Recognizing Players' Activities and Hidden State

Wesley Kerr
School of Information:
Science, Technology and Arts
University of Arizona
Tucson, Arizona, USA
wkerr@email.arizona.edu

Paul R. Cohen
School of Information:
Science, Technology and Arts
University of Arizona
Tucson, Arizona, USA
cohen@cs.arizona.edu

Niall Adams
Department of Mathematics
Imperial College
London, England
n.adams@imperial.ac.uk

ABSTRACT

This paper describes a machine learning approach to classifying the activities of players in games. Instances of activities generally are not identical because they play out in different contexts, so the challenge is to extract the "essences" of activities from instances. We show how this problem may be mapped to a sequence alignment problem, for which there are polynomial-time solutions. The method works well even when some features of activities are not observable (e.g., the emotional states of players). In fact, these features can in some conditions be inferred with high accuracy.

Keywords

activity classification, hidden state, machine learning

1. INTRODUCTION

This paper deals with the problem of recognizing the activities of players in games. At first, this problem seems to apply only to human players, not to non-player characters (NPCs), as the activities of NPCs are controlled by the game and thus are known. However, interactions between players, including NPCs, and their environments can give rise to emergent or "unprogrammed" activities, for which recognizers may be useful.

The main technical problem addressed by this paper is: Given a set, $\mathcal{E}_c$, of instances of activity $\mathcal{C}$, learn a formal representation, $\mathcal{S}_c$, that captures the "essential aspects" of activity $\mathcal{C}$. Generally, instances of activities in $\mathcal{E}_c$ are rich and complex and contain a lot that should not be in $\mathcal{S}_c$; for example, as one escapes from a pursuer, one really doesn't care whether the pursuer is wearing brown shoes or black. Said differently, shoe color is not essential to recognizing instances of "escape." Often, several experiences are required to learn $\mathcal{S}_c$, if only to eliminate from $\mathcal{S}_c$ those aspects of activities $\mathcal{E}_c$ that are not essential.

The structure $\mathcal{S}_c$ is called the *signature* of $\mathcal{C}$. The main technical contribution of this paper is an efficient algorithm for both learning signatures and using them as recognizers of activities.

The problem as described involves *supervised learning* in the sense that the learner can be sure that all the activities in a set $\mathcal{E}_c$ are instances of activity $\mathcal{C}$. For example, $\mathcal{E}_{escape}$ might be a set of instances of a player escaping from another player. We also have developed an unsupervised method to cluster activities of several kinds, and to learn the "average" activity in each cluster [17].

If aspects of activity $\mathcal{C}$ are not accessible to the learner, then they cannot be added to signatures $\mathcal{S}_c$. An illustrative covert aspect of activity is intensional state: the affect, beliefs, desires and intentions of a player. These are hidden in the sense that one player generally cannot "see" them in another. You can see that someone is hurrying down the street but you do not know whether he's late for an appointment, trying to shake a pursuer, or merely out for a brisk walk. Covert aspects of activity are a problem because without them it is difficult to classify activities.

Our approach is to build NPCs that have affect, beliefs, desires and intentions, and make these available as aspects of the experiences, $\mathcal{E}_c$. These aspects may then be learned as part of the signature $\mathcal{S}_c$. The intensional content of an activity such as "escape" might not be learnable by observing other players escaping, but it can be learned by escaping oneself, assuming that one's intensional state is accessible to oneself.

2. REPRESENTATION

We represent activities in $\mathcal{E}_c$ as *propositional multivariate time series* or PMTS's. Think of a PMTS as a matrix in which every row represents a proposition, every column represents a moment in time, and every cell i,t contains 1 or 0 depending on whether proposition P_i is true at time t. Consecutive moments during which proposition P_i is true are called the *fluent* P_i, as illustrated in Figure 1.

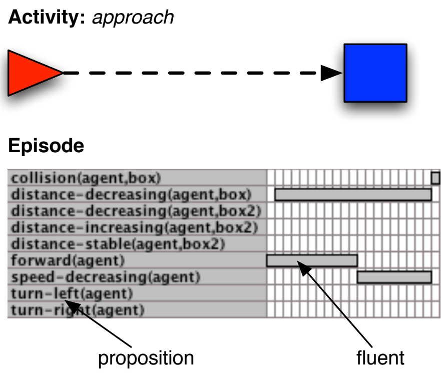

Figure 1: A fluent-based representation of "approach".

