Cloud-Enabled Privacy-Preserving Collaborative Learning for Mobile Sensing*

Bin Liu, Yurong Jiang, Fei Sha, Ramesh Govindan
Department of Computer Science
University of Southern California
{binliu, yurongji, feisha, ramesh}@usc.edu

Abstract

In this paper, we consider the design of a system in which Internet-connected mobile users contribute sensor data as training samples, and collaborate on building a model for classification tasks such as activity or context recognition. Constructing the model can naturally be performed by a service running in the cloud, but users may be more inclined to contribute training samples if the privacy of these data could be ensured. Thus, in this paper, we focus on *privacy-preserving collaborative learning* for the mobile setting, which addresses several competing challenges not previously considered in the literature: supporting complex classification methods like support vector machines, respecting mobile computing and communication constraints, and enabling user-determined privacy levels. Our approach, Pickle, ensures classification accuracy even in the presence of significantly perturbed training samples, is robust to methods that attempt to infer the original data or poison the model, and imposes minimal costs. We validate these claims using a user study, many real-world datasets and two different implementations of Pickle.

Categories and Subject Descriptors

I.1.2 [**Computing Methodologies**]: Symbolic and Algebraic Manipulation—*Algorithms*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Algorithms, Human Factors, Security

Keywords

privacy, mobile sensing, support vector machines

* This research was partially sponsored by National Science Foundation grant CNS-1048824 and CNS-0121778. Any opinions, findings and conclusions or recomendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

SenSys'12, November 6–9, 2012, Toronto, ON, Canada.

1 Introduction

As smartphones proliferate, their sensors are generating a deluge of data. One tool for handling this data deluge is statistical machine classification; classifiers can automatically distinguish activity, context, people, objects, and so forth. We envision classifiers being used extensively in the future in mobile computing applications; already, many pieces of research have used standard machine learning classifiers (Section 5).

One way to build robust classifiers is through *collaborative learning* [42, 8, 5], in which mobile users contribute sensor data as training samples. For example, mobile users may submit statistical summaries (features) extracted from audio recordings of their ambient environment, which can be used to train a model to robustly recognize the environment that a user is in: a mall, an office, riding public transit, and so forth. Collaborative learning can leverage user diversity for robustness, since multiple users can more easily cover a wider variety of scenarios than a single user.

We envision that collaborative learning will be enabled by a software system that efficiently collects training samples from contributors, generates statistical classifiers, and makes these classifiers available to mobile users, or software vendors. In this paper, we address the challenges involved in designing a system for collaborative learning. Such a system must support the popular classifiers (such as Support-Vector Machines or SVMs and k-Nearest Neighbors or kNNs), must scale to hundred or more contributors, and must incentivize user contribution (Section 2). To our knowledge, no prior work has discussed the design of a collaborative learning system with these capabilities.

An impediment to scaling collaborative learning is the computational cost of constructing the classifier from training data. With the advent of cloud computing, a natural way to address this cost is to let mobile users submit their sensor data to a cloud service which performs the classifier construction. In such a design, however, to incentivize users to participate in collaborative learning it is essential to ensure the privacy of the submitted samples. Training samples might accidentally contain sensitive information; features extracted from audio clips can be used to approximately reconstruct the original sound [41, 18], and may reveal over 70% of the words in spoken sentences (Section 4.1).

In this paper, we present Pickle, a novel approach to privacy-preserving collaborative learning. Pickle is based

on the following observation: the popular classifiers rely on computing mathematical relationships such as the inner products and the Euclidean distances between pairs of submitted training samples. Pickle perturbs the training data on the mobile device using lightweight transformations to preserve the privacy of the individual training samples, but regresses these mathematical relationships between training samples in a unique way, thereby preserving the accuracy of classification. Beyond addressing the challenges discussed above, Pickle has many desirable properties: it requires no coordination among users and all communication is between a user and the cloud; it allows users to independently tune the level of privacy for perturbing their submitted training samples; finally, it can be made robust to poisoning attacks and is collusion-resistant. Pickle's design is heavily influenced by the requirements of mobile sensing applications, and occupies a unique niche in the body of work on privacy-preserving methods for classifier construction (Section 5).

A user study demonstrates that Pickle preserves privacy effectively when building a speaker recognition classifier (Section 4.1); less than 2% of words in sentences reconstructed by attacking Pickle transformations were recognizable, and most of these were stop words. Results from a prototype (Section 4.2) show that Pickle communication and storage costs are small, classification decisions can be made quickly on modern smartphones, and model training can be made to scale using parallelism. Finally, using several datasets, we demonstrate (Section 4.3) that Pickle's privacy-preserving perturbation is robust to regression attacks in which the cloud attempts to reconstruct the original training samples. The reconstructed samples are significantly distributionally-different from the original samples. Despite this, Pickle achieves classification accuracy that is within 5%, in many cases, of the accuracy obtained without any privacy transformation.

2 Motivation and Challenges

Modern phones are equipped with a wide variety of sensors. An emerging use of this sensing capability is *collaborative learning*, where multiple users contribute individually collected training samples (usually extracted from raw sensor data) so as to collaboratively construct statistical models for tasks in pattern recognition. In this paper, we explore the design of a system for collaborative learning.

What is Collaborative Learning? As an example of collaborative learning, consider individual users who collect audio clips from their ambient environments. These users may be part of a social network. Alternatively, they may have no knowledge of each other and may have volunteered to provide samples, in much the same way as volunteers sign up for distributed computing efforts like SETI@HOME; in this sense, we focus on *open* collaborative learning. Training samples extracted from these clips are collected and used to build a classifier that can determine characteristics of the environment: e.g., determine whether a user is at home, on a bus, in a mall, or dining at a restaurant, etc. As another example, consider a mobile health application that collects patient vital signs to build a model for classifying diseases.

Collaborative learning results in classifiers of activity, or of environmental or physiological conditions etc. Many proposed systems in the mobile sensing literature (e.g., [35, 8, 10, 43, 4]) have used machine-learning classifiers, often generated by using training samples from a *single* user. Due to the diversity of environments or human physiology, classifiers that use data from a single user may not be robust to a wide range of inputs. Collaborative learning overcomes this limitation by exploiting the diversity in training samples provided by multiple users. More generally, collaborative learning is applicable in cases, such as human activity recognition or SMS spam filtering, where a single user's data is far from being representative.

Designing a system for collaborative learning sounds conceptually straightforward, but has many underlying challenges. Before we describe these challenges, we give the reader a brief introduction to machine-learning classifiers.

The Basics of Classification. The first step in many machine-learning algorithms is *feature extraction*. In this step, the raw sensor data (an image, an audio clip, or other sensor data) are transformed into a vector of features that are most likely to be relevant to the classification task at hand. Examples of features in images include edges, contours, and blobs. For audio clips, the fundamental frequency, the spreads and the peaks of the spectrum, and the number of harmonics are all examples of features.

Classifiers are first trained on a set of training samples denoted by: $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$ where $\boldsymbol{x}_i \in \mathbb{R}^{\mathsf{P}}$ is the i-th training feature vector, and y_i is a categorical variable representing the class to which $\boldsymbol{x}_i$ belongs. For example, $\boldsymbol{x}_i$ may be a list of spectral features of an audio clip, and y_i may identify this audio clip as "bus" (indicating the clip was recorded in a bus). In what follows, we use $\boldsymbol{X}$ to denote the data matrix with $\boldsymbol{x}_i$ as column vectors, and $\mathcal{U}$ or $\mathcal{V}$ to refer to users.

The goal of classification is to construct a classifier using $\mathcal{D}$ such that when presented with a new test feature vector $\boldsymbol{x}$, the classifier outputs a label y that approximates $\boldsymbol{x}$'s true class membership.

One popular, yet simple and powerful, classifier is the k-Nearest-Neighbor (kNN) classifier. Given a feature vector $\boldsymbol{x}$ and a training set $\mathcal{D}$, kNN finds the k training feature vectors which are the closest to $\boldsymbol{x}$, in terms of the Euclidean distance between them:

$$\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} - 2\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}_i + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i. \quad (1)$$

The classifier then outputs the majority of all the nearest neighbors' labels as the label for $\boldsymbol{x}$ (ties broken arbitrarily).

Support Vector Machine (SVM) is another popular and more sophisticated classifier. It leverages a non-linear mapping to map $\boldsymbol{x}$ into a very high-dimensional feature space. In this feature space, it then seeks a linear decision boundary (i.e., a hyperplane) that partitions the feature space into different classes [16]. For the purposes of this paper, two computational aspects of this classifier are most relevant:

- The training process of SVMs rely on computing either the inner product $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$ or the Euclidean distance $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2$ between pairs of training feature vectors.

- The resulting classifier is composed of one or more of the submitted training samples — support vectors.

Design Goals and Challenges. A system for open collaborative learning must support three desirable goals.

First, it must *support the most commonly-used classifiers* such as the Support Vector Machine (SVM) classifier and the k-Nearest Neighbor (kNN) classifier described above. These popular classifiers are used often in the mobile sensing literature for logical localization [7], collaborative video sensing [8], behavioral detection of malware [10], device identification [40] and so on. Other pieces of work, such as CenceMe [43], EEMSS [4], and Nericell [44] could have used SVM to get better classification performance.

Second, the system must *scale* to classifiers constructed using training samples from 100 or more users. At this scale, it is possible to get significant diversity in the training samples in order to enable robust classifiers. A major hurdle for scaling is computational complexity. Especially for SVM, the complexity of constructing the classifier is the dominant computational task, and using the classifier against test feature vectors (i.e., $\boldsymbol{x}$ above) is much less expensive. As we discuss later, it takes a few hours on a modern PC to construct a classifier using data from over 100 users; as such, this is a task well beyond the capabilities of smartphones today. A less crucial, but nevertheless important, scaling concern is network bandwidth usage.

