User-Controllable Learning of Location Privacy Policies With Gaussian Mixture Models

Justin Cranshaw and **Jonathan Mugan** and **Norman Sadeh**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{jcransh,jmugan,sadeh}@cs.cmu.edu

Abstract

With smart-phones becoming increasingly commonplace, there has been a subsequent surge in applications that continuously track the location of users. However, serious privacy concerns arise as people start to widely adopt these applications. Users will need to maintain policies to determine under which circumstances to share their location. Specifying these policies however, is a cumbersome task, suggesting that machine learning might be helpful. In this paper, we present a user-controllable method for learning location sharing policies. We use a classifier based on multivariate Gaussian mixtures that is suitably modified so as to restrict the evolution of the underlying policy to favor incremental and therefore human-understandable changes as new data arrives. We evaluate the model on real location-sharing policies collected from a live location-sharing social network, and we show that our method can learn policies in a user-controllable setting that are just as accurate as policies that do not evolve incrementally. Additionally, we highlight the strength of the generative modeling approach we take, by showing how our model easily extends to the semi-supervised setting.

Introduction

Hundreds of millions of people around the world now carry smart-phones. The rich array of sensors on these devices has allowed web-enabled applications to enter the lives and routines of users in increasingly intimate and unprecedented ways. For example, applications such as Foursquare and Facebook Places allow users to "check-in" to places they visit, broadcasting their location to their social connections on these sites. Although check-in applications require users to initiate the location sharing on a case by case basis, as users grow more comfortable with the technology, more applications will continuously track the location of users. The wide adoption of continuous tracking technology will create both opportunities and challenges for users and application designers alike.

Continuous tracking opens up a whole new range of uses and benefits to users over check-in services. For instance, in social location sharing applications, continuously tracking is better suited for coordination and for encouraging serendipitous encounters than check-in systems, which only ever offer a discrete approximation of users' routines. Examples of continuous social location sharing applications include Latitude, Glympse, and Locaccino[1]. Beyond social location sharing, numerous mobile applications are collecting users' locations for a variety of purposes. For instance, there are now several applications that continuously track users for health purposes, for example by estimating their caloric output based on their movements. Additionally, tracking the locations of users can be used to collaboratively learn recommendations for new places the users might want to visit (Zheng and Xie 2011). However, continuous tracking also raises significant privacy concerns that have to be addressed before such applications gain wider appeal (Beresford and Stajano 2005; Gedik and Liu 2007; Tsai et al. 2010; Benisch et al. 2011). Smart phone OS manufactures, such as Apple and Google, typically attempt to mitigate these concerns by requiring users to manually give permission to every application that requests their location. However, with such sensitive data being collected in such large quantities, these coarse controls are bound to lead to scenarios where unintended, potentially damaging information is leaked to the wrong entities. The potential for abuse of this data is great. Because of these concerns, privacy-conscious applications will need to maintain an access control policy that decides when to disclose the user's location. Although, requiring users to stipulate this policy in advance would give them maximum control, specifying complex sharing preferences can quickly become too difficult for the user, possibly preventing them from adopting the technology (Sadeh et al. 2009; Maxion and Reeder 2005).

If users were to label their access control preferences on a small sample of sharing scenarios, then machine learning could be used given this data to find accurate policy functions. However, this solution also poses significant challenges. In straight-forward applications of machine learning it would be difficult for users to be able to control the

[1] http://www.locaccino.org

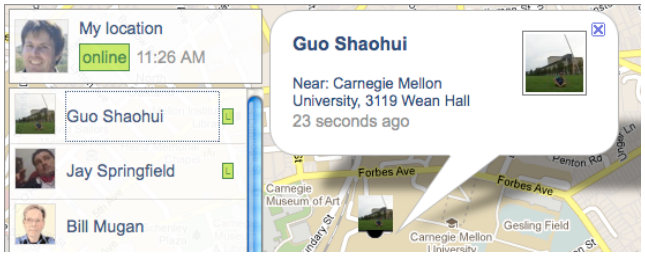

Figure 1: A screenshot of Locaccino.

underlying sharing policy. In order for machine learning to be useful for learning security and privacy policies, the user must always be in control of the underlying policy. This means that the policy suggestions made by the algorithm must be intelligible by the user, and the user must be able to approve or refute any changes the algorithm makes. Additionally, as the user's preferences evolve, she may want to manually change the underlying policy. The algorithm should therefore be able to adapt to this, incorporating new manual policy changes into the underlying model. This suggests an incremental approach to policy learning.

