Learning Activity Predictors from Sensor Data: Algorithms, Evaluation, and Applications

Bryan Minor, Janardhan Rao Doppa, *Member, IEEE,* and Diane J. Cook, *Fellow, IEEE*

Abstract—Recent progress in Internet of Things (IoT) platforms has allowed us to collect large amounts of sensing data. However, there are significant challenges in converting this large-scale sensing data into decisions for real-world applications. Motivated by applications like health monitoring and intervention and home automation, we consider a novel problem called *Activity Prediction*, where the goal is to predict future activity occurrence times from sensor data. In this paper, we make three main contributions. First, we formulate and solve the activity prediction problem in the framework of imitation learning and reduce it to a simple regression learning problem. This approach allows us to leverage powerful regression learners that can reason about the relational structure of the problem with negligible computational overhead. Second, we present several metrics to evaluate activity predictors in the context of real-world applications. Third, we evaluate our approach using real sensor data collected from 24 smart home testbeds. We also embed the learned predictor into a mobile-device-based activity prompter and evaluate the app for 9 participants living in smart homes. Our results indicate that our activity predictor performs better than the baseline methods, and offers a simple approach for predicting activities from sensor data.

Index Terms—Activity prediction; smart environments; digital prompting; regression learning

1 Introduction

Learning and understanding observed activities is at the center of many fields of study. An individual's activities affect that individual, society, and the environment. Over the past decade, the maturing of data mining and pervasive computing technologies has made it possible to automate activity learning from sensor data. This activity information is now commonly utilized in applications from security systems to computer games. As a result of this technology push and application pull, robust approaches exist for labeling activities that occurred in the past or may be occurring in the present. In this paper, we propose to extend this recent work to look at activities that will occur in the future.

We study a novel problem called *Activity Prediction*, where the goal is to predict the future activity occurrence times from sensor data, and introduce a data-driven method for performing activity prediction. Activity prediction is valuable for providing activity-aware services including energy-efficient home automation, prompting-based interventions, and anomaly detection. However, activity prediction faces challenges not found in many other data mining tasks: sensor data are noisy, activity labels provided by activity recognition algorithms are subject to error, and the data contains rich relational and temporal relationships that must be exploited to be able to make highly-accurate predictions.

We formulate and solve the activity prediction problem as an instance of the imitation learning framework, where the training data serves as the demonstrations provided by an expert. We provide a reduction of activity prediction learning to simple regression learning. This reduction allows us to leverage powerful off-the-shelf regression learners to learn an effective activity predictor that can reason about relational and temporal structure among the activities. Our approach naturally facilitates life-long learning, where the predictor can be improved and adapted based on new data from the users.

Selecting performance metrics for activity prediction is also challenging because there are multiple parameters that influence the desirability of the algorithm's performance. We provide several evaluation metrics and discuss their usefulness in the context of real-world applications. We evaluate our activity prediction learning algorithms on twenty-four smart home sensor datasets and find that our proposed imitation-based methods not only outperform baseline predictors but predict a majority of the activities within minutes of their actual occurrence.

In addition, we embed our activity predictor inside an activity prompting algorithm and demonstrate the effectiveness of the prompting app for multiple participants living in smart homes. This prompting application demonstrates a potential real-world use of our prediction algorithms. The predictions are used to remind the users to perform certain daily activities. These interactions are able to help users live more independently and can even positively influence their behavior.

We summarize our contributions as follows:

- Study a novel problem called *Activity Prediction* motivated by diverse real-world applications.
- Novel formulation of learning activity predictors in the framework of imitation learning and reduction to simple regression learning.
- Provide an evaluation methodology to measure the performance of an activity predictor for different

• *B. Minor, J. Doppa, and D. Cook are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, 99164.*
E-mail: bminor@eecs.wsu.edu

applications.
- Demonstrate good experimental results with our imitation-learning-based activity predictor on real sensor data collected from 24 smart home testbeds.
- Embed the learned predictor into a mobile-device-based activity prompter and evaluate the app on multiple participants living in smart homes.

In the next section, we formally describe the problem setup. This is followed by a description of both our independent and recurrent activity predictors. Next, we describe how activity recognition algorithms can be used to generate training data for learning activity predictors. Then, we describe various metrics for measuring the prediction performance, and provide experimental results comparing different predictors and analyze their performance. Finally, we describe an activity prompting mobile application we have used with a number of participants.

2 Problem Setup

We consider the problem of *Activity Prediction* from sensor data. Let $A = \{a_1, a_2, \cdots, a_K\}$ be a set of K activities, where a_j corresponds to the j^{th} activity class. Given features $\mathbf{x} \in \Re^d$ extracted from the sensor data as input, an activity predictor generates $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_K)$ as output, where $\hat{y}_j \in \Re$ is the predicted relative next occurrence time of activity a_j, or the predicted number of time units that will pass until a_j occurs again. Fig. 1 provides an illustration of the activity prediction problem.

Our training data consists of a sequence of raw sensor events $\Lambda = (\lambda_1, \lambda_2, \cdots, \lambda_N)$, where event λ_i corresponds to a sensor reading or sensor value with an associated timestamp t_i. We assume that an activity recognition (AR) algorithm is available to label each sensor event with its corresponding activity class and we use this information to train the activity predictor. An activity recognition algorithm learns a mapping from Λ to the corresponding activity label, a_Λ. We first label the raw sensor events Λ using the activity recognizer and use this to generate training data for activity prediction at each time step t_i as illustrated in Fig. 1. We employ the AR algorithm from Cook et al. [1] that yields 95% recognition accuracy via 3-fold cross validation on the activities evaluated in this paper.

We further assume the availability of a *feature function* Φ that computes a d-dimensional feature vector $\Phi(\lambda_i) \in \Re^d$ for any sensor event λ_i using the context of recent sensor events and a non-negative *loss function* L such that $L(x, \hat{y}, y^*) \in \Re^+$ is the loss associated with labeling a particular input $\mathbf{x} \in \Re^d$ by output $\hat{\mathbf{y}} \in \Re^K$ when the true output is $\mathbf{y}^* \in \Re^K$ (e.g., RMSE). Our goal is to return a function/predictor whose predicted outputs have low expected loss.

3 Learning Algorithms

In this section we describe two algorithms for learning activity predictors: 1) The Independent Predictor (IP), a simple baseline approach, and 2) The Recurrent Activity Predictor (RAP), which is intended to improve on the baseline.

3.1 Independent Predictor

The *Independent Predictor* is our baseline activity predictor. As the name suggests, this predictor completely ignores the relational and temporal structure of the problem, and makes predictions using only the information from the most recent sensor events at a given time. The independent predictor is trained as follows. For each sensor event λ_i in the training sequence Λ, we extract the features $\mathbf{x}_i = \Phi(\lambda_i) \in \Re^d$ listed in Table 1 (input) and the ground-truth activity predictions $\mathbf{y}_i^* \in \Re^K$ (output) from the labeled activity segments (see Fig. 1 for an illustration). We learn an independent regressor Π_j for each activity $a_j \in A$ as follows: for each sensor event λ_i in the training sequence Λ, we collect the input $\mathbf{x}_i$ and output y_{ij}^* (the ground truth prediction for activity a_j), and give the aggregate set of input-output pairs $\left\{\mathbf{x}_i, y_{ij}^*\right\}_{i=1}^N$ (training examples) to a regression learner to minimize the given loss function L.

During testing (or inference), we employ the K learned regressors independently on a given input $\mathbf{x} = \Phi(\lambda) \in \Re^d$ to compute the predicted output $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_K)$, where $\hat{y}_j = \Pi_j(x)$. The test-time complexity of this predictor is very low, which is valuable for making real-time predictions. However, the main weakness of this approach is that the recent sensor events may not provide sufficient context to make highly-accurate activity predictions, and it completely ignores the rich temporal structure of the problem.

3.2 Recurrent Activity Predictor

Notice that the independent predictor only uses the most recent sensor event data at a given time to make its predictions. To address this limitation, one could consider joint models by reasoning about the relationships between different activities, and accounting for the temporal structure of the problem. It is important to note that the activity prediction problem can be viewed as a generalization of sequence labeling, where each output token is a vector of K values corresponding to the next activity occurrence time of each activity (K is the number of activities).

A natural solution would be to define a graphical model to encode the relationships between input and output variables at different time steps and learn the parameters from the training data [2]. However, such a graphical model may be very complex (large branching factor) and can pose severe learning and inference challenges. We may consider simplifying the model to allow for tractable learning and inference, but that can be detrimental to prediction accuracy. An alternate solution is to employ a heuristic inference method (e.g., loopy belief propagation or variational inference) with the complex model. Even though these methods have shown some success in practice, it is very difficult to characterize their solutions and to predict when they will work well for a new problem. Therefore, we provide a much simpler but effective solution that is based on imitation learning.

