One-Class Classification-Based Real-Time Activity Error Detection in Smart Homes

Barnan Das*, Diane J. Cook†, *Fellow, IEEE*, Narayanan C. Krishnan‡, and Maureen Schmitter-Edgecombe§
*Intel Corporation, Santa Clara, CA 95054. barnandas@gmail.com
†School of Electrical Engineering and Computer Science, Washington State University. cook@eecs.wsu.edu
‡Department of Computer Science and Engineering, Indian Institute of Technology, Ropar, India. ckn@iitrpr.ac.in
§Department of Psychology, Washington State University. schmitter-e@wsu.edu

***Abstract*—Caring for individuals with dementia is frequently associated with extreme physical and emotional stress, which often leads to depression. Smart home technology and advances in machine learning techniques can provide innovative solutions to reduce caregiver burden. One key service that caregivers provide is prompting individuals with memory limitations to initiate and complete daily activities. We hypothesize that sensor technologies combined with machine learning techniques can automate the process of providing reminder-based interventions. The first step towards automated interventions is to detect when an individual faces difficulty with activities. We propose machine learning approaches based on one-class classification that learn normal activity patterns. When we apply these classifiers to activity patterns that were not seen before, the classifiers are able to detect activity errors, which represent potential prompt situations. We validate our approaches on smart home sensor data obtained from older adult participants, some of whom faced difficulties performing routine activities and thus committed errors.**

***Index Terms*—Smart homes, machine learning, activity recognition, one-class classification**

I. Introduction

The world's population is aging, with the estimated number of individuals over the age of 85 expected to triple by 2050 [1]. The increase in the number of individuals crossing higher life expectancy thresholds has made a large section of the older population susceptible to cognitive impairments such as Alzheimer's disease and dementia. An estimated 5.5 million Americans have Alzheimer's disease in 2014. By 2050, the global number of people who are 65 and older with some form of cognitive impairment may nearly triple.

There are currently 15.5 million [2] family and friends providing unpaid care to those with Alzheimer's and other dementias. These individuals face extreme physical and emotional stress due to caregiving, resulting in depression. Therefore, we must consider innovative technology-driven solutions to reduce caregiver burden. Moreover, technology-driven long-term care facilities in individual homes can provide a low-cost alternative to spending substantial amount of time and money in hospitals, nursing homes or health clinics.

With recent advances in sensor technologies and machine learning, it is possible to transform a regular home into a *smart home* with bare minimum infrastructural modifications and at a reasonable cost [3]. We hypothesize that solutions derived from smart home technologies can reduce caregiver burden and provide long-term low cost care facilities in individuals' homes. Studies have shown that smart environment technologies can detect errors in activity completion and might be utilized to extend independent living in one's own home without compromising safety [4], [5]. One type of intervention that is valuable for individuals with cognitive impairment is automated prompts that aid with activity initiation and completion. To date, most applications of the smart home technology for intervention have relied upon partial or full expert design of prompting rules [6]. Our approach to automating prompting-based intervention for activity completion is to predict activity errors in real time from streaming sensor data in a smart home while an individual performs everyday activities. In our previous work [7], we designed machine learning algorithms that map activity steps to "prompt" and "no-prompt" labels based on training data provided by actual experimenters who provided needed prompts to the participants. However, this approach assumes that the streaming sensor data collected from a smart home is labeled with the corresponding activity and sub-activity (for example, retrieving a broom from a supply closet is a sub-activity step for the *Sweeping and Dusting* activity). To address this limitation, we propose an unsupervised approach to error detection by utilizing one-class classification-based algorithms that do not require training data for the activity errors.

The proposed approach, DERT, or **D**etecting Activity **E**rrors in **R**eal **T**ime, is trained only on the activity data of the participants for whom the experimenters reported no errors. Data that do not contain activity errors comprise the *target* class for an activity. The remainder of the data for the same activity from the participants who committed errors, also called the *test samples*, are used to validate the efficacy of DERT in detecting activity errors in real time. As with most outlier detection techniques, DERT detects the outliers in the test samples. For an automated prompting application these outliers can be considered as activity errors. We use the sensor data from 580 participants who were part of two smart home studies. Statistical features that are capable of capturing activity errors are extracted from the raw sensor data before training our algorithms. We also postulate that including information about activity errors could improve the performance of our proposed method. In order to further improve error detection, we utilize

knowledge gathered from classifier-identified errors to build ensemble-based error classifiers.

DERT assumes that activity labels are available in real time. This assumption is based on the activity recognition algorithm proposed by Krishnan and Cook [8]. This algorithm has been tested on 12 daily activities that are similar in nature to the ones analyzed in this paper (e.g., cook, eat, clean) but are performed in an *unscripted fashion in actual smart homes* with an average accuracy over 20 real-world smart homes of 99%. In our current experiments, we use activity labels collected from human annotators. In the future, these will be provided by the real-time activity recognizer. As most of the one-class classification features require the activity start time, we assume that the beginning of the activities can be successfully detected using the activity recognition algorithm. In addition, we also assume that the activities are performed when the smart home has one resident and the activities are not interleaved or concurrent. The contribution of this paper is not another activity recognizer or a new one-class SVM. The contribution is activity error detection with the assumption that activity recognition works reliably well in real time. We use Scholkopf's one-class SVM after performing statutory parameter tuning to make it suitable for our application.

II. Related work

Intelligent prompting technologies have been in existence for quite some time [9]. Existing solutions can be broadly classified into four categories that are based on: rules (time and context) [10], reinforcement learning [11], planning and supervised learning [12], [13] Many existing prompting technologies are designed for activity initiation. Automated prompting at the granularity of sub-activities (activity steps), however, is a harder problem. Automated prompting in real-time on streaming sensor data is even harder. There are only a few research groups who have worked on sub-activity prompting. Most of these solutions are either based on video analysis [14], [15] or monitoring interactions with a variety of objects of daily use [16], [17] in the smart home using sensors, such as RFID tags.

Hoey et al. [15] proposed a real-time computer vision system to assist dementia patients with hand washing. The proposed approach combines a Bayesian sequential estimation framework for tracking hands and towel. A decision theoretic framework is used for computing policies of all possible actions. The decision policies which dictate system actions are computed using a POMDP using a point-based approximate solution. The tracking and decision making systems are coupled using a heuristic method for temporally segmenting the input video stream based on the continuity of the belief state. This approach performs prompts well for appropriate activity steps. However, it can cause major privacy concerns to the participants due to the use of video input. Moreover, due to reliance of this technique on video analysis, it is not easily generalizable to other activities of daily living.

On the other hand, some approaches [16], [17], [18] use sensor platforms that can provide deep and precise insight into sub-activity steps. For example, usage of RFID tags with smart home objects of daily use is a very common strategy to gather a 1:1 mapping between sensors and activity steps. Some of these approaches use complex plan recognition models for predicting the probability of a action for a state when the activity is modeled as a stochastic process such as a Markov chain.

