Real-time Prediction of Length of Stay Using Passive Wi-Fi Sensing

Truc Viet Le
School of Information Systems
Singapore Management University
trucviet.le.2012@phdis.smu.edu.sg

Baoyang Song
Department of Applied Mathematics
École Polytechnique
baoyang.song@polytechnique.edu

Laura Wynter
IBM Research
Singapore
lwynter@sg.ibm.com

***Abstract*—The proliferation of wireless technologies in today's everyday life is one of the key drivers of the Internet of Things (IoT). In addition to being an enabler of connectivity, the vast penetration of wireless devices today gives rise to a secondary functionality as a means of tracking and localization of the devices themselves. Indeed, in order to discover and automatically connect to known Wi-Fi networks, mobile devices have to scan and broadcast the so-called *probe requests* on all available channels, which can be captured and analyzed in a non-intrusive manner. Thus, one of the key applications of this feature is the ability to track and analyze human behaviors in real-time directly from the patterns observed from their Wi-Fi-enabled devices. In this paper, we develop such a system to obtain these Wi-Fi signatures in a *completely passive* manner and use the Wi-Fi features it captures within a set of adaptive machine learning techniques to predict in real-time the expected length of stay (LOS) of the device owners at a specific location.**

I. Introduction

The widespread use of Wi-Fi hotspots has become the norm in many daily settings as the number of Wi-Fi-enabled mobile devices (e.g., smartphones, headphones, cameras, etc.) in the consumer market grows exponentially. In order to discover and automatically connect to known Wi-Fi networks, a mobile device has to scan periodically over the Wi-Fi bands by broadcasting the so-called *probe requests* on all available channels [1], [2], [3], [4]. After having successfully associated with a Wi-Fi access point (AP), a mobile device typically continues to send probe requests (even though with less frequency), particularly when the connection is unstable or when the user roams between different APs [1], [4]. Probe requests, which is specified by the IEEE 802.11 protocol [5], carry valuable information about the connectivity such as the device's unique MAC address and the signal strength, among others. It is also noteworthy that a device's probe requests are universally accessible. That is, while only the administrator of the APs can query the system log, anyone can access probe requests sent by mobile devices using an off-the-shelf Wi-Fi sniffer such as `Wireshark` [6], [7], [8]. Moreover, even if no APs are present, probe requests are still being sent by mobile devices and can still be observed. Thus, accessing probe requests can be done in a *completely non-intrusive* manner that requires no additional software installation on the client device.

Thanks to these advantages, real-time Wi-Fi analytics using probe requests have gained much interest among researchers in recent years. Handte *et al.* [4] designed a system to estimate the number of passengers in public transportation vehicles. Musa *et al.* [1] described how to exploit probe requests to infer vehicle trajectories using a hidden Markov model (HMM). Bonné *et al.* [7] built a system on top of a Raspberry Pi to track the user's location at mass events using probe, association and re-association requests. Wang *et al.* [9] studied queue time measurement using a single-point Wi-Fi monitoring approach and designed queue measurement techniques adaptive to different periods of time based on Bayesian networks.

In many retail settings (e.g., coffee shops, fast food restaurants, stores, etc.), it is often desirable to have real-time knowledge of customers' flow and mobility patterns such that goods and services at different locations can be adjusted accordingly to optimize sales. At the same time, numerous research into human behaviors have shown that behaviors are far from being random and, in many cases, are highly predictable using machine learning techniques [10], [2], [11], [9], [8], [12], [13]. To this end, Manweiler *et al.* [11] proposed a system (called "ToGo") that predicts a user's length of stay (LOS) at a specific location. Their system is intrusive because it requires additional software installation on each client device in order to send out the necessary features through Wi-Fi and have their LOS classified and predicted. The system made use of a static support vector machine (SVM) classifier in order to predict the LOS classes of each participating user.

