Automated HPOMDP Construction through Data-mining Techniques in the Intelligent Environment Domain

G. Michael Youngblood, Edwin O. Heierman, Diane J. Cook, and Lawrence B. Holder
Department of Computer Science & Engineering
The University of Texas at Arlington
Arlington, Texas 76019-0015
{youngbld, heierman, cook, holder}@cse.uta.edu

Abstract

Markov models provide a useful representation of system behavioral actions and state observations, but they do not scale well. Utilizing a hierarchy and abstraction as in hierarchical hidden Markov models (HHMMs) improves scalability, but they are usually constructed manually using knowledge engineering techniques. In this paper, we introduce a new method of automatically constructing HHMMs using the output of a sequential data-mining algorithm, Episode Discovery. Repetitive behavioral actions in sensor rich environments can be observed and categorized into periodic episodes through data-mining techniques utilizing the minimum description length principle. From these discovered episodes, we demonstrate an automated technique for creating HHMMs and subsequent hierarchical POMDPs (HPOMDPs) for intelligent environment research. We present the theory of this technique and frame it in a case study involving our MavHome architecture and an on-campus smart apartment with a real inhabitant.

Introduction

As the AI research community seeks to develop work around larger and larger domains, the ability to manually design data structures and model data will become an intractable obstacle. The ability to learn structures and models from data automatically would enable an increased capacity for working with large data domains, and provide fertile areas for autonomous learning without human-influenced bias on structure. We are currently engaged in research that examines the problem of learning human inhabitant behavioral models and their interactions in intelligent environments (i.e., smart homes and offices). By observing large amounts of human-subject interaction data, we can create user models for the development of intelligent automation systems.

Intelligent environments with humans-in-the-loop present a real-world domain with unique features. The domains we work with are sensor rich—having hundreds of sensors registering motion, light, temperature, device interactions, and so forth—but do not provide a fully observable environment. Humans are creatures of habit, so interactions usually follow daily, weekly, and monthly activity patterns with some regularity. However, human behavior can also be unpredictable at times providing a level of uncertainty. In order to automate features of an intelligent environment, we must make automation decisions in these uncertain, partially observable, and individually unique environments.

Work in decision-making under uncertainty has popularized the use of Hierarchical Hidden Markov Models (HHMMs) (Fine, Singer, & Tishby 1998) and Partially Observable Markov Decision Processes (POMDP) (Sondik 1971; Kaelbling, Littman, & Cassandra 1996). Recently there have been many published hierarchical extensions that allow for the partitioning of large domains into a tree of manageable POMDPs (Pineau, Roy, & Thrun 2001; Theocharous, Rohanimanesh, & Mahadevan 2001). Although the Hierarchical POMDP (HPOMDP) fits well for the intelligent environment domain, current work in the field requires *a priori* construction of the HPOMDP. Given the anticipated size of our domain, we need to seed our model with structure automatically derived from observed inhabitant activity data. As a result, we look to the data-mining community for a possible solution to identify data-driven structures in the large amounts of data collected and streaming in an intelligent environment.

The inhabitant interactions can be viewed as a time-ordered sequence, and several works address the problem of discovering patterns in such sequences. Variations of the Apriori property can be used for mining sequential patterns from time-ordered transactions (Agrawal & Srikant 1995). When the data is not transactional in nature, frequent parallel and serial episodes can be discovered by sliding a window of user-defined size over an event stream (Mannila, Toivonen, & Verkamo 1995). However, the important patterns in a time-ordered data stream captured from an intelligent environment may not be the ones that occur with the most frequency, but rather the ones that occur with the most regularity. Therefore, we employ a technique that evaluates patterns for frequency, size, and regularity (Heierman & Cook 2003).

This paper introduces a new technique for the automatic construction of learned HPOMDPs through a data-mining algorithm known as Episode Discovery (ED). This technique provides one possible solution to the problem of creating these structures for advanced AI work without the need for hand-generation.