Imitation Learning Formulation. We formulate and solve the activity prediction problem in the framework of imitation learning. In traditional imitation learning, the goal of the learner is to learn to imitate the behavior of an expert performing a sequential-decision making task (e.g.,

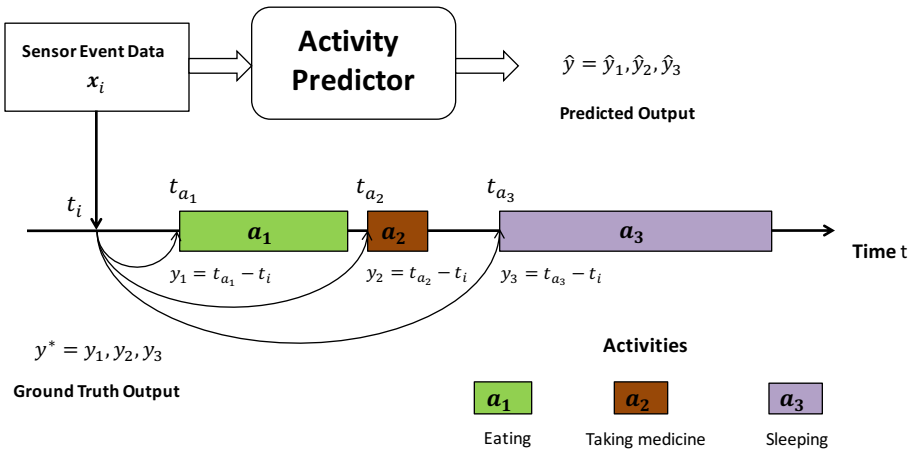

Fig. 1. A high-level overview of the activity prediction problem. Given features $\mathbf{x} \in \Re^d$ extracted from the current sensor event at time t_i as input, the activity predictor needs to predict the relative occurrence time of each activity. In this example, we have three activities: a_1 (eating); a_2 (taking medicine); and a_3 (sleeping). The starting times of activities a_1, a_2, and a_3 are t_{a_1}, t_{a_2}, and t_{a_3}, respectively. Therefore, the ground-truth output is $\mathbf{y}^* = (y_1, y_2, y_3)$, where $y_j = t_{a_j} - t_i$ stands for the correct relative next occurrence time of activity a_j.

playing a video game) in a way that generalizes to similar tasks or situations. Typically this is done by collecting a set of trajectories of the expert's behavior (e.g., games played by the expert) on a set of training tasks. Then supervised learning is used to find a predictor that can replicate the decisions made on those trajectories. Often the supervised learning problem corresponds to learning a mapping from states to actions and off-the-shelf classification tools can be used. In our activity predictor learning problem, the expert corresponds to the loss function L (available for training data) and the expert behavior corresponds to predicting the best output $\mathbf{y}_i^* \in \Re^K$ at each time step i (see Fig. 1). In recent work, imitation learning techniques are successfully applied to solve a variety of structured prediction tasks in natural language processing and computer vision [3], [4], [5], [6], [7], [8], [9]. However, to the best of our knowledge, imitation learning has not been used to solve activity learning tasks.

Main Challenges. The main challenge in applying imitation learning for solving the activity prediction problem is that the different output variables have structural dependencies as activities are related to each other. Therefore, the predictor Π should jointly reason about all the activities to compute the predicted output $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_K)$. Unfortunately, the size of the joint output space is exponential, which renders the straightforward application of the above imitation learning formulation impractical.

Pseudo-Independent Predictors. To address the aforementioned challenges, we employ *pseudo-independent* predictors to achieve tractability without losing accuracy. Specifically, we assign one predictor Π_j to each activity a_j. These predictors are not completely independent, but are pseudo-independent in the sense that each predictor predicts the output for only a single activity but has the previous predictions from all the other predictors available as part of the input (context) while making predictions. The main advantage of this pseudo-independent formulation is that it allows us to encode arbitrary relationships between activities and the temporal structure as context features, and is highly efficient in terms of training and testing without much loss in accuracy.

Algorithm 1 RAP Learning via Exact Imitation

Input: Λ = Training sequence of sensor event data labeled with activity segments, L = Loss function
Output: Π, the recurrent predictor

for each activity predictor $j = 1$ to K **do**
  Initialize the set of regression examples $\mathcal{D}_j = \emptyset$
end for
for each time step $i = 1$ to N **do**
  for each activity predictor $j = 1$ to K **do**
    Compute $\Psi_{local}(i) = \Phi(\lambda_i)$
    Compute $\Psi_{context}(i, j)$
    Joint features $\Psi_{ij} = \Psi_{local}(i) \oplus \Psi_{context}(i, j)$
    Compute best output $y_{ij}^* \in \Re$ using L
    Add regression example $\left(\Psi_{ij}, y_{ij}^*\right)$ to $\mathcal{D}_j$
  end for
end for
for each activity predictor $j = 1$ to K **do**
  Π_j =**Regression-Learner**($\mathcal{D}_j$)
end for
return learned predictor $\Pi = (\Pi_1, \Pi_2, \cdots, \Pi_K)$

Pseudo-Independent Representation. For learning and making predictions, we need a feature representation for each of our pseudo-independent predictors. The predictor Π_j for activity a_j will employ both local features $\Psi_{local}(i) = \Phi(\lambda_i)$ (see Table 1) and prediction context features $\Psi_{context}(i, j)$, including the previous activity predictions from a small history window. The context features $\Psi_{context}(i, j)$ we use in this work include previous predictions $\hat{y}$ for all activities except a_j. If we employ a history context of H previous windows, the context feature vector will be of size $H \cdot (K - 1)$.

Exact-Imitation Approach. Algorithm 1 provides the pseudo-code for our recurrent activity predictor learning via exact imitation of the loss function. At each time step i, for

each activity predictor Π_j, we compute the joint features $\Psi_{ij} = \Psi_{local}(i) \oplus \Psi_{context}(i,j)$ (input) and the best activity prediction $y^*_{ij} \in \Re$ (output) from the training data, where $\oplus$ refers to the vector concatenation operator. Note that for the exact imitation training, context features consist of ground-truth labels from the previous events. The aggregate set of input-output pairs $\left\{\Psi_{ij}, y^*_{ij}\right\}_{i=1}^{N}$ (training examples) for each activity predictor Π_j is given to a regression learner to learn Π_j by minimizing the given loss function L. If we can learn a function $\Pi = (\Pi_1, \Pi_2, \cdots, \Pi_K)$ that is consistent with all these imitation examples, then it can be proved that the learned function will generalize and perform well on new instances [10], [11].

DAGGER **Algorithm.** One issue with exact imitation training is error propagation: errors in early time steps can propagate to downstream decisions and can lead to poor global performance [12]. Therefore, we employ a more advanced imitation learning algorithm called DAGGER (Dataset Aggregation) [11] to learn robust predictors. DAGGER is an iterative algorithm that can be viewed as generating a sequence of predictors (one per iteration), where the first iteration corresponds to exact imitation training. In each subsequent iteration, DAGGER makes decisions with the predictor from the previous iteration, and generates additional training examples to learn to recover from any errors. A new predictor is learned from the aggregate set of training examples. In the end, the final predictor is selected based on a validation set. DAGGER has valuable theoretical properties and can be seen as a no-regret online learning algorithm [11]. If we deploy the learned recurrent predictor in a real-life application, then the predictor can be adapted in real time based on feedback from the users, and the DAGGER algorithm can be employed to naturally facilitate a *life-long learning* setting.

Algorithm 2 provides the pseudo-code for recurrent activity predictor learning via DAGGER. In each iteration k of DAGGER, we combine the predictor from previous iteration $\hat{\Pi}^k$ and the oracle predictor Π^* (predicts correctly using the training data) using a mixture parameter $\beta_k \in [0,1]$ to form the current policy Π^k that will be used to make the decisions. At each decision step, we flip a coin with bias β_k. If the coin turns up heads, we get the correct prediction from the oracle predictor Π^*. Otherwise, we ask predictor Π^k to make the prediction. Intuitively, the mixture parameter β_k allows us to generate diverse training examples.

At each time step i, for each activity predictor Π_j, we use the local features $\Psi_{local}(i)$ (these features do not change during training) and the context features $\Psi_{context}(i,j)$ based on the predictions of the current policy Π^k (these features can change as they depend on the previous predictions) to compute the predicted output $\hat{y}_{ij}$. We compare the predicted output $\hat{y}_{ij}$ with the correct output y^*_{ij} using the error threshold ϵ_j. If the difference between the predicted time and correct time for an activity class a_j is more than the allowed error ϵ_j, we call it a mistake. For each mistake, we use the input vector with context features from Π^k that was used to form the prediction, along with the ground-truth activity time y^*_{ij}, to generate a new training example and add it to the aggregate set of examples $\mathcal{D}_j$.

At the end of each DAGGER iteration k, the aggregate set of training examples $\mathcal{D}_j$ for each activity predictor Π_j is given to a regression learner to learn a new Π_j by minimizing the given loss function L. $\hat{\Pi}^{k+1} = (\Pi_1, \Pi_2, \cdots, \Pi_K)$ is the new recurrent predictor, where Π_j is the new regressor learned from the aggregate set of training examples $\mathcal{D}_j$. In the end, the final recurrent predictor is selected from the sequence of predictors based on the validation data.