In this work, we are interested in developing a non-intrusive and purely passive system to obtain the relevant Wi-Fi mobility information of customers in retail settings. We design and implement a system for *passive* Wi-Fi sensing and *real-time* prediction of LOS using adaptive online machine learning techniques. Specifically, we wish to detect a given customer's behavior without requiring them to install and run any particular application on their device, and then predict *how long* they will stay in the vicinity. We also wish to do that *as soon as possible* before the person is known to have left.

We define the length of stay (LOS) (also called "dwell time") of a mobile device as the *duration of time* it stays *active* at a specific location. "Active" means that the device can be passively detected via Wi-Fi. Suppose LOS can be categorized into discrete class labels (e.g., passer-by, short stay, medium stay, long stay, etc.) Our goal is to develop a non-intrusive system that detects Wi-Fi information from active devices in the vicinity (of a specific location) and predicts in real-time

TABLE I: The retained data fields of each received data frame.

Field	Description
`timestamp`	Date and time of the receipt of the frame
`MAC_addr`	Unique MAC address of the mobile device
`power_mgt`	Power management state (awake/sleep) of the device
`type`	Either 1 (management), 2 (control) or 3 (data)
`subtype`	Additional discrimination between frames
`seq_ctrl`	Counter that identifies message order and eliminates duplicates
`RSSI`	Received Signal Strength Indicator indicating the signal strength
`channel`	Indicates the channel (e.g., ranging 1–14 for 2.4 GHz band)
`data_rate`	Speed of data transmission
`SSID`	Identifier of the AP

the true LOS class of each as soon as possible.

Our main contributions can be summarized as follows:

- We design and develop a completely passive system that requires no additional software installation on the client device for Wi-Fi sensing and feature generation;
- The proposed system uses the generated features to efficiently train an online classifier and predict in real-time the LOS class of each active device;
- We evaluate our system at two different locations and demonstrate its efficacy against a baseline classifier proposed in [11].

II. System Design & Implementation

A. System Architecture

Fig. 1 illustrates the overall architecture of our passive Wi-Fi sensing and LOS prediction system. One (or multiple) passive sniffer(s) collects Wi-Fi data of the active devices in real-time and cache them into a `Redis` server. A cron job is then created to clean, normalize and dump the data from the `Redis` server into a `MySQL` database. After which, a classification and prediction module reads from the `MySQL` database, learns from the examples of inactive devices (i.e., those whose data frames are no longer detected) and predicts the LOS of those still active devices. Finally, a web application interacts with the server as the front end to provide real-time visualization of the number of active mobile devices and their predicted LOS. We elaborate on each of the components as follows.

Wi-Fi Sniffer. The Wi-Fi sniffer captures both probe requests and other data frames (e.g., null, probe responses, etc.) However, unlike probe requests, most other data frames are channel-specific, which means that the sniffer has to listen to the right channel in order to capture them. On the other hand, these data frames are typically sent out by the mobile device after having been successfully associated with an access point (AP). Thus, it is necessary to loop over all the available channels on which the associated AP works in order to capture these data. We call this *channel switching* on the AP.

After parsing the packets, only the data fields listed in Table I are retained for each received data frame. The received data frames are then piped asynchronously to the `Redis` server. Asynchronization is essential because a sniffer typically receives thousands of data frames per second.

Backend. The backend module consist of two sub-modules: a `Redis` server and a `MySQL` database. In the `Redis` server, the following two key-value pairs are used:

- `active_users`: A sorted set of MAC addresses associated with an expired timestamp as the value. A mobile device is *active* when the maximum duration between any two consecutive data frames it sends out is no longer than 10 minutes; otherwise, it is *inactive*, and its last data frame is said to be *expired*. When estimating the number of active mobile devices and their LOS, only those MAC addresses with non-expired values are considered;
- `records`: A list where each element is a serialization of a `JSON` object consisting of the fields mentioned in Table I. A Python script is then used to normalize the fields and dump the list into the `MySQL` database.

Because we are only interested in mobile devices (e.g., smartphones), only those MAC addresses of known mobile phone manufacturers are retained by looking up the latest IEEE OUI table [14]. Those that are not likely from mobile devices (e.g., from laptops, printers, etc.) are discarded.