This figure suggests that the meaning of an activity such as "approach" can be represented as a set of temporal relations between fluents: First, the player starts to move and shortly thereafter the distance between the player and a box begins to decrease. Immediately after the player stops moving forward, its speed starts to decrease. Finally, it collides with the box.

The temporal relations between these fluents can be represented by *Allen relations*, of which there are seven, shown in Figure 2. The Allen relations in any activity (such as a player approaching a box) may be written out sequentially in a canonical form called a *qualitative sequence* [19].

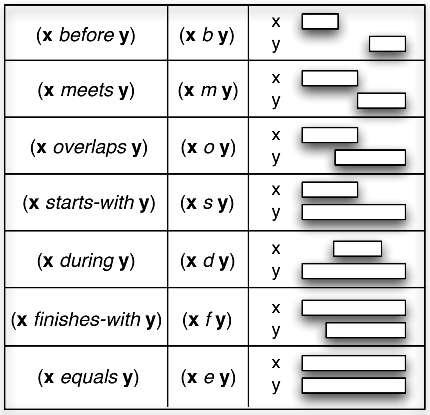
(**x** *before* **y**) (**x** *b* **y**) x y
(**x** *meets* **y**) (**x** *m* **y**) x y
(**x** *overlaps* **y**) (**x** *o* **y**) x y
(**x** *starts-with* **y**) (**x** *s* **y**) x y
(**x** *during* **y**) (**x** *d* **y**) x y
(**x** *finishes-with* **y**) (**x** *f* **y**) x y
(**x** *equals* **y**) (**x** *e* **y**) x y

Figure 2: Allen Relations

To illustrate qualitative sequences (and to keep further examples short) we abbreviate the fluents as capital letters as follows: A = forward(agent), B = decreasing-distance(agent, box), C = speed-decreasing(agent), and D = collision(agent,box). The qualitative sequence representation of Figure 1 is shown in Figure 3.

1	A	*overlaps*	B
2	A	*meets*	C
3	B	*finishes − with*	C
4	A	*before*	D
5	B	*meets*	D
6	C	*meets*	D

Figure 3: The qualitative sequence associated with the fluent-based representation in Figure 1.

3. LEARNING

Signatures are learned by a package of algorithms called CAVE, which we developed for the purposes of classifying and visualizing episodes or activities. Qualitative sequences are the raw materials from which CAVE learns signatures. Let $\mathcal{E}_c = \{S_1, S_2, \ldots, S_k\}$ be a set of qualitative sequences with the same activity label, $\mathcal{C}$. We define the signature of the activity label, $\mathcal{S}_c$, as an ordered sequence of *weighted* Allen relations. (The only difference between a signature and a qualitative sequence is these weights.) The task is to learn $\mathcal{S}_c$ given several qualitative sequences, $\mathcal{E}_c$.

In general, the qualitative sequences associated with an activity will not be identical. They will differ in "accidental" aspects: the color of a box, a spurious object off to one side, the initial distance between the agent and the box, and so on. The task is to learn the "essence" of an activity, the aspects of qualitative sequences that are entailed by the activity, and are not accidental. In practice, qualitative sequences can involve dozens of fluents and hundreds or thousands of Allen relations [19]. Thus, the algorithm that reduces qualitative sequences $\mathcal{E}_c$ to the signature $\mathcal{S}_c$ must be very efficient.

Our approach is based on sequence alignment. A set of qualitative sequences $\mathcal{E}_c$ is aligned to optimize the correspondence between the elements of the sequences. Where elements correspond, their scores are increased. The signature $\mathcal{S}_c$ accumulates the alignment of sequences. After aligning all the sequences, the sequence of the highest-scoring elements in $\mathcal{S}_c$ is the "essence" of activity $\mathcal{C}$.

We select a sequence at random from $\mathcal{E}_c$ to serve as the initial signature, $\mathcal{S}_c$, and initialize all of its weights to 1. After this, $\mathcal{S}_c$ is updated by combining it with the other sequences in $\mathcal{E}_c$, processed one at a time.

Two problems are solved during the processing of the sequences in $\mathcal{E}_c$. First, the sequences are not identical, so $\mathcal{S}_c$ must be constructed to represent the most frequent relations in the sequences. The weights in $\mathcal{S}_c$ are used for this purpose. Second, because a relation can appear more than once in a sequence S_i, there can be more than one way to align S_i with $\mathcal{S}_c$. These problems are related because the frequencies of relations in $\mathcal{S}_c$ depend on how sequences are successively aligned with it.

