Grounding Abstractions in Predictive State Representations

Brian Tanner and **Vadim Bulitko** and **Anna Koop** and **Cosmin Paduraru**
Department of Computing Science
Edmonton, Alberta, Canada T6G 2E8
{btanner, bulitko, anna, cosmin}@cs.ualberta.ca

Abstract

This paper proposes a systematic approach of representing abstract features in terms of low-level, subjective state representations. We demonstrate that a mapping between the agent's predictive state representation and abstract features can be derived automatically from high-level training data supplied by the designer. Our empirical evaluation demonstrates that an experience-oriented state representation built around a single-bit sensor can represent useful abstract features such as "back against a wall", "in a corner", or "in a room". As a result, the agent gains virtual sensors that could be used by its control policy.[1]

1 Introduction

It is often useful for intelligent agents to reason at a level more abstract than their perceptions. Abstractions aggregate many sensory situations into a single configuration of abstract features, allowing behavior expressed in abstract terms that generalize across similar situations. This generalization allow humans designers to encode control policies closer to the level of human reasoning. For instance, using abstractions, an agent's designer can define a policy such as "avoid navigating through open areas" or "don't stop near a doorway".

The desire to write control policies in abstract terms is often thwarted because the policies must be implemented over the set of state variables and sensor readings available to the agent. In simulation, one option is to directly augment the environment with high-level information (e.g., each tree in a forest may be annotated as a possible cover for a soldier agent) [Paull and Darken, 2004]. If access to the environment is not available, a programmer may write code to map low-level state (e.g., coordinates and obstacle detectors) to higher-level abstract terms (e.g., being in a narrow corridor where the unit cannot move sideways) [Orkin, 2005].

As realism of simulations used for real-life training and education [Dini *et al.*, 2006] increases, both approaches are becoming expensive. Indeed, the number of abstract features and the amount of underlying software engineering needed to define them has been compared to construction of the Tower of Babel [McDonald *et al.*, 2006]. Likewise, manually annotating modern games and simulations is prohibitively time-consuming and error-prone.

This paper makes several contributions to the aforementioned problems. We propose an automated approach for learning a mapping from agent state representations to human-defined abstractions using a small set of data labeled by a human expert. We use a "I know it when I see it" approach to defining abstract features; instead of programmatically defining or assigning values to the abstract features, the human expert examines a number of examples and labels them appropriately. A machine learning classification algorithm, such as C4.5 [Quinlan, 1993], can then create a classifier that will appropriately label novel future experiences.

For this approach to be useful, it is important that the agent's state representation allows the values of the abstract features to generalize well between the labeled examples and unseen data. We argue that agents using subjective, grounded representations are well suited to making these generalizations. A subjective, grounded approach represents the agent's state in terms of experience with its sensors and effectors. Specifically, we use a predictive representation of state: the agent's state is stored as a vector of probabilities of outcome sensations given various sequences of action [Littman *et al.*, 2002]. We believe that both concise and vague abstractions can be characterized by patterns of interaction: abstractions learned in terms of a predictive representation are portable to novel situations with similar patterns of interaction.

In a small navigation domain, we demonstrate that an agent using a simple classification algorithm can learn to identify intuitive abstractions like BACK-TO-WALL, CORNER, NARROW-CORRIDOR, and ROOM using both subjective and objective state representations. Further, we show that in the subjective case, the learned classifier allows the agent to identify these abstractions in a novel environment without any additional training.

2 Background and Related Work

In a typical situated agent framework, the agent (decision maker) interacts with an environment at a series of time steps. The agent receives perceptions or *observations* from the environment at each moment and responds with *actions* that may affect the environment. In this paper, we model the world as a

[1]This work was supported in part by RLAI, NSERC, iCORE, and Alberta Ingenuity.

partially observable Markov Decision Process (POMDP). We assume that at each discrete time step, there is a set of data (called the *state*) that is sufficient to make accurate probabilistic predictions of future experience (the Markov assumption). The state is not directly observable by the agent, the observation received from the environment may correspond to multiple different states.