Web Server. The web server is implemented in `Flask` (a Python web framework) to show the number of active mobile devices and their predicted LOS in real-time. Fig. 2 shows the screenshot of the front-end web application.

B. Classification & Prediction Module

The prediction module is implemented in Python using the `Scikit-learn` machine learning library [15]. In this application, instead of predicting the real LOS (in minutes), we classify it into a finite number of class labels (i.e., categories of LOS). This is more useful in most practical use cases in retail settings [11]. How to do the actual classification is specific to the use case and is not discussed here. We, however, give a simple example of how such a classification can be done in Section III-B. In this module, we implement an online classifier for real-time classification of LOS. Online classifier works best because its parameters can be adaptively updated in real-time in light of new stream of learning examples (i.e., ground truths). A new ground truth emerges when a device becomes inactive and has its true LOS recorded.

We propose to implement the linear SVM classifier (i.e., SVM with linear kernel) with stochastic gradient descent (SGD) training due to its ease of implementation and proven efficiency in large-scale online learning [16]. SGD is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions. Each parameter update (i.e., feature weights $\mathbf{w}$ of SVM) of SGD is given by:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta\Big(\alpha\frac{\partial R(\mathbf{w})}{\partial \mathbf{w}} + \frac{\partial L(\mathbf{w}^\top \mathbf{x}_i + b, y_i)}{\partial \mathbf{w}}\Big), \qquad (1)$$

where η is the step size (i.e., the learning rate), $\alpha > 0$ is a non-negative hyperparameter, $R(\mathbf{w})$ is the regularization term, $L(\cdot)$ is a loss function, b is the intercept, and $(\mathbf{x}_i, y_i)$ is a new ground truth. The interested reader may refer to [16] for detailed implementation of SGD.

At each time step t, the module queries the `MySQL` database and generates a list of time-dependent features $\mathbf{x}(t)$ and LOS class labels $\mathbf{y}$. Table II summarizes the feature vector $\mathbf{x}(t)$ of each training example at time step t. Note that 'RSSI' and

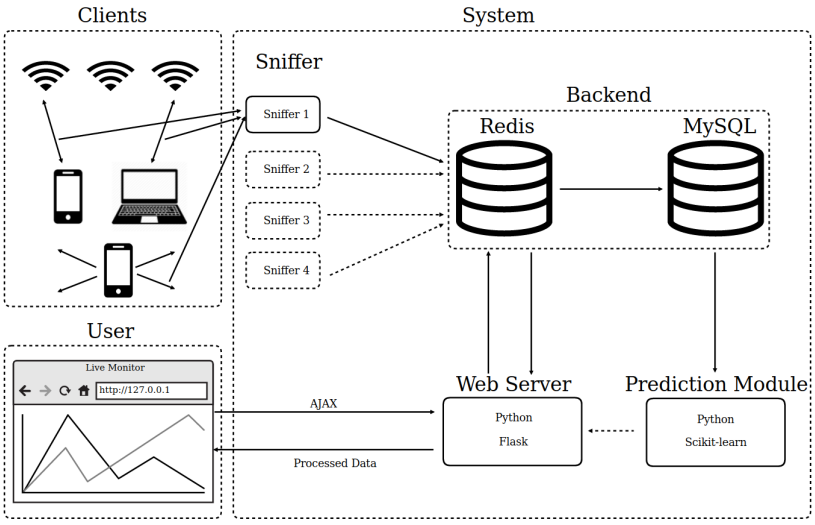

Fig. 1: Overall architecture of our passive sensing and prediction system of length of stay (LOS) using real-time Wi-Fi data.

Fig. 2: Screenshot of the front-end web application monitoring the number of active mobile devices in real-time.

'Data rate' feature each has three values: cumulative mean, standard deviation ('stddev') and histogram.