Algorithm 2 RAP Learning via DAGGER

Input: $\mathcal{D}$ = Training examples, d_{max} = dagger iterations, $\epsilon = (\epsilon_1, \epsilon_2, \cdots, \epsilon_K)$ = error threshold vector
Output: Π, the recurrent predictor

for each activity predictor $j = 1$ to K **do**
  Initialize $\mathcal{D}_j$ with regression examples from exact imitation
  Π_j =**Regression-Learner**($\mathcal{D}_j$)
end for
Initialization: $\hat{\Pi}^1 = (\Pi_1, \Pi_2, \cdots, \Pi_K)$
for each dagger iteration $k = 1$ to d_{max} **do**
  Current Policy: $\Pi^k = \beta_k \Pi^* + (1-\beta_k)\hat{\Pi}^k$ // Π^* *is the oracle predictor*
  for each time step $i = 1$ to N **do**
    for each activity predictor $j = 1$ to K **do**
      Compute $\Psi_{local}(i) = \Phi(\lambda_i)$
      Compute $\Psi_{context}(i,j)$ using context predictions from Π^k
      Joint features $\Psi_{ij} = \Psi_{local}(i) \oplus \Psi_{context}(i,j)$
      Compute predicted output $\hat{y}_{ij} \in \Re$ using Π^k from Ψ_{ij}
      Compute correct output $y^*_{ij} \in \Re$ using oracle predictor Π^*
      if $\Pi^k(\Psi_{ij}) \neq \Pi^*(\Psi_{ij})$ **then**
        Add regression example $\left(\Psi_{ij}, y^*_{ij}\right)$ to $\mathcal{D}_j$
      end if
    end for
  end for
  for each activity predictor $j = 1$ to K **do**
    Π_j =**Regression-Learner**($\mathcal{D}_j$) // *learn from aggregate data*
  end for
  $\hat{\Pi}^{k+1} = (\Pi_1, \Pi_2, \cdots, \Pi_K)$ // *new recurrent predictor*
end for
return best predictor $\hat{\Pi}^k$ on the validation data

Regression Learner. Recall that both of our activity predictor learning algorithms need a cost-sensitive regression learner. We have a hard learning problem at hand, which means linear functions will not suffice. We experimented with standard regression tree learners (constant outputs at leaves), but they could not effectively handle the high variance for some activity times. Hence, we employ a variant of regression trees called model trees, where predictions are made by a learned linear function over all the features at each leaf node. To further improve the models' robustness, we utilize a random forest of 100 model trees in our experiments.

4 TRAINING DATA GENERATION VIA ACTIVITY RECOGNIZER

Time series forecasting uses a model to predict future values of a target variable based on previously observed values of the variable. In the case of activity forecasting, or activity prediction, the target variable is the number of time units that will elapse until a particular activity of interest will occur again. In order to base this prediction on previously observed values, we need to know when the activity occurred in the past. For this we rely on automated activity recognition. The challenge of activity recognition is to map sensor events to a label that indicates the corresponding activity the individual is performing. As with activity prediction, the activity recognition data consists of raw sensor

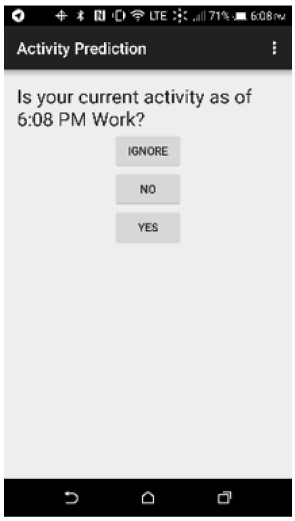

Fig. 2. Example prompt for the EMA app.

events, Λ (see Section 2). Features x_i are extracted from the sensor events at time t_i and a supervised learning algorithm maps the features onto a value from A which indicates the activity that is being performed.

Our activity recognizer builds on the AR algorithm from Krishnan and Cook [13] to perform real-time activity labeling on streaming data. AR extracts features from a sliding window containing w consecutive sensor events, $\lambda_{i-w} \ldots \lambda_i$, and learns a function that maps the feature vector x_i onto an activity label indicating the activity that is performed at the time of the last event in the window, or time t_i. The features that are used for the activity models in this paper are listed in Table 1. While the size of the window can be adjusted based on the most likely current activities, for our experiments we set the window size w to be 30.

Because of the insight that automated activity recognition sheds on human behavior and the valuable context activity labels bring to technology customization, activity recognition is a highly-investigated area of research [14], [15], [16], [17], [18]. Methods have been developed that encompass a diversity of sensor platforms including ambient sensors, wearable or phone sensors, and audio or video data. While AR utilizes a decision tree to learn activity models, these models can be created from a variety of learning approaches including support vector machines, Gaussian mixture models, and probabilistic approaches such as nave Bayes, hidden Markov models, and conditional random fields [14], [15], [19], [20], [21], [22], [23]. These models trade off computational cost, the type of sensor data that can be processed, and recognition accuracy / sensitivity. Unlike many of these earlier efforts, AR handles data that is collected in real-world settings with no data segmentation or participant scripting. It handles recognition in environments with multiple residents and with interwoven activities [13].

To train AR, we provide labels for at least one month of sensor data from each smart home location. Human annotators label the sensor events in each dataset with corresponding activities based upon interviews with the residents, photographs of the home, and a floorplan highlighting the locations of sensors in the space. Sensor events are labeled with the activity that was determined to be occurring in the home at that time. The datasets contain 120 activity classes in total, but many of them appear infrequently. For our experiments, we focus on 11 core activities that occur daily in a majority of the datasets. These activities, listed in Table 3, represent complex activities of daily living that are reflective of the resident's daily health and functioning [24]. Sensor events that do not fit into one of the core activity classes are labeled as "Other Activity" and provide context for AR as well as for the activity predictor. To maximize consistency of ground truth labels, multiple annotators look at the datasets and disagreements between labels are resolved via discussion. The annotators demonstrate an overall interannotator agreement of $\kappa = 0.85$.

We evaluate the accuracy of AR using two methods and utilize the accuracy results in the analysis of our activity

TABLE 1
Activity prediction features.

Feature	*Description*
lastSensorEventHours+*	Hour of day for current event
lastSensorEventSeconds+*	Seconds since the beginning of the day for the current event
windowDuration+*	Window duration (sec)
timeSinceLastSensorEvent+*	Seconds since previous event
prevDominantSensor1+*	Most frequent sensor in the previous window
prevDominantSensor2+*	Most frequent sensor in the window before that
lastSensorID+*	Current event sensor
lastLocation+*	Most recent location sensor
sensorCount+**	Number of events in the window for each sensor
sensorElTime+**	Time since each sensor fired
timeStamp+*	Normalized time since beginning of the day
laggedTimeStamps*	Previous event timeStamps
laggedPredictions***	Previous event predictions
timeSinceLastPrediction****	Time since previous predictions
maximumValue#	Maximum value of sensor
minimumValue#	Minimum value of sensor
sum#	Sum of sensor values
mean#	Mean of sensor values
meanAbsoluteDeviation#	Average difference from mean
medianAbsoluteDeviation#	Avg. difference from median
standardDeviation#	Value standard deviation
coeffVariation#	Coefficient of value variation
numZeroCrossings#	Number of median crossings
percentiles#	Number below which a percentage of values fall
sqSumPercentile#	Sq. sum values $<$ percentile
interQuartileRange#	Difference between 25th and 75th percentiles
binCount#	Values binned into 10 bins
skewness#	Symmetry of values
kurtosis#	Measure of value "peakedness"
signalEnergy#	Sum of squares of values
logSignalEnergy#	Sum of logs of squares
signalPower#	SignalEnergy average
peakToPeak#	Maximum - minimum
avgTimeBetweenPeaks#	Time between local maxima
numPeaks#	Number of peaks

+Used for activity recognition. *Local features Ψ_{local}, one of each. **Local features Ψ_{local}, one sensorCount and one sensorElTime for each sensor used. ***Context features $\Psi_{context}$, one per activity per context spot. ****Context feature $\Psi_{context}$, one per context spot. #Based on window of recent values for each sensor.

predictor. First, we measure AR recognition performance using 10-fold cross validation on the annotated sensor data for the 11 activities. Using this method, AR achieves a 96% accuracy for the datasets analyzed in this paper.

Our second evaluation method pairs AR with ecological momentary assessment (EMA) [25]. EMA is considered a reliable measurement technique in the psychology and sociology literature for recording events and behavior data in a natural setting [26], [27]. The idea is to query participants about the activity they are currently performing in order to obtain the most accurate, in-the-moment information about their behaviors. We collect this information using an EMA app that brings up an activity query at random times throughout the day, as shown in Fig. 2. The queried activity label is provided by AR. If AR does not guess the current activity correctly, the user can optionally provide the correct current activity label in order to better train the model. We collected 291 query responses from 8 smart home sites. From the results we observe an 86% accuracy rate for AR. We will utilize both of these baseline performance results as we evaluate our activity predictors.

5 Evaluation Methodology

In this section, we will review and introduce several metrics to evaluate activity prediction algorithms in the context of real-world applications. To compare the effectiveness of different solution approaches for a given problem, the evaluation metrics must be carefully chosen. The quality and usefulness of a particular metric will vary based on the application and specific evaluation criteria. Many metrics tend to emphasize particular aspects of the results, so choosing multiple metrics can be necessary to completely understand the effectiveness of an approach.

Challenges. Selecting performance metrics for activity prediction is challenging because there are multiple parameters that influence the desirability of the algorithm's performance. Activity predictors can be evaluated in multiple ways, depending upon the type of performance that is desired. First, activity prediction can be viewed as a type of classification task in which any prediction that has non-zero error (or error greater than a threshold) is considered a mis-labeled data point. In this case, traditional classifier-based performance measures can be utilized. Second, activity prediction can be considered as a type of forecasting algorithm. Viewed in this light, error is proportionate to the numeric distance between the predicted and actual values. In addition, activity prediction relies on the effectiveness of an online activity recognition algorithm. The performance of the activity predictor is not anticipated to exceed the reliability of the activity recognizer that is being used to train the predictor. The activity recognizer, in turn, is trained using hand-annotated data which may be inconsistently labeled.

Evaluation Metrics. We introduce several evaluation metrics and employ them to validate our prediction algorithms. Using our previous notation, $\hat{\mathbf{y}}_i$ represents a vector of predicted outputs for a sensor event in the evaluation dataset with elements $\hat{y}_{ij}$. $\mathbf{y}_i^*$ is the vector of true values for the same event with elements y_{ij}^*. Note that we have K activities in total. Each evaluation metric takes a predicted output $\hat{\mathbf{y}}_i$ and ground truth output $\mathbf{y}_i^*$ as input, and returns a real-value indicating the quality of the prediction. One could perform macro-averaging of metric values over different testing instances and datasets to compute aggregate values.