Third, the system must have the right *adoption incentives* to enable disparate users to contribute training samples: (1) The system must ensure the *privacy* of the submitted samples, as we discuss below; (2) It must be robust to *data poisoning*, a legitimate concern in open collaborative learning; (3) It must enable users who have not contributed to the model to use the classifier, but must dis-incentivize *free-riders* who use classifiers directly obtained from other users. Of these, addressing the privacy goal is most intellectually challenging, since the construction of many popular classifiers, like SVM or kNN, requires calculations using accurate feature vectors which may reveal privacy (Section 3).

Consideration of economic incentives for collaborative learning is beyond the scope of this paper; we assume that crowd sourcing frameworks like Amazon Mechanical Turk[1] can be adapted to provide appropriate economic incentives.

Cloud-Enabled, Privacy-Preserving Classification. We propose to use an approach in which mobile users submit training samples (with associated labels) to a cloud, possibly at different times over a period of hours or days; the cloud computes the classifier; the classifier is then sent to mobile phones and used for local classification tasks.

Using the cloud addresses the computational scaling challenge, since classifier construction can be parallelized to take advantage of the cloud's elastic computing capability. The cloud provides a rendezvous point for convenient training data collection from Internet-connected smartphones. Finally, the cloud provides a platform on which it is possible to develop a service that provides collaborative learning of different kinds of models (activity/context recognition, image classification, etc.). Indeed, we can make the following claim: to conveniently scale open collaborative learning, an Internet-connected cluster is *necessary*, and the cloud infrastructure has the right pay-as-you-go economic model since different collaborative learning tasks will have different computational requirements.

However, using the cloud makes it harder to achieve an important design goal discussed above, *privacy*.

Privacy and the Threat Model. In cloud-enabled open collaborative learning, users contribute several training samples to the cloud. Each sample consists of a feature vector $\boldsymbol{x}$ and the associated label y. Both of these may potentially leak private information to the cloud (as we discuss below, in our approach we assume the cloud is untrusted), and we consider each in turn. Before doing so, we note that *using* the classifier itself poses no privacy threat[2], since smartphones have enough compute power to perform the classification locally (Section 4.2).

Depending upon the kind of classifier that the user is contributing to, the label y may leak information. For example, if the model is being used for activity recognition, a label may indicate that the user was walking or running at a given time. In this paper, we do not consider label privacy because the user *willingly* contributes the labeled feature vectors and should have no expectation of privacy with regard to labels.

However, users may (or should, for reasons discussed below) have an expectation of privacy with respect to information that may be leaked by the feature vectors. Feature vectors $\boldsymbol{x}$ often consist of a collection of statistical or spectral features of a signal (e.g., the mean, standard deviation or the fundamental frequency).

Some features can leak private information. Consider features commonly used to distinguish speech from music or noise [35]: the Energy Entropy, Zero-Crossing Rate, Spectral Rolloff, or Spectral Centroid etc. These statistics of the speech signals may, unintentionally, reveal information that can be used to extract, for example, age, gender or speaker identity. In experiments we have conducted on audio clips from the TIMIT dataset [20] (details omitted for brevity), female voices tend to have higher average spectral rolloff and average spectral centroid than male voices, while voices of younger individuals have higher average energy entropy and lower average zero-crossing rate than voices of the aged. Similar age-related differences in measures of repeated activity have also been observed elsewhere [6].

Worse yet, a relatively recent finding has shown that, in some cases, feature vectors can be used to reconstruct the original raw sensor data. Specifically, a commonly used feature vector in speech and music classification is the Mel-frequency cepstrum coefficients (MFCC), which are computed by a sequence of mathematical operations on the frequency spectrum of the audio signals. A couple of works [41, 18] have shown that it is possible to approximately reconstruct the original audio clips, given the MFCC feature vectors. In Section 4.1, we present the results of an experiment that quantifies information leakage by MFCC reconstruction.

[1] https://www.mturk.com/mturk/welcome

[2] Assuming the user can trust the phone software; methods for ensuring this are beyond the scope of this paper.

In the context of collaborative learning, this is an alarming finding. When a user submits a set of feature vectors and labels them as being in a cafe (for example), the cloud may be able to infer far more information than the user intended to convey. When the original audio clips are reconstructed, they may reveal background conversations, the identity of patrons, and possibly even the location of the specific cafe. A recent, equally alarming, finding is that original images may be reconstructed from their feature vectors [54].

Given these findings, we believe it is prudent to ensure the privacy of feature vectors. The alternative approach, avoiding feature vectors that are known or might be suspected to reveal private information, can affect classification accuracy and may not therefore be desirable.

One way to preserve the privacy of $\boldsymbol{x}$ is to generate $\tilde{\boldsymbol{x}}$ from $\boldsymbol{x}$ and send only $\tilde{\boldsymbol{x}}$ the cloud, with the property that, with high likelihood, the cloud cannot reconstruct $\boldsymbol{x}$ from $\tilde{\boldsymbol{x}}$. Our approach randomly perturbs the feature vectors to generate $\tilde{\boldsymbol{x}}$, but is able to reconstruct some of the essential properties of these feature vectors that are required for classifier construction, without significantly sacrificing classifier accuracy. As we show later, approaches that use other methods like homomorphic encryption, secure multi-party communication or differential privacy make restrictive assumptions that do not apply to our setting.

We make the following assumptions about the threat model. The user trusts the software on the mobile device to compute and perturb feature vectors correctly, and to transmit only the perturbed feature vectors and the associated labels. The user does not trust other users who participate in the collaborative learning, nor does she trust any component of the cloud (e.g., the infrastructure, platforms or services) The cloud has probabilistic polynomial-time bounded computing resources and may attempt to reconstruct the original feature vectors. Servers on the cloud may collude with each other, if necessary, to recontruct the original feature vectors. Moreover, the cloud may collude with user A to attempt to reconstruct user B's original feature vectors by directly sending B's perturbed feature vectors to A. Also, user B's perturbed feature vectors may be included in the classifiers sent to A, and A may try to reconstruct B's original feature vectors.

Given that the cloud is untrusted, what incentive is there for the cloud to build the classifier correctly (i.e., why should users trust the cloud for developing accurate classifiers)? We believe market pressures will force providers of the collaborative learning "service" to provide accurate results, especially if there is a revenue opportunity in collaborative learning. Exploring these revenue opportunities is beyond the scope of this work, but we believe there *will* be revenue opportunities, since a service provider can sell accurate classifers (of, for example, context) to a large population (e.g., all Facebook users who may be interested in automatically updating their status based on context).

3 Privacy-Preserving Collaborative Learning

In this section, we discuss a novel approach to preserving the privacy of collaborative learning, called Pickle.

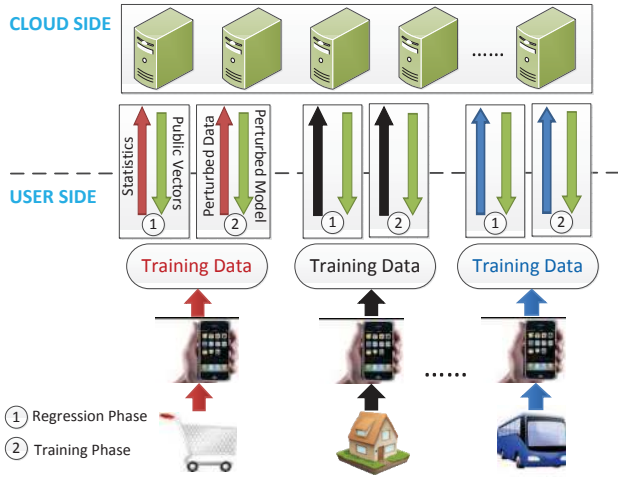


Figure 1—Illustrating Pickle.

3.1 Pickle Overview

In Pickle (Figure 1), each user's mobile phone takes N training feature vectors, where each vector has P elements, and pre-multiplies the resulting $\mathsf{P} \times \mathsf{N}$ matrix by a *private, random* matrix $\boldsymbol{R}$, whose dimensionality is $\mathsf{Q} \times \mathsf{P}$. This multiplication *randomly perturbs* the training feature vectors. Moreover, we set $\mathsf{Q} < \mathsf{P}$, so this *reduces the dimensionality* of each feature vector. A dimensionality-reducing transformation is more resilient to reconstruction attacks than a dimensionality preserving one [32]. In Pickle, $\boldsymbol{R}$ is private to a participant, so is not known to the cloud, nor to other participants (each participant generates his/her own private random matrix). This multiplicative perturbation by a private, random matrix is the key to achieving privacy in Pickle.

A dimensionality-reducing transformation does not preserve important relationships between the feature vectors, such as Euclidean distances and inner products. For instance, the inner product between two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ now becomes $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{x}_j$. This is not identical to $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$ unless $\boldsymbol{R}$ is an orthonormal matrix which necessarily preserves dimensionality. A dimensionality-reducing transformation can *approximately* preserve Euclidean distances [25], but even this property is lost when different participants use different private random matrices; in this case, the Euclidean distances and inner products for perturbed feature vectors from *different users* is no longer approximately preserved. Distortion in these relationships can significantly degrade classification accuracy when used directly as inputs to classifiers (Section 4.3).

In this paper, our focus is on methods that maintain high classification accuracy while preserving privacy. The central contribution of this paper is the design of a novel approach to *approximately reconstruct those relationships using regression, without compromising the privacy of the original feature vectors, while still respecting the processing and communication constraints of mobile devices.*

To do this, Pickle learns a statistical model to compensate for distortions in those relationships, then approximately reconstructs distance or inner-product relationships between

the users' perturbed feature vectors, before finally constructing the classifier. Conceptually, here is how Pickle works.

1. Users generate labeled raw data at their convenience: for example, Pickle software on the phone may collect audio clips, then prompt the user to label the clips.
2. Once a piece of raw data is labeled, the software will extract feature vectors, perturb them using $\boldsymbol{R}$, and upload them, along with the corresponding label, to the cloud; as an optimization, the software may batch the extraction and upload. (In what follows, we use the term *user*, for brevity, to mean the Pickle software running on a user's mobile device. Similarly, we use the term *cloud* to mean the instance of Pickle software running on one or more servers on a public cloud.)
3. When the cloud receives a sufficient number of labeled perturbed feature vectors from contributors (the number may depend on the classification task), it constructs the classifier and sends a copy to each user.