Copyright © 2005, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

Related Work

There are many intelligent environment projects producing valuable research with similar goals to our own. The Georgia Tech Aware Home is working on aspects of inhabitant localization, context-aware computing, and many HCI applications (AHRI 2003; Salber, Dey, & Abowd 1999; Abowd, Battestini, & O'Connell 2002). The AIRE group at the MIT AI Lab is engaged in research involving pervasive computing designs and people-centic applications and have constructed "AIRE spaces" in the forms of an intelligent conference room, intelligent workspaces, kiosks, and "oxgenated" offices (AIRE Group 2004). At Stanford University, the Interactive Workspaces Project is exploring work collaboration technologies in technology-rich environments with a focus on task-oriented work such as design reviews or brainstorming sessions. Their experimental facility is called "iRoom" where they are investigating integration issues with multiple-device, multiple user applications, interaction technologies, deployment software, and component integration (Stanford Interactivity Lab 2003). The Gaia project at UIUC has introduced the idea of using a meta-OS to manage active spaces and has defined an architecture to manage these environments (Gaia Project 2004). Gaia seeks to extend operating system services to physical spaces so that both physical and virtual devices can seamlessly interact. The Adaptive Home at UC-Boulder utilizes a neural network to control the lighting, HVAC, and water temperature in a manner that minimizes operating cost (Mozer 1999). The field of intelligent environment research has many niches. Our project is unique in that it focuses on the entire environment management and not just a single area of control, it utilizes advanced AI techniques in novel ways (e.g., seeding HHMMs with Data Mining techniques), and is designed for long term usage and growth with the inhabitants.

Intelligent Environments

The motivation for this work is the development of systems to automate intelligent environments, such as the apartment setting shown in figure 1. This environment is called the MavPad and is part of the ongoing MavHome (**M**anaging an **A**daptive **V**ersatile Home) Project at The University of Texas at Arlington (mavhome.uta.edu). The MavPad is outfitted with a motion sensor array, environmental sensors (including HVAC), and lighting and appliance control technologies.

The sensing and control capabilities of intelligent environments fit into the generalized models of any sensed and controlled system. Figure 1 shows the sensing and control capabilities of the MavPad. The sensors in our environments are designated with a zone-number combination (e.g., A1) for uniqueness. In our environments, there is a one-to-one correspondence between state and action (e.g., an action such as turning on a light produces the state change of that light being on). Inhabitant interaction in these environments produces a constant stream of time-ordered data.

Our intelligent environment data streams are formally defined as a 6-tuple $\langle\, t, z, n, l, r, p \,\rangle$:

- t is the timestamp of the event (DD-MM-YYYY HH:MM:SS)
- z is the device or sensor zone
- n is the device or sensor number
- l is the level or value of the new state
- r is the source of the event (e.g., sensor network, powerline control)
- p is the specific inhabitant initiating the event (if identifiable)

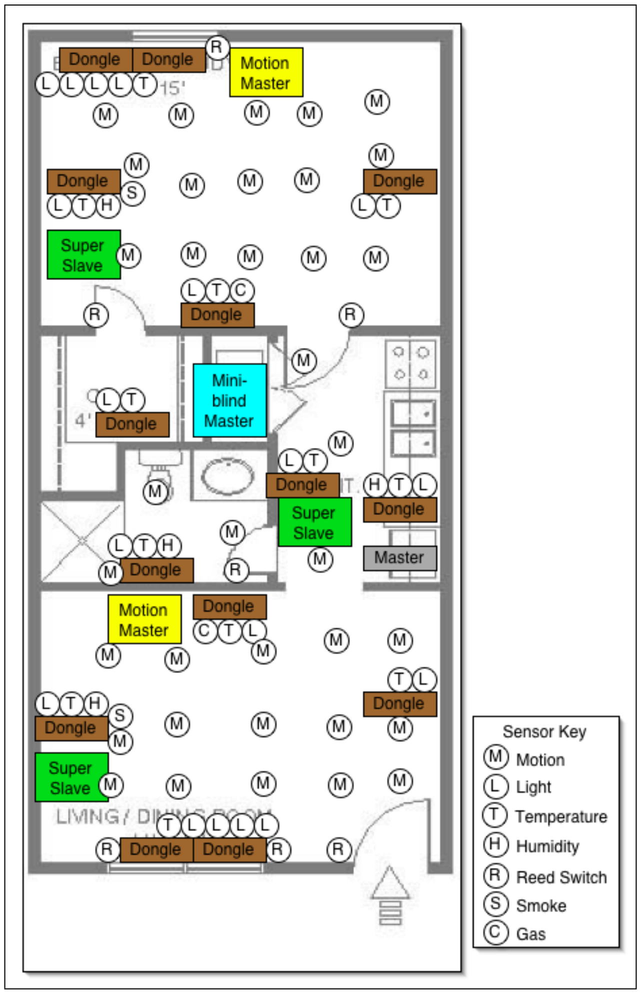

Figure 1: MavPad sensors.

