Efficient Value Function Approximation with Unsupervised Hierarchical Categorization for a Reinforcement Learning Agent

Yongjia Wang & John E. Laird

Computer Science and Engineering, EECS Department
University of Michigan, Ann Arbor, MI 48109, USA
yongjiaw@umich.edu, laird@umich.edu

Abstract

We investigate the problem of reinforcement learning (RL) in a challenging object-oriented environment, where the functional diversity of objects is high, and the agent must learn quickly by generalizing its experience to novel situations. We present a novel two-layer architecture, which can achieve efficient learning of value function for such environments. The algorithm is implemented by integrating an unsupervised, hierarchical clustering component into the Soar cognitive architecture. Our system coherently incorporates several principles in machine learning and knowledge representation including: dimension reduction, competitive learning, hierarchical representation and sparse coding. We also explore the types of prior domain knowledge that can be used to regulate learning based on the characteristics of environment. The system is empirically evaluated in an artificial domain consisting of interacting objects with diverse functional properties and multiple functional roles. The results demonstrate that the flexibility of hierarchical representation naturally integrates with our novel value function approximation scheme and together they can significantly improve the speed of RL.

1. Introduction & Background

In this paper, we consider the problem of how persistent intelligent agents can learn to make decisions in environments populated with diverse types of objects that have multiple functional roles. We assume that an object's perceptual features do not necessarily map directly to the functional utility of the objects. Thus, the agent must learn how perceptual features (such as the color and size of an animal) are related to the functional properties of the objects (such as whether the animal is dangerous or not), and how those functional properties relate to taking actions in the world (such as avoiding the animal or hunting it). Although these are fundamental characteristics of our world, environments with such characteristics have not been studied in machine learning. Furthermore, the agent's persistent, ongoing existence limits learning to be incremental and online.

Our approach is a unique synthesis of two machine learning approaches: unsupervised, hierarchical category learning and behavior adaptation based on reward signals using reinforcement learning (RL) [1].

1.1. Hierarchical Category Learning

Categorization provides an agent with means for generalizing its knowledge from a specific instance to a class of objects. COBWEB [2] and its variants are incremental hierarchical clustering algorithms, which have been successfully applied in various category learning tasks [3, 4]. The algorithm has two important features. First, it automatically organizes instances of objects, each represented by a feature vector, into a hierarchical category structure, as shown in Figure 1. The hierarchical structure makes the algorithm scale well with increasing object diversity. Second, the algorithm is incremental – it learns the hierarchical structure one instance at a time, and is robust to the order of inputs as well as noise in the features. These characteristics make it a good candidate for a persistent agent's category learning component. A previous version of the ICARUS cognitive architecture used a COBWEB-based system, called LABYRINTH [5] for its declarative learning and memory.

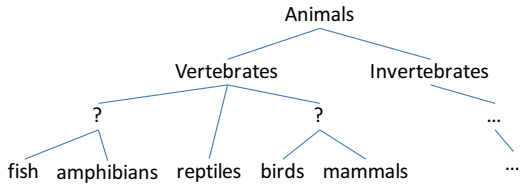

Figure 1: Example hierarchical category structure

978-0-7695-4191-4/10 $26.00 © 2010 IEEE
DOI 10.1109/WI-IAT.2010.16

To date, hierarchical category structures based on functional characteristics in the context of an RL agent has not been empirically investigated. Additionally, COBWEB did not address functional diversity where the same object is used for qualitatively different purposes. For example, an agent can choose to eat an apple, play with the apple, or throw the apple at its enemy. It is unlikely that the categorical structure for the purpose of "playing" is consistent with the categorical structure for the purpose of "eating" because they have distinctive, uncorrelated underlying functional features. Therefore, an agent must have the option to categorize an object based on specific functional context.

