \forall 1 \leq j \leq k$$

$$\sum_{i=1}^{n} w_{j_1 i} w_{j_2 i} = \Theta_u(j_1, j_2) \quad : \quad \forall 1 \leq j_1, j_2 \leq k$$

There are two methods to approximate the above optimization problem. First, the problem can be relaxed into a semi-definite program (SDP) and then the solution of the SDP converted to a feasible solution for our quadratically

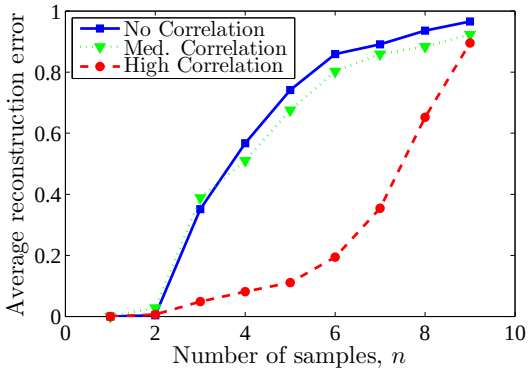


Figure 3. The average reconstruction error of all parameters for different vertical correlations.

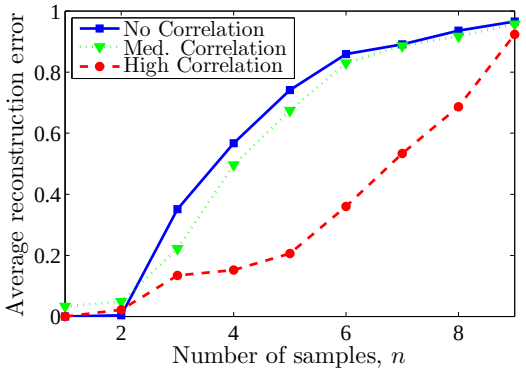


Figure 4. The average reconstruction error of all parameters for different horizontal correlations.

constrained program. However, in most cases, SDP relaxations needs a specific form of constraints. The second approach is to use a non-linear optimizer based on interior-point barrier or sequential quadratic programming. We use KNITRO [28] non-linear solver package in our implementation. This package can be linked to a MATLAB code.

Using our MATLAB implementation, we evaluate the time efficiency of this non-linear solver with respect to the number of samples used for the transformation. Here, we simply assume that we know the value of n for the reconstruction. We experiment with various number of predictors (model inputs), k, ranging between 3 to 5. For any given n and k, we divide the whole data set into groups of n samples. For each group, we perform the transformation and reconstruction. At the end, we report the average number of iterations over all of the groups.

As Figure 2 suggests, there is a peak in the reconstruction time for any given value of k. This is the situation where the number of variables, $n(k+1)$, is closest to the number of constraints, $\frac{k^2+3k}{2}+1$. For smaller values of n, the number of constraints is more than the free variables. A single feasible solution in this case may be the reconstruction result. On the other hand, for larger values of n, the number of free variables is more than the constraints. It is easier then to find a point closer to the mean to maximize the objective function. Based on this we expect the reconstruction error to be very high after $n = k$.

We should note that since the rows of W and Y are not ordered, the order of the reconstructed matrices $\tilde{Y}$ and $\tilde{W}$ may be different from the original data. In our implementation, we try all permutations of the rows to find the best match. This, of course, is only feasible for small values of n, and is just to show that even the best possible reconstruction will not be accurate against the feature matrices.

To empirically demonstrate that accurate reconstruction is unlikely for when n is larger than a threshold and to find that threshold, we generate random input and output values with various correlations. In particular, when no correlation is present, we generate 1000 different data segments with $k = 4$ using independent random values distributed with a normal distribution. We call this distribution the case of no correlation. When correlations are present, they can be classified into two classes: i) the vertical correlation is the correlation among different attributes, and ii) the horizontal correlation is the correlation within a single attribute. To generate an input with some vertical correlation, we multiply our random matrix by a matrix U such that $U^T U = C$, where C is the correlation matrix. In particular, let R be the independent random input:

$$W = RU$$

$$W^T W = U^T R^T RU = U^T U = C$$

The last equation comes from the fact that R is a matrix of independent variables and $R^T R = I$. We simply use several correlation matrices and derive U using a Cholesky decomposition. This is an established method of adding correlation to independent random variables. Finally, we create the highest possible vertical correlation by making each attribute $\mathbf{w_i}$ to be $\mathbf{w_1} + c$ where c is a constant offset.