We call such learning algorithms *user controllable policy learning* algorithms, which were first introduced by Kelly *et al.* (2008). In their work, they proposed a local search method for accurately learning security and privacy policies. The policies learned by their algorithm evolved in an incremental way, keeping the user in control of the learning process. However their work relied on simulations to evaluate their methodology. We take on a similar task, however our work is informed by data collected over the course of several real user studies of a policy-based location sharing system.

Although this work focuses on location sharing policies, the problem of learning policies for user preferences extends beyond location sharing. Fang and LeFevre used active learning (2010) to discover content sharing policies for Facebook. There has been work on learning user preferences for recommender systems (Adomavicius and Tuzhilin 2005; Rashid et al. 2002), and for helping users identify a desired item among a large set of items (Viappiani, Faltings, and Pu 2006).

In this paper we introduce a user-controllable Gaussian mixture classifier that incrementally learns location sharing policies by making local changes to the underlying components of the Gaussian mixture. We evaluate our model on real data collected from a location sharing social network, and we show our model performs well even on complex policies. Our user-controllable methods quickly converge to be just as accurate as standard non-incremental learning, but the user-controllable models make far fewer changes to the policy.

Data Collection and Methods

Our results are based on data collected from Locaccino, an application that allows users to share their current location with their friends subject to user-controllable privacy rule specifications. The users' locations are continuously tracked using software running in the background on their smart phones. Locations are typically accurate to within 50 meters, and are reported back to the Locaccino server in regular intervals. Users of Locaccino create privacy rules to control access to their location. Each rule specifies a set of situations in which the user's location will be viewable by a set of friends. Users can specify locations where they want to allow sharing by creating rectangles on a map, and by specifying which of their friends can see them in each rectangle. They can also specify time rules, for example by specifying which friends are allowed to see them between the hours of 10 AM and 5 PM on Wednesdays. Combinations of location rules and time rules are also allowed. For example, a user can specify the her mother can see her location when she is on campus between the hours of 10 AM and 5 PM on Wednesdays.

In this paper we analyze the policies specified by 43 users who have used Locaccino for at least 2 weeks. During this time, the users set up policies according to their sharing preferences for a total of 674 friends, approximately 15 friends per user on average.

Problem Statement and Model

In this section we formally describe the location sharing setting, and we present our user-controllable Guassian mixture model that we use to learn location sharing policies.

Formal description of location sharing

An observation $x = (lat, lon, t)$ of user u is a 3-tuple consisting of a latitude lat, longitude lon, and a timestamp t such that u was observed by the system at location (lat, lon) at time t. We suppose that u's location is tracked over some period of time, and n discrete observations of u are collected. We denote this set of all observations by $\mathcal{O}_u = \{x_1, \ldots, x_n\}$. We further suppose u has m friends $V_u = \{v_1, \ldots, v_m\}$ with which she wants to share her location. For each friend $v \in V$, u specifies a location sharing policy, which is a function $f_{u,v} : \mathcal{O}_u \rightarrow \{0, 1\}$. For $x = (lat, lon, t)$, $f_{u,v}(x) = 1$ indicates that u wishes to grant v permission to see her location at (lat, lon) at time t, and $f_{u,v}(x) = 0$ indicates that u wishes to deny v permission. We treat $f_{u,v}$ as the ground truth sharing preference of u. Based on this ground truth, u specifies a sharing policy in the system, $\widehat{f_{u,v}}$. In practice $\widehat{f_{u,v}}$ will be different from $f_{u,v}$, as $\widehat{f_{u,v}}$ is constrained by the expressiveness of the system policy language and the burden on u to specify

her true preference. The location sharing is instantiated by the system through a request that v makes of u's location. If x is the system observation of u at the time of the request, then the system will share u's location with v if and only if $\widehat{f_{u,v}}(x) = 1$.

Given this formal setting, our goal in this work is to use machine learning to help the user minimize disagreement between $\widehat{f_{u,v}}$ and $f_{u,v}$. Furthermore, we aim to accomplish this in a manner that keeps the user in control of specifying the policy at all times.