To evaluate the effectiveness of technology-based prompts, Seelye et al. [19] experimented with a hierarchy of audio, video, direct, and indirect prompts to investigate the type of prompts that are effective for assisting individuals with mild cognitive impairment, in completing routine activities. It was found in this study that the technology-based prompts enabled participants to correct critical errors in activities and get back on track. Moreover, user feedback from participants indicated that in general they perceived the prompting technology to be very helpful and appropriately timed.

In our smart home infrastructure, we avoid using any sensor technology that represents a potential threat to privacy. This, however, causes major hurdles in sensing fine-grained activity information. On the other hand, we are motivated to propose an efficient, non-intrusive and generic solution to the automated sub-activity level prompting problem by automatically detecting activity errors. To the best of our knowledge, this problem with the given infrastructural limitations has not been addressed in the past. The same machine learning-based techniques are applicable for any type of sensor-based data, including data from wearable sensors, smart phone sensors, and other sensor platforms.

III. Problem analysis

To determine the ability of machine learning algorithms to detect activity errors, we performed two studies in our smart home test bed with community-dwelling older adults. Our on-campus smart home is equipped with a variety of sensors. Passive infrared motion detectors are used to collect the location information of the participant in the apartment. Magnetic door sensors monitor open and close events of doors, closets, cabinets, microwave and refrigerator. Pressure-based item sensors monitor use of household objects such as water cup, cup noodles and medication dispenser. Accelerometer-based vibration sensors are used to track the use of household objects such as broom, dustpan and brush. In addition, other sensors to measure ambient light level, temperature, water and power use are also used. The sensors are wireless and utilize a Control4 ZigBee wireless mesh network [8].

Every sensor in the smart home logs time stamp, sensor ID and current state, whenever there is a change of its state. For example, when a cabinet door is opened, the door sensor associated with it logs an OPEN state into the database. The sensor logs are stored in a database. Table I shows an example of raw sensor data collected in the smart home.

Clinically-trained psychologists watch over a web camera as the older adult participants perform activities of daily living, such as the following: **Study 1:** Sweeping and Dusting, Medication Dispenser, Writing Birthday Card, Watching DVD,

Data collected from the smart home studies are available at `http://casas.wsu.edu/datasets.html`.

TABLE I
HUMAN ANNOTATED SENSOR EVENTS.

Time	Sensor	Message	Activity	Error
14:59:54.934979	D010	CLOSE	cooking	
14:59:55.213769	M017	ON	cooking	
15:00:02.062455	M017	OFF	none	
15:00:17.348279	M017	ON	cooking	*Error*
15:00:34.006763	M018	ON	cooking	
15:00:35.487639	M051	ON	cooking	

Watering Plants, Answering Phone Cal, Cooking, Selecting Outfit; **Study 2:** Sweeping and Dusting, Cleaning Countertops, Medication Dispenser, Watering Plants, Washing Hands, Cooking. These activities represent instrumental activities of daily living (IADLs) [20] that can be disrupted in Mild Cognitive Impairment (MCI) and can help discriminate healthy aging from MCI [21].

Participants are instructed how to execute an activity. For example, in the *Sweeping and Dusting* activity, the participants were instructed to retrieve a broom, dustpan/brush, and duster from a specified closet and then to sweep the kitchen floor and dust the furniture in the living and dining rooms. For activities such as *Writing Birthday Card*, the location of the activity is tracked with motion sensors and by vibration sensors attached with a box that contains stationaries to write a birthday card, such as pens and cards. During activity execution, the experimenters manually note the presence of an activity error and the corresponding time.

An external annotator determines the activity start and end as well as the timing of activity errors based the sensor data and experimenter-recorded error time and context. Table I shows a snippet of annotated sensor data. An error label is assigned to a single sensor event and indicates the beginning of an error. The error continuous until the participant detects and corrects the error. In this paper, we experiment with errors that could be identified by a human annotator from the sensor data. We focus our experiments to the following activities from Study 1: *Sweeping and Dusting*, *Medication Dispenser*, *Watering Plants*, and *Cooking*; and all activities from Study 2. We note that cameras are only used in our on-campus smart home test bed to record ground truth information, which in turn is only used to train our machine learning models. Once the algorithm is deployed in real homes, the activity recognition and automated prompting systems would work without any human intervention.

IV. DETECTING ACTIVITY ERRORS IN REAL TIME - PROPOSED APPROACH

Activity errors can occur in a variety of ways. Modeling each separate class of errors is impractical. Therefore, we formulate the problem of activity error detection as an outlier detection problem. For a specific activity, the sensor data from participants for whom no errors were reported by the psychology experimenters are used to train the target class. Sensor data for the same activity from participants who made errors is used as the test data to evaluate the efficacy of the proposed approach in accurately detecting activity errors. When a participant performs an activity, the algorithm labels every sensor event occurring in the smart home either as "normal" or "error". Therefore, a data sample in our methodology corresponds to a single sensor event and is represented by a corresponding feature vector. We extract temporal and contextual features from the sensor events. All data samples belonging to the target class are labeled as "normal". However, the test samples obtained from the sensor data of the participants who committed errors, have both "normal" and "error" labels. Specifically, a test sample corresponding to a sensor event that is labeled as the beginning of an error state, is labeled as "error".

In the following sections, we provide details of our proposed feature extraction and error detection algorithms. Our error detection methods start with a simple outlier detection algorithm and then evolve into more complex algorithms by adding additional features and creating ensembles.

A. Feature extraction

We extract statistical features from sensor events that capture contextual information while an activity is performed. The features can be broadly classified into *longitudinal* and *cross-sectional* features. Longitudinal features are calculated from the sensor data of an individual participant collected from the beginning of the activity. On the other hand, cross-sectional features are derived from data across a set of participants. A feature vector, comprising the features listed below, are generated for every sensor event of an activity. An event, e_j, where $1 < j < z$ and $j \in \mathbb{Z}^+$, represents a time step from the beginning of an activity performed by a participant. That is, while e_1 corresponds to the first event of an activity, e_z corresponds to the last event.

1) Longitudinal features:

*a) **Sensor Identifier**:* A nominal feature that uniquely identifies the sensor associated with the current sensor event in the smart home. This feature gives an idea of the sensors that are commonly triggered for a particular activity, as indicated by the target class data instances.

*b) **Event Pause**:* A temporal feature that represents the time elapsed (in milliseconds) between the previous and current sensor events.

*c) **Sensor Pause**:* A temporal feature that represents the time elapsed (in milliseconds) since the last event that was generated by the same sensor and the current event.