The standard POMDP model represents the agent's current state as a probability distribution over a set of unobservable situations that the agent may be in. Each of these unobservable situations is called a *nominal state*; distributions over nominal states are called *belief states*. POMDPs use the conditional probability rule, a nominal state transition model, and an observation model to update the belief state with each new action and observation. The transition and observation models are usually determined by a human expert and then later these parameters are optimized with data.

Alternatively, history-based methods also model partially observable environments by considering either a fixed or variable length window of recent interactions with the environment [McCallum, 1995].

Predictive state representations (PSR) are a third type of representation that model both observable and partially-observable environments [Littman *et al.*, 2002]. Instead of focusing on previous interaction with the environment, the predictive model represents state as a set of predictions about future interactions. In a PSR, the agent's state is represented as answers to questions about future experience in the world. For example, one element of the state representation might be "What is the probability of receiving the sequence of observations $(o_1 o_2 \ldots o_i)$ if I take the sequence of actions $(a_1 a_2 \ldots a_j)$?". Singh *et al.* [2004] proved that linear predictive state representations can represent all environments that can be represented by a finite-state POMDP or a finite-length history method.

A PSR has two components: structure and parameters. The structure is the set of events that the PSR makes predictions about, called the *core tests*. Each core test predicts the probability of observing a particular sequence of observations if a particular sequence of actions is chosen by the agent. The parameters are used to update the core test values after each new action and observation. Algorithms have been proposed for learning the PSR parameters [Singh *et al.*, 2003] and discovering and learning PSR structure and parameters together [McCracken and Bowling, 2006].

3 Abstractions

A state representation stores each state as a configuration of variables. In a traditional MDP, the state is often represented by a single discrete variable with a unique value for each state in the state space. In a POMDP, the state is represented by a vector of n continuous-valued variables $\in [0, 1]$, where n is the number of underlying nominal states. In a linear PSR, the state is also a vector of n continuous-valued variables $\in [0, 1]$, where n is the number of core tests.

In this paper, an abstraction is a many-to-one operator ϕ that maps several configurations of state variables to a single configuration of a different set of variables. For example, consider an MDP for a navigation task where the state is represented using three discrete variables, $\langle x, y, \theta \rangle$. The first two variables, $\langle x, y \rangle$ represent a location, while θ denotes rotation or orientation in a coordinate system. One abstraction ϕ_i might identify the NORTH-EAST-CORNER, defined as all states within 5 units of the $\langle 0, 0 \rangle$ corner of the environment. In this case:

$$\phi_i(\langle x, y, \theta \rangle) = \begin{cases} 1 & \text{if } x \in \{1, 2, 3, 4\} \text{ and } y \in \{1, 2, 3, 4\} \\ 0 & \text{otherwise} \end{cases}$$

We generally refer to the outcomes of abstractions as *abstract features*. Abstract features allow a control algorithm to make useful generalizations: to behave similarly in states that have different state variable combinations but similar abstract features. For example, our agent may know that NORTH-EAST-CORNERis a good place to recharge its batteries.

3.1 Representing Abstractions

Useful abstract features can often be invented by a human expert leveraging knowledge about a particular domain or task. No matter what the agent's representation, these abstract features are understood in the expert's terms. Thus, a mechanism must be found that reliably converts the agent's state representation into the abstract features. This is usually an extensive software engineering effort involving lengthy tuning via trial and error [Paull and Darken, 2004]. In games and virtual reality trainers the resulting abstraction is sometimes referred to simply as "sensors" [Orkin, 2005] or "situation awareness components" [Kunde and Darken, 2006].

The agent's state representation has a strong impact on the complexity and robustness of any script that converts the agent's state into abstract features. For instance, abstract features such as BACK-TO-WALL or NARROW-CORRIDOR are difficult to concisely define as a function of $\langle x, y, \theta \rangle$. These features can have value 1 in various disjoint locations in the map, making their values difficult to know without explicit reasoning about walls or other obstacles. On the contrary, a predictive representation that stores the state as predictions about future perceptions can succinctly capture these abstractions in terms of patterns of interaction with the environment.