Before the classifier is generated, the cloud learns a model to compensate for the distortion introduced by perturbation. Specifically, in this *regression phase*:

1. The cloud sends to each participating user a collection of *public* feature vectors.
2. The user perturbs the cloud-generated feature vectors using its private transformation matrix and returns the result to the cloud.
3. The cloud employs regression methods to learn approximate functions for computing the desired mathematical relationships between feature vectors.

The key intuition behind our approach is as follows. Pattern classifiers can effectively *discriminate* between different classes by leveraging the most important covariance structures in the underlying training data. Our regression phase learns these structures from the transformed representations on public data. However, our privacy transformation sufficiently masks the less important components that would be required to *generate* the original feature vectors. This is why we are able to build accurate classifiers even without being able to regenerate the original feature vectors.

3.2 The Regression Phase

Step 1: Randomly Generate Public Feature Vectors. In this step, the cloud randomly generates M (in our paper, we set M to 3P) public feature vectors as a $\mathsf{P} \times \mathsf{M}$ matrix $\boldsymbol{Z}$ and sends this matrix to each user. The random public feature vectors have the same dimensionality as true feature vectors. In Pickle, the cloud synthesizes random public feature vectors using summary statistics provided by U users. In this method, each user sends to the cloud the mean and the covariance matrix of its private training data, derived from a fixed number (in our paper, 4P) of its feature vectors. The cloud generates a $\boldsymbol{Z}$ that *approximates the statistical characteristics of the training feature vectors of* all *the* U *users*; this matrix, generated using an equally-weighted Gaussian mixture model that simulates the true distribution of user data, is used in the next two steps to learn relationships between the feature vectors and are *not* used to build classifiers.

This method never transmits actual private feature vectors to the cloud, so preserves our privacy goal. Moreover, although the cloud knows the mean and the covariance, this information is far from sufficient to generate accurate individual samples since two random draws from the same continuous distribution have zero probability of being identical. Despite this, it is possible that sample statistics of the feature vectors may leak some private information; to limit this, Pickle generates sample statistics from a very small number (4P) of data samples. Finally, in this step, the public feature vectors need not be labeled.

Step 2: Perturb the Public Feature Vectors. The high-level idea in this step is to perturb $\boldsymbol{Z}$ in exactly the same way as users would perturb the actual training feature vectors. Concretely, a user $\mathcal{U}$ generates a private random matrix $\boldsymbol{R}_u$[3], computes the perturbed public feature vectors $\boldsymbol{R}_u\boldsymbol{Z}$, and sends $\boldsymbol{R}_u\boldsymbol{Z}$ to the cloud.

However, this approach has the following vulnerability. If $\boldsymbol{Z}$ is invertible, the private $\boldsymbol{R}_u$ can be recovered by the cloud when it receives the perturbed vectors $\boldsymbol{R}_u\boldsymbol{Z}$: the cloud simply computes $\boldsymbol{R}_u\boldsymbol{Z}\boldsymbol{Z}^{-1}$.

To raise the bar higher, Pickle computes and sends $\boldsymbol{R}_u(\boldsymbol{Z}+\boldsymbol{\varepsilon}_u)$ to the cloud, where $\boldsymbol{\varepsilon}_u$ is an additive noise matrix. The cloud would then need to know $\boldsymbol{R}_u\boldsymbol{\varepsilon}_u$ in order to apply the inversion to obtain an accurate $\boldsymbol{R}_u$. Unlike the public feature vectors $\boldsymbol{Z}$, however, $\boldsymbol{\varepsilon}_u$ is private to the user. The elements of $\boldsymbol{R}_u$ are drawn randomly from either a Gaussian or a uniform distribution. $\boldsymbol{\varepsilon}_u$ has the distribution of $\mathcal{N}(\boldsymbol{\varepsilon}_u; \boldsymbol{0}, \alpha_u\boldsymbol{\Sigma}_Z)$, where $\boldsymbol{\Sigma}_Z$ is the (sample) covariance matrix of $\boldsymbol{Z}$. α_u is tunable, controlling the *intensity* of the additive noise [11]. As we show in Section 4.3, higher privacy can be obtained by using smaller values of Q (i.e., greater reductions in dimensionality) or bigger values of α_u.

Step 3: Regress. The regression step is executed on the cloud. We describe it for two users; extending it to multiple users is straightforward. Assume users $\mathcal{U}$ and $\mathcal{V}$ have chosen random matrices $\boldsymbol{R}_u$, $\boldsymbol{\varepsilon}_u$ and $\boldsymbol{R}_v$, $\boldsymbol{\varepsilon}_v$ respectively. The cloud receives $\boldsymbol{Z}_u = \boldsymbol{R}_u(\boldsymbol{Z}+\boldsymbol{\varepsilon}_u)$ and $\boldsymbol{Z}_v = \boldsymbol{R}_v(\boldsymbol{Z}+\boldsymbol{\varepsilon}_v)$. The key idea is to use $\boldsymbol{Z}_u$ and $\boldsymbol{Z}_v$ to approximate quantities which are pertinent to classification.

Concretely, let μ and ν be two indicator variables that $\mu, \nu \in \{u, v\}$. Also, let $\boldsymbol{z}_i$ stand for the i-th public feature vector and $\boldsymbol{z}_{\mu i}$ the i-th transformed feature vector by $\boldsymbol{R}_\mu$. In other words, $\boldsymbol{z}_i$ and $\boldsymbol{z}_{\mu i}$ are the i-th columns of $\boldsymbol{Z}$ and $\boldsymbol{Z}_\mu$.

Intuitively, we would like the cloud to be able to recover the original relationships from the perturbed feature vectors. For this, we learn four functions (f_{uu}, f_{uv}, f_{vu} and f_{vv}) in the form of $f_{\mu\nu}(\boldsymbol{z}_{\mu i}, \boldsymbol{z}_{\nu j}; \boldsymbol{\theta}_{\mu\nu})$ that can approximate well a certain function $f(\boldsymbol{z}_i, \boldsymbol{z}_j)$ of (particularly, the distances or the inner products between) $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$. $\boldsymbol{\theta}_{\mu\nu}$ is the parameter vector of the function. Once these functions are learnt, they are applied to actual training data sent by users (Section 3.3).

The parameter vectors $\boldsymbol{\theta}_{\mu\nu}$ are thus of critical importance. To identify the optimal set of parameters, we have used linear regression (LR). We now show how to approximately recover the concatenation of public feature vectors $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ (i.e., $f(\boldsymbol{z}_i, \boldsymbol{z}_j) = [\boldsymbol{z}_i^{\mathrm{T}}, \boldsymbol{z}_j^{\mathrm{T}}]^{\mathrm{T}}$) using LR. The models then can be

[3]The user can choose a task-specific $\boldsymbol{R}_u$. However, once chosen, the matrix is fixed, though private to the user. A dynamically varying $\boldsymbol{R}_u$ will incur high computational cost, due to the Regress phase in the next step.

used to compute inner products and distances *approximately* on transformed actual training feature vectors from users[4]. The approximated quantities will then be supplied to learning algorithms to construct classifiers (Section 3.3).

For each pair of μ and ν, let $\boldsymbol{Q}_{\mu\nu}$ be the matrix whose columns are concatenated $\boldsymbol{z}_{\mu i}$ and $\boldsymbol{z}_{\nu j}$ with M^2 columns (the number of total possible concatenations is M^2, since there are M public feature vectors). Also, let $\boldsymbol{Z}_\mathsf{C}$ denote the matrix whose columns are concatenated $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$. Note that $\boldsymbol{Q}_{\mu\nu}$ has $2\mathsf{Q}$ rows, where Q is the row-dimensionality of each user's private transformation matrix $\boldsymbol{R}_\mu$ or $\boldsymbol{R}_\nu$ (for simplicity of description, we assume the dimensionality is the same for the two users; Pickle allows different users to choose different Q). $\boldsymbol{Z}_\mathsf{C}$ has $2\mathsf{P}$ rows, where P is the row-dimensionality of the public or original feature vectors.

For linear regression, we use this equation to obtain $\boldsymbol{\theta}_{\mu\nu}$ for $f_{\mu\nu}$

$$\boldsymbol{Z}_\mathsf{C} = \boldsymbol{\theta}_{\mu\nu}\boldsymbol{Q}_{\mu\nu} \tag{2}$$

where the parameter $\boldsymbol{\theta}_{\mu\nu}$ is a matrix with the size of $(2\mathsf{P} \times 2\mathsf{Q})$. The optimal parameter set is thus found in closed form as $\boldsymbol{\theta}_{\mu\nu} = \boldsymbol{Z}_\mathsf{C}\boldsymbol{Q}^{+}_{\mu\nu}$, where $^+$ denotes the pseudo-inverse.

Our implementation of Pickle uses one optimization, called *iPoD*. In *iPoD*, the cloud can avoid calculating the regression functions f_{uu} and f_{vv} (i.e., when $\mu = \nu$) by asking users directly for the corresponding inner products calculated from their own feature vectors. These inner products do not reveal the individual feature vectors. This trades off a little communication overhead (quantified in Section 4.2) for improved accuracy.

Instead of linear regression, we could have used Gaussian Process Regression (GPR). We have found in preliminary experiments that GPR marginally improves accuracy over LR but is significantly more compute-intensive, so we omit a detailed description of GPR.

Finally, all the schemes described above extend to multiple users naturally: Pickle simply computes 4 (or 2 when using *iPoD*) regression functions for every pair of users.

3.3 Model Generation and Return

Building the Classifier. After the cloud learns the functions $f_{\mu\nu}$ with the procedure in the previous section, it is ready to construct pattern classifiers using training samples contributed by users. In this step of Pickle, each user $\mathcal{U}$ collects its training feature vectors $\boldsymbol{X}_u$ (in which each column is one feature vector), then perturbs these feature vectors with its private $\boldsymbol{R}_u$. Each perturbed feature vector, together with its label, is then sent to the cloud. Using perturbed feature vectors from each user, the cloud generates the classification model.