In the first experiment, we change the value of n used to create the feature matrices and derive the average reconstruction error for 10 different input matrices with various correlations. As Figure 3 suggests, even in the extreme case of the highest vertical correlation, the reconstruction error is close to 1 when $n > 2k$.

In the next experiment, we try various horizontal correlations while employing a medium vertical correlation. Similarly the highest correlation is obtained when the segments are just a shifted version of each other by a constant offset. We should emphasize that our data segmentation is required to remove any significant horizontal correlation. This experiment is to show what happens if residual temporal correlations exist between segments. Results in Figure 4 show that for $n > 2k$, the error approaches 1 indicating privacy protection against this particular reconstruction.

Figures 3 and 4 suggest that the user can personalize the amount of privacy that is guaranteed based on the number of segments shared. One simple way to implement this is to have multiple threshold values on the number of samples being shared. Each threshold guarantees a higher degree of privacy (a higher minimum reconstruction error).

To derive the relation between a safe value of n and k, in the last experiment we change the value of k and create cor-

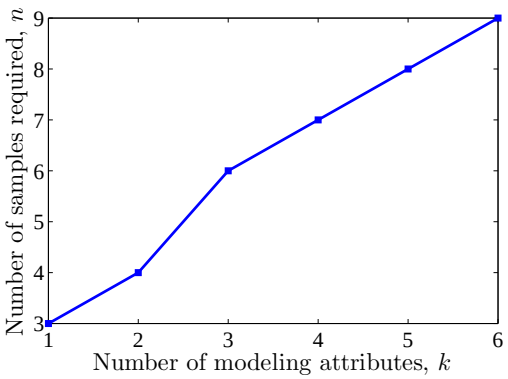


Figure 5. The number of samples required to minimize the change of a privacy breach.

related input samples. Figure 5 plots the minimum number of samples required to guarantee a reconstruction error of at least 0.9. Observe that $n = 2k$ is always a safe bet. The curve suggests that the client-side software should compute neutral features based on a number of segments that is at least twice the number of model inputs.

The authors admit that this rule of thumb is not supported by rigorous theoretical analysis. A formal investigation is beyond the scope of this paper. Here, the conjecture is merely observed without formal proof. In general, however, the client-side software, when asked to share data, can compute the neutral features given the currently available segments then perform the reconstruction itself and estimate reconstruction error. If the error is significantly less than 1, the software may warn the user that privacy may be breached.

6 A Case Study

In this section, we implement our proposed scheme and evaluate it on a participatory sensing data set used for green navigation [21]. In that service, individual drivers install OBD-II scanners in their cars which record the engine sensor parameters along with GPS location information on SD cards. Their service then uses regression modeling to to suggest the most fuel-efficient routes between arbitrary source destination pairs [19] by using a prediction model to estimate the car's expected fuel consumption on each road segment.

There are several privacy-preserving techniques proposed for participatory sensing applications that can are comparable in that they rely on data alteration. Mostly they propose specific noise models to overcome correlations in the data set [22, 17, 34, 20]. Since we have already removed the correlations between data points during segmentation, we compare our technique to one that shares raw segments perturbed by white noise. This simple technique in fact outperforms more complex correlated noise schemes in terms of impact on the final model. We first present the details of our implemented client and modeling server. Next, we compare the level of privacy achieved by our approach to perturbation-based techniques. Finally, we compare the accuracy of modeling of our approach and perturbation-based approaches.

6.1 Implementation

We implemented a desktop client and server as C++ libraries. The client program takes the data trace file and a configuration file as input. Each measurement attribute is a column in the trace file. The configuration file stores the settings of the application: i) A unique application id, ii) the segmentation interval, iii) the segmentation attributes (e.g. time), and iv) predictor functions that map X to W (g_j).

The application id is used for later correspondence with the server to make sure that the feature matrices are used for the correct model. The segmentation interval and segmentation attribute are used by the client to segment the input trace. In particular, the client reads the input trace file row by row and groups the samples based on the segmentation attributes. There can be multiple attributes that are used as the segmentation attribute. For example, location traces from a GPS receiver consists of a longitude and a latitude. The segmentation process simply takes the euclidean distance between the segmentation attributes of two consecutive samples. The increments are then added and when the sum reaches the segmentation interval, this batch of samples are used to produce one segment. All samples for each attribute are added and divided by the segmentation interval to get the average value to be used in the segmented data. These values correspond to $\mathbf{x_i}$ in our notation.