1.2. Reinforcement Learning within an Object Oriented Environment

RL is an established approach for problems that require incremental learning of control behavior while interacting with an environment with uncertainty. The problem is generally presented as learning an optimal policy based on reward signals. The policy is traditionally derived from a value function that maps state-action pairs to future expected rewards, where the state is an unstructured feature vector. Value function approximation in a high dimensional, continuous state space has been a challenge to RL. In an object-oriented environment, however, the state representation contains extra information (constraints) that can be used to improve learning. Diuk *et al*. [6] introduced an object-oriented representation framework to enable generalization and make reinforcement learning feasible in domains where the state space is too large for unstructured representations. However, their focus is on learning action models, and the framework assumes predefined object categories. In environments with diverse functional types of objects, the agent must continuously expand its category vocabulary, which presents a unique challenge to traditional reinforcement learning systems and is one motivation for integrating category learning with RL.

1.3. Our Approach

Figure 2 shows the dynamic data flow through the agent in our general approach. The two dashed boxes represent the learning components we are investigating in this paper. During interaction with the external environment, the category learning system predicts an object's functional category based on perceptual features. The functional category and other features, such as relational features among objects, are then composed into the final state representation used by the RL system for choosing actions.

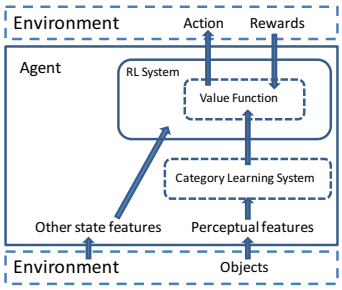

Figure 2: Abstract view of data flow through an agent

2. Evaluation Domain

We present our evaluation domain before describing our system, so that we can illustrate how the system works using a concrete example. Unfortunately, there are no existing benchmark tasks that reflect the challenge of object diversity we are pursuing. Therefore, we created an artificial domain to evaluate our system.

The domain has a simulated environment with a discrete time and discrete location grid world. The agent is equipped with different types of ranged weapons. To attack a prey, the agent must choose an appropriate weapon and distance from which to attack. The efficacy of a weapon depends on the functional properties of the weapon and the prey, as well as the distance to the prey. Moreover, the prey may detect the agent and become alerted before the attack action, which significantly reduces the success rate of hunting. In order to get close enough to the prey without alerting it, the agent can approach the prey from behind static obstacles. The probability of successfully moving towards the prey depends on the sensing capabilities of the prey and the types of obstacles between the prey and agent.

This domain captures the characteristics and challenges that we intend to address using our system. First, there are multiple interacting objects and multiple types of interaction: the interaction between prey and weapons, and the interaction between prey and obstacles. Second, there is diversity for each object type: prey, obstacles, and weapons all have diverse functional properties.

Figure 3 shows a scenario, where the prey (P) is in the middle and there are two types of static obstacles: bush (b) and rock (r), which can coexist in the same cell (b, r). For simplicity, the agent can only approach the prey from eight different directions. The first number for a path indicates bush distance to the prey and the second number indicates rock distance. 0 indicates that the object is absent or out of effective range. There is no additive effect from multiple occurrences of the same obstacle type, so if there are two bushes at both distance 1 and 2, the effect is the same as if there is one bush at distance 1.

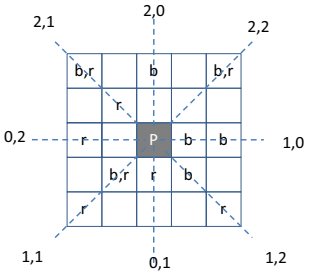

Figure 3: A hunting scenario

Figure 4 shows the model for interaction between objects in the domain. Grey boxes represent physical objects, white boxes with solid lines represent abstract quantities, white boxes with dotted lines represent probabilistic outcomes, large dashed boxes highlight local interactions among objects. Prey have different sensing properties which affect how likely it can sense the agent when blocked by a bush or rock, as highlighted by the "Prey Obstacle Interaction" box. The prey also has physical properties that affect how easily it can be shot and fatally wounded, as highlighted by the "Prey Weapon Interaction" box. Finally, distance to the prey affects both types of interactions. In the task, the agent must choose a weapon, an attacking path and a shooting distance. If the hunting is successful, the agent receives a positive reward

The functional properties of a prey and weapon are represented as continuous numbers in our environment model. We assume the agent can measure and internally represent these quantities as numeric features. Prey sensing properties consist of *sensitivity to bush* and *sensitivity to rock*. Prey physical properties consist of *health* and *size*. Weapon properties consist of *power* and *accuracy*. The probabilistic outcomes are determined by the numeric values of related features. For example, the probability of successfully approaching behind a rock is higher if the prey has higher *sensitivity to bush*. The probability of fatally wounding a prey is higher if the weapon has higher power or the prey has lower health.