Let $\boldsymbol{x}_{ui}$ denote the unperturbed i-th feature vector from user $\mathcal{U}$ and likewise $\boldsymbol{x}_{vj}$ for the user $\mathcal{V}$. Moreover, let

$$\tilde{\boldsymbol{x}}_{ui} = \boldsymbol{R}_u\boldsymbol{x}_{ui}, \qquad \tilde{\boldsymbol{x}}_{vj} = \boldsymbol{R}_v\boldsymbol{x}_{vj} \tag{3}$$

denote the perturbed feature vectors. Using the regression parameters obtained from Equation (2), the cloud first attempts to reconstruct the concatenation of $\boldsymbol{x}_{ui}$ and $\boldsymbol{x}_{vj}$,

$$\begin{bmatrix} \boldsymbol{x}_{ui} \\ \boldsymbol{x}_{vj} \end{bmatrix} \approx f_{uv}(\boldsymbol{x}_{ui}, \boldsymbol{x}_{vj}; \boldsymbol{\theta}_{uv}) = \boldsymbol{\theta}_{uv} \begin{bmatrix} \tilde{\boldsymbol{x}}_{ui} \\ \tilde{\boldsymbol{x}}_{vj} \end{bmatrix} \triangleq \begin{bmatrix} \boldsymbol{r}_{uj} \\ \boldsymbol{r}_{vj} \end{bmatrix}$$

where $\boldsymbol{r}_{uj}$ and $\boldsymbol{r}_{vj}$ are P-dimensional vectors. The cloud then approximates the inner product with the reconstructed feature vectors, $\boldsymbol{x}_{ui}^\mathrm{T}\boldsymbol{x}_{vj} \approx \boldsymbol{r}_{ui}^\mathrm{T}\boldsymbol{r}_{vj}$. Similarly, to approximate the distance between two feature vectors, we use[5]

$$\|\boldsymbol{x}_{ui} - \boldsymbol{x}_{vj}\|_2^2 \approx \boldsymbol{r}_{ui}^\mathrm{T}\boldsymbol{r}_{ui} - 2\boldsymbol{r}_{ui}^\mathrm{T}\boldsymbol{r}_{vj} + \boldsymbol{r}_{vj}^\mathrm{T}\boldsymbol{r}_{vj} \tag{4}$$

Once inner products or distances are approximated, the cloud can build SVM or kNN classifiers using the following simple substitution: whenever these algorithms need the distances or inner products of two feature vectors, the approximated values are used.

Model Return. In this step, the cloud returns the model to users, so that classification may be performed on individual phones; for the details of the classification algorithms, we refer the interested reader to [16]. The information returned depends upon the specific classifier (e.g., when using SVM, the support vectors must be returned), but must include all functions $f_{\mu\nu}$ and associated $\boldsymbol{\theta}_{\mu\nu}$ parameters for every pair of users. These are required because the classification step in many classifiers also computes distances or inner products between the test feature vectors and training feature vectors presented in the model (e.g., the support vectors in SVM); all of these vectors are perturbed so their distances and inner products must be estimated using the $f_{\mu\nu}$ functions.

3.4 Privacy Analysis

Recall that the privacy goal in Pickle is to ensure the computational hardness of de-noising user contributions by the cloud (either by itself, or in collaboration with other users) and thereby inferring $\boldsymbol{X}$. We now show a user $\mathcal{U}$ who follows the steps of the protocol does not leak vital information which can be used to de-noise user contributions. In the protocol, $\mathcal{U}$ sends data to the cloud in Steps 1, 2 and 4 *only*.

In Step 1, $\mathcal{U}$ sends the mean and covariance matrix of a small number of its private training samples. Using this, the cloud can construct synthetic vectors whose first and second-order statistics match that of $\mathcal{U}$'s private data, but clearly cannot reconstruct $\boldsymbol{X}_\boldsymbol{u}$.

In Step 2, $\mathcal{U}$ sends $\boldsymbol{R}(\boldsymbol{Z} + \boldsymbol{\varepsilon})$ to the cloud. One might assume that the cloud can filter out the additive noise $\boldsymbol{R}\boldsymbol{\varepsilon}$ and then recover $\boldsymbol{R}$ by using the known $\boldsymbol{Z}^{-1}$. However, existing additive noise filtering techniques (such as spectral analysis [26], principal component analysis, and Bayes estimation [22]) need to know at least the approximate mean and the approximate covariance of the additive noise. In Pickle, the cloud cannot know, or estimate with any accuracy, the covariance of $\boldsymbol{R}\boldsymbol{\varepsilon}$, since that depends upon $\boldsymbol{R}$, a quantity private to the user.

Finally, in Step 4, $\mathcal{U}$ sends $\boldsymbol{RX}$ to the cloud. The privacy properties of this dimensionality-reducing transform are proven in [32], which shows that $\boldsymbol{X}$ cannot be recovered

[4] It is also possible to directly regress inner products and distances using the functions but we have experimentally found that directly regressing these quantities does not result in improved accuracy over the methods described.

[5] In the *iPoD* optimization, the first and last terms of the RHS in (4) can be obtained directly from the users.

without knowing $\boldsymbol{R}$ — that is because there are infinite factorizations of $\tilde{\boldsymbol{X}}$ in the form of $\boldsymbol{RX}$. In fact, even if $\boldsymbol{R}$ is known, because the resulting system of equations is underdetermined, we can only reconstruct $\boldsymbol{X}$ in the sense of minimum norm.

Given this, using $\boldsymbol{\varepsilon}_u$ provides an additional layer of privacy. $\boldsymbol{\varepsilon}_u$ is a random matrix with real-valued elements, so it is highly infeasible for an adversary to guess its values successfully using brute force. The adversary may attempt to find approximate values for $\boldsymbol{\varepsilon}_u$, but would still be faced with the challenge of determining whether the resulting approximate value for $\boldsymbol{R}_u$ is correct; the only way to do this is to attempt to reconstruct the original feature vectors and see if they reveal (say) meaningful human speech or other recognizable sounds, and this is also computationally hard, as described above.

However, it may be possible for an attacker to *approximate* $\boldsymbol{X}$ using a *reconstruction attacks*. In Section 4.3, we show that Pickle is robust to these attacks as well.

Finally, Pickle is robust to collusion between the cloud and users. Since each user $\mathcal{U}$ *independently* selects a secret $\boldsymbol{R}$, and since its feature vectors are encoded using this secret, another user cannot directly compute any of $\mathcal{U}$'s original feature vectors from perturbed feature vectors it receives from the cloud (for the same reason that the cloud itself cannot compute these). A similar robustness claim holds for collusion between cloud servers.

3.5 Other Properties of Pickle

Besides ensuring the privacy of its training feature vectors, Pickle has several other properties.

Pickle is *computationally-efficient* on mobile devices, and incurs minimal communication cost. It requires two matrix multiplications (one for the regression stage and the other during training); classification steps require computing distances or inner products. It transmits a few matrices, and a classification model over the network. All of these, as we shall validate, require minimal resources on modern phones, and modest resources on the cloud infrastructure.

Pickle *requires no coordination* among participants and provides flexible levels of privacy. Each user can independently choose the privacy transformation matrix $\boldsymbol{R}$, and communicates only with the cloud. Users can also independently set the level of desired privacy by selecting the dimensions of $\boldsymbol{R}$ or the intensity of the additive noise matrix $\boldsymbol{\varepsilon}$. In Section 4.3, we explore the implications of these choices.

Pickle *disincentivizes free-riding*. A user $\mathcal{U}$ who does not contribute training samples, can get the model from other users, but, to use it, must also participate in at least the regression phase so that the cloud can compute f_{uv} and f_{vu} for all other users $\mathcal{V}$ whose vectors are included in the classifier.

Although we have discussed Pickle in the context of classification, it extends easily to other tasks like non-linear regression and estimating distributions; these tasks arise in the context of participatory sensing [1, 12, 23, 17, 5, 46].

Beyond SVM and kNN, Pickle can be applied to all kernel based classification and regression methods that use distances or inner-products to establish relationships between training samples. One can simply replace these distances or inner products with approximations derived by applying Pickle's regression functions.

Finally, Pickle can be *made robust to poisoning attacks* in which a *few* malicious users attempt to inject bogus data in order to render the model unusable.[6] For classification algorithms which build robust statistical models, attackers must inject distributionally different feature vectors in order to succeed. Prior work has examined these kinds of attacks and have proposed a distance-based approach, called Orca, to detecting poisoning attacks [9]. Because Pickle can approximately preserve distances, the cloud can run Orca even though it receives only perturbed data, as shown in Section 4.3.

4 Evaluation of Pickle

In this section, we perform three qualitatively different kinds of evaluation: a *user-study* which brings out the benefits of Pickle, especially for applications like speaker recognition where the un-perturbed feature vectors are known to leak privacy; *measurements on a prototype* that quantify the resource cost of Pickle; and an extensive characterization of the privacy-accuracy tradeoffs in Pickle, together with a comparison of alternatives, using an *evaluation on public data-sets*.

4.1 Pickle Privacy: A User Study

In a previous section, we asserted that a commonly used feature vector for acoustic applications, MFCC, could be used to approximately reconstruct original audio clips. In this section, we demonstrate this using a small-scale user-study on acoustic data, and show that: a) a scheme like Pickle is necessary, since without it, almost the entire audio clip can be reverse-engineered from unperturbed MFCC feature vectors; b) Pickle can mitigate this privacy leakage without significant loss in classification accuracy.

MFCC is widely used in acoustic mobile applications, like [35, 8, 34, 36, 42]. In particular, MFCC can be used to recognize speakers [42, 34] or their genders [36]; collaborative learning can be used to build models for both these applications. To quantify the efficacy of Pickle in MFCC for speaker recognition, we used spoken sentences from four volunteers (two men and two women) in the TIMIT dataset [20], and trained SVM (with RBF) models by extracting the standard 13-dimensional MFCC feature vectors from the audio clips, with and without Pickle. For Pickle feature vectors with a 50% dimensionality reduction and a 0.3 intensity of additive noise (denoted by $(50\%, 0.3)$), recognition accuracy is degraded only by 4.32%! We leave a more detailed discussion of Pickle's impact on accuracy for a later section, but now demonstrate how, with minimal accuracy loss, Pickle can greatly reduce privacy leakage.