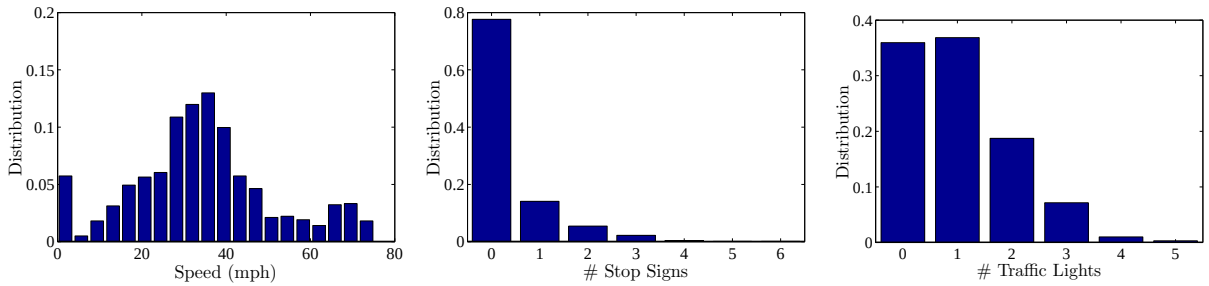
Next, the client checks for the privacy condition and if $n < 2k$, warns the user to use a larger input. The next step in the client is to derive the values of $\mathbf{w_i}$ from $\mathbf{x_i}$. This is done using the configuration file settings. The client calculates ρ_u, ν_u, and θ_u. The feature matrices are transferred over a TCP connection using XML. The XML message contains an application id and the values of matrices ρ_u, Θ_u, ν_u.

The server program maintains a list of models mapped to the application ids. Each model consists of the aggregate values of ρ, ν, and Θ. When an XML message is received from a client, the application id is looked up to find the corresponding model. The feature matrices in the XML are simply added to the aggregate matrices. The client configuration file ensures the same ordering of columns when calculating the matrices. The server derives the model coefficients using Equation (2) and writes them to a file.

6.2 Experimental Setup and Data Set

To evaluate our privacy-aware mechanism, we utilize data from the experimental deployment of a green navigation system. This data consists of geo-tagged engine sensor measurements for a range of vehicles and constitutes a total of over 1000 miles of driving. A total of sixteen users (with different cars) drive over the course of three months [21].

After segmentation, the total number of data segments in the data set are 650. Each segment represents a approximately 2 miles of driving data, although shorter segments are present when routes where shorter than 2 miles or were not a multiple of that distance. There are 5 parameters for each segment, which are inputs to (and output of) a model of vehicular fuel consumption. These are: i) fuel consumption over the segment, $\mathbf{y} = f$, ii) $\mathbf{w_1} = m(ST + vTL)$ where m and v are the mass and the average velocity of the vehicle, ST and TL are the number of stop signs and traffic lights encountered, iii) $\mathbf{w_2} = mv^2$, iv) $\mathbf{w_3} = m$, and v) $\mathbf{w_4} = Av^2$ where A is the frontal area of the car. Using our notation, we can write $\mathbf{x_1} = m$, $\mathbf{x_2} = v$, $\mathbf{x_3} = A$, $\mathbf{x_4} = ST$, $\mathbf{x_5} = TL$. Therefore, we have $k = 4$ and $d = 5$.



(a) (b) (c)

Figure 6. Distribution of various parameters in the green navigation data set.

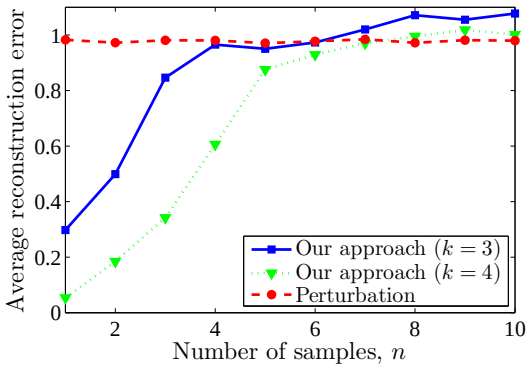


Figure 7. Comparing our approach against data perturbation in terms of the average reconstruction error of all parameters for different values of k.