Updating the signature $\mathcal{S}_c$ with a sequence S_i occurs in two phases. In the first phase, S_i is optimally aligned with $\mathcal{S}_c$. The alignment algorithm, described below, penalizes candidate alignments for elements in $\mathcal{S}_c$ that are not matched by elements in S_i, and rewards matches. These penalties and rewards are functions of the weights stored with the signature. In the second phase, the weights in the signature $\mathcal{S}_c$ are updated. If an element in S_i is aligned with one from $\mathcal{S}_c$, then the weight of this element is incremented by one. Otherwise the weight of the element is initialized to one and it is inserted into $\mathcal{S}_c$ at the location selected by the alignment algorithm.

Updating the signature relies on the Needleman-Wunsch global sequence alignment algorithm [22]. The algorithm uses dynamic programming to find an optimal alignment between two sequences. Alignments are constructed by selecting operators that modify each sequence to look more like the other. Conventionally, sequence alignment is based on three operators: Elements from two sequences may *match*; an element may be *deleted*, or an element may be *inserted*. However, to ensure that no information from sequences in $\mathcal{E}_c$ is lost, we allow the sequence alignment only to insert or match, not delete, elements. The costs of insertion and matching are determined by the weights that are stored with the signature as mentioned above.

$\mathcal{S}_c$ Aligned	S_i Aligned	$\mathcal{S}_c$ Updated	
–	(C *meets* A)	(C *meets* A)	1
–	(C *before* B)	(C *before* B)	1
–	(C *before* C)	(C *before* C)	1
(A *finishes* D)	–	(A *finishes* D)	1
(A *overlaps* B)	(A *overlaps* B)	(A *overlaps* B)	6
(A *meets* C)	(A *meets* C)	(A *meets* C)	6
(D *overlaps* B)	–	(D *overlaps* B)	1
(D *meets* C)	–	(D *meets* C)	1
(B *overlaps* C)	(B *overlaps* C)	(B *overlaps* C)	6

Table 1: Updating the signature $\mathcal{S}_c$.

The process is illustrated in Table 1. Suppose that some sequences have already been aligned with the signature $\mathcal{S}_c$. The sequence in column "S_i Aligned" is first aligned optimally with $\mathcal{S}_c$, as shown; notice that this involves inserting some "empty space" into both $\mathcal{S}_c$ and S_i. Then the two are merged, as shown in column "$\mathcal{S}_c$ Updated," and the weights associated with each Allen relation are updated.

Because the process of updating signatures first aligns and then merges new sequences into $\mathcal{S}_c$, signatures become stuffed with large numbers of elements that occur very infrequently, and thus have low weights. We use a simple heuristic to clean up the signature: After K training episodes, all of the relations in the signature with weights less than or equal to n are removed. All of our experiments use $K = 10$ and $n = 3$, meaning that the signature is pruned of all elements occurring fewer than 4 times after a total of 10 training episodes. The signature is again pruned after 20 training episodes, and so forth.

4. THE USES OF SIGNATURES

Signatures support several functions:

Learning Signatures are learned from qualitative sequences as described above.

Classification Given a qualitative sequence, its corresponding activity should be correctly classified.

Inference Given a qualitative sequence, the entailments of the activity, some of which will not be directly observable in the sequence, should be correctly inferred.

The remainder of this paper describes experiments to test whether signatures support classification of activities and inference of affective and other internal aspects of state.

5. DATA SOURCES

This section describes two virtual environments in which players interact with each other and with objects under the constraints of simulated physics. These environments are simpler than today's computer games, but are sufficient to support complex behaviors and to generate large propositional, multivariate time series. In addition, our signature-learning algorithm has been tested with real-valued multivariate time series in several domains, including recognizing handwritten characters [19].

5.1 Wubble World 3D

Wubble World is a virtual environment with simulated physics, in which softbots, called wubbles, interact with objects [18]. Wubble World is instrumented to collect distances, velocities, locations, colors, sizes, and other sensory information and represent them with propositions such as $Above(wubble, box)$ (the wubble is above the box) and $PVM(wubble)$ (the wubble is experiencing positive vertical motion).

We collected a dataset of episodes of wubbles performing several kinds of tasks: *jumping over* a box, *jumping on* a box, *approaching* a box, *pushing* a box, *moving around* a box *to the left*, *moving around* a box *to the right*. Episodes were generated by manually controlling a wubble to repeatedly perform one of these tasks. Each episode was unique, in that the wubble would start closer or farther from the box, move more or less quickly, and so on. There were 37 episodes labeled *jump over*, 20 episodes labeled *jump on*, and 25 episodes for each of the remaining tasks. The average number of fluents and average number of Allen relations for these episodes are shown in Table 2.