The policy space $\mathcal{F}$

We assume the users' ground truth sharing preferences can be arbitrary functions $f_{u,v}$ and we assume that the system policies that they specify are chosen from a fixed family of functions $\widehat{f_{u,v}} \in \mathcal{F}$, which we call the *policy space*. In this work we assume each $\widehat{f_{u,v}} \in \mathcal{F}$ is of the form $\widehat{f_{u,v}} = \phi_1 \vee \phi_2 \vee \cdots \vee \phi_k$, where each ϕ_i is a *rule* specifying a condition for sharing with v. In this way, if any of the rules are satisfied by a request, then the system will permit sharing, and the default condition (*i.e.* if no rules are satisfied) is to deny the request. Each rule ϕ_i is either a *location rule*, a *time rule* or it is a conjunction of a location rule and a time rule, which we call a *location-time rule*. A location rule is a union of rectangular geographic regions. We say a location rule ϕ_i is satisfied by request $x = (lat, lon, t)$ if the location coordinate (lat, lon) falls within any of the specified regions of ϕ_i. A user constructs a time rule by specifying the hours of the day and days of the week where she wants to allow v to have access to her location (i.e. from 9AM-5PM on Mondays and Tuesdays). Let $h(t)$ and $d(t)$ respectively indicate the time of day and day of the week at timestamp t. Then the time rule ϕ_i is satisfied by $x = (lat, lon, t)$ if $h(t)$ and $d(t)$ fall within any of time regions specified by ϕ_i.

There is a convenient geometric interpretation of $\mathcal{F}$. A single rule ϕ_i can be mapped to an axis-aligned cuboid in the three dimensional space $[-90, 90] \times (-180, 180] \times (0, 168]$, where the first two coordinates correspond to degrees latitude and longitude, and the third coordinate corresponds to the time variable with one unit per every hour of every weekday. We may then view each $\widehat{f_{u,v}} \in \mathcal{F}$ as a set of axis-aligned cuboids this space. The rule space $\mathcal{F}$ is then the collection of all such sets of cuboids. A request $x = (lat, lon, t)$ can then be transformed into a point in this space, and a policy function $\widehat{f_{u,v}}$ accepts the request if the corresponding point lies inside one of the policy's cuboids.

Learning policies with Gaussian mixtures

Throughout we use the convention that capital letters refer to random variable, lowercase letters refer to particular values of random variables, and script letters refer to collections of random variables. We assume that for some subset $\mathcal{X} \subset \mathcal{O}_u$, the user has provided a corresponding labeled set $\mathcal{Y}_v = \{f_{u,v}(x) : x \in \mathcal{X}\}$ indicating her sharing preference for v at each observation. Our central learning objective is to find the most likely $\widehat{f_{u,v}} \in \mathcal{F}$ given the training data. In the geometric setting, this corresponds to finding the most likely set of axis-aligned cuboids in $\mathbb{R}^3$ that include data-points where $Y_v = 1$ but exclude data-points where $Y_v = 0$.

A secondary goal of this work is to empirically explore the generative modeling[2] of location sharing rules. Although discriminative models such as decision trees, tend to be more accurate than generative models, there are advantages to generative modeling that are particular appealing to learning location sharing policies. One main advantage of the generative approach relates to the ease of handling unlabeled data. Typically the set of labeled observations $\mathcal{X}$ is a small subset of the entire set of observations $\mathcal{O}_u$. Furthermore, since there is a heavy burden on the user in collecting the labels $\mathcal{Y}_v$, it is essential that any real system attempt to minimize the required user effort. We envision using techniques from semi-supervised learning to minimize this burden, and generative approaches are often much easier to extend to the semi-supervised setting (Ulusoy and Bishop 2005). Although we don't empirically explore semi-supervised techniques in this work, in later sections we describe how our model might be extended to the semi-supervised setting.

In order to learn a generative model for this problem, we relax the target concept slightly and instead assume that data from each label class was generated by a mixture of axis-aligned (diagonal co-variance matrices) Gaussian distributions in $\mathbb{R}^3$. That is for sharing preference $Y_v = i$, we assume $p(X|Y_v = i) = \sum_{k=1}^{K_i} \pi_i^k \mathcal{N}(X|\mu_i^k, \Sigma_i^k)$ for class parameters μ_i, $\mathbf{\Sigma}_i$, and π_i, which are respectively the means, co-variances, and mixing proportions of the K_i Gaussians. If we know the parameters for each class, then for an unlabeled observation x, we can compute

$$p(Y_v = i|x, \mu_i, \mathbf{\Sigma}_i, \pi_i) = \frac{p(Y_v = i|\mu_i, \mathbf{\Sigma}_i, \pi_i) \sum_{k=1}^{K_i} \pi_i^k \mathcal{N}(x|\mu_i^k, \Sigma_i^k)}{p(x, Y_v|\mu_i, \mathbf{\Sigma}_i, \pi_i)}.$$

We then label x with $\arg\max_i \{p(Y_v = i|x, \mu_i, \mathbf{\Sigma}_i, \pi_i)\}$.