Mean absolute error (MAE), as defined in (1), provides a measure of the average absolute error between the predicted output and ground-truth output. It is similar to another well-known metric, **root mean squared error (RMSE)**, defined in (2). Both of these measures provide the average error in real units and quantify the overall error rate, with a value of zero indicating a perfect predictor and no upper limit. Because RMSE squares each term, it does bring a disadvantage in effectively weighting large errors more heavily than small ones.

$$\text{MAE} = \frac{\sum |\hat{y}_{ij} - y_{ij}^*|}{K} \tag{1}$$

$$\text{RMSE} = \sqrt{\frac{\sum (\hat{y}_{ij} - y_{ij}^*)^2}{K}} \tag{2}$$

If the activities have varying levels of importance, we may want to use an error measure that places more emphasis on some activities than others (e.g., weighted RMSE). This might be the case if a particular activity needs to be predicted very accurately, for example. Additionally, we may need to compare results across activities or datasets where the time spacing between activity occurrences may be different. In these cases, measures such as MAE and RMSE do not give an indication of the *relative* error. For example, an error of 60 minutes in predicting a time-critical activity (e.g., taking medicine) may be unacceptable, but may be acceptable for other activities that do not need to happen at a certain time (e.g., housekeeping). In such situations, we may want to use a normalized error, such as **range-normalized RMSE (NRMSE)**, defined in (3). Here, the minimum and maximum functions are computed over all ground-truth values of the test instances we are evaluating. This metric would usually be applied on each activity or dataset that we wish to separate. While NRMSE is convenient for comparing results from different sets, it does not have a well-defined normalization factor with which we can evaluate the actual magnitude of the errors.

$$\text{NRMSE} = \frac{RMSE}{\max(y_{ij}^*) - \min(y_{ij}^*)} \tag{3}$$

Another useful normalized metric is **mean absolute percentage error (MAPE)**, defined in (4). MAPE normalizes each error value for each prediction by the true value y_{ij}^* we are trying to predict. This metric allows us to normalize the error individually for each prediction. We can also quickly determine approximately how large the error is since it is a percentage of the true activity time. However, as y_{ij}^* approaches zero (i.e., the activity is about to occur), an error of any insignificant amount can cause the element in the summation to become large. This leads to inflation of the MAPE value due to a few outlier cases where the error is small but the true activity time is even smaller.

$$\text{MAPE} = \frac{\sum \frac{|\hat{y}_{ij} - y_{ij}^*|}{y_{ij}^*}}{K} \tag{4}$$

Since the metrics we have listed so far are based on finding the averages of all errors, they are sensitive to possible distortion by outliers. As a result, the metrics can often have large values. In order to analyze the effects of outliers, other evaluation metrics can be used. One metric we introduce for this purpose is the **error threshold fraction (ETF)**, defined in (5). $I(\hat{y}_{ij}, y^*_{ij}) = 1$ if $|\hat{y}_{ij} - y^*_{ij}| \leq v$ and 0 otherwise. Note that the numerator of the fraction is a count of the number of events with error below the threshold v. This metric indicates the fraction of the errors that are below the time threshold v. v should be non-negative, and $\lim_{v\to\infty} \mathrm{ETF}(v) = 1$. By varying v we can ascertain how the errors are distributed; if we find that the ETF does not approach 1 until v is large, this may indicate that there are a significant number of large-error outliers. $\mathrm{ETF}(0)$ indicates the number of predictions which had zero error.

$$\mathrm{ETF}(v) = \frac{\sum I(\hat{y}_{ij}, y^*_{ij})}{K} \tag{5}$$

Yet another metric to consider is **Pearson's** r, i.e., the correlation coefficient between the predicted and actual activity occurrence times. This measure, shown in (6), does not quantify the amount of error but does indicate the relationship between the predicted and actual values.

$$\mathrm{r} = \frac{\sum(\hat{\mathrm{y}}_{\mathrm{ij}} - \overline{\hat{\mathrm{y}}_{\mathrm{ij}}})(\mathrm{y}^*_{\mathrm{ij}} - \overline{\mathrm{y}^*_{\mathrm{ij}}})}{\sqrt{\sum(\hat{\mathrm{y}}_{\mathrm{ij}} - \overline{\hat{\mathrm{y}}_{\mathrm{ij}}})}\sqrt{\sum(\mathrm{y}^*_{\mathrm{ij}} - \overline{\mathrm{y}^*_{\mathrm{ij}}})}} \tag{6}$$

Usually there is no single best evaluation metric for any particular application. We may use multiple metrics in order to evaluate multiple aspects of the performance. In fact, the nature of the data mining we present here is further complicated because error and imprecision occurs in several places:

1) Ground truth labels. Inaccurate class labels represent a source of error that exists in many datasets. We estimate the amount of error in ground truth activity labels by measuring inter-annotator agreement, or the degree of agreement of the activity labels between multiple annotators. This is typically represented using Cohen's kappa [28].

2) Activity recognition accuracy. In general, the activity recognition model is trained using a limited set of training data. The trained model will then be used to generate activity labels for previously-unseen data. The model itself may be subject to error due to representational limitations, small training set, or shortcomings of the employed learning algorithm.

3) Predictor error. In the same way that the activity recognition algorithm will likely experience some error, so also an imperfect activity prediction algorithm will generate erroneous predictions.

Given that there are multiple sources of error, we need to reconsider the standard procedure employed for evaluating predictors. Unlike classification algorithms, where an accuracy of 100% is expected, in this case the expected accuracy will be limited to the quality of the labels. As a result, the evaluation metrics that are discussed here can be κ-*normalized* to reflect the same accuracy range that would be considered for a perfect dataset, while being sensitive to label noise that is known to be present in the data.

TABLE 2
Description of CASAS smart home testbed datasets used to evaluate activity predictors.

ID	*Residents*	*Time Span*	*Sensors*	*Sensor Events*
1	1	2 months	36	219,784
2	1	2 months	54	280,318
3	1	2 months	26	112,169
4	1	2 months	66	344,160
5	1	2 months	60	146,395
6	1	2 months	60	201,735
7	2	1 month	54	199,383
8	1	2 months	54	284,677
9	1	2 months	44	399,135
10	1	1 month	38	98,358
11	1	2 months	54	219,477
12	1	4 months	40	468,477
13	1	12 months	58	1,643,113
14	1	1 month	32	133,874
15	1	10 months	40	1,591,442
16	1	2 months	38	386,887
17	1	12 months	32	767,050
18	1	1 month	46	178,493
19	1	1 month	36	92,000
20	1	2 months	40	217,829
21	2	10 months	62	3,361,406
22	1	2 months	56	247,434
23	1	1 month	32	106,836
24	1	2 months	34	216,245

In our evaluation of the prediction algorithms, we employ MAE, RMSE, and ETF. MAE and RMSE values are provided in seconds, which are the same units used for the predictions. In order to detect outliers in the errors that may negatively affect the RMSE, we also report ETF values. We vary the ETF threshold from one second up to 24 hours to observe the corresponding distribution of errors for each method. Finally, in the prompting application, where the ground truth labels are provided by the actual participants at the time prompts are delivered, we will also consider κ-*normalization* of the prediction results.

6 Experiments and Results

In this section we empirically evaluate and compare our activity predictors on real-world data using the evaluation metrics introduced in Section 5.

6.1 Experimental Setup

Datasets. We evaluate our activity prediction algorithm using sensor and activity data collected from 24 CASAS smart homes[1]. Descriptions of the datasets are provided in Table 2. Each CASAS smart home test bed used in this evaluation includes at least one bedroom, a kitchen, a dining area, and at least one bathroom. While the sizes and layouts of the apartments vary, each home is equipped with combination motion/light sensors on the ceilings as well as combination door/temperature sensors on cabinets and external doors. Sensors unobtrusively and continuously collect data while residents perform their normal daily routines. Fig. 3 shows a sample layout and sensor placement for one of the smart home test beds. Each smart home sensor event is labeled

1. These datasets are available at http://casas.wsu.edu.

TABLE 3
Activity classes.

Activity	*Sensor Events*
Bathe	208,119
Bed-Toilet Transition	174,047
Cook	2,614,836
Eat	585,377
Enter Home	174,486
Leave Home	311,164
Personal Hygiene	1,916,646
Relax	2,031,609
Sleep	732,785
Wash Dishes	1,139,057
Work	2,028,419

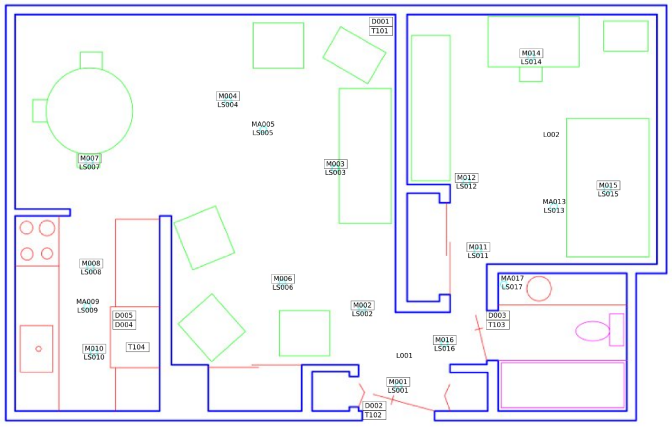

Fig. 3. Floor plan of one CASAS smart home testbed. The location of each sensor is indicated with the corresponding motion (M), light (LS), door (D), or temperature (T) sensor number.

by AR; these labels are collectively used to determine the ground-truth activity predictions $\mathbf{y}^*$ at any given time. The predictors were trained and tested separately for each dataset. The 11 activities used by both AR and the predictors are shown in Table 3.