Given a mobile device, let $\{r_i\}_{i=1}^n$ be its time series of RSSI values. The cumulative mean and standard deviation of $\{r_i\}_{i=1}^n$ are defined as: $\{\mu_i^r\}_{i=1}^n = \left\{\frac{\sum_{j=1}^i r_i}{j}\right\}_{i=1}^n$ and $\{\sigma_i^r\}_{i=1}^n = \left\{\frac{\sum_{j=1}^i r_i^2}{j} - (\mu_i^r)^2\right\}_{i=1}^n$, respectively. The histogram of RSSI is a time series of vectors of the form $\{h_i^r\}_{i=1}^n = \left\{\left(\sum_{j=1}^i 1_{r_j\in(-100,-80)}, \ldots, \sum_{j=1}^i 1_{r_j\in(-20,0)}\right)\right\}$. The instantaneous gradient of $\{r_i\}_{i=1}^n$ is the *backward difference* defined by $\{r_i'\}_{i=2}^n = \left\{\frac{r_i - r_{i-1}}{\Delta t_i}\right\}_{i=2}^n$, where Δt_i is the time difference between the i-th and $i-1$-th received data frame. The cumulative mean, standard deviation, histogram and instantaneous gradient of data rate are similarly defined.

The learning features in Table II are derived from those in Table I by using the *feature importance* functionality of random forests (with LOS classes being the response) [15].

TABLE II: Cumulative feature vector $\mathbf{x}(t)$ of each data frame collected from a mobile device at each time step t.

Feature	Description
`begin_hours`	Integer hour of the day when the device was first detected
RSSI	Cumulative mean, stdev and histogram of RSSI
Data rate	Cumulative mean, stdev and histogram of data rate
`time_spent`	Current LOS (so far) of this device (in minutes)
`num_device`	Current number of *other* devices detected at the location
`rssi_grad`	Instantaneous gradient of RSSI
`data_rate_grad`	Instantaneous gradient of data rate

III. Experiments

A. Data Collection

We deploy our system to passively collect Wi-Fi data at two different office locations in Singapore, which we call Location 1 and Location 2. Because the offices are open space and dynamic environments, where workers and clients keep coming in and out during the day, our test environments resemble and are comparable to realistic retail settings. Moreover, the dynamics at the two locations are also different. Location 1 has more homogeneous population of office workers characterized by longer stays and more typical "office-hour" patterns. The dynamic at Location 2 is more agile with frequent client visits (i.e., short-term stays) and mobile working environment for the employees. Thus, the two datasets represent two distinct customer behavioral and mobility patterns.

Four (4) and seven (7) days of data are collected at Location 1 and Location 2, respectively. (The days of collection are not necessarily continuous.) For each day, we collect Wi-Fi data frames from all mobile devices detected in the office continuously for 9 hours, from 9 a.m. to 6 p.m. Fig. 3 shows the daily number of records (i.e., data frames) collected at each location. On average, there are approximately 800 and 400 unique mobile devices observed per day at Location 1 and Location 2, respectively. Note that on June 20, 2016 at Location 2, we performed channel switching on the APs. We switched over all 14 channels that resulted in a significant

number of packet losses. Starting from July 21, 2016, we only switched over 3 channels of $1, 6$ and 11 where most of the APs work on. Fig. 3 shows that, for both locations, very large numbers of records were collected per day and every moment (i.e., thousands of records per second).

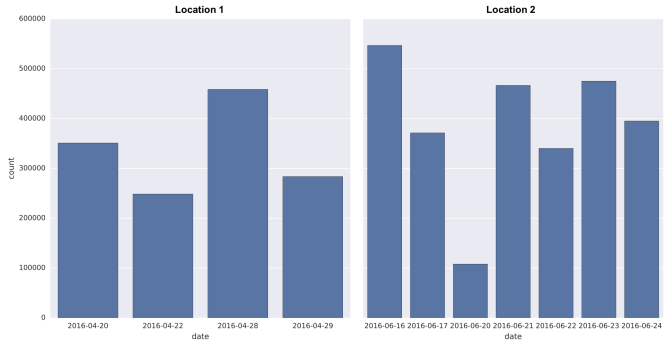

Fig. 3: Daily number of records (data frames) sniffed at two office locations. Dates are in 'year-month-date' format.