The goals of our system are to learn a model of the inhabitants of the intelligent environment, automate devices to the fullest extent possible in order to maximize the comfort of the inhabitants, minimize the consumption of resources, and maintain safety and security. In order to accomplish these goals, we must first learn from the inhabitant, and then incorporate this into an adaptive system for continued learning and control. Data streams from our work are collected and provided first to a data-mining technique, Episode Discovery (ED), in order to learn the inhabitant patterns.

Episode Discovery

Our data-mining algorithm discovers interesting patterns in a time-ordered data stream. ED processes a time-ordered sequence, discovers the interesting episodes that exist within the sequence as an unordered collection, and records the unique occurrences of the discovered patterns. Because the framework is capable of processing the interactions incrementally, it can be used as part of a real-time system. These features make ED a suitable algorithm for mining an intelligent environment data stream.

An input stream *O* processed by ED consists of a time-ordered sequence of events $\{s,t\}$, where *s* is a symbol representing an interaction and *t* is a time stamp. Given an input stream *O*, ED:

1. Partitions *O* into Maximal Episodes *Emax*.
2. Creates Candidates *C* from the Maximal Episodes.
3. Computes a Compression Value *V* for each candidate.
4. Identifies Interesting Episodes *P* by evaluating the Compression Value of the candidates.

Each 6-tuple of the intelligent environment data stream is transformed into an event by constructing a symbol s from the concatenation of *z*, *n*, and *l*, and using *t* as the timestamp. Each atomic device interaction is represented by a unique symbol with this approach.

The algorithm processes the sequence of events as follows. First, ED partitions the input sequence into overlapping clusters, or episodes, of symbols that are closely related in time by collecting the events in a window. Once the window accumulates the maximum number of interactions that occur within the window time frame, the contents are converted to a maximal episode. The maximum number of interactions accumulated by the window, as well as the time span of the window, is computed by analyzing the compression statistics computed by the algorithm. A candidate is created to represent each unique maximal episode, and additional candidates are generated from the subsets of these candidates (Heierman, Youngblood, & Cook 2004).

Next, a periodic compression value is computed for each candidate by the use of an evaluation function, based on Minimum Description Length (MDL) principles (Rissanen 1989), that incorporates regularity (predictable periodicity, such as daily, weekly, or monthly), in addition to frequency and size. Using autocorrelation techniques, the algorithm identifies the best periodic pattern that describes the episodes represented by the candidate. A compression value is then computed based on this periodic pattern. The larger the potential amount of compression a pattern provides, the greater the regularity demonstrated by the pattern, and the greater the impact that results from automating the pattern.

Once all of the candidates have been evaluated and sorted in descending order, the algorithm greedily identifies the periodic episodes by selecting from the sorted list the candidate with the largest compression value. After selecting a candidate, the algorithm marks the events represented by the interesting episode. To avoid selecting overlapping candidates, the second and subsequent candidates are rejected if any of the interactions represented by the candidate are already marked. These steps are repeated until all candidates have been processed, resulting in a collection of interesting episodes. Because periodicity is evaluated, the interesting episodes that are periodic can be identified.

