LEARNING IMITATION STRATEGIES USING COST-BASED POLICY MAPPING AND TASK REWARDS

Srichandan V. Gudla and Manfred Huber
University of Texas at Arlington
Arlington, TX 76019
United States of America
{gudla, huber}@cse.uta.edu

ABSTRACT

Learning by imitation represents a powerful approach for efficient learning and low-overhead programming. An important part of the imitation process is the mapping of observations to an executable control strategy. This is particularly important if the capabilities of the imitating and the demonstrating agent differ significantly. This paper presents an approach that addresses this problem by optimizing a cost function. The result is an executable strategy that as closely as possible resembles the observed effects of the demonstrator on the environment. To ensure that the imitating agent replicates the important aspects of the observed task, a learning component is introduced which learns the appropriate cost function from rewards obtained while executing the imitation strategy. The performance of this approach is illustrated within the context of a simulated multi-agent environment.

KEY WORDS

Imitation, Reinforcement Learning, Policy Mapping

1. Introduction

As computer and robot systems move into more complex and unstructured environments, it becomes increasingly important that these systems have adaptive capabilities, can interact with humans, and are easy to program even by users who are not skilled computer programmers. In such situations it becomes essential that advanced means of programming or autonomous learning capabilities are available. Imitation, or learning from demonstration is a technique that takes an intermediate stance between fully autonomous learning and direct programming. In this paradigm new behaviors are acquired by observing other agents, be they human or artificial, operating in the environment. This framework can be seen either as a learning paradigm that provides the learning robot with richer information, or alternatively as a simpler approach to programming that permits to communicate new behaviors to the system by demonstrating them.

Robot Imitation and learning from demonstration have received significant interest in recent years [1],[4],[8],[9]. Within the field of robotics most of this has focused on imitation in humanoid systems. Most approaches in this domain address imitation by observing the demonstrator's joint angles and then attempting to execute the same angle sequence on the kinematic structure of the robot. If the structure of the imitator is not identical or very similar to the one of the imitating system, however, such approaches often lead to unsatisfactory results and the observed sequences have to be adapted on-line to address this problem. In addition, imitation at such a low level often limits its application to relative small task domains and does generally not generalize to the re-use of the acquired strategy when the environmental conditions change.

Other, more symbolic approaches to learning from demonstration have been developed where the imitating agent attempts to learn the internal policy model of the demonstrating agent [2],[5]. While this permits to address larger tasks, most of these approaches require that the agents have identical representations and behavioral repertoires and that the imitator can observe the actions chosen by the other agent. In most real-world systems, however, the demonstrating and the imitating agent can have significantly different capabilities and only the effects of actions are observable.

The imitation approach presented here is aimed at imitation at a functional level and focuses on the mapping from an observed state sequence to an executable control strategy, largely ignoring the perceptual challenges involved in translating sensory and in particular visual input into representations of the state of the environment. Functional imitation here implies that the goal is not to copy action sequences but rather to attempt to achieve similar sequences of effects on the state of the environment irrespective of the particular actions. The resulting imitation strategies are mappings from the observed states of the world to the behavioral capabilities of the imitating agent. The result is a control strategy that matches the behavioral repertoire of the imitating agent and that exactly or closely matches the functional effects of the observed actions even in situations where behavioral capabilities of the imitator and the demonstrator are dissimilar. To achieve the mapping between observation and imitation, the approach presented here uses a distance metric which represents the deviation of the imitation strategy from the functional intent of the demonstrator.

To permit the system to determine automatically which aspects of the observed task are important in the particular environment, this approach is combined with a reinforcement learning component [7] which attempts to acquire an optimal cost function using feedback obtained while executing the mapped policy. This incrementally increases the quality of imitation even for new tasks and demonstrations that have the same or similar objectives.

The techniques introduced here are illustrated using the WISE simulation environment [6] which is based on the Wumpus world computer game.