B. Experimental Design

For each day, we divide the 9-hour timeline into $\lceil \frac{9}{\Delta} \rceil$ intervals of length (at most) Δ hours each, where Δ is varied from 1 to 6 hours. (The last interval is $9 - \lfloor \frac{9}{\Delta} \rfloor \times \Delta$ hours.) The first interval is always used for training, then the second interval is used for testing, and then after that we alternate between training and testing until the last interval. The last interval is *always* used for testing no matter what. That is, if the second-to-last interval is test, the last one is still test.

For each training interval, we artificially defined 5 classes of LOS ranging from 1 to 5, where 1 corresponds to the shortest stay and 5 the longest. We define the boundaries of each class by evenly dividing the training interval into 5 sub-intervals. That is, if the training interval is of length 1 hour, then LOS class 1 is $(0, 12]$ minutes, class 2 is $(12, 24]$ minutes, class 3 is $(24, 36]$ minutes, class 4 is $(36, 48]$ minutes, and class 5 is $(48, \infty)$ minutes. Hence, the LOS class of each mobile device (detected during the training interval) is determined by which sub-interval its *last* data frame (i.e., final LOS) falls on. The last data frame of a mobile device is received when it has not been detected at the location for at least 10 minutes since the last receipt. We say the device is absent from (or has left) the location. Fig. 4 illustrates the design of our experiments.

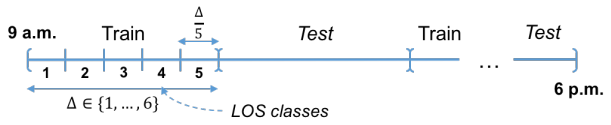

Fig. 4: Design of our experiments on each day at each location.

It is important to note that, for each training interval, only those devices that are present and and later become absent in the *same* interval are taken as ground truths for learning, i.e., those who have their true LOS observed. Those that later become absent in other subsequent intervals are discarded from learning. For each test interval, all active mobile devices that become absent in the interval are included for testing, whether or not they have been present in the intervals before.

For each training example, we collect the cumulative data frame features $\mathbf{x}(t)$ of the device (as shown in Table II) every 15 seconds until its last data frame. Corresponding to each feature vector $\mathbf{x}(t)$ is the true LOS class of the device as observed in the interval using the defined class boundaries.

For testing, for each present mobile device i, we make a prediction of its final LOS class using the observed data frame features $\mathbf{x}_i(t)$ every 15 seconds during the test interval (via a trained classifier). At time t, we call such prediction $\hat{C}_i(t)^{pred}$. We also have the current class label of the device based on its current LOS and the defined class boundaries (as illustrated in Fig. 4). We call this $C_i(t)^{current}$. Thus, the *adjusted* predicted class of LOS at time t is given by:

$$C_i(t)^{pred} = \max\{\hat{C}_i(t)^{pred}, C_i(t)^{current}\}. \quad (2)$$

Hence, if the last class label (i.e., class 5) is present in the training set, then it would always be correctly "predicted" in the test set once the current LOS of the device crosses over to the lower boundary of the last class (since it would be the only class left considering the device's current LOS).

C. Evaluation

To evaluate the LOS prediction accuracy, we use the `mean_misprediction` metric defined as follows. For each device i, suppose that i's final true class of LOS is C_i^{true}. At any time t, our adjusted prediction of i's true class is $C_i(t)^{pred}$. The instantaneous misclassification error for i can be expressed as: $D_i(t) = |C_i^{true} - C_i(t)^{pred}|$. We evaluate the mean misclassification rate across $N(t)$ active mobile devices present at time t as:

$$\texttt{mean\_misprediction}(t) = \frac{\sum_{i=1}^{N} |D_i(t)|}{N(t)}. \quad (3)$$

We can then further average the `mean_misprediction` error rate across all 15-second time steps of all test intervals for a given day and across all the days at the given location. The following classifiers are evaluated in our experiments:

- **SVM.** The static linear SVM classifier that is trained from the examples in the *immediate* previous training interval. This is the classifier implemented in the framework proposed by Manweiler *et al.* [11].
- **Stochastic Gradient Descent (SGD).** The online linear SVM classifier that is trained from the examples in the immediate previous training interval and is continuously updated in real-time during testing as new ground truths are learned. Refer to Section II-B for implementation.