We call the above numeric features *functional features*, as they represent an object's perceivable functional properties that have intrinsic meanings to the agent. For many functional features, the values are "expensive" to obtain in nature because they have to be tested out by actual interactions with the object. There are other non-functional *perceptual features* (not shown in the figure) of prey that are more easily perceivable, such as visual, smell, and sound features, and they can be correlated with certain functional features. These perceptual features are useful for the agent to predict the functional features when they are not directly available. For example, the agent cannot directly observe a prey's sensing properties before choosing the action, and has to make predictions based

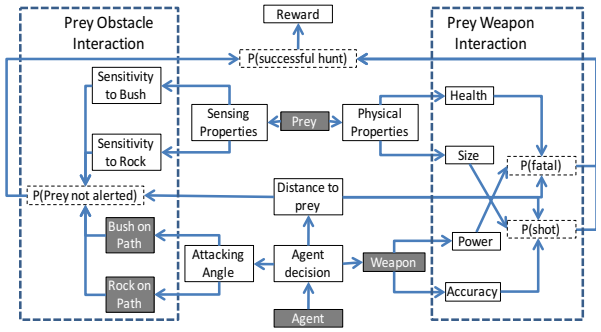

Figure 4: Interaction model of the domain

on correlated perceptual features such as the size and shape of eye, nose, and ear.

3. Implementation

Our approach integrates a category learning (CL) system based on COBWEB and the RL system within the Soar cognitive architecture [7, 8]. The RL system uses the hierarchical structure from the CL system as a representational basis for value function approximation.

3.1. Category Learning System

The default behavior in COBWEB is to create a single category hierarchy regardless of the functional context, which leads to poor performance when an environment is populated with objects with multiple functional roles. We have extended COBWEB to support categorization based on functional roles and functional features. For example, two prey that have similar *health* and *size* can have very different *sensitivity to bush* and *sensitivity to rock*. In our system, we have the option to equip the agent with prior knowledge so that it creates multiple functional category hierarchies for a single object (in this case the prey). In the hunting domain, the agent can create three separate hierarchies for: prey sensing categories, prey physical categories, and weapon categories. In Figure 5, a hierarchy is associated with a specific functional role, and is learned based on specific functional features. The hierarchical structure for prey physical categories is expanded in the figure. Although only two levels of the hierarchy are shown, the CL system can create a deeper hierarchy to partition the continuous multi-dimensional feature space. For integration purposes, each category node in the hierarchy is automatically assigned a unique internal identifier. For example, the prey physical category represented by '2' may have larger size and higher health than the category represented by '1'. '2.1', '2.2' and '2.3' represent three more specific categories under the

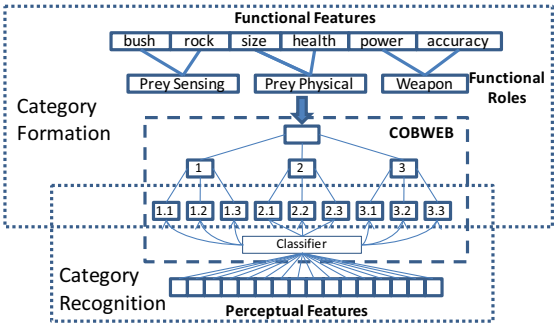

Figure 5: Category learning system

super-category '2'. The identifiers themselves do not have an intrinsic meaning, and the only requirement for them is uniqueness. The identifiers are used as state features in the RL system to acquire semantics in a specific functional context, a process described in the next section.