3.2 Learning Abstractions

We propose a machine learning approach for automatically finding a mapping between an agent's state representation and an expert's understanding of abstract features. The expert can look at a number of situations and appropriately label the abstract feature values in each case. We call this set of labeled examples the *training set*. For example, in the case of our mobile agent, the expert looks at a map of a particular environment and labels examples of the agent being in a narrow corridor. The agent then uses an off-the-shelf machine learning algorithm such as a decision tree learner to learn a set classifier that takes state variables as inputs and emits abstract feature values as outputs.

4 Experimental Results

For the demonstration, we chose a simple model of a path-finding environment that allows us to evaluate our approach

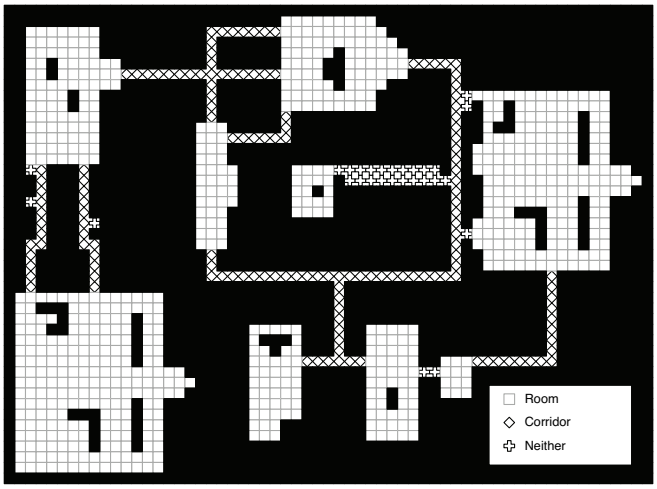

Figure 1: Map A, used to train the agent's virtual sensors to recognize various abstract features.

in detail. Specifically, our agent has four actions and a single, binary observation. It is situated in a discrete grid environment where some cells are open and some are blocked. The agent occupies exactly one cell and has four possible orientations (up, down, left, and right). The agent's actions are: go forward (F), turn left (L), turn right (R) , and go backward (B). If the agent moves forward or backward and collides with an obstacle, the agent's position remains unchanged. At any time step, the agent's sensor reports 1 if the agent faces a blocked cell and 0 otherwise.

We chose to make the grid world deterministic: the actions always succeed. Determinism greatly simplifies the performance measures and visualization of our experiments without weakening the results.

We use two different maps in our experiments. The first, Map A (Figure 1), has a total of 943 reachable grid locations with four orientations each, for a total of 3,772 states. The second environment, Map B (Figure 2), has a total of 1,063 reachable grid locations with four orientations each, for a total of 4,252 states.

State Representations

The agent uses two different state representations, one that is based on objective coordinates $\langle x, y, \theta \rangle$ and another that is a predictive representation. As we have discussed, the objective $\langle x, y, \theta \rangle$ coordinate system is useful for characterizing certain abstractions that are tied to physical location on a map. If certain areas of the map have a certain abstract feature value (such as "in a room"), then a classifier learned over the $\langle x, y, \theta \rangle$ state will correctly classify other states within the same room with very few training examples. However, if the agent has seen no examples of a particular room, the classifier will have no means to identify it even if it is identical in size and shape to another room seen previously.

The agent's predictive representation is a vector of test values. Each test value corresponds to the probability of the test's observations matching those generated by the environment, conditioned on the test's actions being executed.