5.2 Wubble World 2D

	NumFluents	NumAllenRelns
approach	34.84	572.72
jump-on	80.45	1356.70
jump-over	51.35	1156.51
left	95.24	3228.16
push	99.40	3802.20
right	95.04	3276.44

Table 2: Average values for the episodes in the Wubble World 3D dataset.

Like Wubble World 3D, Wubble World 2D is a virtual environment with simulated 2D physics (Fig. 4). Agents in the Wubble World 2D simulation were endowed with a cognitive system based on a blackboard architecture [9]. Every agent has a limited perceptual system comprised of a visual system with a 90° field of view and a limited range of sight, an auditory system that provides omnidirectional sensing with a limited range, and an olfactory system that is also omnidirectional and limited in range. Agents have a fixed amount of energy that can be replenished by consuming bits of food scattered throughout the environment, and can expend energy by "running" or by colliding with other agents. All agents have the ability to increase their speed so as to run or sprint for a short period of time, and doing so will drain the agent's energy. Wubbles have a primitive two-dimensional computational model of emotion, inspired by those in [6, 21]. One dimension, called *valence*, loosely corresponds to the happiness of the agent. The second dimension, called *arousal*, loosely corresponds to the excitement level of the agent. The emotional system is autonomic and over time tends towards a neutral emotional state.

The decision making process of an agent is guided by the sensory system as well as an arbiter that selects between competing goals. Agents in Wubble World 2D can: wander around, pursue other agents, flee from other agents, kick inanimate objects, eat and defend food, or sit idly waiting for something to happen to them. At every moment in time the arbiter determines the insistence of each possible goal. The insistence of a goal is affected by the energy level, the arousal level, and valence of the agent as well as the currently perceived state of the world. For instance, if a smaller agent is in front of an agent who has bullying tendencies it will have a high desire to pursue the smaller agent. Once the arbiter calculates all of the insistence values, it selects to carry out the the goal with the highest insistence. Goals are mapped into actions via a finite state machine. Once the agent has selected a goal to achieve, it initializes the corresponding FSM and begins executing the commands that will ultimately achieve the goal. Sometimes the FSMs for a goal can be very complex while others, such as wandering, are very simple.

Like the original Wubble World, all of the interactions for an agent are recorded for later analysis. Each agent has its own unique view of the world based on its perceptual system. In Wubble World 3D the wubbles had a global view of the world, whereas in Wubble World 2D the agents have a egocentric view of the world and if the agent cannot sense an object, then nothing is recorded about that object. An egocentric view of the world helps focus the agent's attention on the things within its sphere of influence and reduces the dimensionality of the time series generated.

Unlike Wubble World 3D, in the 2D version most of the sensors recorded are real-valued. These values are converted to propositions with an algorithm like SAX [20]. At every time step we record the current position, speed and heading of an agent, as well as the internal state of the agent consisting of its energy level, arousal, valence, active goal, and the active state of the executing FSM. For

	NumFluents	NumAllenRelns
ball	697.55	137283.20
chase	366.55	41906.10
column	637.25	125309.90
eat	130.80	6069.80
fight	1319.65	527603.75
flee	337.0	36758.45

Table 3: Average values by activity for the episodes in the Wubble World 2D dataset.

every other agent or object within our sensing area we record the relative position, relative velocity, distance, whether or not there is currently a collision between the agent and the object, and whether the object was seen, smelt, or heard.

The average numbers of fluents and average numbers of Allen relations for these episodes are shown in Table 3. It's worth noting the scale of the learning problem: On average, *fight* activities generated qualitative sequences of more than half a million Allen relations.

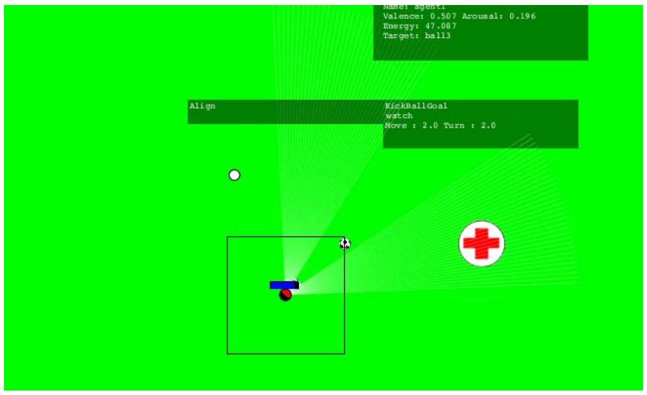

Figure 4: Agents and objects in Wubble World 2D (see text)

6. CLASSIFICATION

For classification purposes, a novel sequence is given the activity label that corresponds to the signature it matches best. Classification accuracy was assessed using 10-fold cross validation. Results for Wubble World 3D and 2D are shown in Table 4. For both datasets, the classification accuracies of learned signatures are very high. Most of the errors occur when one activity contains a part that looks like another. For example, in Wubble World 3D, some activities were mistaken for *approach*, as all activities start with the wubbles approaching objects.