To recognize the functional category membership of an input instance, the original COBWEB algorithm descends the hierarchy to search for the best match based on its perceptual features. In our system, category recognition is separated from category formation as represented by the two large dotted boxes. All the available perceptual features are used for recognition, which is based on a supervised Naïve Bayesian classifier targeting a set of basic categories [9]. Currently, the algorithm chooses these basic categories by picking the level where the within-group total variance of functional features drops below a predefined threshold.

3.2. Reinforcement Learning System

Soar-RL [8] encodes the value function as a set of rules, with an expressive syntax equivalent to first-order logic. The left-hand side of a rule tests state and action features, while the right-hand side generates the expected value for the matching state action pair. The expected value of an action is the sum of the values of all rules matching the current state and that action. Figure 6 shows how the CL system is integrated with the RL system to achieve sparse-coarse coding [10]. The entire system has a two-layer structure to approximate the utility function of hunting different prey with different weapons. The first layer represents the CL system layer as described in Figure 5. Each hierarchy is shown to have two branches per level for demonstration purposes. In our system, the branching factor can be controlled within a specified range, such as from 2 to 5. When functional features are detected, the system automatically grows the corresponding hierarchies. As discussed earlier, functional features are usually not available when the agent wants to make a decision, so the recognition process is activated in such situations to predict the functional category based

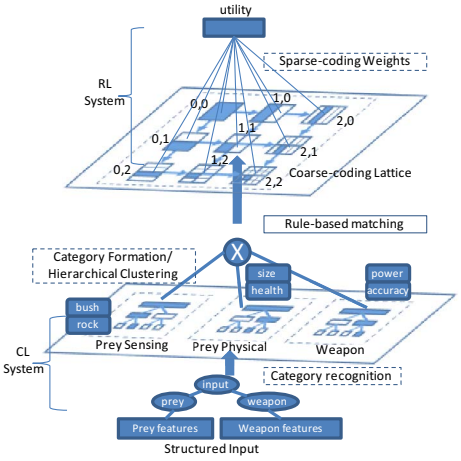

Figure 6: Overall structure of the learning system viewed as a two-layer network

on perceptual features. The dark colored nodes in the hierarchies represent symbolic categories matching with the input objects. These symbolic categories are used in the state representation and are matched by rules in the RL system. Rules are represented as cells in the coarse-coding layer. Dark colored cells represent rules that match the current state. The numbers on each grid indicates the hierarchy levels for component hierarchies, which will be explained later. The grids form an emerging lattice structure, and only a 2D projection of the lattice is shown in the figure. The complete lattice consists of all three hierarchies and has a 3D cubic shape instead of being 2D square.

We formally describe the algorithm below. To learn a target function, the system first maps the input objects into a vector of functional roles R, which represents the argument types of the target function. The vector O represents the input objects binding with R:

$$R = (r_1, r_2, \ldots, r_n)$$
$$O = (o_1, o_2, \ldots, o_n)$$

For example, the function is to predict the utility of hunting some prey with some weapon by shooting behind a rock from 2 units of distance to the prey. The inputs are two objects: rabbit and bow. If the system has no prior knowledge about the objects, then it creates a single functional role for all objects. According to our notation, input to the system will look like R=(generic-object, generic-object), O=(rabbit, bow). If we know that prey and weapon play distinctive functional roles with different set of functional features, then the input will look like R=(prey, weapon), O=(rabbit, bow). Furthermore, if we know more details about the interaction model: there are two components, one is about how to get close to the prey, the other is about how to choose the most effective weapon, then the input will look like R=(prey-sensing, prey-physical, weapon), and