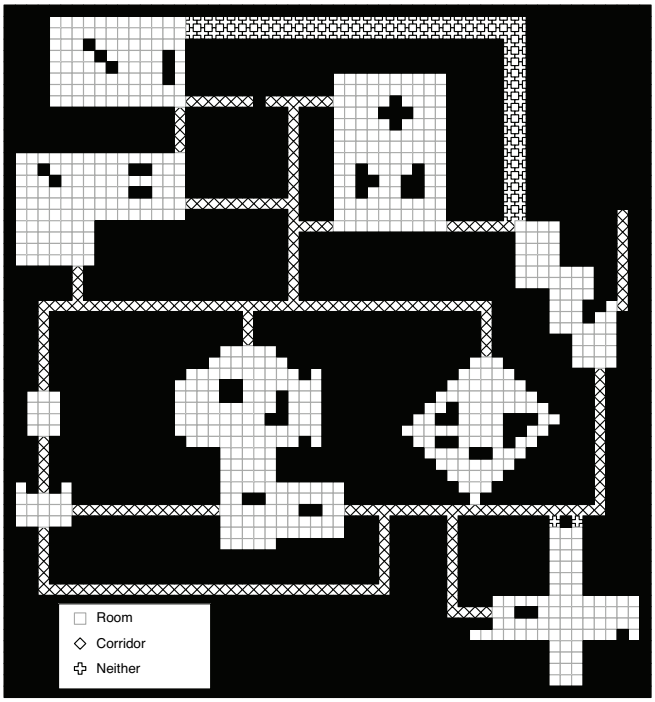

Figure 2: Map B, used for testing the agent's virtual sensors.

As an example, consider a predictive state representation with only two core tests. Core test t_1 is defined as seeing (1) after executing action (F). The value of t_1 will be 1 if and only if the agent is facing an obstacle or can reach it within one step forward. Core test t_2 is defined as seeing $(1, 1, 1)$ while executing (R, R, R). The value of t_2 will be 1 if the agent had obstacles on its right, behind it, and on the left side of it. In a deterministic environment any core test has the value of 1 or 0. Thus, this micro representation has four states: $[t_1 = 0, t_2 = 0]$, $[t_1 = 0, t_2 = 1]$, $[t_1 = 1, t_2 = 0]$, and $[t_1 = 1, t_2 = 1]$. These tests are compactly denoted as F1 and R1R1R1.

We construct the PSR representation from an accurate POMDP model of the environment using an adaptation of an algorithm introduced by Littman *et al.* for creating linear PSRs from POMDP models [2002]. The PSR construction algorithm is a polynomial algorithm that uses depth-first search to find the set of PSR core tests and their values in all of the POMDP nominal states. Our straightforward adaptation of Littman *et al.*'s algorithm uses breadth-first rather than depth-first search, yielding considerably shorter core tests.

Abstract Features

We chose four abstract features for our experiments, ranging from the formally and succinctly defined BACK-TO-WALL to the more vague ROOM:

1. BACK-TO-WALL is a feature that is on when the agent would hit a wall if the backward action was taken, and off otherwise.
2. CORNER is a feature that is on in every grid location that has a corner where two blocked grid cells meet.

3. ROOM is a feature that is on when the agent is in a room. This is a difficult feature to program manually due to the fact that "room" is a vague human-level concept. In our approach we circumvent this problem by simply allowing a human designer to label certain states as room without formally defining it (Figures 1 and 2).

4. NARROW-CORRIDOR is a feature that is on in all 1-cell wide corridors, labeled by a human designer. This and the ROOM abstraction are not labeled entirely consistently within or across the two environments because of subjective decisions made by the human designer.

5. Finally, some grid cells are labeled as "neither" because the expert knew they should not be classified as ROOM or NARROW-CORRIDOR, but there was no other abstract feature to label them with.

4.1 Experiment 1: $\langle x, y, \theta \rangle$ vs. PSR on Map A

In this experiment, we compare the classification accuracy of decision trees learned from varying amounts of training data within a single map (Map A) using both representations. We hypothesize that with few training examples, the $\langle x, y, \theta \rangle$ representation will generally perform better on abstract features that are well characterized by spatial locality (e.g., the ROOM feature). Conversely, we expect the predictive representation to perform well for abstract features that are well characterized by patterns of interaction (e.g., BACK-TO-WALL).

Experimental Method

All states in Map A are labeled as either positive or negative examples of each abstract feature. The agent then uses the J48 (similar to C4.5 [Quinlan, 1993]) classification algorithm to create a separate decision tree for each abstract feature in both representations using the WEKA machine learning software package [Witten and Frank, 2005]. In all of our experiments, the reported results are for a PSR created using some subset of the core tests generated by our POMDP to PSR conversion algorithm. This subset is the first k tests found by the algorithm. Using the full set of thousands of core tests is simply not necessary. For various sample sizes $P = \{1\%, 10\%, 75\%\}$, P of the examples are randomly sampled from Map A and put into the training set, the other examples are left out and used as a test set. The classifier learned on the training set is then applied to the test set and classification accuracy is measured. The proportion of positive examples to negative examples is often quite biased, so all or our results report the accuracy of a baseline classifier that simply predicts the majority class from the training data. Precision, recall, and specificity statistics were also measured but are not reported as they exhibited similar trends to the accuracy. This procedure is repeated 10 times and the mean and standard deviation of the accuracies is reported.