The values of m, v, A, can easily be obtained by knowing parameters $w2$, $w3$, and $w4$. The values of ST and TL can also be determined since they are integers and the number of their value combinations are less than 20. Using a brute force approach the values of ST and TL are derived from $w1$. In our experiments, we use all the five parameters by default. This makes 1 output attribute and 4 predictors, hence $k = 4$. In some experiments, we drop the last parameter from the data set and denote it by $k = 3$. In this case, we only obtain the values for f m, v, and A.

The attacker (reconstruction algorithm) is implemented as a MATLAB code as presented in Section 5. Again, to eliminate differences in the order of reconstructed segments, all permutations of the segments are compared against the original set and the minimum is reported as the reconstruction error. In order to derive the objective function in our nonlinear reconstruction, we use the sample mean and variance from the data set. Figure 6 shows the distribution of three parameters collected by the users of the system. Until otherwise specified, we use the same noise variance as the variance of each parameter.

6.3 Privacy Evaluation

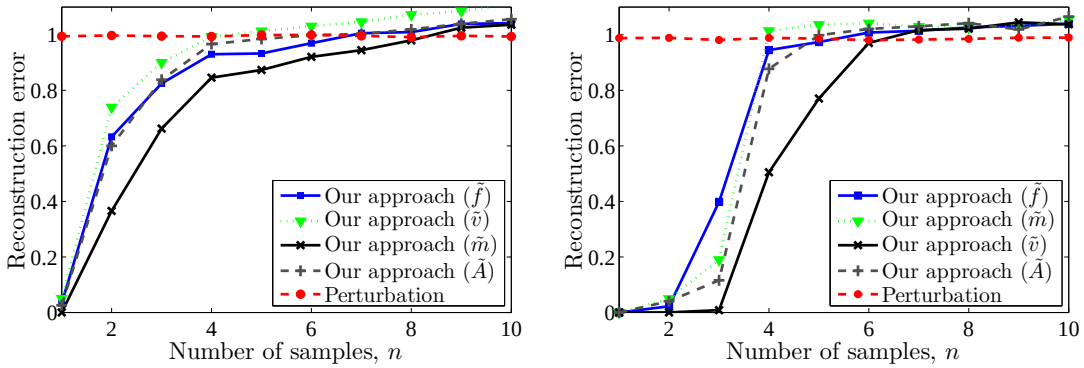
Our evaluation of privacy uses the definition of privacy presented in Section 5. In the first experiment, we compare the reconstruction error resulting from using the two approaches when the number of samples shared by each user, n, varies. The total number of segments are the constant, 650. This divided by n is the number of users who share the data. For every n, we average the resulting reconstruction over all users data and report the results in Figure 7. We do this for both $k = 3$ and $k = 4$. The result from the perturbation scheme does neither change with k nor with n. Therefore, we only present a single line.

Next, we repeat the same experiment while studying the reconstruction error of each individual parameter $\mathbf{x_i}$. Using all of the predictor variables we report the average reconstruction error of each parameter in Figure 8(a). The results show that reconstruction errors of 0.8 and 1.0 are achieved for all parameters when $n = k = 4$ and $n \geq 2k = 8$ respectively. Figure 8(b) repeats the same experiment and reports the error corresponding to the most accurate reconstruction among different samples instead of the average. This is the worst case privacy for in a particular set of data points. The results still show the complete privacy protection for $n \geq 2k$.

In order to evaluate the effect of the number of predictors on privacy, we also evaluate individual parameter reconstruction errors for $k = 3$, by dropping the last predictor variable from the data set. Figures 9(a) and 9(b) shows the average and worst-case privacy obtained by using our approach. The result verifies that a privacy-breach is unlikely if the number of samples is larger than the heuristic threshold.