O=(rabbit, rabbit, bow). These different levels of prior knowledge can be conveniently encoded as rules in our system. After matching objects with functional roles, the category learning system incrementally builds a set of hierarchies H correspondingly:

$$H = (h_1, h_2, \dots, h_n)$$

Let *height*(h_i) denote in the height of the hierarchy h_i, and k_i denote a cluster/node within the hierarchy. Let *level*(k_i) denote the level of cluster k_i in hierarchy h_i, with the root level being 0. Cells, grids and their relations, shown in Figure 6, are defined as following:

$$Cells = \{C_K, K = (k_1, k_2, \dots, k_n) | k_i \in clusters\ in\ h_i\}$$
$$Grids = \{G_L, L = (l_1, l_2, \dots, l_n) | 0 \le l_i \le height(h_i)\}$$
$$C_K\ belongs\ to\ G_L \equiv \forall i \in [1,n], level(k_i) = l_i$$
$$G_{L1} \prec G_{L2} \equiv \forall i \in [1,n], l1_i \le l2_i$$

More intuitively, each cell represents a rule in our RL system. A set of cells are composed into a grid that partitions the state space at a specific level of resolution. There is a lattice structure among the grids with the transitive relation *coarser-than* ($\prec$). Then given the set of input objects, the activation of a cluster k_i is denoted as $a(k_i)$:

$$a(k_i) = \begin{cases} 1\ if\ o_i \in k_i \\ 0\ if\ o_i \notin k_i \end{cases}$$

The mapping from o_i to k_i is achieved via category recognition, and only a single path of clusters are activated for a particular input as shown in Figure 6. $a(k_i)$=1 means an object in the current state, bound to the corresponding functional role, is an instance of the category represented by that cluster. The activation of a cell, $a(C_K)$, is defined as:

$$a(C_K) = \prod_{i=1}^{n} a(k_i)$$

$a(C_K)$=1 means the rule matches the current state and will be fired to participate in predating and learning the target value. The weight, $w(C_K)$, from the cell to the output unit is represented as a numeric value associated with the rule in the RL system. The learning algorithm updates the weights according to the *delta rule* for the identity activation function used in our RL system, where α is the learning rate, and t is the target value:

$$y = \sum_{C_K} w(C_K) a(C_K)$$
$$\Delta w(C_K) = \frac{\alpha}{\sum_{C_K} a(C_K)} (t - y) a(C_K)$$

The connection between the coarse-coding layer and the output unit is always sparse, since, for any input, only one cell from each grid in the lattice has non-zero activation. This is due to the competitive learning nature of the hierarchical clustering layer – only one cluster is activated at each level.

3.3. Time Complexity Analysis

The time and space cost of our system are reasonably bounded under practical assumptions. As in COBWEB, each hierarchy in our system takes O(logN) time, where N is the number of leaf nodes, for both predicting and assimilating a new instance, given bounded branching factor and fixed dimensions in input features [2]. Rule matching in Soar is in constant time given bounded changes in working memory. The remaining time cost is determined by the number of grids in the lattice, which is:

$$|Grids| = \prod_{i=1}^{n} height(h_i)$$

Since there always exists some level beyond which functional differences are too small to be meaningful, we can assume a small fixed height by keeping a limited number of leaf nodes for each hierarchy. Furthermore, based on the observation that the cardinality of object interaction is usually small, we can also assume that the number of functional roles in a target function is bounded by a small constant, and therefore the time cost of each update and prediction is practically constant. The space cost for each hierarchy is O(N). The space cost for the coarse coding lattice is $O(N^{|R|})$ where |R| is the number of functional roles in the target function. Given the above assumption about bounded number of leaf nodes and fixed small number of functional roles, the space cost is also constant.

4. Empirical Evaluation and Analysis

We focus on evaluating two novel features that are important for our system design. The first is to evaluate the successful integration of hierarchical representations in Soar-RL's existing value function approximation scheme. The second is to confirm the importance of supporting multiple functional roles and functional features, which is part of our extension to COBWEB. We do not intend to empirically characterize the category learning system's capabilities of incremental learning, noise tolerance, and flexibility in scaling up with increasing object diversity, as those have been previously established in research on COBWEB. Empirically evaluating the entire system by comparing it to alternative value function approximation approaches is the subject of future work. The evaluation task used in this paper is one specific configuration of the stochastic hunting environment based on some realistic considerations. For example, prey with larger *sizes* tend to have higher *health*. Prey with lower *sensitivity to bush* tend to have

higher *sensitivity to rock*. Weapons with higher power tend to have lower accuracy. We use sigmoid shaped functions to generate probabilistic outcomes from numeric functional features. Although the choice of parameters is arbitrary, it should not affect the conclusions of our evaluations, which are based on qualitative characteristics of the environment: multiple interacting objects with diverse functional properties and multiple functional roles. The data has a 3X3 structure as shown in Figure 5 for each hierarchy, and we do not present the details of the interaction model and data model.