*d) **Sensor Count**:* The contextual information of a participant's progress in an activity is obtained by keeping a count of the number of times each sensor generated an event, from the beginning of the activity up through the current sensor event. This information is stored in the multi-dimensional feature *Sensor Count*, where every dimension corresponds to a specific sensor. For an event e_j in a particular activity, if l represents the total number of sensors associated with the activity, *Sensor Count* is the number of times sensors s_i (where $i = 1...l$) was triggered until the current event.

2) Cross-sectional features:

*a) **Support**:* Not all sensors in a smart home are triggered frequently for a particular activity. If a sensor, that appears infrequently in normal occurrences of an activity, appears

when a participant performs that activity, it could possibly indicate an activity error. This property is captured by *support*. Thus, for a specific activity, support is the ratio between the number of participants that triggered a specific sensor, and the total number of participants. Thus,

$$support(s_i) = \frac{\text{Number of participants that triggered sensor } s_i}{\text{Total number of participants}} \quad (1)$$

*b) **Probability of Event Number**:* This feature represents the probability of the elapsed time from the activity beginning until the current event. We assume that the amount of time participants spend from the beginning of an activity to a certain event number e_j, represented as $time_{e_j}$, is distributed normally across all the participants. Thus,

$$P(e_j) = \frac{\text{Number of participants who reached event } e_j}{\text{Total number of participants}} \quad (2)$$

*c) **Probability of Event Time**:* This feature also represents the probability of elapsed time from activity beginning until the current event. We assume that the amount of time participants spend from the beginning of an activity to a certain event number e_j, represented as $time_{e_j}$, is distributed normally across all the participants. Therefore, *Probability of Event Time* can be represented as follows:

$$P(time_{e_j}; \mu_{e_j}, \sigma^2_{e_j}) = \frac{1}{\sigma^2_{e_j}\sqrt{2\pi}} exp\left(-\frac{(time_{e_j} - \mu_{e_j})^2}{2\sigma^2_{e_j}}\right) \quad (3)$$

*d) **Poisson Probability of Sensor Count**:* The longitudinal feature *Sensor Count* captures the count of events for all sensors through the current event. However, as the training data representing the target class samples consists of feature vectors for all sensor events of the participants, there is a very high variance in the Sensor Count feature values. This is because the feature vectors representing sensor events at the beginning of an activity with low sensor event count are put together in the same target class with feature vectors corresponding to sensor events towards the end of the activity that have high sensor event count. Therefore, we need additional features that take the event number into consideration, as the event number indicates the progress in the activity. As the sensor counts are discrete, we assume that the counts of events of the sensors until a certain event across all the participants follows a Poisson distribution. *Poisson Probability of Sensor Count* represents the probabilities of the counts of events for all sensors for a specific event number calculated using the Poisson probability mass function.

$$P(k_{s_i,e_j}; \lambda_{s_i,e_j}) = \frac{(\lambda_{s_i,e_j})^{k_{s_i,e_j}}}{(k_{s_i,e_j})!} exp(-\lambda_{s_i,e_j}) \quad (4)$$

where, k_{s_i,e_j} is the count for sensor s_i at event e_j, λ_{s_i,e_j} is the mean of counts for sensor s_i at event e_j for all participants, and i belongs to the set of all relevant sensors for an activity.

B. Baseline

DERT detects activity errors on streaming sensor data where a data sample at time t has a strong correlation with the sample at time $(t-1)$. As we take an outlier detection based approach to predict errors, the streaming aspect of the sensor data should be taken into consideration. No standard outlier detection algorithm fits this criterion. Therefore, we design a simple error detection algorithm that we refer to as *Baseline*, to provide a basis for comparison with alternative approaches.

Algorithm 1: Baseline

Train:

Assume that the sensor counts follow a Poisson distribution, i.e., $x_i \sim Pois(\lambda_i)$ and are conditioned over event e_j

Estimate λ_i for $i = 1$ to l and all events e_j

Test:

for every data sample $\mathbf{x} = \{x_1, ..., x_l\}$ **do**

  for i=1 to l **do**

    Find the probability of a sensor count using the Poisson probability mass function $P(x_i; \lambda_i) = \frac{\lambda_i^{x_i}}{(x_i)!} exp(-\lambda_i)$

  end for

  Find the *log* of the product of $P(x_i; \lambda_i)$ for all i:

$$log(P(\mathbf{x})) = log\left(\prod_{i=1}^{l} P(x_i; \lambda_i)\right) = log\left(\prod_{i=1}^{l} \frac{\lambda_i^{x_i}}{(x_i)!} exp(-\lambda_i)\right)$$
$$= \sum_{i=1}^{l} \frac{\lambda_i^{x_i}}{(x_i)!} exp(-\lambda_i)$$

  if $log(P(\mathbf{x})) > \tau$ **then**

    $\mathbf{x} \in$ *target* class

  else

    $\mathbf{x} \in$ *outlier* class

  end if

end for

Fig. 1. Baseline algorithm for outlier detection.

Baseline uses only the *Sensor Counts* features corresponding to sensor events to detect activity errors. While training the target class, using the activity data of the participants who did not commit any errors, the data samples corresponding to an event, e_j, are assumed to follow a multivariate Poisson distribution across all participants. We estimate the Poisson distribution parameter, λ, corresponding to all l sensors of an activity, represented as $\lambda_{e_j} = \{\lambda_1, ..., \lambda_l\}$. A test sample, $\mathbf{x}$, obtained from the activity data of those participants who committed errors, is evaluated on the basis of the probability derived from the Poisson probability mass function:

$$P(x_i; \lambda_i) = \frac{\lambda_i^{x_i}}{(x_i)!} exp(-\lambda_i), \text{ where } 1 < i < l \quad (5)$$

If the product of all probabilities, $\prod_{i=1}^{l} P(x_i; \lambda_i)$, is less than an empirically derived threshold τ, then the test sample $\mathbf{x}$ is classified as an error; otherwise, it belongs to the target class. The threshold corresponds to the maximum F_1-score optimized on a validation set that consists of both target and outlier class samples. The Baseline approach is illustrated algorithmically in Figure 1.