The *k* uniquely identified interesting episodes, *P*, output by ED are formally defined as a 3-tuple $\langle \Phi, \rho, \Psi \rangle$:

- Φ is the unordered set of symbols represented by *P*
- ρ is the number of instances of *P* in the dataset
- Ψ is the set of unique ordered occurrences of *P* in the dataset, and is defined as a 2-tuple $\langle \Upsilon, \varphi \rangle$:
  - Υ is an ordered set of symbols
  - φ is the number of occurrences of this unique pattern out of the total number of instances, ρ

Integrating Episodes into an HHMM

As review, an HHMM is defined as a set of states *S*, which include *production* states (leaves) that produce observations, *abstract* states which are hidden states (unobservable) representing entire stochastic processes, and *end* (child) states which return control to a parent node; a horizontal transition matrix *T*, mapping the probability of transition between child nodes of the same parent; a vertical transition vector Π, that assigns the probability of transition from an internal node to its child nodes; a set of observations *Z*; and a mapping of a distribution probability set of observations in each product state *O*. For detailed discussions refer to (Fine, Singer, & Tishby 1998; Theocharous, Rohanimanesh, & Mahadevan 2001).

From a collection of data that contains many repetitive patterns, data-mining techniques can automatically discover these patterns and provide statistical data on pattern permutations within a given set of members and over the entire data set. This information can be utilized to create abstract states from the identified patterns and production states from the pattern sequences. Aggregating abstract states based on set similarities can be used to increase hierarchical organization.

The 3-tuple data provided by ED is converted to a HHMM by taking each periodic episode *k* and performing the following steps:

1. Create an episode labeled *abstract* node
2. For each set in Ψ
   (a) Create an observation labeled *product* node for each observation in Υ
   (b) Assign the horizontal transition matrix value between each product node $T^s(c(s,i), c(s,j)) = \varphi/\rho$
   (c) Create an *end* node
   (d) Assign the last product node in the sequence a horizontal transition value to the *end* node of
      i. $T^s(c(s,i), c(end)) = 1$ unless another transition exists from this node
      ii. otherwise, $T^s(c(s,i), c(end)) = \varphi/\rho$
3. Assign the vertical transition vector value
   (a) if $s = \Psi^i(\Upsilon^0)$ then $\Pi^s = \Psi^i(\varphi/\rho)$

(b) else $\Pi^s = 0$

4. Connect abstract node to the root node

The vertical transition vector values between abstract nodes from the root to the episodes are assigned from episode occurrence information from ED. Horizontal transition matrix data between abstract nodes of the same level is captured by repeating the episode discovery process on the discovered episodes on each level in order to learn the abstractions of the next higher level. This can be repeated until no abstractions for a level are found, in which case this is the root level. Each abstract state is partially represented by the observation sequences it contains in its child nodes. Due to overlap in these observation sequences between parent abstract nodes, abstract nodes can be grouped into hierarchies. After this process a n-tier HHMM is automatically created from learned data. Hierarchy is data driven and learned.

This technique accounts for the automatic construction of the HHMM *S*, *T*, and Π. *Z* is the set of all observations in Φ^k. From the last paragraph discussion on abstract state representations of observations, it can be seen how *O* can be easily calculated (where $O(s, z) \rightarrow (0, 1)$: the probability of observing z in state s). This is a usable HHMM; however, we desire a framework where we can perform automation and possibly perform additional learning.

HHMM into a HPOMDP

A HPOMDP follows the same base definition as an HHMM with the addition of a set of actions *A* that is used to transition between states, and a reward function *R* defined on the product states. For detailed discussions refer to (Theocharous, Rohanimanesh, & Mahadevan 2001).

Due to the distinguishing characteristic of a one-to-one correspondence between action and observation, the transition action is merely the action of that observation (e.g., if the observation is that the light came on then the action is to turn the light on). In the intelligent environments domain, there are two types of observations: those that have corresponding actions and those that do not. For example, we may control lights and appliances, but we cannot control motion sensing or opening doors (yet). For those transitions that do not have a corresponding action, the action is to do nothing. These actions are mutually exclusive; thus, extending the derived HHMM from the last section to a HPOMDP is performed first by adding the obvious actions to the state transitions and then by establishing a reward (or cost) for transition. This reward can be fixed, heuristic-based, or function-based (e.g., a function of utility cost changes for an action).

Usage of this model entails using belief states based on current environmental observations to map to states in the HPOMDP. We are using ED in an online, stream mining role to provide a set of membership belief states. We are also using a predictor, ALZ (described in the next section), to provide additional belief. From the given current belief state, the action with the largest reward should be chosen (Kaelbling, Littman, & Cassandra 1996). In our work, we look for the nearest automatable action. Additional learning from feedback or interaction data may be used to update the transition probabilities of the model with the goal of improving convergence to the desired model (Theocharous, Rohanimanesh, & Mahadevan 2001).