D. Results

Fig. 5 summarizes all our experimental results. Figures 5a and 5b show the time series of the mean misprediction averaged over all test intervals and all test days at Location 1 and

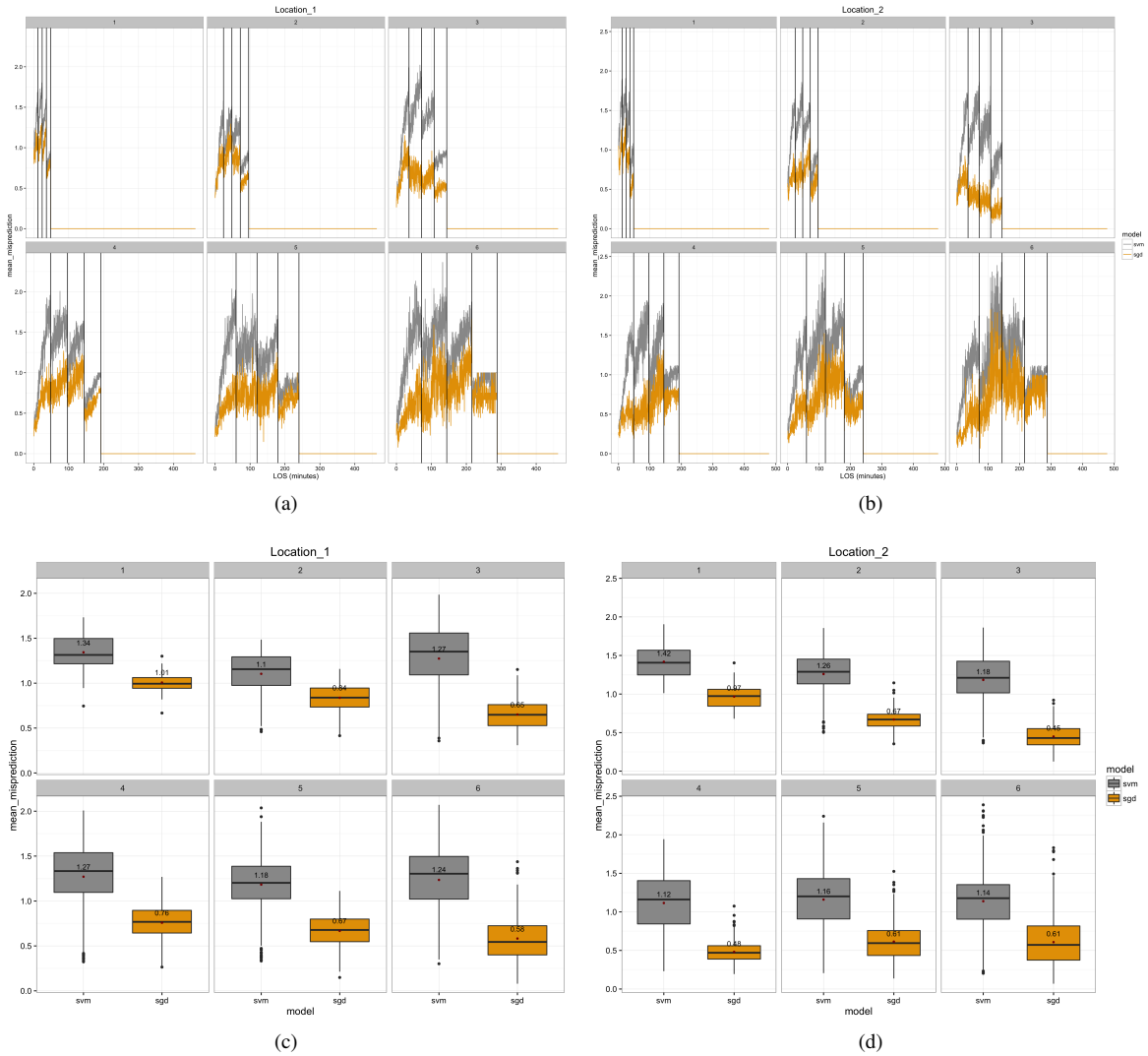

Fig. 5: **Top row:** Time series of the mean misprediction rate averaged over all test intervals and test days for each Δ ($1 \leq \Delta \leq 6$) for: (a) Location 1, and (b) Location 2. Horizontal axes show the timeline of an average test day. Vertical bars illustrate the defined class boundaries for each Δ. **Bottom row:** Box plots showing the mean misprediction distribution averaged over all time steps of all test intervals and test days $\forall\Delta$ for: (c) Location 1, and (d) Location 2. Mean values of the distributions are annotated in the 'boxes' as red dots. Prediction errors for class 5 are excluded once a device crosses the class's lower bound.

2, respectively. Each time series is an averaged prediction error rate made every 15 seconds throughout the entire day. Each day is also divided into 6 subplots for 6 different values of Δ. We additionally plot the vertical bars illustrating the defined class boundaries on each subplot. Through the figures, we can see quite consistently that the time series of SVM is almost always above that of SGD. In other words, SGD consistently outperforms (i.e., makes more accurate classifications than) SVM for all Δ at both locations over all time steps.

Irrespective of what classifier (SVM or SGD) and Δ, all time series share certain common characteristics. For any given sub-interval, the mean error rate almost always increases from the lower boundary to the upper boundary. This is because there is always less uncertainty in the beginning of the sub-interval (at the lower boundary), but as the device continues to stay on longer, there is more uncertainty whether the device would belong to this class or the next (if it eventually crosses the upper boundary). This progression of uncertainty is also true for the whole time series: there is less uncertainty initially (class 1), the level of uncertainty increases as the device stays longer (class 2 and 3). This is why the end of class 2 is where the curve typically 'peaks'. However, once the device crosses over to class 4, the level of uncertainty drops again (as there are only two classes left). As soon as it crosses over to class 5, there is no more uncertainty (as there is only one class left), and the error rate automatically drops to zero.