C. One-class support vector machine

We use a one-class support vector machine for our second method. The OCSVM, one-class support vector machine, proposed by Schölkopf [22] is the core component of this

approach. OCSVM finds a hyper-plane that separates all target class samples from the origin with maximum margin during training. Given n target class samples $\mathbf{x} = \{x_1, x_2, ..., x_n\}$, the objective is to separate the dataset from the origin by solving the following quadratic program:

$$\min_{w \in F, \xi \in \mathbb{R}^n, \rho \in \mathbb{R}} \frac{1}{2} \parallel w \parallel^2 + \frac{1}{\nu n} \sum_i \xi_i - \rho \quad \text{subject to } (w \cdot \Phi(\mathbf{x}_i)) \geq \rho - \xi_i,\ \xi_i \geq 0 \tag{6}$$

Here, $\nu \in (0, 1]$ is an upper bound on the fraction of outliers and a lower bound on the fraction of support vectors (SV). More explanation on the role of ν is offered later. $\Phi : \mathcal{X} \longrightarrow F$ is a kernel map which transforms the training samples to a high-dimensional feature space and (w, ρ) represents the kernel's weight vectors and offset parameterizing a hyperplane in the feature space. Non-zero slack variables ξ_i are penalized in the objective function. Therefore, if w and ρ solve this problem, then the decision function

$$f(\mathbf{x}) = sgn((w \cdot \Phi(\mathbf{x}) - \rho) \tag{7}$$

will be positive for most examples x_i contained in the training set. Because many non-linear problems may be linearly separable after proper transformations, kernel transformations $\Phi(\cdot)$ are normally employed to transfer an input sample from one feature space to another. Φ is a feature map $\mathcal{X} \longrightarrow F$ into a high-dimensional feature space F such that the inner product of the transformed samples in F can be computed by evaluating a simple kernel function

$$k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})), \tag{8}$$

such as the Gaussian kernel

$$k(\mathbf{x}, \mathbf{y}) = exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{c}\right) \tag{9}$$

For new sample $\mathbf{x}_t$, if $f(\mathbf{x}) > 0$, it is classified as a target class sample, otherwise it is regarded as an outlier.

According to Schölkopf's proposition [22], if the solution of Equation 6 satisfies $\rho \neq 0$, the following statements hold true for the parameter ν:

1) ν is an upper bound on the fraction of outliers
2) ν is a lower bound on the fraction of SVs
3) If the data $\{x_1, ...x_n\}$ is generated independently from a distribution $P(\mathbf{x})$ which does not contain discrete components, and if the kernel is analytic and non-constant, with probability 1, asymptotically, ν equals both the fraction of SVs and the fraction of outliers.

The parameter ν plays an important role in determining what fraction of the test data, with both target class and outlier samples, will be classified as outliers. In our application domain, the fraction of outliers defines the false positive rate of OCSVM in predicting errors or prompt situations in an activity. A larger fraction of outlier would mean more frequent interventions, which limits the practicality of the system. Therefore, a balance between outlier and target sample prediction is desired. The fraction of outliers on the test data is also dependent on the kernel parameter $\gamma = \frac{1}{c}$. We perform experiments on all activities separately to choose appropriate values for ν and γ.

D. Activity error classification and ensembles

Currently, there is a limited understanding of the course of functional change that occurs between normal aging and dementia [23]. To better characterize the nature of this change, it is important to evaluate the error types. We coded activity errors for Studies 1 and 2 based on previously-published criteria [24] and categorized them into 4 and 9 different types, respectively. For example, a *Substitution* error is coded when an alternate object, or a correct object but an incorrect gesture, is used and disrupts accurate completion of the activity. Dusting the kitchen instead of the living room is an example of a *Substitution* error for the *Sweeping and Dusting* activity. Descriptions of the error types are given in Table II. Note that the error types in Study 2 in some cases represent more detailed versions of the error categories used for Study 1.

We use smart home sensor data to classify the errors into different error types. The distribution of error data samples among all the error types is given in Table II. Classifiers such as decision trees, naive Bayes, nearest neighbors, logistic regression and support vector machine are used in our experiments. The results are described in Section V-C.

TABLE II
DISTRIBUTION OF ERROR DATA SAMPLES ON ACTIVITY ERROR TYPES.

Study 1		
Error Type	**Description**	**#samples**
Omission	When a step or subtask necessary for accurate task completion is not performed	71
Substitution	When an alternate object, or a correct object but an incorrect gesture, is used and disrupts accurate activity completion	7
Irrelevant action	When an action that is unrelated to the activity, and completely unnecessary for activity completion, is performed	9
Inefficient action	When an action that slows down or compromises the efficiency of task completion is performed	125
Study 2		
Substitution	When an alternate object, or a correct object but an incorrect gesture, is used and disrupts accurate completion of the activity	179
Additional Task Related Activity	When a participant is engaging in a task that appears to be task or goal related, but might not be necessary	214
Additional Non-Task Related Activity	When a participant engages in an activity that is not related to the goal of the overall task	15
Perseveration	When a participant engages in task after it has been completed	19
Searching	When a participant is actively searching through cupboards/drawers/closets for an item	110
Help	When participant asks the experimenter a question or for help on a task	6
Wandering	When a participant is wandering around apartment and appears directionless	2
Microslip	When a participant initiates and terminates an incorrect action before the error is completed	14
Error Detection	When a participant completes an error, recognizes that she made an error and rectifies it before the end of the task	22

We also explore if the knowledge of error types can be harnessed to boost the performance of OCSVM in accurately predicting the errors or outliers. Here, the knowledge of error types together with normal data samples is used to learn an ensemble that consists of OCSVM and the error classification model. We use two methodologies to build this error classification model. The first method considers all error samples from different activities to belong to a single class and a one-class SVM is trained on this error class. The second method considers the error samples to have error-type class labels and a multi-class classifier is trained on all the error samples. Thus, we introduce two ensemble techniques:

OCSVM + OCEM This technique is an ensemble of OCSVM trained on normal activity data and OCEM, **O**ne-**C**lass **E**rror **M**odel, which is a second OCSVM trained on only error data samples. The model selection for OCEM is done in the same way as OCSVM. That is, the ν and γ parameters are determined empirically.

OCSVM + MCEM It is the ensemble of OCSVM trained on normal activity data and a classifier trained on MCEM, **M**ulti-**C**lass **E**rror **M**odel. Conventional multi-class classifiers output a distribution of membership probability of a test sample on all the classes the classifier is trained upon. We use this property of the classifiers to devise a technique that determines if a test sample belongs to any of the error classes, or if it is "out-of-vocabulary", that is, if the test sample is actually a normal data sample. This is done by assigning a threshold value on the membership probabilities. Specifically, if the membership probability of the test sample on any of the error classes is greater than the threshold, then the test sample is considered to belong to that class; else, the test sample is not an error. The threshold value of the membership probability is determined empirically from a randomly selected validation set. F_1-scores obtained by the ensemble is plotted against the threshold values between 0 and 1. The threshold value that gives the highest F_1-score of OCSVM+MCEM is chosen final threshold value.

The final class label of a test sample is decided as a logical "and" operation between the OCSVM and either of the classifiers for the error model.