Converting between policy functions and GMMs

In order for GMMs to be a viable option for user-controllable learning, there must be a way to translate between user specified location rules, and the machine generated GMM policy suggestions. Here we provide a map-

[2]Generative models maximize the joint likelihood $p(\mathcal{X}, \mathcal{Y}_v)$ the data, whereas discriminative models directly maximize the conditional likelihood $p(\mathcal{Y}_v|\mathcal{X})$.

ping between the axis-aligned rectangular rules of Locaccino and the learned GMMs. Given the parameters of a GMM $\mu_i, \mathbf{\Sigma}_i, \pi_i$, we can construct a rectangular time and space rule by taking taking intervals around the mean in each dimension. For example, given $\mu_1, \mathbf{\Sigma}_1, \pi_1$, we can form a rectangular allow rule for each Gaussian component $k = 1, \ldots, K_1$ by forming intervals

$$I_j = (\mu_1^k[j] - c\Sigma_1^k[j,j], \mu_1^k[j] + c\Sigma_1^k[j,j])$$

where $j \in \{1, 2, 3\}$ indicates the dimension in the policy space (*i.e.* latitude, longitude, or time), and c is a parameter determining the width. A corresponding axis-aligned allow rule is then given by $I_1 \times I_2 \times I_3$. We can similarly form rectangular deny rules from the parameters $\mu_0, \mathbf{\Sigma}_0, \pi_0$. Since Locaccino policies are default deny, the suggested rectangular rules that we would present to the users are the set difference between the allow regions minus the deny regions.

User-controllable learning with GMMs

We now present our method for user-controllable policy learning based on Gaussian mixtures. Our approach iteratively learns the user's location privacy rules in an incremental manner by making local changes to the mixture components in each round. This ensures that the underlying policy recommended by the model evolves in a gentle manner, as we believe this is essential to thus maximizing the understandability of the suggestions the algorithm makes.

We assume that the learning proceeds for N rounds. With each round j, the algorithm receives a new set of observations $\mathcal{X}_j$ and corresponding set of labels Y_j. Our goal is to learn for each round j, a Gaussian mixture classifier $M_j = (\theta_{0,j}, \theta_{1,j})$ from the data $(\mathcal{X}_1, \mathcal{Y}_1), \ldots, (\mathcal{X}_j, \mathcal{Y}_j)$ where $\theta_{0,j} = (\mu_{0,j}, \mathbf{\Sigma}_{0,j}, \pi_{0,j})$, and $\theta_{1,j} = (\mu_{1,j}, \mathbf{\Sigma}_{1,j}, \pi_{1,j})$ are the mixture models for deny and allow, respectively. To maximize understandability, we restrict M_j so that it is formed through incremental and atomic changes to M_{j-1}.

Let $M_j^* = (\theta_{0,j}^*, \theta_{1,j}^*)$ be the *unrestricted* maximum likelihood estimate of the model given the data $(\mathcal{X}_1, \mathcal{Y}_1), \ldots, (\mathcal{X}_j, \mathcal{Y}_j)$. Note that M_j^* could differ drastically from M_{j-1}. This is because as new data arrives, directly fitting a new model to the data could cause large, and from the users perspective, unintelligible shifts in the underlying GMM, even if we initialize the learning of the new model from the current model (for instance via EM). In order to maximize the intelligibility of the updates, we instead form the new model M_j by performing local operations on the components of M_{j-1} that move it incrementally closer to M_j^*. This will ensure that we still increase the likelihood with each step, yet we do so only through small atomic changes to M_{j-1}, ensuring maximum usability.

We allow for three three local operations on the parameter space of the GMMs:

- $\theta_{i,j} = delete(\theta_{i,j-1}, k)$ then $\theta_{i,j}$ is formed by removing the kth Gaussian from the mixture in $\theta_{i,j-1}$
- $\theta_{i,j} = add(\theta_{i,j-1}, \theta_{i,j}^*, k^*)$ the $\theta_{i,j}$ is formed by adding the k^*th Gaussian in $\theta_{i,j}^*$ to the mixture in $\theta_{i,j-1}$
- $\theta_{i,j} = swap(\theta_{i,j-1}, \theta_{i,j}^*, k, k^*)$ then $\theta_{i,j}$ is formed by swapping the kth Gaussian from $\theta_{i,j-1}$ with the k^*th Gaussian from $\theta_{i,j}^*$

In all cases after performing the operation we renormalize the mixture parameters $\pi_{i,j}$ accordingly. We allow only one such local operation per algorithm round. Roughly, add is equivalent to adding a location rule, $delete$ is equivalent to $deleting$ a location rule, and $swap$ is equivalent to modifying a location rule (for instance by extending the boundary of the rule).