Activity Prediction Algorithms. We evaluate two forms of our Recurrent Activity Predictor (RAP). The first is **RAP with Exact Imitation (RAP-EI)**, which uses the exact-imitation approach described in Section 3.2. The second version is **RAP with** DAGGER **(RAP-**DAGGER**)**, using the DAGGER approach to improve the predictor with new training values. We also evaluate the **Independent Predictor (IP)** as a informed baseline.

For all three algorithms, we employ the local features $\Psi_{local}(i)$ described in Table 1. These features are generated directly from recent sensor events and highlight information contained in those events. For the RAP-EI and RAP-DAGGER algorithms, we also use context features $\Psi_{context}(i,j)$ listed in Table 1. For each activity's predictor model, these context features consist of the previous predictions for all activities *except for the activity being predicted*. The number of previous predictions used is determined by the context size H. To account for uneven time spacing between events, the time since each set of previous predictions is also included.

We also create a second baseline called **Exponential**, which is uninformed. This method does not learn a complex model of activity times. Instead, it models the relative times of each occurrence for each activity as an exponential distribution. The Exponential method then samples from the distribution in order to generate activity predictions.

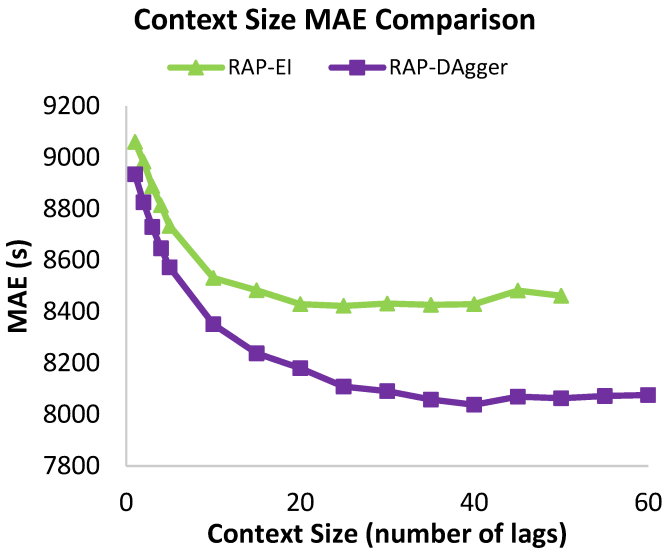

Fig. 4. Average MAE based on context size with one-month datasets.

6.2 Evaluation Procedure

To evaluate the performance of our predictors on these temporal datasets, we employ a *sliding window* validation procedure. This method is similar to k-fold cross-validation, but allows us to maintain the temporal ordering of the sensor event data. We select a window of N=2000 events which we use along with the corresponding ground-truth values as the training examples $\{\mathbf{x}_i, \mathbf{y}_i^*\}_{i=1}^{N=2000}$. We learn a predictor from this training data and employ it to make predictions for the next 5000 events after the window. The window is then shifted forward by 1000 events and the process is repeated. For exact-imitation training, the lag (context) values are provided using the ground-truth values from the training data, while the predicted values are employed during testing.

6.3 Varying Context Size

The performance of the recurrent predictors varies based on the the context size. In order to determine the optimal context size, we vary the size and observe the effects on the RAP-EI and RAP-DAGGER predictors. These tests are performed using the first month of data from each dataset.

The average MAE for both methods is shown in Fig. 4. As context size increases, performance for both RAP-EI and RAP-DAGGER improve until performance plateaus. Beyond this point, adding more context features does not further improve performance. For RAP-EI, this point is around $H = 20$. RAP-DAGGER continues to see improvement up until $H = 40$, however the most of the improvement has been achieved by a context size of $H = 20$.

Based on these results, we find a context size of $H = 20$ previous events to be a good choice. This provides the

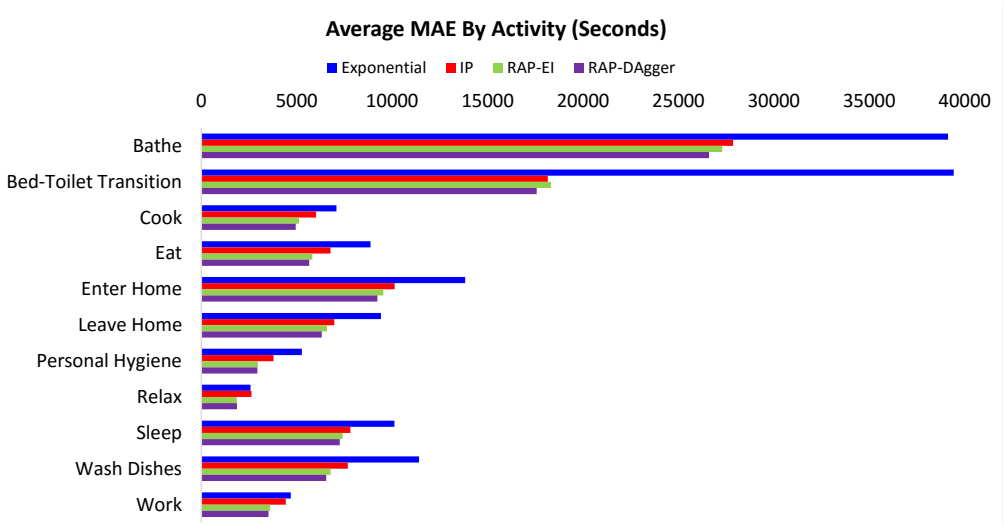

Fig. 5. Average MAE for each activity. These values were averaged for each activity across all datasets.

majority of the benefit gained from a larger context size. Increasing the context size further may provide a small benefit, however the complexity increases as the context size increases.

6.4 Comparison of Different Predictors

Building on the results of the previous section, we now compare the performance of each predictor method using a context size of $H = 20$ previous predictions for the recurrent predictors. For these tests, we use the full datasets described in Table 2. We compare the following four methods:

- Exponential Baseline
- Independent Predictor (IP)
- RAP-EI with context $H = 20$
- RAP-DAGGER with $d_{max} = 5$ iterations, $\beta = 0$, and $H = 20$

IP vs RAP-EI. Table 4 shows the average MAE and RMSE results for each of the methods. IP had an average MAE of 9,294 seconds (about 2.5 hours). The RAP-EI method improves on this, reducing the MAE by about 630 seconds. The RMSE is improved by about 1,900 seconds, or about one half-hour. We also note that the RMSE values can be dramatically influenced by the outliers. There are a few datapoints in which the predicted activity time is off by almost a day. RMSE squares each error there by the average performance measure can be biased by these few outliers. We conclude that to examine the overall performance of the predictors, MAE is a better measure.

The average MAE results for each activity are shown in Fig. 5. RAP-EI outperforms IP on all activities except for Bed-Toilet Transition. This activity is one which can occur at different times during the night, usually unrelated to other activities. Hence, the activity context features used in RAP-EI do not provide any additional information for predicting Bed-Toilet Transitions and may result in higher error. We note that the performance of RAP-EI shows increased improvement over IP for activities such as Cook, Eat, Personal Hygiene, and Wash Dishes. These activities tend to be highly related to each other (e.g., the resident cooks and then eats their meal, followed closely by washing dishes and then brushing their teeth). The context features allow the recurrent predictor to discover these activity relationships to improve its performance over the independent predictor. As with the overall results, these MAE values show that

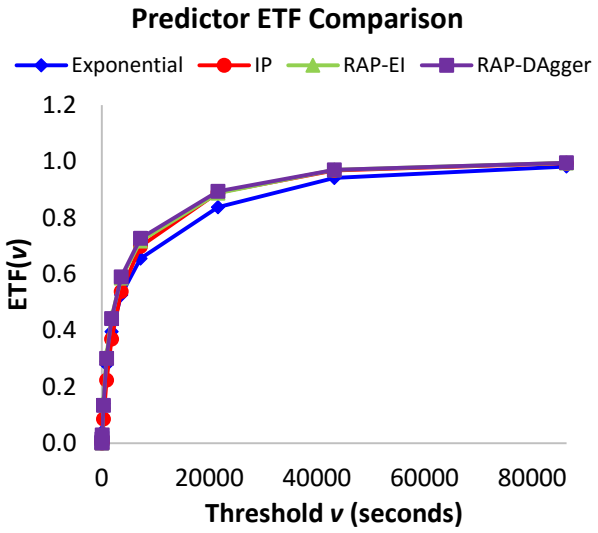

Fig. 6. ETF plotted for each predictor. Threshold values range from one second up to one day.

TABLE 4
Overall MAE and RMSE results for the different predictors (in seconds). The RAP predictors have context size $H = 20$. These values were found by averaging the individual metrics across all the datasets. A one-way ANOVA indicates that the differences in performance are extremely significant ($p < .01$).

Method	*MAE*	*RMSE*
Exponential	13,709.05	27,772.89
IP	9,294.48	21,370.18
RAP-EI	8,661.30	19,439.62
RAP-DAGGER	8,402.01	19,114.13

RAP-EI generally has better performance compared to IP.