E. Performance measures for evaluation

The performance of the proposed approach is evaluated using Recall, Precision and $F-1$-score. In addition, as this approach is intended to work on streaming sensor data in real-time, we need an idea of the temporal accuracy, that is, how far (in time) before or beyond an actual prompt situation, the algorithm predicted a prompt. There could be multiple prompt predictions in the temporal vicinity of the actual prompt as shown in Figure 2. For every actual prompt, we consider temporal accuracy to be the minimum between (a) the time difference between the actual prompt and the beginning of a series of prompt predictions before that, and (b) the time difference between the actual prompt and the beginning of a series of prompt predictions after that. That is, in Figure 2, the minimum between t_{prev} and t_{next} is considered as the temporal accuracy of the proposed algorithm. If the algorithm fails to do any prompt prediction for a particular participant, temporal accuracy is considered as the maximum between: (a) the time differences between the beginning of the activity to actual prompt, and (b) the time difference between the actual prompt to end of the activity. Temporal accuracy of the algorithm on all test participant data can be calculated by the root mean squared errors given below:

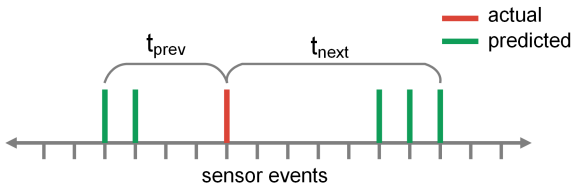

Fig. 2. Temporal accuracy.

- **RMSE(seconds)** Root mean square error of prompt prediction in seconds is calculated as $\sqrt{\frac{\sum_{t=1}^{n}(\textit{Predicted}\text{ clock time}-\textit{Actual}\text{ clock time})^2}{n}}$.
- **RMSE(events)** Root mean squared error of prompt prediction in events is calculated as $\sqrt{\frac{\sum_{t=1}^{n}(\textit{Predicted}\text{ event\#}-\textit{Actual}\text{ event\#})^2}{n}}$.

Here, n represents the total number of test participants.

V. Results

A. Model selection for OCSVM

The sensitivity of OCSVM on the outlier class is dependent on the parameters ν and γ. While ν is an upper bound on the fraction of outliers, γ is the inverse of kernel width c. Appropriate values for ν and γ are obtained by performing an empirical test on the training data with various values of ν and γ and monitoring the fraction of support vectors (SVs) chosen by the OCSVM. ν values are varied from 1.0 to 0.01, γ values are varied exponentially from the set $\{2^5, 2^3, ..., 2^{-1}, 2^{-3}, ..., 2^{-59}\}$. The point in the ν and γ plane, beyond which there is no significant decrease in the fraction of SVs, is chosen for the OCSVM activity model. Model selection for each activity is done independently.

B. Baseline vs OCSVM

We first compare the performance of OCSVM trained on "normal" activity data with the performance of Baseline in correctly predicting the activity errors. Figure 3 compares the performance of Baseline and OCSVM in terms of recall and precision on datasets available from the two studies. The results obtained from Study 1 indicate that OCSVM performs significantly better ($p < 0.05$) than Baseline in terms of recall on the *Sweeping and Dusting* and *Cooking* activities. The recall of OCSVM for the *Medication Dispenser* activity is at par with Baseline. This means that the temporal features such as Event Pause and Sensor Pause do not play any role in predicting *Medication Dispenser* activity errors. However, these features do play an important role in reducing the number of false positives, and thus precision of OCSVM on *Medication Dispenser* is better than Baseline. In fact, in terms

of precision, OCSVM performs significantly better ($p < 0.05$) than Baseline for all activities except *Cooking*.

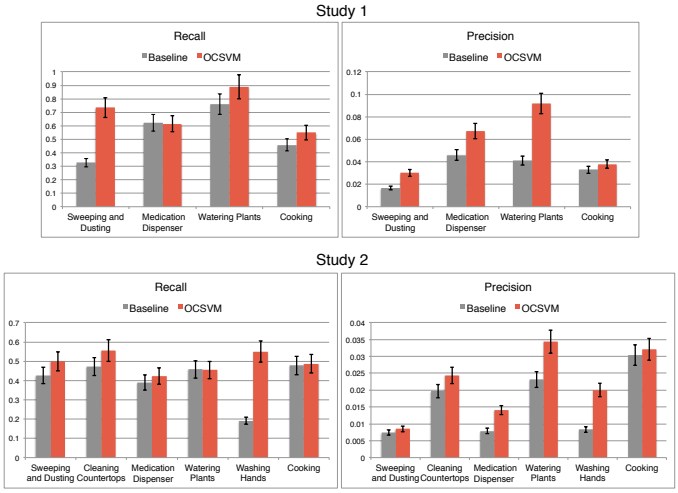

Fig. 3. (*left*) Recall and (*right*) precision values for tested activities.

In Study 2, OCSVM achieves statistically significant improvement over Baseline in terms of recall only for *Washing Hands* and *Cleaning Kitchen Countertops* activities. For the rest of the activities, the performance of OCSVM is almost at par with Baseline. The precision of OCSVM achieves significant improvement over Baseline for all classes except *Sweeping and Dusting* and *Cooking*.

From these observations we can conclude that, although Baseline does a fairly good job of predicting activity errors just on the basis of Poisson probability values for various sensors, low precision on all activities implies that Baseline gives a high false positive rate. The high false positive rate in our application domain implies a higher rate of intervention for the smart home residents, which might be unnecessary. The additional set of features that include temporal features and importance of a specific sensor in an activity (represented by *support*), play a very important role in reducing the number of false positives. Also, for some of the activities, these features help in predicting the activity errors better than Baseline.

In spite of the better performance of OCSVM over Baseline in terms of precision, the precision values are still quite low. This can be attributed to the extreme imbalance of class distribution between normal and outlier data samples. For example, only $\mathbf{0.67}\%$ of all test samples in the *Sweeping and Dusting* task from Study 2 are actual outliers. Therefore, the number of false positives is much higher than the number of true positives. As precision is calculated as $\frac{TP}{TP+FP}$, a high value of FP causes the precision to be low.

We also evaluate the proposed approaches in terms of temporal accuracy. This represents how close in time the OCSVM error prediction is to the actual error occurrence. Temporal accuracies, represented in RMSE values in seconds and number of sensor events, obtained by OCSVM and Baseline are compared in Figure 4. The results show that most of the errors predicted by OCSVM are in the window of 4 sensor events before or after the actual error. In terms of time, this is < 20 seconds before or after the actual error occurrence for most of the activities.

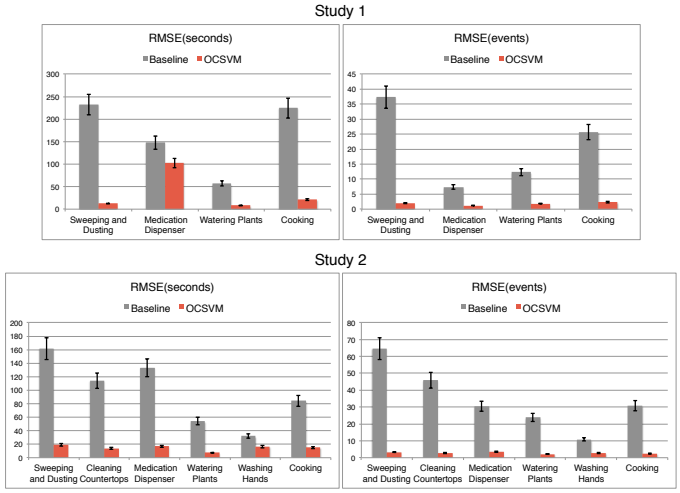

Fig. 4. RMSE values in (*left*) seconds and (*right*) number of events for tested activities.