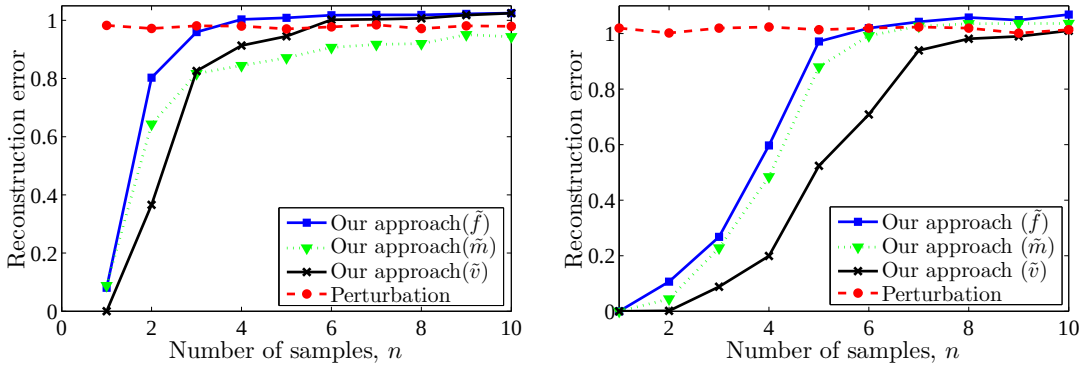
6.4 Prediction Accuracy

In this set of experiments, we evaluate the prediction accuracy of the constructed model at the server and compare it against a perturbation scheme. To this end, we perform a cross validation by leaving out a single segment from the data set and use the rest to construct the model which is then used to predict fuel consumption of the segment in question. To emulate multiple users, we divide (the rest of) the data set into groups of 10 segments and share them using our privacy-aware client firewalls. The regression coefficients computed on the server are then used to estimate the fuel consumption on the single segment that was left out. We add the estimated fuel consumption values on all segments to obtain a total fuel estimate. We calculate the relative error (in percents) between this estimated value and the sum of actual fuel consumption values. This calculation is called *cumulative error*. To calculate the prediction error of the perturbation scheme, we use the same cross-validation and error calculation scheme, except that the model is computed from



(a) (b)

Figure 8. The (a) average, and (b) minimum reconstruction error of various attributes for $k = 4$. Fuel consumption, f, weight (m), speed (v), and frontal area (A).



(a) (b)

Figure 9. The (a) average, and (b) minimum reconstruction error of various attributes for $k = 3$. Fuel consumption, f, weight (m), speed (v), and frontal area (A).

individually shared segments perturbed with additive noise. Thus, to estimate fuel consumption of a segment, we simply add white noise (of a given energy) to the data set with one segment removed and calculate the regression coefficients of the noisy data. The power consumption for that segment is then estimated from the obtained model and the process is repeated for all other segments in turn.

Our first experiment to evaluate the prediction accuracy compares the two schemes when the amount of noise energy used for perturbation changes. Figure 10 shows the results where the prediction error is represented using the mean squared error between the estimates and the actual fuel consumption values where all values in the dataset are normalized to $[-1,+1]$ range. Note that, the prediction accuracy of our approach is presented as a reference and remains constant since we do not use any noise. The x-axis shows the ratio of the noise variance to the variance of data parameters.

Note that, prediction error linearly changes with noise energy. However, it is clear from the figure that even with low noise energy values, the prediction error of the perturbation scheme is much higher than ours. This is despite the fact that our scheme is always providing privacy.

The next experiment studies the effect of the total number of segments that are collected for modeling. We fix $n = 10$ and noise variance ratio to be 1 and change the total number of available segments from 10 to 500. Figure 11 depicts that the prediction error decreases as the number of samples increases. This is simply because the constructed model becomes more accurate. However, when the number of segments is very small, the perturbation scheme shows an additional prediction error because the noise values do not cancel out when the number of samples are small.

In the final experiment, we evaluate the prediction accuracy of the fuel consumption for each individual car and report it in Table 2. For each car, only the data set that corresponds to the car is used for the prediction. The segments are similarly left out one by one for cross-validation. This result simply shows that the prediction error increases at least by four times when using perturbation. In the green navigation service which relies on accurate prediction of energy consumed on various segments in order to compute the most fuel-efficient routes and where different routes often differ in less than 5% to 6% of the total fuel consumption, 2% prediction error certainly affects the routing decisions made by the service. Our scheme on the other hand, enables the application to have the same accuracy as that achieved in the absence of privacy-enabling techniques.

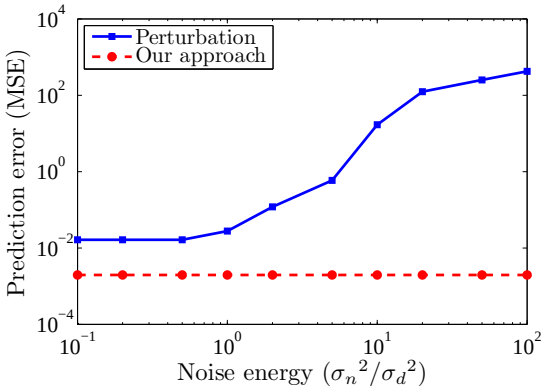


Figure 10. Comparing our approach against data perturbation with varying noise energies.