To this end, we conducted a user study which used eight testing sentences (81 words) from the training set used to construct the classifier. For each sentence, users were asked

[6] Attacks in which a majority of contributors poison the model require other mechanisms. Such attacks can render a model completely useless for the corresponding classification task. In that case, a company that sells these collaboratively-designed models may offer monetary incentives to contributors, but only if the resulting model is shown to be accurate. Discussion of such mechanisms is beyond the scope of the paper.

to listen to three versions of this sentence in the following order: (i) a *Pickle-MFCC* audio clip generated by first applying a reconstruction attack (Section 4.3.3) to $(50\%, 0.3)$ Pickle-transformed MFCC feature vectors[7], and then applying [18] to reverse-engineer the audio clip from the estimated feature vector; (ii) an *MFCC* audio clip generated directly from the MFCC feature vectors using the method described in [18]; and (iii) the *original* audio clip. Users were asked to write down all the words they recognized in each of the clips. Seventeen users participated in the study, having varying levels of proficiency in English.

For each participant, we calculated two *Recognition Ratios (RRs)* for each sentence: Pickle-MFCC RR, is the ratio of the number of words recognized from the *Pickle-MFCC* clip divided by the number of words recognized in the original clip; and MFCC RR, is the ratio of the number of words recognized in the *MFCC* clip to that recognized in the original clip. As Figure 4.1 shows, Pickle offers very good privacy protection; averaged across all sentences, Pickle has an RR of only 1.75%, while the MFCC-reconstructed clips have an RR of 73.88%. Of the words recognized in Pickled clips, most were articles, prepositions or linking verbs, but three users recognized the phrase "below expectations" in one sentence, and one user recognized the words "informative prospective buyer" in another sentence. These words provide minimal information about the original sentences, since they lack vital context information.

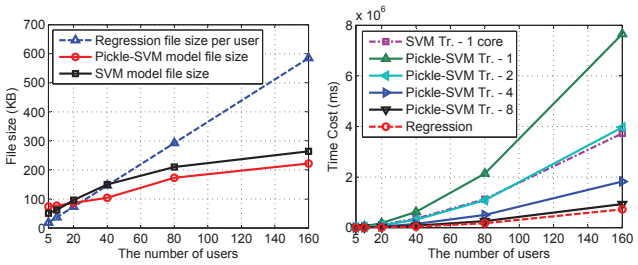
While a more extensive user study is the subject of future work, our study shows that, without Pickle, a collaborative learning task for speaker recognition can leak a majority of words in audio clips when MFCC is used as a feature vector; using Pickle, a negligible minority is leaked.

4.2 Pickle Resource Costs: Prototype Evaluation

Prototype System. We have implemented a prototype of Pickle (Figure 4) which consists of two components: software for *Android 2.3.4* (about 8K lines of Java code, about half of which is the Pickle-SVM engine) and *Windows Mobile 6.5* (about 11K lines of C# code, about 40% of which is the Pickle-SVM engine), and software for *the cloud*, written with .Net 4.0 framework (about 8K lines of code in C#, of which the Pickle-SVM engine is shared with the phone code). The prototype supports all functions required by Pickle, including regression parameter construction and interaction, raw data collection, feature vector generation, transformation, upload and classification, user labeling, outlier detection, model training, and model return. The phone software supports the collection and processing of acceleration and acoustic data, and the cloud component builds a Pickle-SVM classifier with four optional kernels. Support for other sensors and other classifiers is left to future work.

Experimental Methodology. Using this prototype, we have collected 16,000 accelerometer-based feature vectors, collected using smartphones, for the purpose of evaluating the resource costs for collaborative learning. For evaluating collaborative training, we cluster these feature vectors among

[7]The MFCC feature vectors were generated using 25ms overlapping frames with an inter-frame "hop" length of 10ms



(a) Communication Cost (b) Processing Cost

Figure 5—Processing and Communication Cost

160 users each of whom submits, on average, 100 training samples. As shown in Figure 3, when each feature vector has 16 dimensions, the resulting classifier has an accuracy of over 97% even when feature vectors are perturbed by $(50\%, 0.3)$. A more detailed analysis of classifier accuracy is discussed in Section 4.3. Our data set is adversarially chosen; prior work on mobile-phone based sound classification [35] has used half the number of dimensions and an order of magnitude smaller training data set. Thus, from our experiments, we hope to understand how high Pickle's resource costs can be in practice.

We report on experiments conducted on both a Nexus One and an HTC Pure phone and our "cloud" is emulated by a cluster of four Intel(R) Core(TM)2 Duo 2.53 GHz PCs, with 3GB RAM, running Windows 7.

Communication Overhead. In Pickle, each user needs to send the perturbed data, $\boldsymbol{R}_u\boldsymbol{X}_u$, and inner products calculated from her own feature vectors, $\boldsymbol{X}_u^T\boldsymbol{X}_u$ to the cloud, which incurs communication overhead. (The overhead of sending the public feature vectors $\boldsymbol{Z}$ and having each user return $\boldsymbol{R}_u(\boldsymbol{Z}+\boldsymbol{\varepsilon}_u)$ to the cloud is relatively small since the number of feature vectors is small (Section 3.2), so we do not report the cost of this operation). Since our privacy transformation reduces dimensionality, the communication cost of sending the perturbed data is actually lower than the cost of the original data. In our experiment, we use a privacy transformation, with relatively higher communication cost, which reduces dimensionality by only 25%, and adds 0.3 intensity additive noise. In our implementation, each user's perturbed training samples only requires 15KB for the transformed feature vectors with labels and 94KB for the inner products. For comparison, the original training samples without perturbation require 21KB.

The final component of the communication cost is the size of the returned model. This cost has two components for Pickle-SVM: the set of model parameters and perturbed support vectors, and the collection of regression coefficients (each user needs to download only her own regression coefficients, not the entire set of coefficients). For 160 users, the former is 222 KB (Figure 5(a)), and the latter is 585 KB per user. For comparison, the model size for 160 users without Pickle is 264 KB. Pickle's dimensionality-reduction results in a smaller model size.

Overall, these numbers are well within the realm of practicality, especially because our evaluation setting is adversarial and our implementation is un-optimized. For example, simply compressing the data, a factor of 2-3 reduction in transfer overhead can be obtained.

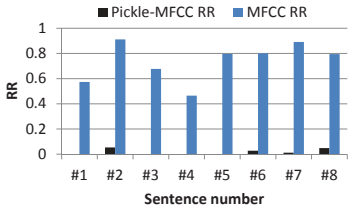


Figure 2—User Study Results

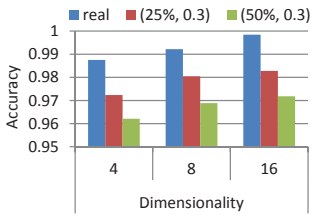


Figure 3—Accuracy of accelerometer-based classifier construction

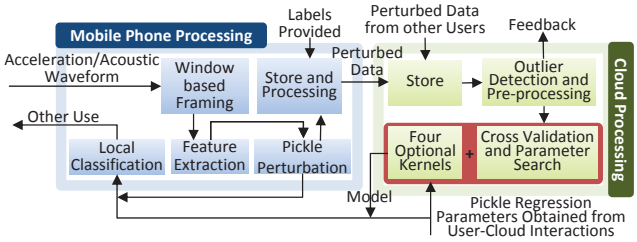


Figure 4—Architecture of the prototype system

Computational Cost. On the mobile phone, it takes on average (over 50 runs) of less than 1 ms on both the Nexus One and the HTC Pure to multiplicatively transform a feature vector. Classifying a test vector on large model constructed from 160 users takes on average 266.7 ms and 477.6 ms on the two phones respectively. For comparison, classifying on a model generated from pure SVM (without perturbation) takes on average 128.1 ms and 231.6 ms on the two phones. The main overhead of Pickle comes from using the regression coefficients to estimate distances from the vector to be classified to the support vectors. Both of these numbers are reasonable, especially since our dataset is large.

On the cloud, the processing of outlier detection is very fast – only 10.55 ms on average (all numbers averaged over 10 runs). Computing regression coefficients for pairs of users is shown in Figure 5(b), the average cost increasing from 0.55 s to 723s as the number of users increases from 5 to 160. However, the cost of model generation on the cloud is significantly higher, on average about 2.13 hours on a single core. Without Pickle, model generation is a factor of 2 faster (Figure 5(b)). Again, Pickle's overhead mainly comes from the regression calculations.

However, a major component of model generation, performing a grid search to estimate optimal parameters for Pickle-SVM, can be parallelized, and we have implemented this. As shown in Figure 5(b), as the number of cores increases, an almost linear speed-up can be obtained; with 8 cores, model generation time was reduced to 0.26 hours. Devising more efficient parallel algorithms for model generation is left to future work.

Finally, as discussed in Section 3, a user who has not contributed feature vectors can use the generated model, but the cloud needs to compute regression coefficients for this new user, relative to other users whose vectors are included in the classifier model. This computation can be performed incrementally, requiring only 8.71s in our 160 users experiment, and adding 160 regression coefficient entries (222 KB) that need to be downloaded only by the new user.

4.3 Accuracy/Privacy Tradeoffs and Comparisons: Dataset Evaluation

In this section, we evaluate Pickle's ability to preserve privacy without sacrificing classification accuracy by analyzing public datasets. We also explore the sensitivity of our results to the different design choices presented in Section 3.

4.3.1 Methodology

Data Sets. We use four datasets to validate Pickle: *Iris*, *Pima Indians Diabetes*, *Wine*, and *Vehicle Silhouettes*. The datasets are from the UCI Machine Learning Repository[8], and are some of the most widely-used datasets in the machine-learning community. All the feature values in each dataset are scaled between 0 and 1.

Users. We partition each data set into several parts to simulate multiple users with private data. To do this, we clustered the feature vectors in each data set using the standard K-means clustering algorithm and assigned each cluster to one "user". (Random partitions would not have been adversarial enough as our main goal is to collaboratively learn from data with *disparate* statistics.) Using this method, the number of users is 2, 5, 2, 5 for the four datasets respectively.