C. Error classification

As mentioned earlier, successful classification of activity errors into predefined error types can provide better insight into the changes dementia brings into one's daily life. Therefore, we first treat classification of errors as an independent problem to see how well these errors can be classified based on the sensor data obtained from the smart home when the everyday activities were performed by the participants. We perform a 5-fold cross validation of five commonly used classifiers on error samples labeled with error type classes. The classifiers that are used in this experiment are as follows: C4.5 decision tree, naive Bayes, k-nearest neighbor, logistic regression and SMO support vector machine. Tables III and IV report the performance of the classifiers on all error types separately.

We find that most of the classifiers produce fairly good results on the Study 1 dataset. In spite of the fact that *Substitution* and *Irrelevant Action* errors have very few samples which results in an imbalanced class distribution, the weighted average F_1-score is more than 0.73 for C4.5, kNN and SMO classifiers. Study 2 dataset has nine different error types which makes the classification problem harder. Out of the nine classes, *Help*, *Wandering*, *Microslip* and *Error Detection* are in the minority. Therefore, most of the classifiers do not adequately learn these classes from the training data. The highest F_1-score is 0.53, achieved by the kNN.

It should be noted that *Microslip* is not a critical activity error and can be excluded from the error classes we analyze in this paper. It is coded when a participant initiates and terminates an incorrect action before the error is completed. For example, in the *Cooking* activity, when the participant begins to make action toward the sink as if she is going to fill water with faucet water but then corrects behavior and obtains the pitcher from the refrigerator. The total number of training examples available for the *Microslip* class is only 14. As this class is highly under-represented in the dataset, the classifiers do not learn any common pattern. Thus, the C4.5 decision

TABLE III
ERROR CLASSIFICATION RESULTS FOR STUDY 1 DATASET.

	Error Type	C4.5	Naive Bayes	kNN	Logistic Regression	SMO
Precision	Omission	0.671	0.667	0.658	0.539	0.781
	Substitution	0.600	0.107	0.250	0.091	0.500
	Irrelevant Action	0.600	0.119	0.800	0.500	0.667
	Inefficient Action	0.790	0.766	0.795	0.765	0.815
	Weighted Avg.	**0.736**	**0.684**	**0.731**	**0.656**	**0.787**
Recall	Omission	0.690	0.169	0.704	0.577	0.704
	Substitution	0.429	0.429	0.143	0.143	0.286
	Irrelevant Action	0.667	0.556	0.444	0.556	0.667
	Inefficient Action	0.784	0.760	0.808	0.704	0.880
	Weighted Avg.	**0.736**	**0.542**	**0.736**	**0.637**	**0.792**
F_1-*score*	Omission	0.681	0.270	0.680	0.558	0.741
	Substitution	0.500	0.171	0.182	0.111	0.364
	Irrelevant Action	0.632	0.196	0.571	0.526	0.667
	Inefficient Action	0.787	0.763	0.802	0.733	0.846
	Weighted Avg.	**0.735**	**0.554**	**0.731**	**0.645**	**0.787**

tree yields 0.00% true positive rate, 0.01% false positive rate, 100.00% false negative rate, and 98.94% true negative rate for the *Microslip* class.

Among all classifiers, C4.5 seems to perform consistently well and is computationally less expensive than classifiers such as logistic regression or SMO. Therefore, we use a C4.5 decision tree as the base classifier for the multi-class error model in the ensemble OCSVM + MCEM. We report the performance of the ensembles in the following section.

D. Ensembles

In this section, we explore the knowledge of error types to boost the performance of OCSVM by building an ensemble of OCSVM and the error type classification model. The primary motivation of using an ensemble is to improve the precision of error prediction on all activities. As mentioned in Section IV-D, we use two ensembles that differ in the error classification model. The first ensemble, OCSVM+OCEM, is an ensemble of the primary OCSVM and a second OCSVM trained on all error samples. The second ensemble, OCSVM+MCEM, is an ensemble of the primary OCSVM and a C4.5 decision tree trained on multi-class error samples. A test sample is evaluated by both the primary OCSVM and the classifier designated for the error classification model. If both classifiers predict the test sample as an "error", then the final outcome of the ensemble is "error" as well; otherwise, it is "normal". This approach helps in restricting the number of false positives.

Figure 5 compares the performance of the ensembles with OCSVM and Baseline in terms of recall and precision. The ensembles achieve similar precision scores as that of OCSVM, apart from a couple of exceptions. OCSVM+MCEM and OCSVM+OCEM perform better ($p < 0.01$) than OCSVM for the *Medication Dispenser* and *Watering Plants* activities,

TABLE IV
ERROR CLASSIFICATION RESULTS FOR STUDY 2 DATASET.

	Error Type	C4.5	Naive Bayes	kNN	Logistic Regression	SMO
Precision	Substitution	0.537	0.491	0.518	0.392	0.528
	Additional TRA	0.643	0.548	0.632	0.539	0.682
	Additional NTRA	0.455	0.026	0.000	0.056	0.200
	Perseveration	0.333	0.170	0.000	0.088	0.286
	Searching	0.426	0.308	0.597	0.331	0.571
	Help	0.000	0.000	0.000	0.063	0.000
	Wandering	0.000	0.000	0.000	0.000	0.000
	Microslip	0.000	0.028	0.000	0.000	0.000
	Error Detection	0.000	0.120	0.000	0.176	0.182
	Weighted Avg.	**0.505**	**0.423**	**0.506**	**0.393**	**0.544**
Recall	Substitution	0.615	0.307	0.721	0.324	0.682
	Additional TRA	0.706	0.107	0.738	0.453	0.743
	Additional NTRA	0.333	0.067	0.000	0.133	0.067
	Perseveration	0.105	0.421	0.000	0.158	0.105
	Searching	0.418	0.445	0.418	0.373	0.473
	Help	0.000	0.000	0.000	0.167	0.000
	Wandering	0.000	0.000	0.000	0.000	0.000
	Microslip	0.000	0.286	0.000	0.000	0.000
	Error Detection	0.000	0.136	0.000	0.136	0.091
	Weighted Avg.	**0.540**	**0.246**	**0.573**	**0.353**	**0.582**
F_1-*score*	Substitution	0.573	0.378	0.603	0.355	0.595
	Additional TRA	0.673	0.180	0.681	0.492	0.711
	Additional NTRA	0.385	0.038	0.000	0.078	0.100
	Perseveration	0.160	0.242	0.000	0.113	0.154
	Searching	0.422	0.364	0.492	0.350	0.517
	Help	0.000	0.000	0.000	0.091	0.000
	Wandering	0.000	0.000	0.000	0.000	0.000
	Microslip	0.000	0.050	0.000	0.000	0.000
	Error Detection	0.000	0.128	0.000	0.154	0.121
	Weighted Avg.	**0.519**	**0.267**	**0.530**	**0.369**	**0.556**