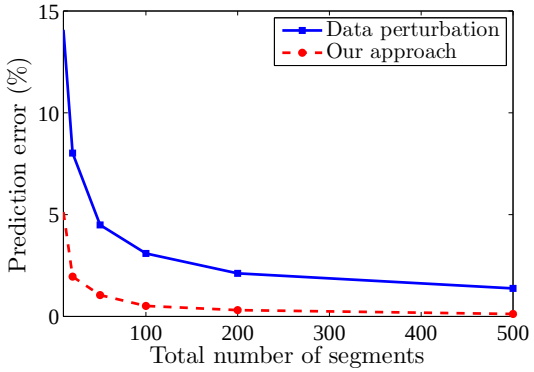


Figure 11. Comparing our approach against data perturbation with varying number of total segments collected.

Car Make	Car Model	Car Year	Our Appr. % error	Perturb. % error
Honda	Accord	2003	0.46	7.86
Ford	Contour	1999	0.58	2.12
Toyota	Corolla	2009	0.36	6.52
Ford	Focus	2009	0.11	2.25
Hyundai	Santa Fe	2008	0.39	2.43
Ford	Taurus	2001	0.18	1.75

Table 2. The modeling error induced by the perturbation comparing to our scheme when using individual car models.

7 Discussion

In the previous sections, we discussed how our technique matched the two design goals presented in the beginning. Although the perfect modeling criterion is analytically shown to be satisfied in all situations, our work lacks a rigorous validation of the perfect neutrality condition.

The privacy argument presented in the paper has two major shortcomings: i) The quantification of privacy does not capture all forms of privacy breaches, and ii) this scheme has not been strictly shown to guarantee privacy, as defined by our measure of privacy. This section discusses these limitations in detail and presents the cases where the approach may perform poorly to satisfy the neutrality condition.

Our quantification of privacy captures a specific form of information about the user, i.e. the exact data point values in users' data traces. Sometimes, more high-level information about the user can be deduced without reconstructing the exact data point values. For example, the magnitude of values in the feature matrix may reveal information about the range of values in the original trace. Changes in the correlations can also be used to infer information about how and when a user changes behavior (e.g. in energy consumption). To reflect those concerns, various definitions including Bayes-Optimal and differential privacy have been studied in the literature [14, 9]. Understanding the performance of our scheme under such notions helps a more concrete analysis of the privacy properties.

Based on our privacy-measure, we identify two major factors that can affect the possibility of revealing data after applying the privacy firewall. First, the distribution of the original data: a discrete and narrow distribution leaves a fewer number of choices for the reconstruction. Narrow distributions means that the value of a variable is approximately known to the attacker even before sharing any data. It can be the case that some data points are estimated rather accurately while the total reconstruction error is still high.

The second factor is the correlation values: For example, if the correlation values are all 1s, the whole data trace is a constant and therefore can be reconstructed perfectly if the average value for the user is revealed. Generally, a higher correlation makes the process of reconstructing the variables easier as the search space shrinks.

8 Related Work

In recent years, a number of techniques have been proposed for modifying or transforming data in such a way so as to preserve privacy. Such methods can be classified into four main categories described in detail below.

First, *randomization techniques* that add noise to the original data points have been used to hide the real value of sensitive data and other attributes (e.g., the trend of the data over time) [2, 6]. They traditionally distort data for methods such as surveys which have an evasive answer bias because of privacy concerns [40, 31]. Fuller [18] and Kim and Winkler [27] showed that some simple statistical information (e.g., means and correlations) can be preserved by adding random noise. In [4, 3], independent random noise (e.g., Gaussian) with high enough power is used to perturb user data. However, high noise power might decrease the utility of the shared data as well and the authors do not quantify this trade off. Recently, Ganti et al. [20] proposed that correlated noise, which has the same distribution as real data, can be used to perturb time-series data. This perturbation method is resilient to traditional filtering techniques, such as Kalman filter [24], and Spectral filtering [25]. Using randomization techniques in the context of privacy preserving regression, however, will introduce error in the regression model.

Another set of randomization techniques preserve *differential privacy* using randomized aggregation functions [14, 9, 36]. When an aggregate value is derived by a trustworthy entity or the user client, differential privacy is preserved if adding or removing a data item does not significantly change the output (aggregate) probability distribution. Like data point perturbation, differential privacy methods rely on ran-

domization that introduces noise to the regression model.