Although these numbers of users are small relative to our targeted scale, we note that the level of privacy and the classification accuracy are *not* likely to become worse with more users. If anything, classification accuracy will improve with more users since one has more and diverse training data.

In our experiments, we use all four datasets to evaluate the performance with 2 users, and also use the Diabetes and Vehicle datasets to test the performance with 5 users. After partitioning the data across users, we randomly select 80% of the labeled feature vectors from each user as the *training data*, and use the remaining for *testing*.

Classifiers. We evaluate the effectiveness of Pickle using two common pattern classifiers: Support Vector Machine (SVM) and k-Nearest Neighbor (kNN). We experiment SVM with the most widely-used RBF kernel as well as the Linear kernel and tune SVM parameters using standard techniques like cross-validation and grid-search. We use a more accurate variant of kNN called LMNN [53] which uses Mahalonobis distances instead of Euclidean distances.

4.3.2 Evaluation Metrics

We use two metrics to evaluate the effectiveness of Pickle. The first assesses how much privacy is preserved and how likely users' private data are to be compromised. The second measures how much Pickle's accuracy is affected by its privacy transformations.

The Privacy Metric. Pickle distorts feature vectors to "hide" them. One way to measure privacy is to quantify the extent of this distortion. We use a slightly more adversarial privacy metric from prior work [2, 49], which measures

[8] http://archive.ics.uci.edu/ml

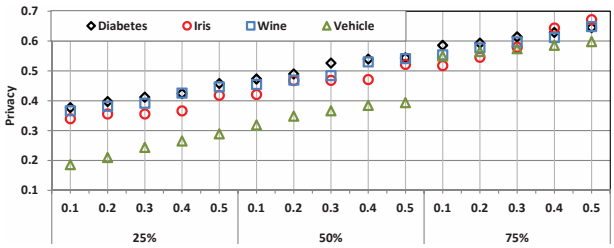


Figure 6—Effect of reconstruction attack on privacy

the "distance" between the original feature vector and an estimate for that vector derived by mounting a *reconstruction attack*. Specifically, let $\boldsymbol{x}_u^d$ stand for the d-th dimension of the feature vector $\boldsymbol{x}_u$, and $\boldsymbol{h}_u^d$ be the corresponding dimension in the reconstructed vector. Then, we can define ℓ_{ud} ($0 \leq \ell_{ud} \leq 1$) to be the difference in the *distributions* of these two quantities, and the privacy metric ℓ ($0 \leq \ell \leq 1$) as ℓ_{ud} averaged over all users and dimensions.

Intuitively, the larger the ℓ, the more confident we are that privacy is preserved. When ℓ is zero, we are less confident. Note that we cannot infer directly that privacy is violated when $\ell = 0$, as the metric only measures *difference* in expectation. Furthermore, the metric is not perfect, since if the original and reconstructed vectors are distributionally different then, regardless of the magnitude of this difference, ℓ is 1. Finally, we emphasize that ℓ is defined with respect to a specific attack on the perturbed feature vectors.

Classification Accuracy. A privacy transformation can adversely affect the classification accuracy, so we are interested in measuring classification accuracy under different privacy levels. We compute the accuracy in a standard way, as the percentage of correctly classified test feature vectors among all test feature vectors. All reported results are averaged over 20 random splits of training, validation and testing data sets.

4.3.3 Attack models and Privacy

In Section 3, we had already discussed a few attack strategies, to which Pickle is resilient. We now discuss somewhat more sophisticated attacks that are based on an intimate knowledge of how Pickle works.

The Reconstruction Attack. Dimensionality-reduction techniques can be attacked by *approximate* reconstruction. By reconstructing original data to the extent possible, these attacks function as a preprocessing step to other types of attacks. In Pickle, the cloud sends the public data $\boldsymbol{Z}$ to a user $\mathcal{U}$ and receives transformed ones $\boldsymbol{Z}_u = \boldsymbol{R}_u(\boldsymbol{Z} + \boldsymbol{\varepsilon}_u)$. While the cloud cannot decipher $\boldsymbol{R}_u$ and $\boldsymbol{\varepsilon}_u$, can the cloud use its knowledge to infer important statistical properties of these variables to approximately *reconstruct* the user's data when she sends actual training vectors for building classifiers? One possible approach is to build a regression model such that $\boldsymbol{Z} \approx h_u(\boldsymbol{Z}_u; \boldsymbol{\beta})$. When the user sends $\boldsymbol{R}_u\boldsymbol{X}_u$, the cloud applies the regression model and tries to recover $\boldsymbol{H}_u \approx h_u(\boldsymbol{R}_u\boldsymbol{X}_u; \boldsymbol{\beta})$.

Figure 6 shows that, even when this attack uses Gaussian Process Regression, Pickle still provides significant privacy. To generate the plot, we compute ℓ for this attack, for various combinations of multiplicative and additive transformations: reducing the dimensionality for the multiplicative transform by 25%, 50% and 75% of the original dimensionality, and adding noise with intensities (Section 3.2) ranging from 0.1 to 0.5 in steps of 0.1. The figure shows the resulting privacy-level metric for each combination of additive and multiplicative transforms under the attack; the resulting privacy levels range from 0.1-0.7. Thus, depending on the degree to which the training data have been transformed, Pickle can be significantly robust to this attack.

The intuition for why Pickle is robust to this reconstruction attack is as follows. Pickle's regression phase learns about relationships between users enough to *discriminate* amongst them. However, the regression is not powerful enough to *generate* the original samples; intuitively, much more information is necessary for generation than for discrimination.

Followup ICA Attack. The cloud can also improve its estimate $\boldsymbol{H}_u$ with a followup strategy. For example, ICA can be used for this purpose [32, 15]. However, we have experimentally verified that this strategy is unsuccessful with Pickle – the ICA algorithm fails to converge to meaningful solutions.

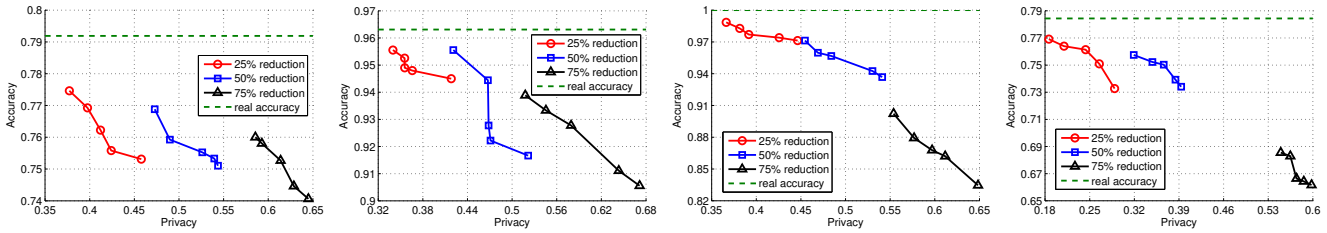
4.3.4 Classifier Accuracy

In this section, we discuss results for the classification accuracy of SVM (with RBF and Linear kernels) and LMNN, using Pickle for 2 users from each dataset. Results for Diabetes and Vehicle with 5 users are omitted but are qualitatively similar except that they have higher classification accuracy because they have a larger training set. These experiments on each of our four data sets use a *baseline configuration* which uses synthesized public feature vectors and *iPoD*. In subsequent sections, we deviate from this baseline configuration to examine different design choices.

Figure 7 plots the classification accuracy for each data set as a function of the privacy-level, for the SVM classifier with the RBF kernel. In this plot, the horizontal line shows the classification accuracy without Pickle. For this classifier, across all four data sets, the loss in classification accuracy is less than 6% for privacy levels up to 0.5; in the worst case (Wine) classification accuracy drops by 15% for a privacy-level of 0.65. This is an important result of the paper: *even when Pickle transforms data so that reconstructed feature vectors are distributionally different from the original ones*, classification accuracy is only modestly affected.

Other features are evident from this figure. In general, classification accuracy drops with privacy-level, but the relationship is non-monotonic: for example, for the Diabetes dataset, 50% reduction with 0.1 intensity of additive noise has higher privacy, but also higher accuracy than 25% with 0.5 intensity. Second, the RBF kernel outperforms the Linear kernel (graphs are omitted to save space) for which a 0.5 privacy-level results in a 10% reduction in classification accuracy over all datasets, and nearly 20% in the worst case.

Finally, Pickle performs well even for nearest neighbor classification (figures omitted for space reasons). For LMNN with $k = 5$, Pickle is within 5% of the actual classification accuracy for each data set for privacy levels to 0.5, and in the worst case incurs a degradation of about 15%. Moreover, for LMNN, in some cases Pickle is even more accurate than without any privacy transformations. This is likely due to the regularization effect caused by noise (either additive or



(a) Diabetes (b) Iris (c) Wine (d) Vehicle

Figure 7—Accuracy-Privacy tradeoff of SVM with RBF Kernel

as a result of regression), which prevents overfitting of the models.

4.3.5 *Comparison*

In this section, we compare Pickle for SVM (with RBF kernel[9]), using the baseline configuration discussed above, against three previously-proposed approaches for preserving the privacy of feature vectors. As we show below, compared to Pickle, these approaches either do not preserve privacy adequately, or are significantly inaccurate.

The first algorithm only adds additive noise [11] and uses Bayes estimation [22] to attack the perturbed feature vectors. For this alternative, we compute the privacy-level based on the Bayesian reconstruction. This alternative is chosen to understand the performance of a simpler additive perturbation. The second algorithm uses the Random Projection (RP) [32] in which each user transforms feature vectors using the same multiplicative noise matrix **R**. To be robust to inversion, the dimensionality of **R** is reduced by more than 50% relative to the dimensions of the original feature vectors. For this case, we derive the privacy-levels by using a pseudo-inverse based attack [32]. Our third algorithm is a KDE-based method ([49]), in which users never send true data, but only send synthetic data drawn from the estimated feature vector distributions. For this case, we compute the privacy-levels using the transformed feature vectors.