Case Study: MavPad

We have been collecting real-world, real-time inhabitant data 24 hours, 7 days a week for the past 6 months in our MavPad on-campus student apartment setting (shown in Figure 1). Experiments have proven the ability of this approach to automatically create HPOMDPs from inhabitant interaction data. We have collected data sets containing months of inhabitant data by students living in the MavPad. The data sets thus represent activity patterns using real sensor data. For example, we establish activity patterns that show someone entering the MavPad, watching TV, going to the kitchen, leaving, entering, and so forth. We ask the inhabitants to live their life normally, so the patterns they generate are those required by their own lifestyle. The MavHome system is deployed in the MavPad to learn from the inhabitant.

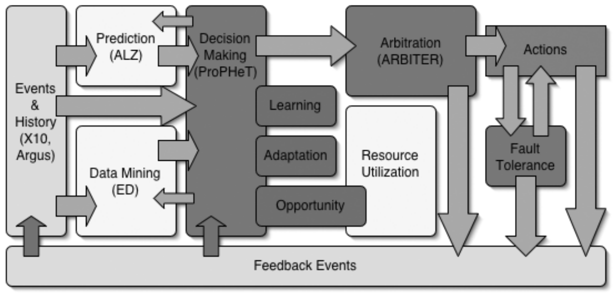

Figure 2: MavHome Architecture.

MavHome Architecture

The MavHome core architecture is composed of several layers. Figure 2 shows the components which include sensor and actuator systems, prediction, data-mining, decision making, learning, and evaluation mechanisms.

An intelligent environment must be able to acquire and apply knowledge about its inhabitants in order to adapt to the inhabitants and meet the goals of comfort and efficiency. These capabilities rely upon effective prediction algorithms. Given a prediction of inhabitant activities, MavHome can decide whether or not to automate the activity or even find a way to improve the activity to meet the system goals.

Specifically, the MavHome system needs to predict the inhabitant's next action in order to automate selected repetitive tasks for the inhabitant. The system will need to make this prediction based only on previously-seen inhabitant interaction with various devices. It is essential that the number of prediction errors be kept to a minimum–not only would it be annoying for the inhabitant to reverse system decisions, but prediction errors can lead to excessive resource consumption. Another desirable characteristic of a prediction algorithm is that predictions be delivered in real time without resorting to an offline prediction scheme.

Based upon our past investigations, MavHome uses the *Active-LeZi* (ALZ) algorithm (Gopalratnam & Cook 2003) to meet our prediction requirements. By characterizing inhabitant-device interaction as a Markov chain of events, we utilize a sequential prediction scheme that has been shown to be optimal in terms of predictive accuracy. Active-LeZi is also inherently an online algorithm, since it is based on the incremental LZ78 data compression algorithm.

The MavHome approach to state space reduction from the large number of potential environment observations is to abstract inhabitant activity to episodes that represent the current task of involvement. Given the inhabitant task episode, observations not related to the task can be pruned. A difficult problem is how to discover these episodes. After discovering the episodes, it is also desirable to be able to classify streamed observations to episodes in real time with the same service. MavHome uses the *Episode Discovery* (ED) algorithm (Heierman & Cook 2003) for finding inhabitant episodes in the collected data and for episode classification of streamed observations.

Decision making is performed in the ProPHeT (**Pro**viding Partially-observable **H**idden (HMM/POMDP) based d**e**cision **T**asks) component. This is the implementation mechanism for the work presented in the earlier sections of this paper.

Before an action is executed it is checked against the policies in the policy engine, ARBITER (A Rule-Based InitiaTor of Efficient Resolutions). These policies contain designed safety and security knowledge and inhabitant standing rules. Through the policy engine the system is prevented from engaging in erroneous actions that may perform such activities as turning the heater to 120° F or from violating the inhabitant's stated wishes (e.g., a standing rule to never turn off the inhabitant's night light).

These system work in concert to learn, adapt, and automate the inhabitants' lives in an intelligent environment.