Second, the *k-anonymity model* [30] was developed because of the possibility of indirect identification of records from public databases. For example, the identity of a patient can be inferred from their home address or cellphone number. In the k-anonymity method, the granularity of data is reduced using techniques such as generalization and suppression. The l-diversity model [33] was designed to handle some weaknesses in the k-anonymity model including the the cases where there is homogeneity of the sensitive values within a group. Many variants of the above methods exist in current literature. A good survey of the corresponding algorithms may be found in [10]. Although k-anonymity is widely used to hide user identity in large database, it can not be applied into this problem because useful information for regression will be lost during generalization and suppression.

Distributed privacy preservation [42, 35] is used to derive aggregate results from data sets which are partitioned across these entities. While the individual may not desire to share their entire data set, they may consent to limited information sharing with the use of a variety of protocols. The overall effect of such methods is to maintain privacy for each individual, while allowing the aggregate results to be correctly computed over an entire group. Our proposed technique in this paper falls under this category. For this purpose, the data sets may either be *horizontally partitioned* or be *vertically partitioned*. In horizontal partitioning [11, 12, 32], the individual records are spread out across multiple entities, each of which have the same set of attributes. In vertically partitioned data sets [11, 39], the individual entities have different attributes of the same set of records. There have been also a lot of efforts in finding good privacy preserving regression methods [38, 13, 26]. None of them, however, quantify the achieved privacy.

Finally, several recent works have proposed cryptographic solutions for privacy-preserving modeling or aggregation of distributed data [8, 15, 36]. Some of these techniques, like our technique, do not perturb data and hence enable exact computation of certain aggregate functions. These solutions allow data sources to publish data in encrypted form, which hides the raw data values. They generally use homomorphic encryption so that the central server can compute aggregate functions directly on encrypted values. However, such encryptions are extremely expensive (up to $30\times$ more expensive than regular encryption) and only a small class of functions can be computed on such encrypted data. Our solution is computationally cheaper and more general than these cryptographic solutions.

Some previous work study how to provide different levels of privacy to different users. Xiao and Tao [41] proposed the notion of *personalized privacy* to allow each data owner to specify her own privacy level. However, the technique works in a centralized setting and for string attributes organized in a taxonomy tree. In contrast, the technique in this paper is targeted towards numerical data in a distributed setting.

9 Conclusion

This paper presented a privacy-aware scheme that enables regression modeling of participatory sensing data. We consider participatory sensing applications that develop general prediction models from data. The novelty of this work comes from the fact that it constructs regression models exactly the same as if private data where available to the server.

The main idea of the paper is to transform user data traces into neutral features that can be shared with the community and with the aggregation server while minimizing the threat to privacy. The time-series traces that are sensed by the user are first segmented into intervals of larger scale. The next step is to convert the segmented data into matrices that give little insight into the original data.

We show how given the transformed matrices, a server can construct the same regression model as if it has access to the data traces. The privacy-enabling properties of our transformation technique are studied. Intuitive arguments and empirical evidence suggest that when a large enough number of samples is transformed into a single feature and shared by a user, a privacy-breach is unlikely. Also, we show how the a personalized level of privacy is achieved by varying the number of samples included in the neutral features.

A data set from a green navigation service is used as a case study. We implement our approach as a client library and a server that enables privacy-enabling model construction in this application. Experimental results show that our technique does not change the accuracy of navigation while trying to make the user data private. In comparison, traditional data perturbation schemes introduce a significant amount of prediction error.

Future work on this topic will include a more formal quantification of privacy properties and their dependency on the user's data set, the number of segments used, and the number of model inputs available. Deployment data will be collected from example applications to investigate usability issues and issue of client interface. Finally, we shall investigate extensions of this approach to other modeling frameworks, besides linear regression.

10 References

[1] T. Abdelzaher et al. Mobiscopes for human spaces. *IEEE Pervasive Computing*, 6(2):20–29, 2007.

[2] N. R. Adam and J. C. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Comput. Surv.*, 21(4):515–556, 1989.

[3] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proc. of SIGMOD'01*, pages 247–255, 2001.

[4] R. Agrawal and R. Srikant. Privacy preserving data mining. In *Proc. of SIGMOD'00*, pages 439–450, 2000.

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[6] R. Brand. Microdata protection through noise addition. In *Inference Control in Statistical Databases, From Theory to Practice*, pages 97–116, 2002.

[7] J. Burke et al. Participatory sensing. In *Proc. of the World Sensor Web Workshop, in conjunction with SenSys'06*, Nov. 2006.