4.1. Generalization Effect from Categorization

To focus on evaluating the performance of value function approximation, we simplified the exploration strategy of the RL agent: we set exploration rate to be 100% during training, and set it to 0 during testing. Learning rate is set to 0.3. The reward received for successful hunting is 1, and for failure it is 0. Each trial involves incrementally training the agent and recording the average of 100 independent testing episodes at different points of training. The final results shown in the plots are the average of 100 such independent trials.

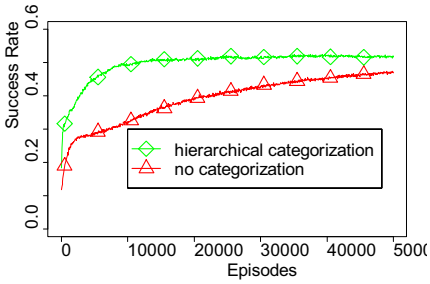

Figure 7: Comparing hierarchical categorization with no categorization

In Figure 7, we compare the learning performance of an agent using hierarchical categorization of objects with a baseline agent that uses the raw functional features without categorization. When there are only 9 unique instances for each hierarchy, a 2-level 3- branch structure is sufficient, and the baseline agent is equivalent to only using the gird $G_{(2,2,2)}$. The results demonstrate that hierarchical categorization leads to faster learning because the coarser grids capture shared information and aid generalization. When there are more unique instances (more than 9) for each hierarchy, "hierarchical categorization" is not affected, while the baseline is worse.

4.2. Flexibility of Hierarchical Representation

An advantage of using a hierarchy compared to flat categorization, such as k-means clustering, is the flexibility of representing categorical boundaries at different resolution levels, without forcing the system to make categorization decisions that are either over-general or over-specific. The next experiment tests whether such flexibility is naturally incorporated into our value function approximation scheme, in which all matched rules with different levels of generality are simultaneously updated.

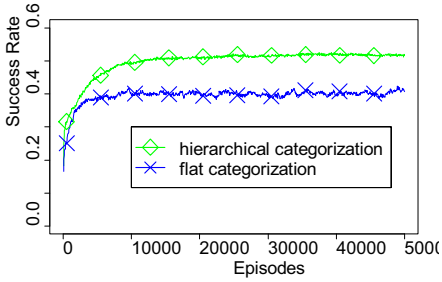

Figure 8: Comparing categorization with and without hierarchical structure

Figure 8 compares using hierarchical categorization to a baseline using single-level flat categorizations. In the baseline, the number of categories is set to 3 so that it is equivalent to only using grid $G_{(1,1,1)}$. Since such categorization does not capture detailed differences within each general category, it cannot improve performance beyond a certain point. We can choose to make a finer-grained flat categorization, which will lead to better asymptotic performance. However, that will inevitably lead to a worse improvement rate in the initial stage. In the extreme case, it will be the same as the performance generated by "no categorization" shown in Figure 7. In conclusion, using hierarchical categorization in our value function approximation scheme, reconciles the tradeoff between over-general and over-specific categorizations.

4.3. Importance of Functional Roles and Functional Features

To test if using functional roles and functional features can be beneficial, Figure 9 compares with a baseline using only a single functional role for prey and a monolithic category hierarchy based on all perceptual features of prey. The single hierarchy creates four levels with three branches at each level to represent all 81 types of prey – combinations of 9 sensing types and 9 physical types.

In Figure 9, the reason for poorer performance using the single monolithic hierarchy is that irrelevant perceptual features are included in the category formation process. These features act as noise and disrupt learning of the optimal category structure. By providing information about functional roles and functional features, there is less noise in categorization, leading to better performance. Our system supports this capability via domain specific rules.