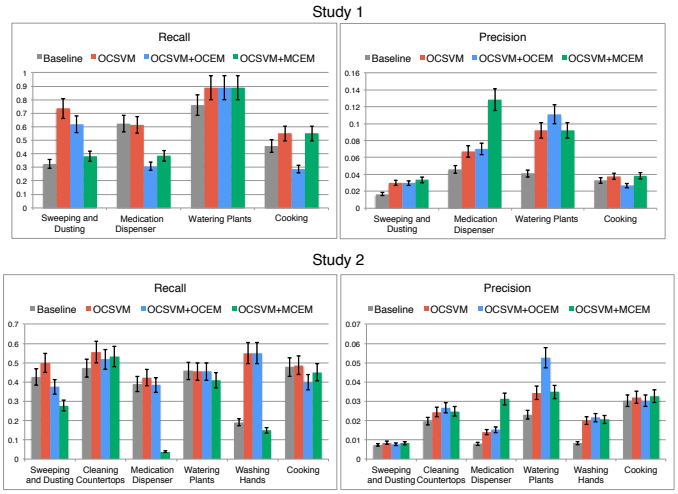

Fig. 5. (*left*) Recall and (*right*) precision values for all methods.

TABLE V
F_1-SCORES FOR STUDY 1 AND STUDY 2 DATASETS.

	Activity	Baseline	OCSVM	OCSVM + OCEM	OCSVM + MCEM
Study 1	Sweeping and Dusting	0.027	0.057	0.056	0.062
	Medication Dispenser	0.085	0.121	0.114	0.192
	Watering Plants	0.077	0.167	0.197	0.167
	Cooking	0.059	0.071	0.049	0.071
Study 2	Sweeping and Dusting	0.014	0.016	0.015	0.016
	Cleaning Countertops	0.033	0.046	0.051	0.047
	Medication Dispenser	0.015	0.027	0.029	0.034
	Watering Plants	0.044	0.064	0.094	0.064
	Washing Hands	0.015	0.038	0.041	0.036
	Cooking	0.056	0.06	0.056	0.061

respectively, in both datasets. However, there is no improvement in recall. In fact, the ensembles either perform worse or are at par with OCSVM. In some cases, such as *Medication Dispenser* in Study 1 and *Sweeping and Dusting* in Study 2, the ensembles yield lower recall than Baseline. The F_1-score for all the techniques are reported in Table V.

Between OCSVM+OCEM and OCSVM+MCEM there is no clear winner. OCSVM+ MCEM achieves higher precision than OCSVM+OCEM for *Medication Dispenser* in both datasets. However, the situation is just the opposite for *Watering Plants* where OCSVM+OCEM performs best. These observations reinforce the fact that due to the differences in the ways daily activities are usually performed, there is no one-size-fits-all solution for all activities.

The temporal accuracy of the ensembles is not an improvement over OCSVM either. RMSE values in terms of seconds and number of events is given in Figure 6. In most of the cases, the RMSE of ensembles is at par with OCSVM and worse than OCSVM in some cases. However, the performance is definitely better than Baseline.

VI. DISCUSSION

A. High false positive rate

The results obtained by the proposed approaches validate our hypothesis that machine learning techniques, particularly, outlier detection algorithms, are capable of recognizing activity errors from sensor data in smart homes. We get an average of 60% recall in predicting activity errors. However, the average false positive rate of our approach on all activities is about 45%. We believe, the reason behind a fairly high false positive rate is rooted in the way activity errors were annotated in the sensor data. As mentioned in Section III, annotations of activity errors only give an idea of when the errors start. However, the duration of the activity error varies vastly. For example, opening the wrong cabinet during *Medication Dispenser* is an error that corresponds to a single event of the door sensor attached to the wrong cabinet. On the other hand, errors such as wandering in the apartment or failing to return items used in *Cooking*, can last for quite a while. In our approache, every sensor event is considered as a data sample, thus the events corresponding to an error after it has already started, are labeled as "normal". However, in reality these sensor events are errors as well. Moreover, annotation of sensor data also depends on the perspective of the annotators. The inter-annotator consistency that we found for activity annotations was roughly 80%. Therefore, if we are able to overcome all of these limitations, we can achieve a much lower false positive rate.

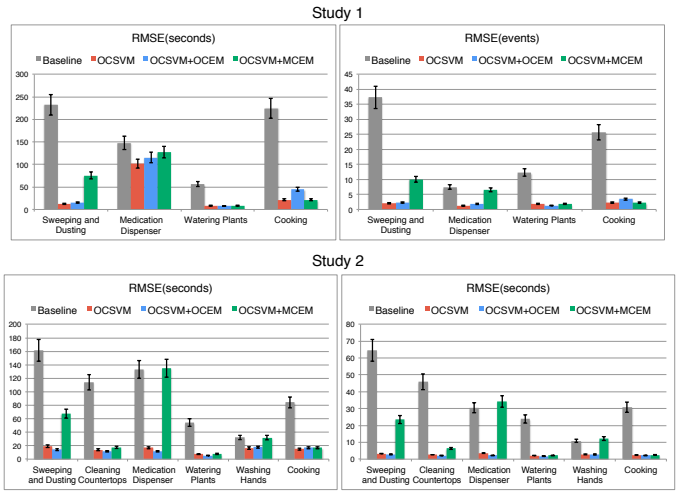

Fig. 6. RMSE in (*left*) seconds and (*right*) number of events for all methods.

B. Analysis of false positives

To better understand the reasons for DERT generating false positives in error detection, we enlisted the help of a psychologist who is involved in the smart home studies conducted with older adults. We randomly chose five participants each for the *Sweeping and Dusting* and *Cooking* activities from Study 1, for whom activity errors were reported by the psychology experimenters. We evaluate this data subset with OCSVM. The top 33% of the false positives, ranked on the basis of ν and γ values, are used for evaluation.

The psychologist watched the smart home video recordings for the chosen participants. Out of a total of 45 false positives, 15 were found to be continuation of errors that had not been annotated - only the beginning of errors were annotated. In 8 additional cases, the psychologist thought that they should have been reported as errors. The remainder of the 22 false positives were part of regular activity steps.

From this clinical evaluation we can conclude that a precise annotation of the error start and end times is crucial for evaluating the performance of our algorithms. This will reduce the false positives that correspond to the continuation of previously occurred errors. Moreover, an experimenter's perspective in reporting errors also plays an important role. For instance, some of the "false positive" cases were actually true positives that were not initially caught by the experimenters. This indicates that automated approaches to error detection could potentially improve upon human-only methods for detecting errors and prompting individuals with memory limitations to correctly complete critical daily activities.