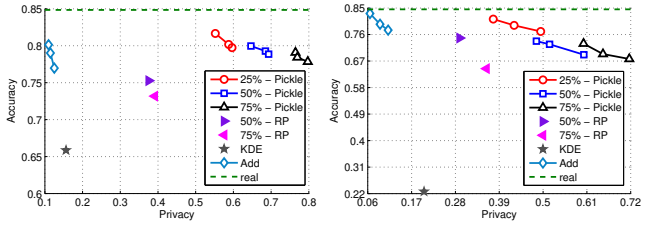
As Figure 8 shows, on the Diabetes and Vehicle datasets with 5 users, Pickle outperforms all alternatives. The additive noise based approach[10] produces acceptable accuracy, but almost no privacy. The KDE-based method offers a little bit more privacy than the additive noise method, but with a significantly degraded accuracy. Finally, the RP method provides, in general, lower privacy than Pickle, and also lower accuracy for data points with comparable privacy. The same results are true for all the four datasets with 2 users, so we have omitted these for lack of space.

4.3.6 *Impact of Design Choices*

Is Regression Necessary? Our main contribution is the use of regression to learn function relationships. For all our datasets and classifiers, turning off regression and using the transformed feature vectors directly for computing distances and inner products, leads to 15-35% accuracy degradations compared to Pickle (graphs are omitted for space reasons).

[9]Results for Linear kernel and LMNN are omitted but are qualitatively similar

[10]For this approach and each dimensionality setting of Pickle, we changes the additive noise intensity from 0.1 to 0.5 in steps of 0.2.



(a) Diabetes dataset (b) Vehicle dataset

Figure 8—Comparison of Pickle to several alternatives

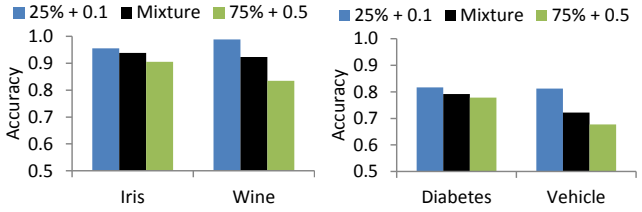
This drop is unacceptable for most applications, and motivates the importance of our approach.

Other Design Choices. Disabling the *iPoD* extension can reduce accuracy up to 7% for SVM and up to 6% for LMNN, so it is beneficial for Pickle to use *iPoD*. As we have discussed, the bandwidth cost of transmitting this matrix is very small. We have also experimented with other public feature vector generation methods: a *DIRECT* method in which the cloud obtains a few unperturbed feature vectors from users; a *NOISY* method which adds additive noise to the vectors of the *DIRECT* method; and an *ARBITRARY* method in which the cloud arbitrarily generates public vectors. We find that our *SYNTHESIS* method occupies a sweet spot: it is significantly more accurate, but not much less private, than *ARBITRARY*, and provides higher privacy, without sacrificing accuracy, than the other two methods.

4.3.7 *Illustrating Other Features of Pickle*

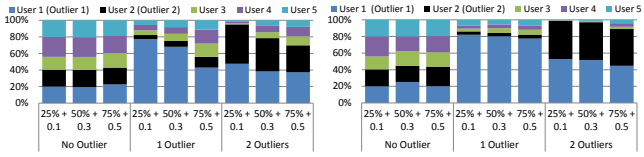
User Diversity. Pickle allows users to independently tune their own privacy transformations. Using SVM with RBF kernel (results for Linear kernel and LMNN are omitted but are qualitatively similar), Figure 9 considers the case of two different privacy settings: a 25% dimensionality reduction with 0.1 intensity additive noise and a 75% dimensionality reduction with 0.5 intensity additive noise. It plots the classification accuracy for three cases: when all users use the first setting, when all users use the second setting, and when users use a mixture of those two settings. The resulting classification accuracy is intermediate for the mixture setting, relative to the other settings. This is encouraging: a less encouraging outcome would have been if the accuracy of the mixture setting was closer to the second setting, since this would mean that users with high privacy requirements could dictate the performance of Pickle.

Robustness to Poisoning. We have implemented the Orca [9] outlier (malicious user) detection algorithm as dis-



(a) 2 users (b) 5 users

Figure 9—Effect of user diversity on accuracy



(a) Diabetes (b) Vehicle

Figure 10—Outlier detection against model poisoning

cussed in Section 3.5), and use our estimates of Euclidean distance in that algorithm. Orca essentially ranks suspicious feature vectors, so we conduct three experiments in which there are 5 users and 0, 1 and 2 of them (respectively) attempt to poison the model generation process by injecting completely random data. In Figure 10, we plot the fraction of the top-100 suspicious feature vectors that belong to each user. When there are no outliers, the distribution is uniform across all five users. However, in the presence of outliers, their feature vectors occupy a disproportionate number of the top hundred suspicious feature vectors. This experiment shows that Pickle can be easily retrofitted into an existing poisoning detector.

In our experiments, we simply discard all outliers before building the classifier. However, it is also possible that a small amount of noisy data (e.g., mislabeled training samples) is contained in the user's training data, but does not affect the data's overall distribution. In this case, the classifier construction process can finally filter these non-representative samples by only selecting the most useful feature vectors for classification.

5 Related Work

Privacy-preserving SVM. There have been several pieces of work on privacy-preserving SVM classifier construction, but each lacks support for a design dimension that is crucial for collaborative SVM classifier construction using mobile sensing.

Feature-perturbation: Closest to our work is the body of work that perturbs feature vectors before transmitting them to a server/cloud. The work of Lin and Chen [30, 29, 31] only considers privacy-preserving classifier training for a single user, but Pickle explicitly supports multiple users. Some approaches require that all participants share a common perturbation matrix [38, 47], while Pickle does not. Other approaches [57, 39, 21] focus on vertically partitioned data, where elements of the feature vector are spread among participants; by contrast, in our setting, the data is horizontally partitioned. An approach that could have been plausibly used for collaborative learning [49] generates synthetic feature vectors whose statistics match the original feature vectors [49]; we have compared Pickle against this and shown that it can result in poor accuracy.

Differential Privacy: Beyond perturbing the input feature vectors, some approaches have explored the use of the differential privacy framework for privacy-preserving SVM construction. In these approaches [48, 13], the classifier construction assumes all the *original* feature vectors are available (unlike Pickle, which perturbs the original feature vectors) and the outputs of the classifiers are perturbed such that individual features are not exposed as a result of small differences in two databases (such as two different versions of training samples). This is achieved by adding noise either to the classifier's parameter vector after optimization or to the objective function itself, thus prior to optimization. Intuitively, these approaches attempt to make it difficult to infer who might have contributed feature vectors, while Pickle hides the content of the feature vector itself. Thus, the two approaches are complementary, and exploring a combination of these two methods is left to future work.

Other Cryptographic Methods: Other methods have attempted to use cryptographic techniques to preserve privacy in SVM construction. A few use homomorphic encryption, but either discuss only SVM construction for two participants [28] or would require peer-to-peer communication [59, 37], whereas Pickle permits multiple users and does not require them to communicate with each other. Finally, several pieces of work [58, 27, 51] use a form of secure multiparty communication, but assume that participants do not collude, an assumption that Pickle does not make. (Of course, not all secure multi-party communication methods assume participants do not collude, but, when applied to the Pickle setting, these methods have the drawback that all participants must be online when *any* participant wishes to *use* the classifier, an unwieldy assumption at best.)

In summary, in the space of prior work on privacy-preserving SVM, Pickle occupies a unique niche largely driven by the requirements and constraints of collaborative learning using sensor data generated from mobile phones.

Other Related Work. Navia-Vasquez et al. [45] consider distributed SVM classifier construction, but do not consider privacy. Many pieces of research in the mobile sensing literature have used machine-learning classifiers for various applications (e.g., [35, 8, 10, 43, 4], and SVM is often a popular choice. A few have examined collaborative learning. Closest to our work is that of Ahmadi *et al.* [5] who consider the problem of accurately estimating a linear regression model from user contributed sensor data, while still ensuring the privacy of the contributions. While this is an instance of privacy-preserving collaborative learning, it is unclear how to extend the approach to nonlinear classifiers; as we have discussed above, for such classifiers it is necessary to carefully design privacy transforms that preserve certain relationships between contributed feature vectors. MoVi [8] is an application in which users within a social group collaboratively, using the cloud, sense their environment and recognize interesting events. However, MoVi assumes that users within a group trust each other, and that the cloud can

be trusted not to reveal data from one group to third parties. Finally, Darwin [42] directly addresses collaborative learning, but does not address privacy and assumes a trustworthy cloud.

Privacy-preservation has, in general, received much more attention in the data mining community which has considered cryptographic methods (e.g., [50, 24]) for clustering and other mining operations. In general, these methods do not scale to many users and require computationally-intensive encoding and decoding operations. That community has also considered anonymization of structured data (such as relational tables) to ensure privacy of individual entries without significantly compromising query results. By now, it is well known that anonymization is vulnerable to composition attacks using side information [19].

Preserving privacy through *perturbation* or *randomization* is most relevant to our work. One body of work has considered data perturbation techniques for datasets using various methods [56, 52, 55, 33] for dataset exchange between two parties; it is unclear how to extend this body of work to Pickle's multi-party setting where the parties are assumed to be able to collude. Additive-noise based randomization perturbs the original data with additive noise (e.g., [3, 2]), but is susceptible to *reconstruction attacks*, in which the spectral properties of the perturbed data can be used to filter the additive noise and recover the original data [26]. Multiplicative noise based perturbation (e.g., [14, 32]) can be robust to these reconstruction attacks. In some approaches (e.g., [14]), the multiplicative noise is dimensionality-preserving while in others [32], it is not. Dimensionality-preserving transformations can preserve inner products and Euclidean distances. Unfortunately, a dimensionality-preserving multiplicative transformation is susceptible to *approximate reconstruction* [32]. Furthermore, if this method is applied to collaborative learning, then participants must agree upon the matrix $\boldsymbol{R}$, and collusion attacks may succeed. It is for this reason that Pickle uses a dimensionality-reducing transformation using per-user private matrices, and then uses a regression phase to recover inter-user relationships so that it can approximately infer Euclidean distances and inner products.