Figures 5c and 5d show the distributions of the mean error rates averaged across all time steps and test days for all values

of Δ at Location 1 and 2, respectively. Note that in these figures, the prediction errors for class 5 are excluded once a device has stayed passed its lower boundary (since the error rate then automatically becomes 0). The figures confirm what we have observed in the time series: SGD outperforms static SVM for all configurations. Not only are the mean and median of the mean misprediction of SGD lower, but the spread of its distributions are also smaller for most cases. Thus, not only does SGD make more accurate predictions, its predictions are also more stable (i.e., having smaller spread).

E. Discussion

As mentioned in Section III-A, Location 2 has more dynamic environment and short-term stays than Location 1. This implies that online learning in real-time should benefit more in Location 2 (as the model parameters are more frequently updated in light of recent training examples). Our experiments demonstrate that. Note that in Fig. 5d (Location 2), the difference between the distribution mean of SVM and that of adaptive SGD is almost always greater than the corresponding difference in Fig. 5c (Location 1), except for $\Delta = 6$. This demonstrates the efficacy of the proposed online SGD classifier for dynamic and rapidly changing environments.

From Figures 5c and 5d and considering only the adaptive SGD classifier, we see that, as Δ increases from 1 to 6, the prediction error rate behaves differently at two locations. The minimum error rate occurs at $\Delta = 6$ for Location 1 and at $\Delta = 3$ for Location 2. This matches with our observation that Location 2 is more dynamic such that (lower value of) $\Delta = 3$ best partitions the data into training and test sets. Location 1, on the other hand, has more regular (office) mobility patterns; thus, $\Delta = 6$ best partitions the data there. We also see that the minimum error rate at Location 2 is lower than that at Location 1, which strengthens our argument that adaptive SGD classifier is more suitable for more dynamic environments.