Fig. 6 shows the ETF values for varying thresholds. The independent predictor has about 9% of its errors below 30 seconds. RAP-EI has about twice as many errors (17%) below 30 seconds. Similarly, 45% of RAP-EI errors are below 30 minutes, compared to only 37% of IP errors. These results indicate that not only does RAP-EI provide improved overall performance, but it also is able to have a greater number of its errors below one hour compared to IP. We also note that the majority of both IP and RAP-EI predictions are within one hour of the actual time, which is sufficient for many applications.

We note that both IP and RAP-EI outperform the random Exponential predictor in most respects. They show significant improvement in terms of overall MAE and RMSE. The Exponential predictor outperforms IP on the Relax activity. However, this is likely due to the fact that the Relax activity occurs frequently throughout the day, resulting in the actual activity times being similar to the mean of the random distribution. Consequentially, the Exponential predictor is able to form predictions that are, on average, closer to the actual times. However, the ETF values indicate that more of the Exponential errors are larger than one hour, as the baseline does not actively adjust its predictions for smaller time differences. Thus, the informed predictions from IP and the RAP predictors show an improvement over the uninformed baseline prediction.

RAP-IE vs RAP-DAGGER**.** We also examine the performance improvement that can be gained using DAGGER to refine the exact imitation predictor. RAP-DAGGER shows a greater overall performance in terms of both MAE and RMSE compared to RAP-EI. This indicates that the diversification of training examples gained through DAGGER allows the predictor to increase its ability to correct from errors. This is also demonstrated by the results in Fig. 5, where RAP-DAGGER outperforms RAP-EI for almost all activities. We note that the performance improvement is small compared to that between IP and RAP-EI. However, adding more DAGGER iterations or increasing the context size should lead to increased performance.

The ETF plot of Fig. 6 indicates that RAP-EI slightly outperforms RAP-DAGGER for smaller error sizes. However, RAP-DAGGER has more of its errors below larger thresholds such as 6 hours. This indicates that the improvements from DAGGER reduce the number of larger outlier errors. Furthermore, the fraction differences between the two predictors are small for the lower thresholds.

Area Under ETF Curves. We note the similarity between the ETF curves in Fig. 6 and a standard ROC curve. In this case, the discrimination threshold is based on the time-based threshold for prediction error. As with the Area Under a ROC Curve, a perfect predictor will have an Area Under the ETF Curve (AUETF) of 1.0. The AUETF values for our predictors are given in Table 5. Consistent with the ETF plots, the RAP-EI and RAP-DAGGER methods outperform IP and all informed predictors outperform the Exponential method.

Behavior Over Time. It is also of interest to examine how the error for each method changes as we move further from the training window. Fig. 7 shows the average MAE at each test horizon against how far that horizon was from the training window. For all methods except the Exponential, the average error is relatively low just after the training window (around 5 minutes). The error generally increases as the test event gets further from the training window. IP has a slightly higher error than RAP-EI, which in turn has a slightly higher error than RAP-DAGGER. All three have much lower error at all event horizons compared to the random Exponential.

TABLE 5
Area under the ETF curve (AUETF) values for each predictor.

Method	*AUETF*
Exponential	0.8672
IP	0.8946
RAP-EI	0.9008
RAP-DAGGER	0.9036

For all predictors, the error tends to increase as the event horizon moves away from the training window. However, IP and both RAP predictors stabilize after about 2,000 events at error values of about 2 hours, respectively. We suspect that the error rates are partially related to the size of the training window used. While the event frequency is different for each dataset, 5000 events is approximately a day or two in length. At 2000 events, the training window is relatively small compared to the size of the datasets, but this window size was chosen to provide a sufficient number of test windows for computing the evaluation metrics. It is likely that increasing the training window size (and thus allowing more of the residents' activities to be observed) may reduce the error rates for the predictors. This hypothesis is supported by the results from the prompting app evaluation, shown in the next section. The predictors used for the app were trained with more than a month of data and had error rates below an hour even at more than two weeks beyond the training window.

7 ACTIVITY PROMPTING

The ability to predict, or forecast, future occurrences of activities can play a central role in activity prompting. Activity prompting can be used to remind a memory-impaired individual of an activity they typically perform or to encourage integration of a new healthy behavior into a normal routine. Prompting technologies have been shown to increase adherence to medical interventions, decrease errors in activities of daily living, and increase both independence and engagement for individuals with cognitive impairment [29], [30], [31], [32]. While limited context-based prompt methods have been explored for medication adherence and activity initiation [33], [34], [35], [36], [37], [38], [39], [40], [41], these approaches do not employ knowledge of current and upcoming activities when delivering prompts. Similarly, while commercially-available time or location-based prompting may help individuals use the aid, they typically require that the user learn how to program the prompts [31], [35], [42], [43] which may reduce use [44]. Activity awareness may reduce these limitations of current prompting systems [45].

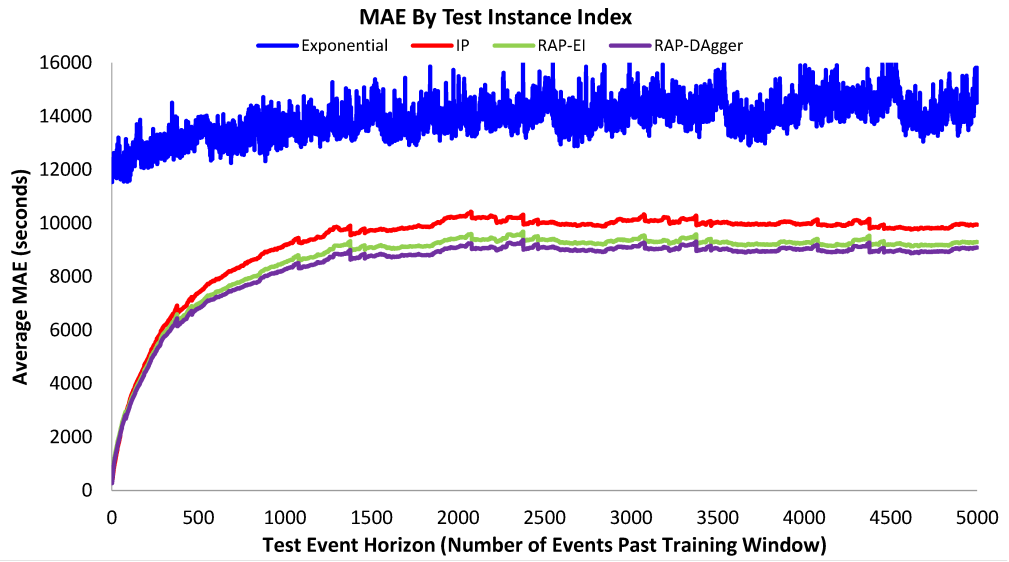

Fig. 7. MAE plotted for each predictor against the test horizon. The test horizon indicates how far (in number of events) the test event is from the end of the training window. MAE values are averaged over all activities and datasets at each test horizon.

We evaluated our IP activity predictor in the context of an activity prompting app called CAFE (CASAS Activity Forecasting Environment). Rather than relying on manual setting of reminder times or hand construction of reminder rules [37], [46], CAFE prompts individuals based on the predicted times that the activities will occur. The iOS-based app periodically queries a server for the predicted times of selected activities. An activity recognition algorithm [13] and our activity predictor both reside on the server and generate real-time labels and predictions as sensor data arrive from the smart homes. When the predicted occurrence time is reached, CAFE issues a notification, as shown in Fig. 8.

We evaluate CAFE with two sets of smart home residents. The first set (Group 1) consisted of two smart homes with young adult participants, while the second (Group 2) consisted of seven smart homes with adult residents over age 65, as described in Table 6. These homes are instrumented with sensors for motion, temperature, light, and door usage. Sensor data is automatically labeled using the AR activity recognition algorithm. For all homes, we utilize the generalized AR model that was trained from the datasets described in Table 2 to generate training data. None of the homes were part of the training set, so the training labels rely on the generalization power of the learned activity recognizer. Using the training data labeled by AR, we then trained a separate independent activity prediction model for each site to form predictions for use in CAFE. The participants responded to CAFE activity prompts over periods that ranged from one week to one month.

The participants in Group 1 were prompted for seven activities. The participants in Group 2 were each prompted for a different set of two or three activities chosen by the participants. They provided a total of 1619 responses, distributed as shown in Fig. 9.

We note that delays may occur between the activity occurring and the prompt being generated. This is partly due to the fact that the database is updated every 15 minutes, after which new prompts are generated. Furthermore, participants would often not respond to a prompt for a minute or so. As a result, they sometimes had already started an activity before they responded to the prompt, even if the prompt arrived at the correct time. Therefore, they respond with "I already did it" or "I will do it later".

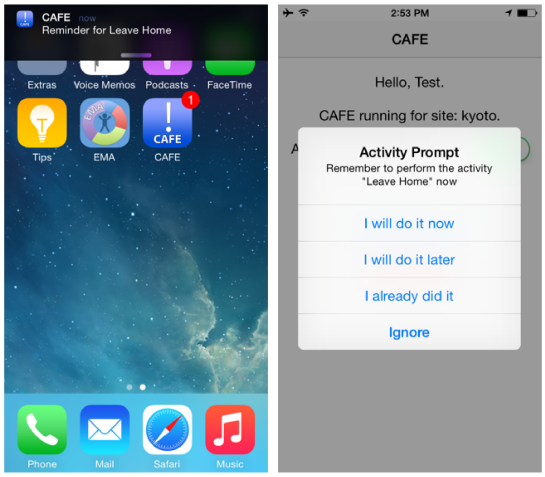

Fig. 8. Interface for the CAFE app.