	Mean	SD
WW 3D	98.73%	2.7
WW 2D	95%	7.03

Table 4: Mean and standard deviation classification accuracy for CAVE signatures for the datasets described in the text.

7. INFERRING HIDDEN STATE

We are interested in activities that have elements that are not accessible to an observer. For such activities, the observable elements must serve as a basis for inferring unobservable elements. For example, if one sees a player moving toward a goal in a hurry, one might infer that the player has seen the goal and wishes to acquire it.

Figure 5 illustrates how this works. We assume that a player has access to its own internal state and learns signatures that include internal state (illustrated as dark bars in Fig. 5). Then, the observable activities of *another* player, denoted **S** in Figure 5, are classified, and a signature is retrieved. Note that **S** contains only observable fluents (i.e., no dark bars) and the classification of **S** is done with only these observables. However, the signature that is retrieved contains internal state, and these are attributed to or "projected onto" the other player.

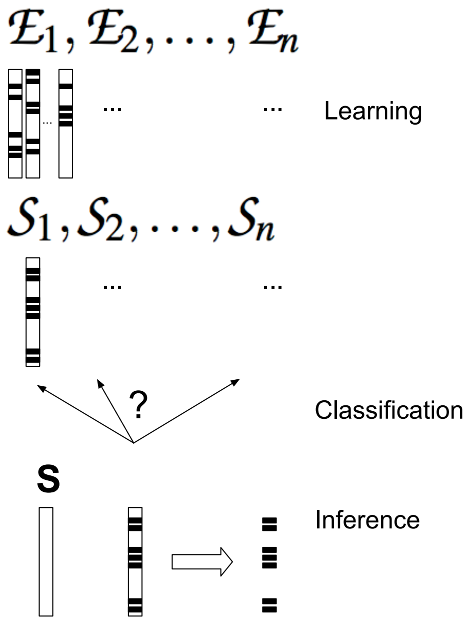
$\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$
Learning
$\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$
?
Classification
S
Inference

Figure 5: The learning, classification and inference pipeline

We implemented this idea in Wubble World 2D as follows:

1. Build wubbles that have externally observable fluents P_e and internal fluents that are not observable, P_i (the dark bars in Fig. 5). All fluents are both influenced by their environments and influence the wubbles' activities;
2. Make the wubbles engage in activities in which P_e and P_i get a workout.
3. Learn signatures for these activities. Note that the signatures will include both P_e and P_i and Allen relations between them.
4. When observing a novel activity, to infer its hidden content (i.e., the P_i fluents)
   (a) transform it into a qualitative sequence, **S**, based only on observable (P_e) fluents,
   (b) find the signature that best matches the qualitative sequence based only on observables
   (c) read out the unobservable, P_i, parts of the signature as the inferred fluents

7.1 Inference

For each learned signature, we select the $\alpha = 10$ most frequent Allen relations that contain at least one unobservable fluent. For example, the ten most frequent Allen relations in the *chase* signature that include at least one hidden fluent are shown in Table 5. These are relations that will be inferred if an episode is classified as *chase*. We call them the "will be inferred" relations.

```
((novel(agent1 agent2) s (goal(agent1) PursueGoal)))
(((goal(agent1) PursueGoal) c (distance(agent1 agent2) 1)))
(((goal(agent1) PursueGoal) c (arousal(agent1) up)))
(((goal(agent1) PursueGoal) c (distance(agent1 agent2) down)))
(((goal(agent1) PursueGoal) c (energy(agent1) stable)))
(((goal(agent1) PursueGoal) c (valence(agent1) stable)))
(((state(agent1) charge) c (arousal(agent1) up)))
(((state(agent1) charge) c (distance(agent1 agent2) down)))
(((state(agent1) charge) c (energy(agent1) stable)))
(((goal(agent1) PursueGoal) c (heading(agent2) up)))
```

Table 5: The top 10 Allen relations that include hidden fluents in the signature for *chase*. In the example, *s* and *c* correspond to the Allen relations *starts-with* and *contains*.