We now introduce some notation. Let K_i and K_i^* be the number of components of $\theta_{i,j-1}$ and $\theta_{i,j}^*$ respectively. Let k_i and k_i^* refer to an arbitrary component of each mixture. Let $D[k_i, k_i^*]$ denote the symmetrized KL divergences between the k_ith Gaussian in $\theta_{i,j-1}$ and the k_i^*th Gaussian in $\theta_{i,j}^*$. We view the $(K_i)2(K_i^*)$ entries of D as weights to a bipartite graph, with the components of each Gaussian as nodes. We let m be the solution to the minimum cost bipartite matching on this graph, which we solve using the Hungarian Method. Then $m[k_i]$ gives the Gaussian in $\theta_{i,j}^*$ that is assigned to the k_ith Gaussian in $\theta_{i,j-1}$, and the cost of this assignment is $D[k_i, m[k_i]]$.

The central idea here is that in each round, independently of $\theta_{i,j-1}^*$ and $\theta_{i,j-1}$, we learn a new GMM $\theta_{i,j}^*$ given $(\mathcal{X}_1, \mathcal{Y}_1), \ldots, (\mathcal{X}_j, \mathcal{Y}_j)$. To make sure that the model we actually suggest to the user $\theta_{i,j}$, does not differ drastically from the model in the previous round $\theta_{i,j-1}$, we form $\theta_{i,j}$ by incrementally moving $\theta_{i,j-1}$ towards $\theta_{i,j}^*$ given the above operations. In general, there will be components of $\theta_{i,j}^*$ that are similar to $\theta_{i,j-1}$, since there is a great deal of overlap in the data that each model was trained on. We use the matching m in the KL-divergence matrix to guide the local operations we make to $\theta_{i,j-1}$. The matching m provides us with a correspondence between components of $\theta_{i,j-1}$ and components of $\theta_{i,j}^*$. Whenever we perform swap operations, we will swap only along edges in the matching. This ensures that the swap operations are roughly equivalent to modifying a single rule in $\theta_{i,j-1}$.

Now we introduce four algorithms for applying local operations to construct M_j and we compare them to two baseline algorithms.

MaxMatchedKL If $K_i^* > K_i$, then $\theta_{i,j}^{temp}$ is formed by adding a Gaussian to $\theta_{i,j-1}$. For each unmatched Gaussian k_i^* in $\theta_{i,j}^*$ we compute $\max_{k_i} D[k_i, k_i^*]$, and we add the k_i^* with largest maximum. If $K_i^* < K_i$ then $\theta_{i,j}^{temp}$ is formed by deleting a Gaussian from $\theta_{i,j}^*$. We choose the k_i to delete in an identical way, except now we maximize over unmatched k_i in $\theta_{i,j-1}$. If $K_i^* = K_i$, then we set $\theta_{i,j}^{temp} \leftarrow swap(\theta_{i,j-1}, \theta_{i,j}^*, k, m[k])$ where $k =$

$\arg\max_{k'} D[k, m[k]]$. For each i, we set each $\theta_{i,j} \leftarrow \theta^{temp}_{i,j}$ only if the likelihood of $\theta^{temp}_{i,j}$ is greater than that of $\theta_{i,j-1}$ given the data. Otherwise we set $\theta_{i,j} \leftarrow \theta_{i,j-1}$. We then return $M_j = (\theta_{0,j}, \theta_{1,j})$.

RandomMatchedKL RandomMatchedKL is a randomized version of MaxMatchedKL. We sample the Gaussian to add, delete or swap proportional to the given objective in each case. In the case of a swap, we would sample a k_i with probability proportional to $D[k_i, m[k_i]]$.

MaxMatchedWeightedKL In this approach we choose to *add*, *delete*, or *swap* randomly. We first flip a coin with probability $\max\{\frac{K^*_i - K_i}{K^*_i}, 0\}$ to decide whether or not to *add*. If we do not *add*, we then flip a coin with probability $\max\{\frac{K_i - K^*_i}{K_i}, 0\}$ to decide to *delete*. If we do not delete, then we *swap*. If we choose to *add*, then we select $k^*_i = \arg\max\{\pi^*_i[k^*_i] \times \min_{k_i} D[k_i, k^*_i]\}$, that is we weight the objective function by how important the k^*_ith component is in the mixture. Similarly, if we choose to delete, we select $k_i = \arg\max\{\pi_i[k_i] \times \min_{k^*_i} D[k_i, k^*_i]\}$. If we choose to *swap*, then we select $k = \arg\max \pi_i[k_i] \times \pi^*_i[m[k_i]] \times D[k_i, m[k_i]]$, and $k^*_i = m[k_i]$. After this procedure we get new Gaussians $\theta^{temp}_{0,j}$ and $\theta^{temp}_{1,j}$. As above, we only update $\theta_{i,j}$ to $\theta^{temp}_{i,j}$ if it results in an increase in the likelihood.