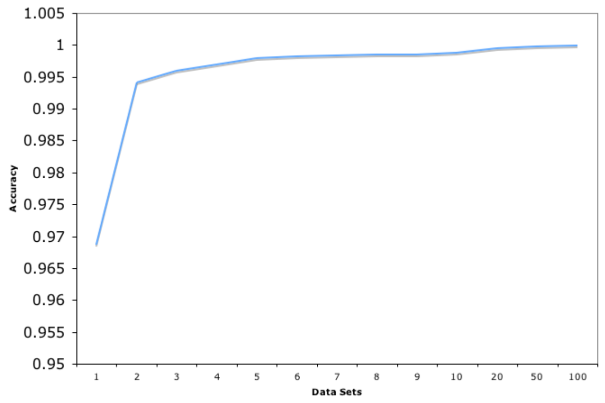

Figure 3: ALZ learning.

A Day in the Life of the Inhabitant

As an illustration of these techniques and the MavHome system deployed in the MavPad, we have evaluated a typical day in the inhabitant's life with the goal of reducing the inhabitant's interactions with the lighting in the MavPad. The data was restricted to just motion and lighting interactions which account for an average of 10,310 events per day. There are on average 18 lighting device interactions a day with the remainder being motion information. Using our ResiSim (Residentual Simulator) tool which exactly replicates the real MavPad, we trained ALZ and ED on real data and then repeated a typical MavPad inhabitant day in the simulator to determine if the system could automate the lights throughout the day in real-time.

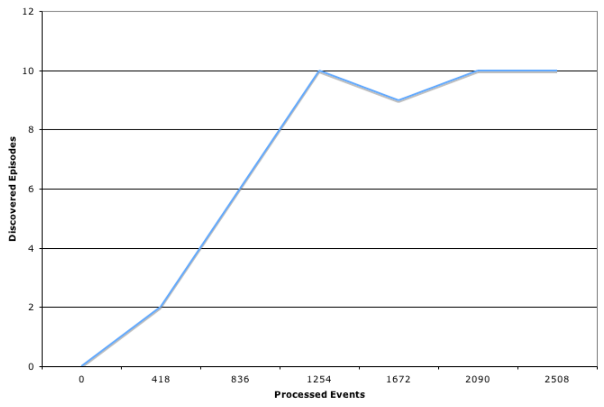

Figure 4: ED learning.

ALZ processed the data and converged to 99.99% accuracy on test data from the data set as illustrated in Figure 3. When the system was run with automation decisions being made by ALZ alone, it was able to reduce interactions by one event as shown in Figure 5. ALZ performance on streaming data maintained between 40-60% accuracy.

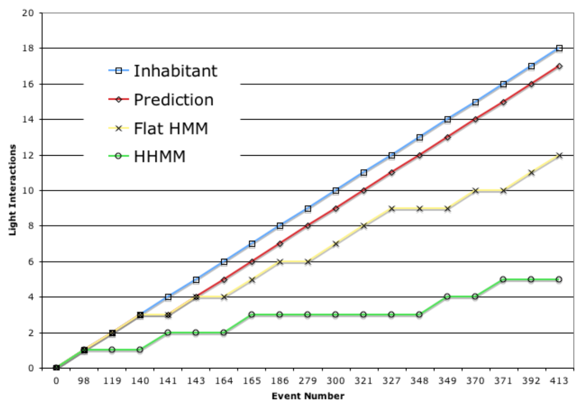

Figure 5: Interaction reduction.

ED processed the data and found 10 interesting episodes that correspond to automatable actions. Figure 4 shows the learning rate of ED for production states. This was abstracted through ED to three abstract nodes. A HPOMDP was constructed in ProPHeT as shown in Figure 6. This system was able to reduce interactions by 72.2% to five interactions. As a comparison, the HHMM produced was flattened and the abstract nodes removed to produce a flat HMM. This HMM was still able to reduce interactions by 33.3% to 12. Comparative results are shown in Figure 5.