This experiment was performed on all four abstract features, here we present the two results that represent either end of the performance spectrum: ROOM and BACK-TO-WALL, shown in Table 1.

	% Data Used for Training		
	1%	10%	75%
ROOM			
Baseline	82.0% ± 0.0	82.0% ± 0.0	82.0% ± 0.0
$\langle x, y, \theta \rangle$	80.1% ± 3.9	95.8% ± 2.6	99.9% ± 0.2
750 tests	90.7% ± 2.5	94.0% ± 1.0	96.7% ± 0.5
BACK-TO-WALL			
Baseline	79.1% ± 0.0	79.1% ± 0.0	79.1% ± 0.0
$\langle x, y, \theta \rangle$	76.8% ± 5.2	79.1% ± 0.1	83.6% ± 1.7
750 tests	93.0% ± 3.7	99.0% ± 0.5	99.6% ± 0.2

Table 1: Accuracy of a classifier learned with $\langle x, y, \theta \rangle$ representation against a classifier learned with a 750 core test PSR on the ROOM and BACK-TO-WALL abstract features. Test and training data sets are disjoint.

Discussion

As expected, the $\langle x, y, \theta \rangle$ classifier was generally more accurate than the PSR classifier on the ROOM feature. Surprisingly, for small amounts of data, the PSR classifier is actually more accurate. This indicates that the subjective state representation is able to identify the most canonical examples of the ROOM feature with very little data.

When testing the BACK-TO-WALL feature, the PSR classifier was substantially better than the $\langle x, y, \theta \rangle$ classifier. BACK-TO-WALL can be characterized by a very simple pattern of interaction even though it occurs in various locations on the map in all orientations. This makes it an ideal candidate for being represented in a subjective representation instead of a coordinate-based one. The decision tree that is learned for the PSR classifier with 750 core tests is noticeably lengthier (31 leaf nodes) than a manually crafted classifier would be (3 leaf nodes). We believe this to be due to overfitting. When the same experiment is run on the BACK-TO-WALL feature using fewer core tests (250 or fewer), the 3-node decision tree is found (Figure 3).

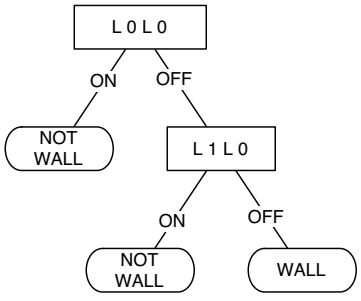

Figure 3: Decision tree classifier for the BACK-TO-WALL feature learned using a predictive state representation with 250 or fewer core tests.

4.2 Experiment 2: Transfer using PSR

In this experiment, we investigate the degradation in classification accuracy when a decision tree learned on training data from Map A is used to predict the abstract feature values on

Map B. Good performance in Map B is evidence that a classifier learned over a subjective state representation can be successful with novel experiences that share patterns of interaction with previous experiences. Note that we will not compare the PSR classifier's ability to transfer to $\langle x, y, \theta \rangle$ classifier in this experiment because the $\langle x, y, \theta \rangle$ decision tree has no basis for being used in a different map. Indeed, $\langle x, y, \theta \rangle$ locations for the abstract features are highly map-specific and not even translation or rotation invariant.

We perform this set of experiments using varying amounts of core tests as features for the classifier. We anticipate that some abstractions (BACK-TO-WALL and CORNER) will be easily identified with a small number of short tests as features. The vague, larger area abstractions like ROOM and NARROW-CORRIDOR may require more, longer tests. If this hypothesis holds, accuracy on the ROOM and NARROW-CORRIDOR abstractions should be worse for classifiers built using fewer core tests, and better when more core tests are used.