[8] C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Mobiquitous*, pages 109–117, 2005.

[9] B.-C. Chen, D. Kifer, K. LeFevre, and A. Machanavajjhala. Privacy-preserving data publishing. *Foundations and Trends in Databases*, 2:1–167, 2009.

[10] V. Ciriani, S. D. C. di Vimercati, S. Foresti, and P. Samarati. k-anonymity. In *Secure Data Management in Decentralized Systems*, pages 323–353. 2007.

[11] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explor. Newsl.*, 4(2):28–34, 2002.

[12] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *Workshop on New security paradigms (NSPW'01)*, pages 13–22, Cloudcroft, NM, 2001.

[13] W. Du, S. Chen, and Y. S. Han. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proc. of 4th SIAM International Conference on Data Mining*, pages 222–233, 2004.

[14] C. Dwork. Differential privacy: a survey of results. In *Proc. of 5th conference on Theory and applications of models of computation (TAMC'08)*, pages 1–19, Xi'an, China, 2008.

[15] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Eurocrypt*, pages 486–503, 2006.

[16] S. B. Eisenman et al. The bikenet mobile sensing system for cyclist experience mapping. In *SenSys'07*, Sydney, Australia, Nov. 2007.

[17] A. Evfimievski. Randomization in privacy preserving data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):43–48, December 2002.

[18] W. A. Fuller. Masking procedures for microdata disclosure limitation. *Official Statistics*, 9(2):383–406, 1993.

[19] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher. Greengps: a participatory sensing fuel-efficient maps application. In *Proc. of MobiSys'10*, pages 151–164, San Francisco, CA, 2010.

[20] R. K. Ganti, N. Pham, Y. Tsai, and T. F. Abdelzaher. Poolview: Stream privacy for grassroots participatory sensing. In *SenSys'08*, pages 281–294, 2008.

[21] GreenGPS Data Set. Fuel consumption data set. http://smart-attire.cs.uiuc.edu/ raghukiran/obd/.

[22] Z. Huang, W. Du, and B. Chen. Deriving private information from randomized data. In *Proc. ACM SIGMOD*, pages 37–48, 2005.

[23] B. Hull et al. Cartel: a distributed mobile sensor computing system. In *SenSys'06*, pages 125–138, 2006.

[24] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. of the ASME–Journal of Basic Eng.*, 82(Series D):35–45, 1960.

[25] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proc. of ICDM*, page 99, 2003.

[26] A. F. Karr et al. Secure regression on distributed databases. *J. Computational and Graphical Statist*, 14:263–279, 2004.

[27] J. J. Kim and W. E. Winkler. Masking microdata files. In *Survey Research Methods Section, American Statistical Association*, pages 114–119, 1995.

[28] KNITRO. http://www.ziena.com/knitro.htm.

[29] M. Kutner, C. Nachtsheim, J. Neter, and W. Li. *Applied Linear Statistical Models*. McGraw-Hill, 2005.

[30] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[31] C. K. Liew, U. J. Choi, and C. J. Liew. A data distortion by probability distribution. *ACM Trans. Database Syst.*, 10(3):395–411, 1985.

[32] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Proc. of 20th Cryptology Conference on Advances in Cryptology (CRYPTO'00)*, pages 36–54, London, UK, 2000.

[33] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3, 2007.

[34] N. Pham, R. Ganti, M. Y. Uddin, S. Nath, and T. Abdelzaher. Privacy-preserving reconstruction of multidimensional data maps in vehicular participatory sensing. In *Proc. of EWSN'10*, Coimbra, Portugal, Feb. 2010.

[35] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *SIGKDD Explor. Newsl.*, 4(2):12–19, 2002.

[36] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proc. of SIGMOD'10*, 2010.

[37] S. Sahni. Computationally related problems. *SIAM Journal on Computing*, 3(4):262–279, 1974.

[38] A. P. Sanil, A. F. Karr, X. Lin, and J. P. Reiter. Privacy preserving regression modelling via distributed computation. In *Proc. of KDD'04*, pages 677–682, 2004.

[39] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proc. of KDD'02*, pages 639–644, Edmonton, Canada, 2002.

[40] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Stat. Assoc.*, 60(309):63–69, 1965.

[41] X. Xiao and Y. Tao. Personalized privacy preservation. In *Proc. of SIGMOD'06*, pages 229–240, June 2006.

[42] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.