VII. CONCLUSION

In this paper, we propose a novel one-class classification-based approach to detect activity errors. Our activity error detection approach, DERT, automatically detects activity errors

in real time while an individual performs an activity. Successful detection of activity errors can help us identify situations to prompt older adults to complete their daily activities. Sensor-based daily activity data from two studies that were conducted on 580 participants are used to train the one-class models for the activities. DERT learns only from the normal activity patterns of the participants and does not require any training examples for the activity errors. Activity data from participants who committed errors are used to evaluate the efficiency of DERT. Test samples that are classified as outliers, are potential prompt situations. We also use smart home sensor data to classify various error types. This information is important for psychologists and gerontologists to characterize the nature of functional changes among older adults due to dementia.

ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation, under grants 1262814 and 1064628. The authors would like to thank Carolyn Parsey, Christa Simon, and Alyssa Weakley for their help with data collection, activity error coding, and evaluation of algorithm-generated false positive errors.

REFERENCES

[1] G. K. Vincent and V. A. Velkoff, *The next four decades: The older population in the United States: 2010 to 2050*. US Department of Commerce, Economics and Statistics Administration, US Census Bureau, 2010.

[2] ALZ, "The Alzheimer's Association 2014 report on Alzheimer's disease facts and figures," Alzheimer's Association, 2014.

[3] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan, "CASAS: a smart home in a box," *Computer*, vol. 46, no. 7, pp. 62–69, July 2013.

[4] D. J. Cook and M. Schmitter-Edgecombe, "Assessing the quality of activities in a smart environment," *Methods of Information in Medicine*, vol. 48, no. 5, p. 480, 2010.

[5] F. Doctor, H. Hagras, and V. Callaghan, "A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 35, no. 1, pp. 55–65, 2005.

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

[7] B. Das, C. Chen, A. M. Seelye, and D. J. Cook, "An automated prompting system for smart environments," in *Toward Useful Services for Elderly and People with Disabilities*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2011, vol. 6719, pp. 9–16.

[8] N. C. Krishnan and D. J. Cook, "Activity recognition on streaming sensor data," *Journal of Pervasive and Mobile Computing*, vol. 10, no. B, pp. 138–154, 2014.

[9] A. M. Seelye, M. Schmitter-Edgecombe, B. Das, and D. J. Cook, "Application of cognitive rehabilitation theory to the development of smart prompting technologies," *IEEE Reviews in Biomedical Engineering*, vol. 5, pp. 29–44, 2012.

[10] M. Lim, J. Choi, D. Kim, and S. Park, "A smart medication prompting system and context reasoning in home environments," in *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on*, vol. 1. IEEE, 2008, pp. 115–118.

[11] M. Rudary, S. Singh, and M. E. Pollack, "Adaptive cognitive orthotics: combining reinforcement learning and constraint-based temporal reasoning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 91.

[12] M. E. Pollack, L. Brown, D. Colbry, C. E. McCarthy, C. Orosz, B. Peintner, S. Ramakrishnan, and I. Tsamardinos, "Autominder: An intelligent cognitive orthotic system for people with memory impairment," *Robotics and Autonomous Systems*, vol. 44, no. 3-4, pp. 273–282, 2003.

[13] 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 *IJCAI*. Citeseer, 2005, pp. 1293–1299.

[14] 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," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 10, no. 2, pp. 323–333, 2006.

[15] J. Hoey, P. Poupart, A. v. 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, no. 5, pp. 503–519, 2010.

[16] S. Zhang, S. McClean, B. Scotney, X. Hong, C. Nugent, and M. Mulvenna, "Decision support for alzheimer's patients in smart homes," in *Computer-Based Medical Systems, 2008. CBMS'08. 21st IEEE International Symposium on*. IEEE, 2008, pp. 236–241.

[17] M. A. Feki, J. Biswas, and A. Tolstikov, "Model and algorithmic framework for detection and correction of cognitive errors," *Technology and Health Care*, vol. 17, no. 3, pp. 203–219, 2009.

[18] Y. Chu, Y. C. Song, H. Kautz, and R. Levinson, "When did you start doing that thing that you do? interactive activity recognition and prompting," in *AAAI 2011 Workshop on Artificial Intelligence and Smarter Living: The Conquest of Complexity*, 2011, p. 7.

[19] A. M. Seelye, M. Schmitter-Edgecombe, D. J. Cook, and A. S. Crandall, "Naturalistic assessment of everyday activities and prompting technologies in mild cognitive impairment." *Journal of the International Neuropsychological Society*, vol. 19, no. 4, pp. 442–452, 2013.

[20] M. B. Tracey Holsinger, Janie Deveau and J. W. Williams., "Does this patient have dementia," *Jama*, vol. 297, no. 21, pp. 2391–2404, 2007.

[21] e. a. K. Peres, "Restriction in complex activities of daily living in mci impact on outcome," *Neurology*, vol. 67, no. 3, pp. 461–466, 2006.

[22] B. Schölkopf, J. C. Platt, and A. J. Smola, "Kernel method for percentile feature extraction," 2000.

[23] C. W. C. Tam, L. C. W. Lam, H. F. Chiu, and V. W. Lui, "Characteristic profiles of instrumental activities of daily living in chinese older persons with mild cognitive impairment," *American journal of Alzheimer's disease and other dementias*, vol. 22, no. 3, pp. 211–217, 2007.

[24] M. Schmitter-Edgecombe and C. Parsey, "Assessment of functional change and cognitive correlates in the progression from healthy cognitive aging to dementia," 2014, Neuropsychology.

Barnan Das is a Senior R&D Engineer at Intel Corporation. He received his Bachelor of Technology degree from West Bengal University of Technology and a Ph.D. from Washington State University. Dr. Das's research interests include machine learning, pervasive computing and computer vision.

Diane J. Cook is a Huie-Rogers Chair Professor at Washington State University. Dr. Cook received a B.S. degree from Wheaton College and M.S. and Ph.D. degrees from the University of Illinois. Her research interests include machine learning, graph-based relational data mining, and smart environments. Dr. Cook is an IEEE Fellow.

Narayanan C. Krishnan is an Assistant faculty at Indian Institute of Technology Ropar. He received his B.Sc and M.Sc degrees from Sri Sathya Sai Institute of Higher Learning and a Ph.D. degree in Computer Science from Arizona State University in 2010. His primary research interests include data mining and human behavior modeling from pervasive sensors.

Maureen Schmitter-Edgecombe received the B.S. degree from Bucknell University and the M.S. and Ph.D. degrees from the University of Memphis. She is a Professor at Washington State University. Her research focuses on evaluating attention, memory, and executive functioning issues with the goal of designing and assessing rehabilitation.