Given the nature of the current notification generation, we also evaluated the prompt timings based on MAE and range-normalized MAE, as summarized in Table 7. These values are calculated by comparing the predicted activity times with the activity start times labeled by the activity recognizer. Each activity occurred at least once a day and MAE values were normalized based on a maximum error of 43,200 seconds, or half of a day. As shown in Table 7, the MAE averaged over all participants in both groups is 933.73 seconds (about 16 minutes). The average prediction error aligns closely with the infrastructure-related delays on average. To further assess the error we calculate the ETF value using 30 minutes, the maximum infrastructure-related delay, as our threshold value.

TABLE 6
Description of CAFE testbeds.

Testbed	*Residents*	*Sensors*	*Training Data*	*Prompting Time*	*Activities*	*Prompts*
Group 1: Young Adults						
kyoto	2	81	2 months	2 weeks	Bathe, Cook, Eat, Leave Home, Relax, Sleep, Work	22
navan	1	28	4 months	2 weeks		90
Group 2: Older Adults						
ihs06	2	26	3 weeks	1 month	Bed-Toilet Transition, Relax, Work	615
ihs07	1	29	3 weeks	3 weeks	Cook, Leave Home	226
ihs08	2	28	3 weeks	1 month	Bathe, Sleep	173
ihs09	2	22	3 weeks	1 month	Eat, Sleep	414
ihs12	1 + 1 pet	23	5 weeks	2 weeks	Eat, Wash Dishes	27
ihs14	4 + 2 pets	29	4 weeks	1 week	Cook, Leave Home	8
ihs21	1	23	4 weeks	3 weeks	Leave Home, Relax	44

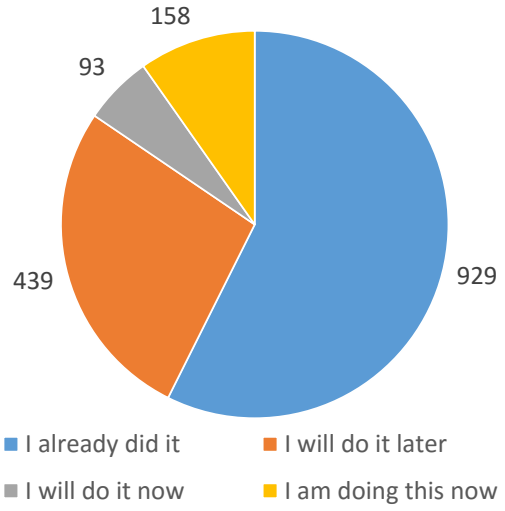

Fig. 9. Distribution of CAFE responses for all participants.

Finally, we obtained ground truth activity labels during the same period that participants in Group 1 received activity prompts. This was accomplished through the use of the EMA app. Using the EMA app, participants are queried every 15 minutes about the activity they are currently performing. The responses are stored in the database and used to validate our activity recognition algorithms. From the collected responses, we report AR accuracy of 92% for the seven activities used. We use this information to generate the κ-normalized values summarized in Table 7.

Interestingly, the participants noted that the app sometimes actually created a modification in their behavior. One resident recalled a time when he was debating between leaving home to get groceries or watching television. Upon receiving a CAFE prompt to leave home, he left immediately to perform his errands. On another occasion, a participant started working earlier than originally planned due to the prompt notification. Integrating activity prompts into daily behavioral routines thus raises interesting challenges for intervention design that need to be carefully considered in future work.

TABLE 7
Evaluation of CAFE prompts (in seconds).

MAE	*Normalized MAE*	*ETF*	*κ-Normalized ETF*
933.73	0.02	0.84	0.92

8 Related Work

Activity learning has been investigated over the last decade for a plethora of sensor platforms, including ambient sensors, wearable sensors, phone sensors, and audio/video data [14], [17]. In addition to activity recognition, activity discovery has been extensively studied utilizing techniques such as zero-shot learning, topic models, and sequence discovery [1], [47], [48], [49], [50]. In the work by Koppula and Saxena, the predicted next activity was supplied to robots in order to provide better assistance.

While activity prediction is not as heavily investigated as these other areas of activity learning, there are some representative first efforts in this area. Many of these techniques focus on sequence prediction, which can be adapted to predict the label of the activity that will occur next in the sequence. This work includes the Active LeZi algorithm [51] which is used to predict the identifier of the sensor in a home that will generate the next event. Other researchers including Hawkins et al. [52], Kitani et al. [53], and Koppula and Saxena [54] have investigated the use of probabilistic graph models to predict next events in video data.

In contrast with the emerging area of activity prediction, activity prompting systems have been developed and evaluated for quite a while. The majority of existing approaches are rule-driven or rely upon manually-generated user schedules [37], [40], [55]. Some recent work has focused on providing prompts to keep an individual on task when they fail to complete critical activities [31], [39], [42], [43], [56], [57], although these require extensive training for each activity. Other approaches [41] use sensed locations to generate reminders that are associated with those places. While these systems may adjust prompts based on user activities, they require substantial manually-generated information about an individual's routine and locations where activities are performed. In contrast with these approaches, the method we propose is data-driven. By utilizing activity-

labeled sensor events to learn an individual's normal routine, our algorithm can automatically generate activity predictions and associated activity prompts.

As an area for continued research, we hypothesize that activity prediction techniques can benefit from segmenting sensor data into sequences that represent single activity types. A number of techniques have been proposed for this task. Some approaches are unsupervised and utilize object-use fingerprints [58], [59] or statistical change point detection [60], [61].

9 Summary and Future Work

We studied a data-driven approach for predicting future occurrences of activities from sensor data. We showed how powerful regression learners can be leveraged to learn an effective activity predictor that can reason about relational and temporal structure among the activities in an efficient manner. Our extensive experiments on twenty-four smart home datasets validate that our recurrent activity predictor is effective at forecasting upcoming activity occurrences. Additionally, we illustrated the use of the predictor as part of our CAFE activity prompting app.

In the future, we will enhance our approach by incorporating iterative prediction refinement as well as smoothing, and perform a large-scale user study. In this paper, we limited our evaluation to consider only activity initiation, but an exciting direction will be to predict the length of the activity as well as individual activity steps. We will study the theoretical properties of our approach and apply it to more general prediction problems.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants 0900781 and 1262814 and by the National Institute of Biomedical Imaging and Bioengineering under Grant R01EB015853.

References

[1] D. J. Cook, N. Krishnan, and P. Rashidi, "Activity discovery and activity recognition: A new partnership," *IEEE Transactions on Systems, Man, and Cybernstics, Part B*, vol. 43, no. 3, pp. 820–828, 2013.

[2] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001, pp. 282–289.

[3] Hal Daumé III, J. Langford, and D. Marcu, "Search-based structured prediction," *MLJ*, vol. 75, no. 3, pp. 297–325, 2009.

[4] J. R. Doppa, A. Fern, and P. Tadepalli, "Structured prediction via output space search," *JMLR*, vol. 15, pp. 1317–1350, 2014.

[5] ——, "HC-Search: A learning framework for search-based structured prediction," *JAIR*, vol. 50, pp. 369–407, 2014.

[6] C. Ma, J. R. Doppa, W. Orr, P. Mannem, X. Fern, T. Dietterich, and P. Tadepalli, "Prune-and-Score: Learning for greedy coreference resolution," in *EMNLP*, 2014.

[7] J. Xie, C. Ma, J. R. Doppa, P. Mannem, X. Fern, T. Dietterich, and P. Tadepalli, "Learning greedy policies for the easy-first framework," in *AAAI*, 2015.

[8] M. Lam, J. R. Doppa, S. Todorovic, and T. Dietterich, "Learning to detect basal tubules of nematocysts in sem images," in *ICCV Workshop on Computer Vision for Accelerated Biosciences*, 2013.

[9] ——, "HC-Search for structured prediction in computer vision," in *CVPR*, 2015.

[10] R. Khardon, "Learning to take actions," *MLJ*, vol. 35, no. 1, pp. 57–90, 1999.

[11] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011.

[12] M. Kääriäinen, "Lower bounds for reductions," in *Atomic Learning Workshop*, 2006.

[13] N. Krishnan and D. J. Cook, "Activity recognition on streaming sensor data," *Pervasive and Mobile Computing*, vol. 10, pp. 138–154, 2014.

[14] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys*, vol. 46, pp. 107–140, 2015.

[15] O. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communication Survey Tutorials*, vol. 15, pp. 1195–1209, 2013.

[16] N. Yala, B. Fergani, and A. Fleury, "Feature extraction for human activity recognition on streaming data," in *International Symposium on Innovations in Intelligence Systems and Applications*, 2015, pp. 1–6.

[17] Y. Zheng, W.-K. Wong, X. Guan, and S. Trost, "Physical activity recognition from accelerometer data using a multi-scale ensemble method," in *Proceedings of the Innovative Applications of Artificial Intelligence Conference*, 2013, pp. 1575–1581.

[18] J.-H. Hong, J. Ramos, and A. K. Dey, "Toward personalized activity recognition systems with a semipopulation approach," *IEEE Trans. Human-Machine Syst.*, vol. 46, no. 1, pp. 101–112, 2016.

[19] T. Barger, D. Brown, and M. Alwan, "Health status monitoring through analysis of behavioral patterns," *IEEE Trans. Syst. Man, Cybern. Part A*, vol. 35, no. 1, pp. 22–27, 2005.

[20] J. K. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Comput. Surv.*, vol. 43, no. 3, pp. 1–47, 2011.

[21] S. Ke, H. Thuc, Y. Lee, J. Hwang, J. Yoo, and K. Choi, "A review on video-based human activity recognition," *Computers*, vol. 2, no. 2, pp. 88–131, 2013.