6 Conclusions

In this paper, we have described Pickle, an approach to preserving privacy in mobile collaborative learning. Pickle perturbs training feature vectors submitted by users, but uses a novel regression technique to learn relationships between training data that are required to maintain classifier accuracy. Pickle is robust, by design, to many kinds of attacks including direct inversion, collusion, reconstruction, and poisoning. Despite this, Pickle shows remarkable classification accuracy for the most commonly used classifiers, SVM and kNN. Finally, Pickle requires minimal computing resources on the mobile device, and modest resources on the cloud. Many avenues for future work remain, including an exploration of more sophisticated regression methods and other classifiers, an extension of applying Pickle to participatory sensing, a more extensive and refined design of user study, and a cryptanalysis of our dimensionality-reduction.

Acknowledgements. We thank the anonymous referees and our shepherd Srdjan Capkun for their comments. We are grateful to members of the Embedded Networks Laboratory who participated in the user study.

7 References

[1] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for human spaces. *IEEE Pervasive Computing*, 6(2), 2007.

[2] D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '01)*, 2001.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM Special Interest Group on Management Of Data (SIGMOD '00)*, 2000.

[4] H. Ahmadi, N. Pham, R. Ganti, T. Abdelzaher, S. Nath, and J. Han. A framework of energy efficient mobile sensing for automatic human state recognition. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services (MobiSys '09)*, 2009.

[5] H. Ahmadi, N. Pham, R. Ganti, T. Abdelzaher, S. Nath, and J. Han. Privacy-aware regression modeling of participatory sensing data. In *Proceedings of the 8th ACM Conference on Embedded Network Sensor Systems (SenSys '10)*, 2010.

[6] M. Almeida, G. Cavalheiro, A. Pereira, and A. Andrade. Investigation of age-related changes in physiological kinetic tremor. *Annals of Biomedical Engineering*, 38(11), 2010.

[7] M. Azizyan, I. Constandache, and R. Choudhury. Surroundsense: mobile phone localization via ambience fingerprinting. In *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (Mobicom '09)*, 2009.

[8] X. Bao and R. Choudhury. Movi: mobile phone based video highlights via collaborative sensing. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys '10)*, 2010.

[9] S. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the 9th ACM Special Interest Group on Knowledge Discovery and Data mining (SIGKDD '03)*, 2003.

[10] A. Bose, X. Hu, K. Shin, and T. Park. Behavioral detection of malware on mobile handsets. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys '08)*, 2008.

[11] R. Brand. Microdata protection through noise addition. *Inference Control in Statistical Databases*, 2002.

[12] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. Srivastava. Participatory sensing. In *Proceedings of the 2006 World Sensor Web Workshop*, 2006.

[13] K. Chaudhuri, C. Monteleoni, and A. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12, 2011.

[14] K. Chen and L. Liu. Privacy-preserving data classification with rotation perturbation. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM '05)*, 2005.

[15] K. Chen and L. Liu. Towards attack-resilient geometric data perturbation. In *SIAM International Conference on Data Mining (SDM '07)*, 2007.

[16] C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20(3):273–297, 1995.

[17] S. Eisenman, E. Miluzzo, N. Lane, R. Peterson, G. Ahn, and A. Campbell. The bikenet mobile sensing system for cyclist experience mapping. In *Proceedings of the 5th ACM Conference on Embedded Network Sensor Systems (SenSys '07)*, 2007.

[18] D. Ellis. PLP and RASTA (and MFCC, and inversion) in Matlab, 2005. online web resource.

[19] S. Ganta, S. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '08)*, 2008.

[20] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, and N. Dahlgren. Darpa timit acoustic phonetic continuous speech corpus cdrom, 1993.

[21] Y. Hu, F. Liang, and G. He. Privacy-preserving svm classification on vertically partitioned data without secure multi-party computation. In *Proceedings of the 5th International Conference on Natural Computation (ICNC '09)*, 2009.

[22] Z. Huang, W. Du, and B. Chen. Deriving private information from randomized data. In *Proceedings of the 2005 ACM Special Interest Group on Management Of Data (SIGMOD '05)*, 2005.

[23] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th ACM Conference on Embedded Network Sensor Systems (SenSys '06)*, 2006.

[24] G. Jagannathan and R. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the 11th ACM Special Interest Group on Knowledge Discovery and Data mining (SIGKDD '05)*, 2005.

[25] W. Johnson and J. Lindenstrauss. Extensions of lipshitz mapping into hilbert space. *Contemp. Maths.*, 26, 1984.

[26] S. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM '03)*, 2003.

[27] D. Kim, M. Azim, and J. Park. Privacy preserving support vector machines in wireless sensor networks. In *Proceedings of the 3rd International Conference on Availability, Reliability and Security (ARES '08)*, 2008.

[28] S. Laur, H. Lipmaa, and T. Mielikainen. Cryptographically private support vector machines. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '06)*, 2006.

[29] K. Lin and M. Chen. Releasing the svm classifier with privacy-preservation. In *Prodeedings of the 8th IEEE International Conference on Data Mining (ICDM '08)*, 2008.

[30] K. Lin and M. Chen. Privacy-preserving outsourcing support vector machines with random transformation. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '10)*, 2010.

[31] K. Lin and M. Chen. On the design and analysis of the privacy-preserving svm classifier. *IEEE Transactions on Knowledge and Data Engineering*, 23(11), 2011.

[32] K. Liu, H. Kargupta, and J. Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Trans. on Knowledge and Data Engineering*, 18(1), 2006.

[33] L. Liu, J. Wang, and J. Zhang. Wavelet-based data perturbation for simultaneous privacy-preserving and statistics-preserving. In *Proceedings of the 8th IEEE International Conference on Data Mining Workshops*, 2008.

[34] H. Lu, B. Brush, B. Priyantha, A. Karlson, and J. Liu. Speakersense: Energy efficient unobtrusive speaker identification on mobile phones. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys '11)*, 2011.

[35] H. Lu, W. Pan, N. Lane, T. Choudhury, and A. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services (MobiSys '09)*, 2009.

[36] T. Maekawa, Y. Yanagisawa, Y. Kishino, K. Ishiguro, K. Kamei, Y. Sakurai, and T. Okadome. Object-based activity recognition with heterogeneous sensors on wrist. 2010.

[37] E. Magkos, M. Maragoudakis, V. Chrissikopoulos, and S. Gritzalis. Accurate and large-scale privacy-preserving data mining using the election paradigm. *Data and Knowledge Engineering*, 68(11), 2009.

[38] O. Mangasarian and E. Wild. Privacy-preserving classification of horizontally partitioned data via random kernels. In *Prodeedings of the 8th IEEE International Conference on Data Mining (ICDM '08)*, 2008.

[39] O. Mangasarian, E. Wild, and G. Fung. Privacy-preserving classification of vertically partitioned data via random kernels. *ACM Transactions on Knowledge Discovery from Data*, 2(3), 2008.

[40] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik. Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking (Mobicom '08)*, 2008.

[41] B. Milner and X. Shao. Clean speech reconstruction from MFCC vectors and fundamental frequency using an integrated front-end. *Speech Communication*, 48(6), 2006.

[42] E. Miluzzo, C. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys '10)*, 2010.

[43] E. Miluzzo, N. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. Eisenman, X. Zheng, and A. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys '08)*, 2008.

[44] P. Mohan, V. Padmanabhan, and R. Ramjee. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys '08)*, 2008.

[45] A. Navia-Vazquez, D. Gutierrez-Gonzalez, E. Parrado-Hernandez, and J. Navarro-Abellan. Distributed support vector machines. *IEEE Transactions on Neural Networks*, 2006.

[46] N. Pham, R. Ganti, Y. Uddin, S. Nath, and T. Abdelzaher. Privacy-preserving reconstruction of multidimensional data maps in vehicular participatory sensing. In *Wireless Sensor Networks*, 2011.

[47] J. Qiang, B. Yang, Q. Li, and L. Jing. Privacy-preserving svm of horizontally partitioned data for linear classification. In *Prodeedings of the 4th International Congress on Image and Signal Processing (CISP '11)*, 2011.

[48] B. Rubinstein, P. Bartlett, L. Huang, and N. Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *Arxiv Preprint arXiv:0911.5708*, 2009.

[49] V. Tan and S. Ng. Privacy-preserving sharing of horizontally-distributed private data for constructing accurate classifiers. In *Proceedings of the 1st ACM SIGKDD International Conference on Privacy, Security, and Trust in KDD*, 2007.

[50] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the 9th ACM Special Interest Group on Knowledge Discovery and Data mining (SIGKDD '03)*, 2003.

[51] J. Vaidya, H. Yu, and X. Jiang. Privacy preserving svm classification. *Knowledge and Information Systems*, 14(2), 2008.

[52] J. Wang and J. Zhang. Nnmf-based factorization techniques for high-accuracy privacy protection on non-negative-valued datasets. In *Proceedings of the 6th IEEE International Conference on Data Mining Workshops*, 2006.

[53] K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(2), 2009.

[54] P. Weinzaepfel, H. Jegou, and P. Perez. Reconstructing an image from its local descriptors. In *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '11)*, 2011.

[55] S. Xu and S. Lai. Fast fourier transform based data perturbation method for privacy protection. In *IEEE Intelligence and Security Informatics*, 2007.

[56] S. Xu, J. Zhang, D. Han, and J. Wang. Singular value decomposition based data distortion strategy for privacy protection. *Knowledge and Information Systems*, 10(3), 2006.

[57] H. Yu, J. Vaidya, and X. Jiang. Privacy-preserving svm classification on vertically partitioned data. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '06)*, 2006.

[58] H. Yu, J. Vaidya, and X. Jiang. Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. In *Proceedings of the 21st Annual ACM Symposium on Applied Computing (SAC '06)*, 2006.

[59] J. Zhan, L. Chang, and S. Matwin. Privacy-preserving support vector machines learning. In *Proceedings of the 2005 International Conference on Electronic Business (ICEB '05)*, 2005.