RandomMatchedWeightedKL RandomMatchedWeightedKL mirrors MaxMatchedWeightedKL but selects k_i and k^*_i randomly proportional to the objective, instead of deterministically selecting the maximum.

Baseline: Random This baseline is to test if the heuristics used in the four models actually perform well at making incremental changes. In this model we simply choose local operations completely uniformly at random. As above, we only update $\theta_{i,j}$ if it results in an increase in the likelihood.

Basline: NewEachRound This baseline is to test the accuracy of the incremental models against a model that picks the best Gaussian at each round. We simply set $M_j \leftarrow M^*_j$. The accuracy of this model should be an upper bound on the accuracy of the experimental models we test, since it does not have the added restriction requiring incremental changes. The goal of this work is to show that the accuracy of the experimental models converges to that of NewEachRound.

Parameter Estimation

We use the Expectation Maximization (EM) algorithm to approximate the unrestricted maximum likelihood estimate for $M^*_j = (\theta^*_{0,j}, \theta^*_{1,j})$. We use EM once per each i, fitting the model $\theta^*_{i,j}$ to the data from class i. We use the mclust implementation in R (Fraley and Raftery 2010), which uses a round of hierarchical clustering to initialize the EM algorithm, and it uses BIC to determine the number of clusters.

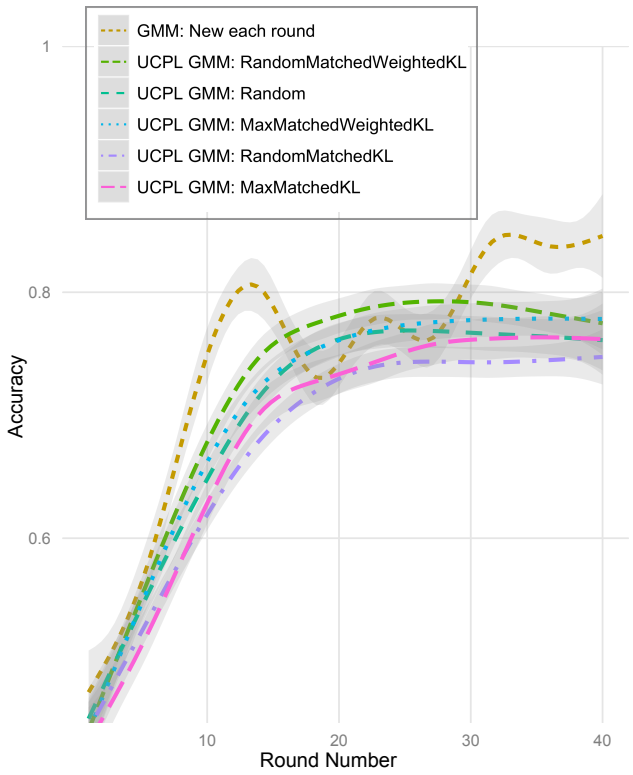

Figure 2: User controllable learning of 124 location sharing policies. The curves are non-parametric regressions of the experimental accuracy per each round of evaluation. Tests were performed on the 8 users with the most complex policies, who cumulatively specified policies for 124 friends.

Training, Evaluation and Results

To test our models, we designed an experiment that mirrors a real-world policy suggestion scenario. We selected 43 Loccaccino users from whom we had collected at least 2000 location observations (roughly at least two weeks worth of data per person) and who had created at least 1 location sharing rule. These users specified sharing policies for a total of 674 friends, roughly 6 friends on average per person. We then sampled 2000 points (uniformly, without replacement) from the set of observations of the user to form the set O_u, and we evaluated the users sharing policy for each friend v at every $o \in O_u$. We use the results of these policy evaluations as the label for training and testing of our methods.