[22] A. Reiss, G. Hendeby, and D. Stricker, "Towards Robust Activity Recognition for Everyday Life : Methods and Evaluation," *Proceedings of 7th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, no. May, pp. 25–32, 2013.

[23] S. Vishwakarma and A. Agrawal, "A survey on activity recognition and behavior understanding in video surveillance," in *Visual Computer*, vol. 29, no. 10, 2013, pp. 983–1009.

[24] V. G. Wadley, O. Okonkwo, M. Crowe, and L. A. Ross-Meadows, "Mild cognitive impairment and everyday function: Evidence of reduced speed in performing instrumental activities of daily living," *The American Journal of Geriatric Psychiatry*, vol. 15, pp. 416–424, 2008.

[25] B. Minor, J. R. Doppa, and D. J. Cook, "Data-driven activity prediction: Algorithms, evaluation methodology, and applications," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: ACM, 2015, pp. 805–814. [Online]. Available: http://doi.acm.org/10.1145/2783258.2783408

[26] K. E. Heron and J. M. Smyth, "Ecological momentary interventions: Incorporating mobile technology into psychosocial and health behavior treatment," *Journal of Health Psychology*, vol. 15, pp. 1–39, 2010.

[27] S. Shiffman, A. a. Stone, and M. R. Hufford, "Ecological momentary assessment," *Annual review of clinical psychology*, vol. 4, no. 1, pp. 1–32, 2008.

[28] K. L. Gwet, *Handbook of Inter-Rater Reliability*. Advanced Analytics, LLC, 2014.

[29] J. Boger and A. Mihailidis, "The future of intelligent assistive technologies for cognition: Devices under development to support independent living and aging-with-choice," *NeuroRehabilitation*, vol. 28, no. 3, pp. 271–280, 2011.

[30] M. W. Bewernitz, W. C. Mann, P. Dasler, and P. Belchior, "Feasibility of Machine-based Prompting to Assist Persons With Dementia," *Assistive Technology*, vol. 21, no. 4, pp. 196–207, 2009.

[31] N. Epstein, M. G. Willis, C. K. Conners, and D. E. Johnson, "Use of technological prompting device to aid a student with attention deficit hyperactivity disorder to initiate and complete daily activities: An exploratory study," *Journal of Special Education Technology*, vol. 16, pp. 19–28, 2001.

[32] M. Schmitter-Edgecome, S. Pavawalla, J. T. Howard, L. Howell, and A. Rueda, *Dyadic interventions for Persons with Early-Stage Dementia: A Cognitive Rehabilitative Focus*. Nova Science Publishers, 2009, ch. 3, pp. 39–56.

[33] J. Hoey, P. Poupart, A. von Bertoldi, T. Craig, C. Boutilier, and A. Mihailidis, "Automated handwashing assistance for persons with dementia using video and a partially observable Markov decision process," *Computer Vision and Image Understanding*, vol. 114, pp. 503–519, 2010.

[34] T. Hart, R. Buchhofer, and M. Vaccaro, "Portable electronic devices as memory and organizational aids after traumatic brain injury: A consumer survey study," *Journal of Head Trauma Rehabilitation.*, vol. 19, no. 5, pp. 351–365, 2004.

[35] B. Wilson, J. Evans, H. Emslie, and V. Malinek, "Evaluation of NeuroPage: a new memory aid," *Journal of Neurology, Neurosurgery, and Psychiatry*, vol. 63, no. 1, pp. 113–115, 1997.

[36] T. L. Hayes, K. Cobbinah, T. Dishongh, J. a. Kaye, J. Kimel, M. Labhard, T. Leen, J. Lundell, U. Ozertem, M. Pavel, M. Philipose, K. Rhodes, and S. Vurgun, "A study of medication-taking and unobtrusive, intelligent reminding." *Telemedicine journal and e-health : the official journal of the American Telemedicine Association*, vol. 15, no. 8, pp. 770–6, 2009.

[37] P. Kaushik, S. S. Intille, and K. Larson, "User-adaptive reminders for home-based medical tasks: A case study," *Methods of Information in Medicine*, vol. 47, pp. 203–207, 2008.

[38] J. S. Weber and M. E. Pollack, "Evaluating user preferences for adaptive reminding," *Human Factors in Computing Systems*, pp. 2949–2954, 2008.

[39] S. Vurgun, M. Philipose, and M. Pavel, "A statistical reasoning system for medication prompting," in *International Conference on Ubiquitous Computing*, 2007, pp. 1–18.

[40] J. Modayll, R. Levinson, C. Harman, D. Halper, and H. Kautz, "Integrating sensing and cueing for more effective activity reminders," *AI in Eldercare*, 2008.

[41] T. Sohn, K. Li, G. Lee, and I. Smith, "Place-its: A study of location-based reminders on mobile phones," in *UbiComp'05 Proceedings of the 7th international conference on Ubiquitous Computing*, 2005, pp. 232–250.

[42] G. E. Lancioni, R. Coninx, N. Manders, M. Driessen, J. V. Dijk, and T. Visser, "Reducing breaks in performance of multihandicapped students through automatic prompting or peer supervision," *J. Dev. Phys. Disabil.*, vol. 3, pp. 115–128, 1991.

[43] H. H. Hsu, C. N. Lee, and Y. F. Chen, "An RFID-based reminder system for smart home," in *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, 2011, pp. 264–269.

[44] H. Ferguson, B. S. Myles, and T. Hagiwara, "Using a Personal Digital Assistant to Enhance the Independence of an Adolescent with Asperger Syndrome," *Education And Training*, vol. 40, no. 6, pp. 60 – 67, 2011.

[45] A. M. Seelye, M. Schmitter-Edgecombe, B. Das, and D. J. Cook, "Application of cognitive rehabilitation theory to the development of smart prompting technologies," pp. 29–44, 2012.

[46] J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis, "A decision-theoretic approach to task assistance for persons with dementia," in *International Joint Conference on Artificial Intelligence*, 2005, pp. 1293–1299.

[47] H.-T. Cheng, M. Griss, P. Davis, J. Li, and D. You, "Towards zero-shot learning for human activity recognition using semantic attribute sequence model," in *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2013, pp. 355–358.

[48] J. Seiter, O. Amft, M. Rossi, and G. Troster, "Discovery of activity composites using topic models: An analysis of unsupervised methods," *Pervasive and Mobile Computing*, vol. 15, pp. 215–227, 2014.

[49] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *International Conference on Human Behavior Understanding*, 2011, pp. 29–39.

[50] P. Rashidi, D. J. Cook, L. Holder, and M. Schmitter-Edgecombe, "Discovering activities to recognize and track in a smart environment," *IEEE TKDE*, vol. 23, no. 4, pp. 527–539, 2011.

[51] K. Gopalratnam and D. J. Cook, "Online sequential prediction via incremental parsing: The active lezi algorithm," *IEEE Intelligent Systems*, vol. 22, pp. 52–58, 2007.

[52] K. P. Hawkins, N. Vo, S. Bansal, and A. Bobick, "Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration," in *IEEE-RAS International Conference on Humanoid Robots*, 2013, pp. 499–506.

[53] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *Proceedings of the European Conference on Computer Vision*, 2012.

[54] H. S. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," in *Robotics: Sciences and Systems*, 2013.

[55] J. Boger, J. Hoey, P. Poupart, C. Boutilier, G. Fernie, and A. Mihailidis, "A planning system based on Markov decision processes to guide people with dementia through activities of daily living." *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 10, no. 2, pp. 323–333, 2006.

[56] B. Das, D. J. Cook, N. Krishnan, and M. Schmitter-Edgecombe, "One-class classification-based real-time activity error detection in smart homes," *IEEE J. Sel. Top. Signal Process.*, 2016.

[57] A. Mihailidis, J. N. Boger, T. Craig, and J. Hoey, "The COACH prompting system to assist older adults with dementia through handwashing: an efficacy study." *Geriatrics*, vol. 8, no. 1, pp. 28–46, 2008.

[58] T. Gu, S. Chen, X. Tao, and J. Lu, "An unsupervised approach to activity recognition and segmentation based on object-use fingerprints," *Data and Knowledge Engineering*, vol. 69, no. 6, pp. 533–544, 2010.

[59] P. Palmes, H. K. Pung, T. Gu, W. Xue, and S. Chen, "Object relevance weight pattern mining for activity recognition and segmentation," *Pervasive and Mobile Computing*, vol. 6, no. 1, pp. 43–57, 2010.

[60] I. Cleland, M. Han, C. Nugent, H. Lee, S. McClean, S. Zhang, and S. Lee, "Evaluation of prompted annotation of activity data recorded from a smart phone," *Sensors (Switzerland)*, vol. 14, no. 9, pp. 15 861–15 879, 2014.

[61] K. D. Feuz, D. J. Cook, C. Rosasco, K. Robertson, and M. Schmitter-Edgecombe, "Automated Detection of Activity Transitions for Prompting," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 5, pp. 575–585, 2015.

Bryan Minor received his B.S. and Ph.D. from Washington State University. He is a research associate at Washington State University. His research interests include activity prediction, smart environments, and human-computer interaction.

Janardhan Rao Doppa received his M.Tech. from IIT Kanpur and his Ph.D. from the Oregon State University. He is an Assistant Professor at Washington State University. His research interests include artificial intelligence, machine learning, and data-driven science and engineering. His work on structured prediction received an outstanding paper award at the AAAI 2013 conference.

Diane J. Cook received her B.A. from Wheaton College and her M.S. and Ph.D. from the University of Illinois. She is a Huie-Rogers Chair Professor at Washington State University. Her research interests include machine learning, smart environments, and automated health assessment and intervention.