To test the inference of hidden Allen relations, a qualitative sequence **S** is selected at random, then classified as one of six activities. Next, we calculate the degree of overlap between the "will be inferred" relations for the classified activity and the *actual* Allen relations that include at least one hidden fluent in **S**.

The results are shown in Table 6. The diagonal entries are cases where the qualitative sequence **S** was classified correctly and the off-diagonal entries are for cases where **S** was classified incorrectly. The number in the cells of Table 6 are the proportions of "will be inferred" relations that are actually in **S**. For example, when **S** is an instance of chasing, and is correctly classified as such, then 98% of the relations in Table 5 are actually in **S**. (The reason that the proportion is 98%, when there are only ten relations to be found, is that 98% is the *average* proportion of these ten relations that were found in repeated iterations in a cross-validation study.)

So, when a qualitative sequence **S** is classified correctly, there is a very high probability that its associated "to be inferred" relations actually occur. Interestingly the specificity of these inferences is also high: when **S** is classified incorrectly, its inferred relations are rarely correct. The confusions, however, are telling. When **S** is kicking a column, and it is incorrectly classified as kicking a ball, then 81% of its inferred relations are correct. This occurs because kicking a column and kicking a ball have a lot of the same affective components. However, if kicking a column is incorrectly classified as a chase, then none of the inferred relations will be correct.

This experiment and others with the 3D wubbles strongly support the claim that if an agent can correctly classify the activity of another agent, it can select the α most frequent relations as inferred hidden relations, and they will be correct with high probability.

8. RELATED WORK

The problem that motivated this work was illustrated decades ago in a study by Heider and Simmel [14]. These investigators showed a short stop-action animation of three geometric shapes interacting and asked subjects to describe what was going on. Invariably, the subjects told elaborate stories that attributed agency, goals and affective states to the geometric shapes. How is it that seeing one geometric shape move rapidly toward another, which recoils, gives rise to the words "attack" and "hit" and "bully" and the like?

In [5], human subjects controlled simulated insects on networked machines to perform a variety of activities. The authors trained a three-layer neural network to classify the activities based on observable information, such as relative distance between the insects, relative velocity, absolute velocity, relative heading, etc. The trained neural network and heuristic based algorithm outperformed human subjects on a classification task. Intention was not explicitly modeled in this system even though they system could detect it through motion cues. In a followup study [3], Barrett et al. provide results that suggest that our ability to infer the intentionality of agents (even simple triangles) appears as young as 4 years of age, and we become more adept at it as we age. Given a forced choice task with similar behaviors to the original study, subjects correctly identified the right behavior in 80% of the samples.

Baker et al. [2] demonstrate that goal inference can be accomplished by Bayesian inverse planning. Furthermore, the learned Bayesian model accords well with human judgments of the intentions of a simple goal seeking agent.

Signatures and signature learning are a novel approach to learning models of activities, but there are many related approaches. Firoiu and Cohen [13] describe a system composed of HMMs each trained on a batch of robot episodes. The robot represents its experiences with the HMM and learns the number of hidden states within the HMM via HMM state splitting. In [23], the authors perform an experiment in which time series recordings are made of the sensors of a Pioneer-1 robot performing different actions. The resulting time series are clustered using Dynamic Time Warping as a distance metric between the time series. Although they do not explicitly model entire activities, the authors show that the clusters found by agglomerative clustering match those constructed by an expert. Similarly in [26] the authors propose to cluster the sensor recordings for a mobile robot. In this instance, they first look for abrupt changes within sensor values as indication that an event has occurred, and after finding events they cluster the time series to generate signatures for the interaction.

More recently, Crick and Scasselati developed a system for recognizing activities and simultaneously the intentions of the actors, all from a folk physics-based interpretation of position and motion [11, 12]. This work is an extension of previous work where the authors developed a system that could correctly identify which player was "it" in a game of tag [10]. They found that when the robot was able to participate in playground games (like tag), then it could recognize the intentions of other agents towards itself much more quickly than if it was simply a passive observer. This was because it was able to test its hypotheses about the other agents intentions towards it by approaching them.