Finally, in terms of runtime performance, both static SVM and online SGD are very efficient: SVM takes no more than 3 (s) and SGD less than 1 (s) to train on average for all Δ. Each parameter update of Eqn. (1) takes constant time [16].

IV. Conclusion

In this paper, we propose, design and test a passive Wi-Fi sensing system to monitor and predict in real-time information about people's movements. Such a system has many interesting applications in retail settings in particular. We discuss specifically the application of predicting the length of stay (LOS) of an individual's mobile device in real-time at a specific location. The system automatically generates a number of features derived from probe requests and other frames obtained in a non-intrusive way. The features are used to train a linear SVM classifier as well as an online SGD update mechanism to take into account the dynamics of the environment and adaptive changes to the classification parameters. Of interest to the future work would be to explore other applications that can be derived from this type of system.

References

[1] A. Musa and J. Eriksson, "Tracking unmodified smartphones using Wi-Fi monitors," in *Proceedings of the 10th ACM conference on embedded network sensor systems*. ACM, 2012, pp. 281–294.

[2] M. V. Barbera, A. Epasto, A. Mei, V. C. Perta, and J. Stefa, "Signals from the crowd: Uncovering social relationships through smartphone probes," in *Proceedings of the 2013 Conference on Internet Measurement*. ACM, 2013, pp. 265–276.

[3] M. Cunche, M.-A. Kaafar, and R. Boreli, "Linking wireless devices using information contained in Wi-Fi probe requests," *Pervasive and Mobile Computing*, vol. 11, pp. 56–69, 2014.

[4] M. Handte, M. U. Iqbal, S. Wagner, W. Apolinarski, P. J. Marrón, E. M. M. Navarro, S. Martinez, S. I. Barthelemy, and M. G. Fernández, "Crowd density estimation for public transport vehicles," in *EDBT/ICDT Workshops*, 2014, pp. 315–322.

[5] I. L. S. Committee *et al.*, "IEEE 802.11 – Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," in *IEEE, June*, 2012.

[6] N. Cheng, P. Mohapatra, M. Cunche, M. A. Kaafar, R. Boreli, and S. Krishnamurthy, "Inferring user relationship from hidden information in WLANs," in *MILCOM 2012-2012 IEEE Military Communications Conference*. IEEE, 2012, pp. 1–6.

[7] B. Bonné, A. Barzan, P. Quax, and W. Lamotte, "WiFiPi: Involuntary tracking of visitors at mass events," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*. IEEE, 2013, pp. 1–6.

[8] W. Xi, J. Zhao, X.-Y. Li, K. Zhao, S. Tang, X. Liu, and Z. Jiang, "Electronic frog eye: Counting crowd using Wi-Fi," in *IEEE INFOCOM 2014 – IEEE Conference on Computer Communications*. IEEE, 2014, pp. 361–369.

[9] Y. Wang, J. Yang, Y. Chen, H. Liu, M. Gruteser, and R. P. Martin, "Tracking human queues using single-point signal monitoring," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2014, pp. 42–54.

[10] M. Cunche, M. A. Kaafar, and R. Boreli, "I know who you will meet this evening! Linking wireless devices using Wi-Fi probe requests," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*. IEEE, 2012, pp. 1–9.

[11] J. Manweiler, N. Santhapuri, R. R. Choudhury, and S. Nelakuditi, "Predicting length of stay at Wi-Fi hotspots," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 3102–3110.

[12] W. Zheng and D. Zhang, "Handbutton: Gesture recognition of transceiver-free object by using wireless networks," in *2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 6640–6645.

[13] T. V. Le, S. Liu, and H. C. Lau, "A reinforcement learning framework for trajectory prediction under uncertainty and budget constraint," in *ECAI 2016: 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands-Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, vol. 285. IOS Press, 2016, p. 347.

[14] I. S. Association *et al.*, "IEEE OUI and company ID assignments," http://standards.ieee.org/develop/regauth/oui/oui.txt, 2016.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[16] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.