First, we separate the data into a training set and a testing set. We choose this partition to respect the temporal ordering of the observations, that is we're always testing on *future* observations with respect to the training data. This is an important distinction, since any classifier will necessarily pay a price associated with the randomness of a user's schedule. We use the first 60% of observations of each user as training, and the last 40% as testing. To emulate the loca-

Model	Mean KL-Div	Std Dev
GMM: NewEachRound	0.90	1.56
UC-GMM: Random	1.66	2.91
UC-GMM: RandomMatchedWeightedKL	0.48	1.39
UC-GMM: MaxMatchedWeightedKL	0.54	1.38
UC-GMM: RandomMatchedKL	0.49	1.31
UC-GMM: MaxMatchedKL	0.41	1.15

Table 1: Mean change in model per round as measured by the symmetrized KL-divergence. Means are taken over non-zero KL-divergence values between each model in round i and the updated model in round $i+1$ for all rounds i. Differences in mean are statistically significant between the group of models in the upper section of the table and the lower section of the table (t-tests with p-values < 0.001).

tion policy suggestion scenario, we evaluate the models in rounds. With each round i, the models see some new labeled data X_i chosen from the training set. Our choice of X_i is sequential, that is in each round, we see the *next* set of observations from the user. In round i each model will build a classifier C_i given the training data $X_1, \ldots, X_i$. In the case of the user-controllable classifiers, C_i will also depend on C_{i-1}. The accuracy of the classifiers is estimated by testing them at each round on the test data (note that only the training data changes with each round, the test data is a fixed set for every u and v). In our experiments each X_i was chosen to be the next 75 observations from the training set, and we iterated this procedure for 30 rounds. We also measure KL-divergence for each model between the underlying distributions in C_{i-1} and C_i. If a model has a small KL-divergence in the update from C_{i-1} to C_i, this indicates the changes made to the classifier at that round are more incremental.

We evaluate the effectiveness of our user-controllable models on the 124 policies (from the full set of 674) that were more complex than always allow or always deny. These policies are taken from a total of 8 users. Accuracies for each round are shown in Figure 2, and the average KL-divergences for each round are shown in Table 1. Observe that all of the user controllable models except for MaxMatchedKL are converging to the accuracy rate of the non user-controllable model NewEachRound. At the end of 30 rounds, the average accuracy of the non-user controllable baseline NewEachRound was 0.87. The most accurate user-controllable model after 30 rounds was RandomMatchedWeightedKL, which had average accuracy 0.85. The least accurate user-controllable model was MaxMatchedKL, which had average accuracy 0.76. Additionally, observe that, with the exception of the baseline Random, the updates made by the user-controllable models are less significant that those made by NewEachRound, suggesting that the changes are indeed more incremental than those made by simply learning a new model in each round.

Extensions to Semi-supervised Learning

In deploying machine learning systems that require user feedback, it is essential for the usability of the system to minimize the amount of feedback required, otherwise the user might get frustrated and stop using the system. This concern motivated our choice of Gaussian Mixture Models for learning location privacy policies, as there are natural extensions of GMM that can help with this burden.

In semi-supervised learning, we try to leverage the structure of unlabeled data to improve the model performance, thereby decreasing the number of labels required to learn an accurate model. Formally, we assume there is a labeled set of obervations $\mathcal{X}^L \subset \mathcal{O}_u$ with corresponding labels $\mathcal{Y}_v^L = \left\{ f_{u,v}(x) : x \in \mathcal{X}^L \right\}$. In semi-supervised learning, we also make use of unlabeled observations $\mathcal{X}^U = \mathcal{O}_u \backslash \mathcal{X}^L$. We call $(\mathcal{X}^L, \mathcal{Y}_v^L)$ the labeled data and $\mathcal{X}^U$ the unlabeled data. In practice, the number of unlabeled data points $|\mathcal{X}^U|$ is often much greater than the number of labeled data points $|\mathcal{X}^L|$. The underlying principal behind semi-supervised learning is that there is often informative structure to the unlabeled data. In the case of user mobility traces, observations are typically clustered around important locations (i.e. home and work), and often exhibit strong temporal regularities. Within these clusters, the sharing preferences are often very predictable.

The advantage of a generative approach to modeling location privacy rules is that it only requires slight modifications to our learning methods to take this unlabeled data into account. We can learn a model that takes this unlabeled data into account by finding parameters that maximize the joint likelihood over both labelled and unlabeled data, $p(\mathcal{X}^U, \mathcal{X}^L, \mathcal{Y}_v^L | \theta)$. This is typically done with the Expectation Maximization algorithm.

Discussion

Our chief motivation for using Gaussian mixtures in this task was the great amount of flexibility the model provides for reasoning about the user's sharing policy. For instance, if the model discovers an "allow" Gaussian component that has a much larger (relative) variance along the time dimension than the latitude and longitude dimensions, this would most likely correspond to a location rule, since it would grant access to a specific location over a wide variety of times. Similarly, if the Gaussian has much greater relative variances along the latitude and longitude dimensions than the time dimension, this would most likely correspond to a time rule, since it grants access to a specific time range over a wide variety of locations. Additionally, since Gaussian mixture models are generative, they provide a model of the joint distribution over $\mathbf{X}$ and Y, allowing us to probabilistically reason about both the target rule and the set of observations of the user.