Pautler et al. [25] propose a similar folk physics-based interpretation of intention. In addition to recognizing intentions, the authors also focus on developing agents capable of extracting explanations for these intentions. Explanations are generated when the agent's

	Signatures					
	column	ball	chase	flee	fight	eat
column	0.97	0.81	0.00	0.01	0.23	0.13
ball	0.18	0.97	0.00	0.01	0.16	0.13
chase	0.01	0.07	0.98	0.05	0.04	0.02
flee	0.05	0.17	0	0.97	0.06	0.11
fight	0.13	0.19	0.63	0.06	0.96	0.19
eat	0	0.02	0	0.03	0	0.97

Table 6: A matrix showing the percent overlap between the α most frequent hidden relations in the signature and the hidden relations that exist in the test set, but are not observable for the 2D wubble dataset.

assumptions about the environment are violated and other agents are not acting in accordance to their perceived intentions. In both [12] and [25], the ability to recognize intentions and the set of intentions that can be recognized are built into the agent, not learned.

In [7], Breazeal describes an implementation of facial imitation, which is much more focused than general activity learning, but they have similar motivations that guided the implementation. They argue that robots need to infer the mental states of others in order to interact with them in a humanlike way, and they believe that this will come from the observable behavior.

Real time behavior recognition of video game opponents has received a fair amount of attention in the past few years. Kabanza et al [16] adopt a hidden-markov model (HMM) based approach to plan recognition of human opponents in the real-time strategy game Starcraft. The authors constructed a library of potential Hierarchical Task Networks, or plans, that can potentially be executed by the human opponent. As the game unfolds the system selects the plan with the highest probability and from there infers the intentions of the player. In order to avoid developing a complete library of plans, Cheng and Thawanmos [8] recommend a case-based approach to plan recognition. This method treats the world more similar to our PMTS representation, but makes no attempt to extract a common pattern for a plan.

Recent research has proposed the mirror neuron system within humans as a neural mechanism for recognizing and understanding the intentions of other people [15, 1]. We do not claim that signatures are a software version of mirror neurons, but they certainly are a representation of both external and internal aspects of behavior and can effectively infer the latter given the former.

9. DISCUSSION AND FUTURE WORK

The experiments reported here, and others in [19], demonstrate remarkably high classification accuracies for signatures. These results are not limited to Wubble World, but are equally robust for classifying handwritten characters based on the dynamics of pen movements, as well as multivariate time series with carefully controlled covariances between the variables. We know from these experiments that writing PTMSs as qualitative sequences of Allen relations increases classification accuracy. Apparently representing a pattern of fluents as a sequence of Allen relations between fluents captures much of the structure of activities, and seems robust to the errors that sequence alignment can potentially introduce while constructing a signature.

Although the CAVE algorithm was originally designed for propositional sequences, it works well on real-valued sequences by first transforming them to propositional form, apparently with little loss of accuracy.

CAVE seems a promising algorithm for learning signatures of activities in online games much more sophisticated than Wubble World. Unlike some "bottom up" algorithms that learn by composing larger patterns from smaller ones [4, 24], the computational complexity of CAVE depends only weakly on the number of propositions or real-valued fluents that are used to describe activities. More propositions means more Allen relations, which means longer qualitative sequences, and a somewhat harder job for sequence alignment. In contrast, bottom up approaches quickly drown in the combinatorics of pairs, triples, and n-tuples of propositions, each pair of which can be in seven Allen relations. CAVE can easily handle activities that involve, on average, more than a thousand fluents and half a million Allen relations.

Although CAVE is currently a supervised algorithm, meaning that someone must label the instances of activities, an unsupervised version has been implemented and works well. The basic idea is to cluster qualitative sequences with the *k*-means algorithm. Each cluster centroid becomes a signature, then each qualitative sequence "moves" toward the centroid it matches best (according to sequence alignment) and then the centroids (signatures) are recalculated. This process continues until the qualitative sequences no longer move from one signature to a closer one.

We are enthusiastic about unsupervised learning of signatures in the context of games because it gives us a way to learn *emergent* activities that were not directly programmed or anticipated by game designers. As games increase in complexity, and especially as more human players and NPCs interact, the ability to find unanticipated activities in gameplay may help to explain why some games are more engaging and successful than others.

10. ACKNOWLEDGMENTS

This work is supported by a grant from the Office of Naval Research entitled "Cognitive Semantics for Soar Wubbles." Additional funding was provided by DARPA under contracts 13251 and W911NF1020064. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research or DARPA.

11. REFERENCES

[1] Z. K. Agnew, K. K. Bhakoo, and B. K. Puri. The Human Mirror System: A Motor Resonance Theory of Mind-Reading. *Brain research reviews*, 54(2):286–93, 2007.

[2] C. L. Baker, R. Saxe, and J. B. Tenenbaum. Action Understanding as Inverse Planning. *Cognition*, 113(3):329–49, 2009.