Experimental Setup

All states in Map A and B are labeled as either positive or negative examples of each abstract feature. Like the previous experiment, the agent then uses the J48 algorithm to induce decision tree classifiers from the training set, in this case all of the states from Map A. The decision tree is then tested on all examples from Map B. Because the agent has access to all of the available training and testing data, only a single learning trial is used and no standard deviation values are reported in Table 2. We present the results of using 10, 50, 250, and 750 core tests.

The results labeled Baseline correspond to predicting the value of the abstract feature that occurred most frequently in the training data: it is the accuracy that a classifier would get with no information about each training example except its label.

	ROOM	CORRIDOR	BACK-WALL	CORNER
Baseline	70.1%	78.9%	75%	90.6%
10 tests	87.1%	96.2%	100%	98.6%
50 tests	88.3%	97.6%	100%	100%
250 tests	91.5%	97.6%	100%	99.2%
750 tests	94.8%	97.6%	99.5%	99.8%

Table 2: Accuracy of PSR-based decision trees when training on Map A and testing on Map B with various amounts of tests.

Discussion

The transfer results, shown in Table 2, show the accuracy of the decision trees for all four abstract features when tested on Map B. All of the accuracies are very good, close to perfect with the exception of the ROOM abstraction. ROOM is particularly hard, not only because it is vague, but because the ROOM feature is locally aliased. That is, there are places in Map B labeled as not ROOM that look very similar to areas in Map A labeled as ROOM. This is evident in the bottom-left and middle-right rooms in Figure 1. Both of these rooms have sections that look locally very much like the double wide area from the top of Map B in Figure 2, which is labeled as not ROOM.

The results in Table 2 are produced using a classifier learned over different numbers of core tests. These tests are the first k found by a breadth-first search, meaning they may be quite short and ask questions only about states in the immediate neighborhood. Most of the abstract features (BACK-TO-WALL, NARROW-CORRIDOR and CORNER) are well characterized by short tests as evidenced by their high accuracy with only a 10 test PSR.

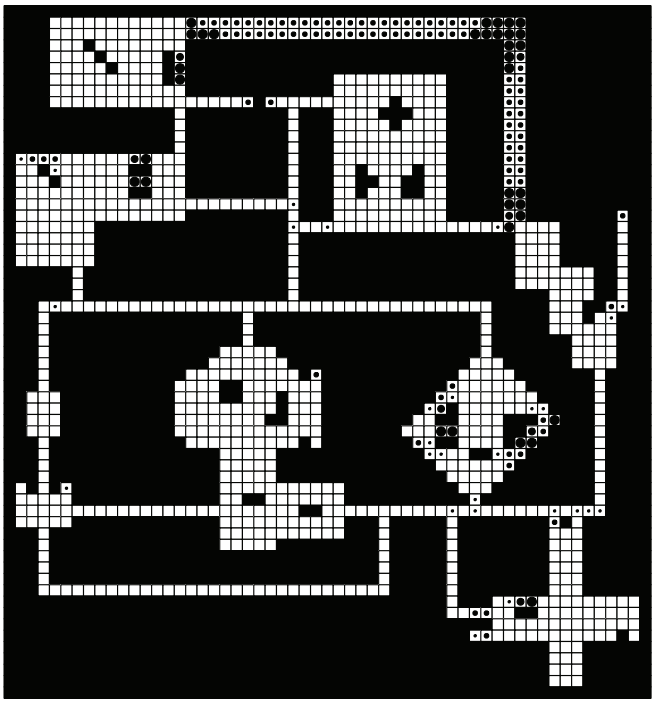

Figure 4: Errors identifying the value of the abstract feature ROOM. Grid locations with marks indicate errors; ranging from small circles (error in a single orientation) to large circles (error in all four orientations).

The bottom rows in Table 2 (50, 250, and 750 tests) provide empirical support for the conjecture that larger area abstractions may need more, longer core tests. Classification accuracy of the ROOM abstraction increases substantially as more core test values are provided to the classification algorithm. The number of leaf nodes in the decision tree grows from 9 leaves with 10 core tests, to 20, 44, and finally 55 with 50, 250, and 750 core tests respectively. This seems to indicate that it is not the sheer volume of tests that is improving the quality of the classifier: which tests are used makes a difference in the accuracy of the classifier.