This added richness over discriminative classifiers gives us

the ability to simultaneously make statements about important times and locations in the person's weekly routine, as well as statements about the most likely target policy. For example, the Gaussian component with highest mixture probability most likely corresponds to the user's home. Finally, by applying some very natural heuristics to the learned distributions, we are able to translate back and forth between the actual policy specified by the use, and the underlying Gaussian mixtures. For instance, we can project a window of, say two standard deviations of each "allow" Gaussian on to the time axis to find windows of time where the user wants to share. Similarly, we can project the Gaussians onto the latitude and longitude to find rectangles where the user wants to share. In this way we can directly build Locaccino rules from our model.

Our method opens up many possibilities for future work. Currently, the model focuses on places that the users have gone, and makes no attempt to infer or incorporate places the users will go or might go. Additionally, our method of getting feedback from users was non-adaptive. One can imagine adaptive, active-learning approaches, such as that used by (Fang and LeFevre 2010), that could select the points of feedback from the users that are most useful to the learning algorithm. Our work also suggests a follow-on user study to directly measure the user's professed understandability of the policy as it changes over time.

Conclusion

We present a method for incrementally learning policies for location sharing that uses a Gaussian mixture model to represent the user's sharing preferences. As our model processes new data, we restrict the evolution of the model so that only incremental changes are made, thus reducing the amount of change per round needed to converge to an accurate policy. Using real user data, we show that our method was able to incrementally learn policies for location-sharing that were not significantly less accurate than those learned without the incremental restriction. Our next step in this work is to explicitly evaluate the usability of the polices that are created by our methods.

Acknowledgments

This work has been supported by NSF grants CNS-0627513, CNS-0905562, and CNS-1012763, and by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389 and W911NF-09-1-0273 from the Army Research Office. Additional support has been provided by Google, Nokia, France Telecom, and the CMU/Portugal ICTI. The authors would also like to thank the many members of the Locaccino team for their support.

References

Adomavicius, G., and Tuzhilin, A. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* 734–749.

Benisch, M.; Kelley, P.; Sadeh, N.; and Cranor, L. 2011. Capturing Location-Privacy Preferences: Quantifying Accuracy and User-Burden Tradeoffs. *Journal of Personal and Ubiquitous Computing*.

Beresford, A., and Stajano, F. 2005. Location privacy in pervasive computing. *Pervasive Computing, IEEE* 2(1):46–55.

Fang, L., and LeFevre, K. 2010. Privacy wizards for social networking sites. In *Proceedings of the 19th international conference on world wide web*, 351–360. ACM.

Fraley, C., and Raftery, A. E. 2010. MCLUST Version 3 for R: Normal Mixture Modeling and Model-Based Clustering. Technical Report 504, Department of Statistics, University of Washington.

Gedik, B., and Liu, L. 2007. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing* 1–18.

Kelley, P.; Hankes Drielsma, P.; Sadeh, N.; and Cranor, L. 2008. User-controllable learning of security and privacy policies. In *Proceedings of the 1st ACM workshop on Workshop on AISec*, 11–18. ACM.

Maxion, R., and Reeder, R. 2005. Improving user-interface dependability through mitigation of human error. *International Journal of Human-Computer Studies* 63(1-2):25–50.

Rashid, A.; Albert, I.; Cosley, D.; Lam, S.; McNee, S.; Konstan, J.; and Riedl, J. 2002. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*, 127–134. ACM.

Sadeh, N.; Hong, J.; Cranor, L.; Fette, I.; Kelley, P.; Prabaker, M.; and Rao, J. 2009. Understanding and capturing peoples privacy policies in a mobile social networking application. *Personal and Ubiquitous Computing* 13(6):401–412.

Tsai, J.; Kelley, P.; Cranor, L.; and Sadeh, N. 2010. Location-sharing technologies: Privacy risks and controls. *ISJLP* 6(2):119–151.

Ulusoy, I., and Bishop, C. M. 2005. Generative versus discriminative methods for object recognition. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, 258–265. Washington, DC, USA: IEEE Computer Society.

Viappiani, P.; Faltings, B.; and Pu, P. 2006. Preference-based search using example-critiquing with suggestions. *Journal of Artificial Intelligence Research* 27(1):465–503.

Zheng, Y., and Xie, X. 2011. Learning travel recommendations from user-generated GPS traces. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2(1):2.