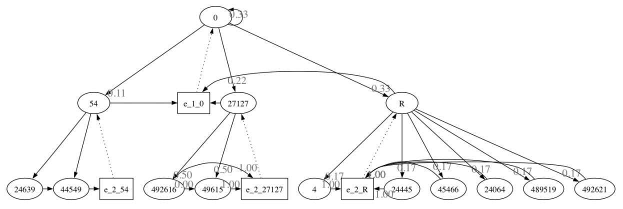

Figure 6: Learned HHMM from MavPad inhabitant data.

The additional abstractions in the hierarchy coupled with a next state produced by ALZ and a probability of membership from ED to provide input to the belief state create a system that improves automation performance over a flat model or prediction alone. Problems in the automation decisions appear around interactions that occur within a short time-frame and are currently under investigation.

Conclusions

This paper presents a new technique for the automatic construction of learned HPOMDPs through a data-mining algorithm, Episode Discovery (ED). This technique provides one possible solution to creating these structures for advanced AI work without the need for hand-generation. The flow and characterization of data from stream through episode discovery to HHMM and then HPOMDP is presented. A short description of the MavHome architecture and small case study of our initial results from a typical MavPad inhabitant day illustrates the viability of this approach.

Future Work

We are continuing to refine our work, evaluating increasingly complex patterns, improving automation usage, expanding the sensor resolution, and are investigating hierarchical policy gradient algorithms to apply hierarchical reinforcement learning to the HPOMDP in order to improve and adapt our systems automatically over time. Our goal is to develop techniques for the intelligent environment to continually adapt and learn with the inhabitants. Current experimentation involves more data from the MavPad and from a multiple-inhabitant study in the MavLab (our research lab).

Acknowledgements

This work was supported by National Science Foundation grants IIS-0121297 and EIA-9820440.

References

Abowd, G. D.; Battestini, A.; and O'Connell, T. 2002. The Location Service: A Framework for Handling Multiple Location Sensing Technologies.

Agrawal, R., and Srikant, R. 1995. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering*, 3–14.

AHRI. 2003. [AHRI] - Aware Home Research Initiative.

AIRE Group. 2004. MIT Project AIRE – About Us.

Fine, S.; Singer, Y.; and Tishby, N. 1998. The Hierarchical Hidden Markov Model: Analysis and Applications. *Machine Learning* 32(1):41–62.

Gaia Project. 2004. Gaia homepage. http://gaia.cs.uiuc.edu.

Gopalratnam, K., and Cook, D. J. 2003. Active LeZi: An Incremental Parsing Algorithm for Device Usage Prediction in the Smart Home. In *Proceedings of the Florida Artificial Intelligence Research Symposium*, 38–42.

Heierman, E., and Cook, D. J. 2003. Improving Home Automation by Discovering Regularly Occurring Device Usage Patterns. In *Proceedings of the International Conference on Data Mining*.

Heierman, E.; Youngblood, G. M.; and Cook, D. 2004. Mining temporal sequences to discover interesting patterns. In *3rd Workshop on Mining Temporal and Sequential Data (TDM'04)*.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1996. Planning and Acting in Partially Observable Stochastic Domains. Technical Report CS-96-08, Brown University, Providence, RI.

Mannila, H.; Toivonen, H.; and Verkamo, A. 1995. Discovering frequent episodes in sequences. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining*, 210–215.

Mozer, M. 1999. An Intelligent Environment must be Adaptive. *IEEE Intelligent Systems* 14(2):11–13.

Pineau, J.; Roy, N.; and Thrun, S. 2001. A Hierarchical Approach to POMDP Planning and Execution. Workshop on Hierarchy and Memory in Reinforcement Learning (ICML).

Rissanen, J. 1989. *Stochastic Complexity in Statistical inquiry*. World Scientific Publishing Company.

Salber, D.; Dey, A. K.; and Abowd, G. D. 1999. The context toolkit: Aiding the development of context-enabled applications. In *CHI*, 434–441.

Sondik, E. J. 1971. The optimal control of partially observable Markov Decision Processes. PhD thesis, Stanford University, Palo Alto, CA.

Stanford Interactivity Lab. 2003. Interactive Workspaces.

Theocharous, G.; Rohanimanesh, K.; and Mahadevan, S. 2001. Learning Hierarchical Partially Observable Markov Decision Processes for Robot Navigation. IEEE Conference on Robotics and Automation.