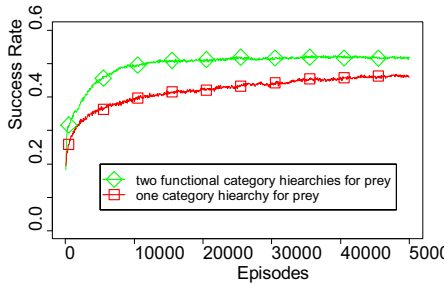

Figure 9: Comparing hierarchies obtained with and without functional features

4.4. Summary of Results and Analysis

Comparing all three figures, even using the monolithic hierarchy (lower line in Figure 9) leads to faster initial learning than no categorization at all (lower line in Figure 7), as well as better asymptotic performance than flat categorization (lower line in Figure 8). This is because the monolithic hierarchy still captures certain amount of functional similarity. With the help of functional roles and functional features, the learned category hierarchies more accurately reflect functional similarities and leads to better performance.

5. Discussion

We created a complex domain that stresses the types of challenges we are trying to address. There are many parameters to configure in the evaluation environment, including the underlying functions defining the probabilistic outcomes as well as the detailed data model. The results are based on a particular setting of the parameters, and part of the future work is to do evaluations in the space of possible environments. For example, when the differences among subtypes of functional categories are smaller in terms of determining the probabilistic outcomes, the performance differences between using full hierarchy and using flat categorization will shrink. We can also increase the object diversity by adding more noise to functional features, which will result in deeper as well as more dynamically changing hierarchies (we developed techniques to keep track of category identifiers during dynamic growth of the hierarchy, but that is not evaluated here). However, these quantitative changes are not expected to qualitatively change our conclusions.

Supporting the use of functional roles and functional features as a form of prior knowledge in our system can be practically useful despite its overhead. The observation is that only a relatively small set of functional roles and functional features are required to connect categorization with the general goals and basic needs of an agent. The benefit of providing such knowledge is that categorization more accurately reflects the functional similarities of objects without distractions from perceptual noise, and leads to significant improvement in RL. The origin of these functional roles and features is a more profound issue, and how to automatically learn them in a utility-driven manner is an important topic for future research.

6. Related Work

Our novel approach combines sparse coarse-coding and hierarchical structure in the context of object-oriented state representations. Like tile-coding based value function approximation [10], our approach relies on learning piecewise-constant local basis functions. The advantage of using local basis function approximators, compared to a global function approximator such as the classic multi-layer perceptron (MLP) or linear models [11], is better stability and convergence when used in TD learning [10]. This is because global function approximators are designed to fit relatively smooth functions, with linear model being the extreme case. Although MLP is a universal function approximator, its performance degrades when there are increasingly more intensive interference among nearby regions in the input feature space. Interference is the negative side-effect of generalization and can be catastrophic for learning arbitrarily non-smooth functions [12]. Having sparser connections in MLP can reduce interference. However, French [13] has noted that reducing overlap avoids catastrophic interference at the cost of a dramatic reduction in the exploitation of shared structure. The insight is that we need some structure in the connections. Our approach can be viewed as one approach to reconciling interference and generalization by combining competitive learning, hierarchical representation and sparse-coding in a multi-layer network, which can be further regulated by rule-based symbolic domain knowledge. Competitive learning via hierarchical clustering generates symbolic categories, which server as primitive structures to restrict interferences within local regions. On the other hand, sparse-coding with a hierarchical representation results in an emerging lattice structure, which regulates generalization by keeping the necessary connectivity, and at the same time minimizing interferences among unrelated regions. Such quicker and more stable learning inevitably shifts the cost to somewhere else: comparing to a fully connected MLP, more units (cells) are required in our system to achieve higher resolution in the value function approximation, although we have shown it is not a practical concern.