A small amount of classification accuracy was lost as we moved up to 750 core tests, presumably because the low ratio of features to training examples allowed the classifier to overfit the training data.

Figure 4 shows the location of misclassified ROOM feature in Map B. The decision tree induced from 250 tests makes mistakes in cells that appear locally similar but have the opposite labeling in Map A. The decision tree also misclassifies certain

states that are not similar to anything seen in Map A.

5 Limitations

Our results provide evidence that subjective representations have desirable generalization properties for representing abstractions. However, our experiments were limited to a simple grid world and a linear predictive state representation. Further substantiation of our claim will require wider experimentation with more abstractions, various environments, and more general subjective representations.

Our approach does not eliminate human experts: they are still required to identify which abstractions will be useful. This expert must also manually label a small amount of data before the learning algorithm can be applied. In the near future, an automated approach like reinforcement learning may be used to learn the control policies for these agents. When the expert is no longer creating control policies manually, it will not be necessary for these abstractions to have an interpretation suitable for humans. In this case, it would be very useful to have the agent automatically discover useful abstractions through its own experience [Grollman *et al.*, 2006].

6 Conclusion

In this paper we have proposed a method for automatically learning a mapping from low-level state representations to high-level abstract features. In particular, we have demonstrated that a subjective, predictive state representation can generalize from a small sample of designer-labeled examples to novel experience. Our experiments also show that a classifier learned over a predictive representation can effectively classify new experience in a different environment with similar high-level characteristics.

This work is a step along the path to the automatic construction of virtual sensors that will be useful for creating strong control policies. This generalization ability will enable the design of control policies applicable in a wide variety of situations, allowing for richer domains and more complex agents.

7 Acknowledgments

The authors gratefully acknowledge the ideas and encouragement they received from Richard Sutton, Mike Bowling, Dan Lizotte, Mark Ring and especially Martha Lednicky for helping compile the results.

References

[Dini *et al.*, 2006] Don M. Dini, Michael Van Lent, Paul Carpenter, and Kumar Iyer. Building robust planning and execution systems for virtual worlds. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment conference (AIIDE)*, Marina del Rey, California, 2006.

[Grollman *et al.*, 2006] D. H. Grollman, O. C. Jenkins, and F. Wood. Discovering natural kinds of robot sensory experiences in unstructured environments. *Journal of Field Robotics*, In Press, 2006.

[Kunde and Darken, 2006] Dietmar Kunde and Christian J. Darken. A mental simulation-based decision-making architecture applied to ground combat. In *Proceedings of the Behavior Representation in Modeling & Simulation (BRIMS)*, 2006.

[Littman *et al.*, 2002] Michael L. Littman, Richard S. Sutton, and Satinder Singh. Predictive representations of state. In *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[McCallum, 1995] Andrew Kachites McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, Rochester, New York, 1995.

[McCracken and Bowling, 2006] Peter McCracken and Michael Bowling. Online discovery and learning of predictive state representations. In *Advances in Neural Information Processing Systems 18*, pages 875–882, 2006.

[McDonald *et al.*, 2006] David McDonald, Alice Leung, William Ferguson, and Talib Hussain. An abstraction framework for cooperation among agents and people in a virtual world. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment conference (AIIDE)*, Marina del Rey, California, 2006.

[Orkin, 2005] Jeff Orkin. Agent architecture considerations for real-time planning in games. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment conference (AIIDE)*, 2005.

[Paull and Darken, 2004] Gregory H. Paull and Christian J. Darken. Integrated on- and off-line cover finding and exploitation. In *Proceedings of GAME ON conference*, 2004.

[Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA, USA, 1993.

[Singh *et al.*, 2003] Satinder Singh, Michael L. Littman, Nicholas Jong, David Pardoe, and Peter Stone. Learning predictive state representations. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 712–719, 2003.

[Singh *et al.*, 2004] Satinder Singh, Michael R. James, and Matthew R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *Proceedings of the Twenthieth Conference on Uncertainty in Artificial Intelligence*, pages 512–519, 2004.

[Witten and Frank, 2005] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition edition, 2005.