[3] H. Barrett, P. Todd, G. Miller, and P. Blythe. Accurate Judgments of Intention From Motion Cues Alone: A Cross-Cultural Study. *Evolution and Human Behavior*, 26(4):313–331, 2005.

[4] I. Batal, L. Sacchi, R. Bellazzi, and M. Hauskrecht. Multivariate Time Series Classification with Temporal Abstractions. *Florida Artificial Intelligence Research Society Conference*, 2009.

[5] P. W. Blythe, P. M. Todd, and G. F. Miller. How Motion Reveals Intention: Categorizing Social Interactions. In G. Gigerenzer and P. M. Todd, editors, *Simple Heuristics That Make Us Smart*, pages 257–285. Oxford University Press, New York, 1999.

[6] C. Breazeal. Emotion and Sociable Humanoid Robots. *International Journal of Human-Computer Studies*, 59(1-2):119–155, 2003.

[7] C. Breazeal, D. Buchsbaum, J. Gray, D. Gatenby, and B. Blumberg. Learning From and About Others: Towards Using Imitation to Bootstrap the Social Understanding of Others by Robots. *Artificial life*, 11(1-2):31–62, 2005.

[8] D. C. Cheng and R. Thawonmas. Case-Based Plan Recognition for Real-Time Strategy Games. In *Proceedings of the 5th Game-On International Conference*, pages 36–40, 2004.

[9] D. D. Corkill. Blackboard Systems. *AI Expert*, 6(9):40–47, 1991.

[10] C. Crick, M. Doniec, and B. Scassellati. Who is IT? Inferring Role and Intent from Agent Motion. In *Proceedings of the 6th IEEE International Conference on Development and Learning*, pages 134–139, 2007.

[11] C. Crick and B. Scassellati. Inferring Narrative and Intention from Playground Games. In *Proceedings of the 7th IEEE*

Conference on Development and Learning, pages 13–18, 2008.

[12] C. Crick and B. Scassellati. Controlling a Robot with Intention Derived from Motion. *Topics in Cognitive Science*, 2(1):114–126, 2010.

[13] L. Firoiu and P. R. Cohen. Abstracting from Robot Sensor Data using Hidden Markov Models. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 106–114, 1999.

[14] F. Heider and M. Simmel. An Experimental Study of Apparent Behavior. *The American Journal of Psychology*, 57(2):243, 1944.

[15] M. Iacoboni, I. Molnar-Szakacs, V. Gallese, G. Buccino, J. C. Mazziotta, and G. Rizzolatti. Grasping the Intentions of Others with One's Own Mirror Neuron System. *PLoS biology*, 3(3):e79, 2005.

[16] F. Kabanza, P. Bellefeuille, F. Bisson, A. R. Benaskeur, and H. Irandoust. Opponent Behaviour Recognition for Real-Time Strategy Games. In *AAAI Workshops*, 2010.

[17] W. Kerr, P. Cohen, and N. Adams. Beyond Pattern Mining for Interval Time Series. *Advances in Data Analysis and Classification*, 2011 (submitted).

[18] W. Kerr, P. Cohen, and Y.-h. Chang. Learning and Playing in Wubble World. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 66–71, 2008.

[19] W. N. Kerr. *Learning to Recognize Agent Activities and Intentions*. PhD thesis, University of Arizona, 2010.

[20] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing SAX: a Novel Symbolic Representation of Time Series. *Data Mining and Knowledge Discovery*, 15:107–144, 2007.

[21] M. Masuch, K. Hartman, and G. Schuster. Emotional Agents for Interactive Environments. In *Fourth International Conference on Creating, Connecting and Collaborating through Computing (C5'06)*, pages 96–102, 2006.

[22] S. B. Needleman and C. D. Wunsch. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of molecular biology*, 48(3):443–453, Mar. 1970.

[23] T. Oates, M. D. Schmill, and P. R. Cohen. A Method for Clustering the Experiences of a Mobile Robot that Accords with Human Judgments. In *AAAI-00*, 2000.

[24] D. Patel, W. Hsu, and M. L. Lee. Mining Relationships Amont Interval-based Events for Classification. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD'08)*, pages 393–404, 2008.

[25] D. Pautler, B. Koenig, B. Quek, and A. Ortony. Inferring Intention and Causality from 2D Animations. *Submitted*, 2009.

[26] M. T. Rosenstein and P. R. Cohen. Continuous Categories for a Mobile Robot. *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 634–640, 1999.