There are other approaches to combining hierarchy with local approximators, such as in kd-Q-Learning [14] and adaptive tile-coding [15]. However, these approaches lack the regulating structures in our approach because they are not designed for object-oriented state representations. Furthermore, these approaches face the "curse of dimensionality". For environments involving objects, each object is represented by multi-dimensional features, and the total dimension of the feature space can easily become prohibitively expensive for learning if using unstructured state vectors that concatenate all the features. Our Category learning system performs perceptual processing and dimension reduction: each hierarchy reduces the multi-dimensional subspace of corresponding functional features into a single dimension of hierarchical structure. Prior domain knowledge about functional roles and functional features helps to regulate such dimension reductions for object-based representations.

7. Conclusion

We presented a novel, two-layer architecture for efficient value function approximation by integrating unsupervised hierarchical categorization with the existing RL system in Soar. Our system has two unique features. First, the value function approximation algorithm utilizes hierarchical structures to smoothly reconcile the tradeoff between over-specific and over-general categorizations, so that learning can be quicker and more accurate at the same time. Second, our system supports the use of prior domain knowledge about functional roles and functional features of objects to regulate learning. These properties are valuable for autonomous learning agents in a novel, complex, object-oriented environment. The empirical results have confirmed our hypothesis and point out promising paths for future research.

8. Acknowledgement

This research was supported in part by the Ground Robotics Reliability Center (GRRC) at the University of Michigan, with funding from government contract DoD-DoA W56H2V-04-2-0001 through the Joint Center for Robotics.

9. References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction,* MIT Press, 1998.

[2] D. H. Fisher, "Knowledge Acquisition via Incremental Conceptual Clustering," *Machine Learning*, vol. 2, no. 2, pp. 139-172, 1987.

[3] W. F. Iba, and P. Langley, "Unsupervised Learning of Probabilistic Concept Hierarchies," *Machine Learning and Its Applications, Lecture Notes in Computer Science*, vol. 2049, pp. 39-70, 2001.

[4] N. Sahoo, and G. Duncan, "Incremental Hierarchical Clustering of Text Documents," in *16th Conf. on Information and Knowledge Management*, 2006, pp. 357-366.

[5] P. Langley, K.B. McKusick, J.A. Allen, W.F. Iba, and K. Thompson, "A Design for the ICARUS Architecture," *SIGART Bull.*, vol. 2, no. 4, pp. 104-109, 1991.

[6] C. Diuk, A. Cohen, and M.L. Littman, "An Object-Oriented Representation for Efficient Reinforcement Learning," *in Proc. 25th Int. Conf. on Machine learning*, 2008, pp. 240-247.

[7] J. E. Laird, "Extending the Soar Cognitive Architecture," in *Proc.1st Artificial General Intelligence Conf.*, 2008, pp. 224-235.

[8] S. Nason, and J. E. Laird, "Soar-RL: Integrating Reinforcement Learning with Soar," *Cognitive Systems Research*, vol. 6, no. 1, pp. 51-59, 2005.

[9] E. Rosch, "Principles of Categorization," in *Cognition and Categorization*, John Wiley & Sons Inc, 1978, pp. 27-48.

[10] R.S. Sutton, "Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding," in *Advances in Neural Information Processing Systems 8*, pp. 1038-1044, MIT Press, 1996

[11] S.J, Bradtke, and A.G. Barto, "Linear Least-Squares Algorithms for Temporal Difference Learning," *Machine Learning*, vol. 22, no. 1, pp. 33-57, 1996

[12] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are Complementary Learning Systems in the Hippocampus and Neocortex," *Psych Review*, vol. 102, no. 3, pp. 419-457, 1995

[13] R. M. French, "Using Semi-Distributed Representations to Overcome Catastrophic Forgetting in Connectionist Networks," in *Proc. 13th Annual Cognitive Science Conf.*, 1991, pp.173-178,.

[14] H. Vollbrecht, "Hierarchic Function Approximation in kd-Q-Learning," in *Proc. 4th Int. Conf. on Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, 2000, pp. 466-469

[15] S. Whiteson, M. E. Taylor, and P. Stone, "Adaptive Tile Coding for Value Function Approximation," AI Tech Report AI-TR-07-339, University of Texas at Austin, 2007.