2. Functional Imitation Using Cost-Functions

Imitation takes place when an agent learns a task from observing its execution by a teacher or demonstrator. In general, the demonstrator can here be another artificial agent or ideally a human, while the imitator is a robot or an artificial computer agent. This general definition implies that the behavioral capabilities of the two agents involved can be substantially different and that the imitator might not be capable of performing the precise action sequence of the demonstrating agent. For example, if a mobile robot with a simple gripper is to imitate a human demonstrator for a house cleaning task, it will generally not be capable of performing all aspects of the demonstration in the same fashion. It might, for example, not be capable to reach on top of a book shelf due to its limited size. Similarly, it will not be able to perform aspects of the task in the same fashion as the human. For example, to pick a magazine off the floor, the human will bend down and reach forward. To achieve the same functional outcome, the mobile robot will have to drive up to the magazine and then close its gripper on it, thus executing an action sequence that behaviorally differs substantially from the one observed. The approach presented here addresses this by establishing a lowest cost approximation to the observed sequence. The result is an agent that approximately repeats the observed task.

Underlying this approach is a view that sees imitation as a three step process leading from perceptual observations to the execution and storage of a corresponding, executable control policy as illustrated in Figure 1.

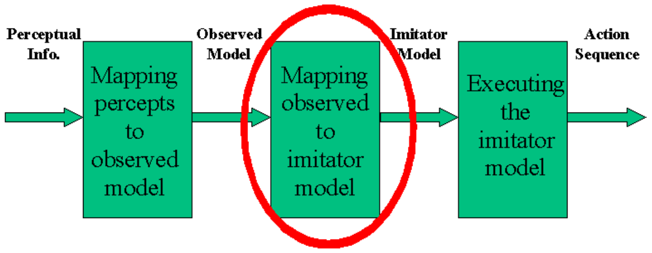

Perceptual Info.
Mapping percepts to observed model
Observed Model
Mapping observed to imitator model
Imitator Model
Executing the imitator model
Action Sequence

Figure 1: Imitation Process

The first step involves translating the perceptual input stream into a discrete representation of the sequence of the observed events. The resulting model of the observed task takes the form of a discrete Markov chain where states represent the observed state of the demonstrator and the environment and transitions occur when an observable change in the state is detected. Since actions are not directly observable, no actions are associated with the transitions. Similarly, aspects of the state that can not be observed are not represented in the observed task model.

The second step is concerned with mapping the observed behavior model onto the internal model of the imitating agent. The internal model is again represented as a discrete Markov model where states represent states of the environment and of the imitating agent. In contrast to the observed model, however, the internal model is a complete representation of the behavioral capabilities of the imitator. States occurring in the model represent all possible states that can be achieved actively by the agent using all options in its behavioral repertoire. Transitions of the internal model correspond to the effects of the execution of a particular action. In general, this model can be learned by the agent by exploring the range of actions at its disposal or can be pre-programmed by the designer using the available knowledge about the operation of the agent. The goal of the model mapping is to find the policy, i.e. the mapping from states to actions in the internal model that leads to the state sequence that most closely matches the observed state sequence and thus most closely reproduces the functional outcomes of the observed task.

In the third step, the imitating agent executes the policy identified in the second step. If the internal model is an accurate representation of the behavioral capabilities of the imitator, policy execution should be straightforward.

This paper focuses on the second step and thus assumes that the perceptual capabilities to generate the model of the observations are available and that the model of the observed task is already constructed. The main task addressed here is the mapping from the observed model to the internal model. In general, this will require identifying correspondences between states in the observed and in the internal model and searching for a state and transition sequence that matches the one observed. For the purpose of this paper it is assumed that the states in the observed and in the internal model are represented in terms of the same state attributes, facilitating the computation of a state distance measure. However, since the behavioral capabilities of the demonstrator and the imitator are generally not identical, the mapping process does not result in the exact same state sequence for the imitator, requiring the identification of the closest matching sequence which might include additional transitions or might not include certain observed states because they can not be achieved by the imitator or prevent it from achieving the remainder of the task. To identify the best policy, the approach presented here searches for the best match using a cost function defined on the state and action. The cost function here captures which aspects of the observed task are functionally important and as a result, changes in the cost function can directly affect the resulting imitation strategy.

Figure 2 illustrates the basic model mapping parameters used. Here, the observed model states (dark states) are mapped to internal states (light states) using a cost criterion consisting of an adaptable distance metric between the states and the cost of the actions.

2.1 Cost-Based Model Mapping

To map the state and transition sequence of the observed model to the internal model of the agent, the approach taken here has to address two main parts: i) Mapping the start state of the observed sequence to a corresponding start state in the imitator's model. ii) Mapping each transition in the observed model onto transitions in the imitator's internal model such as to produce the closest matching

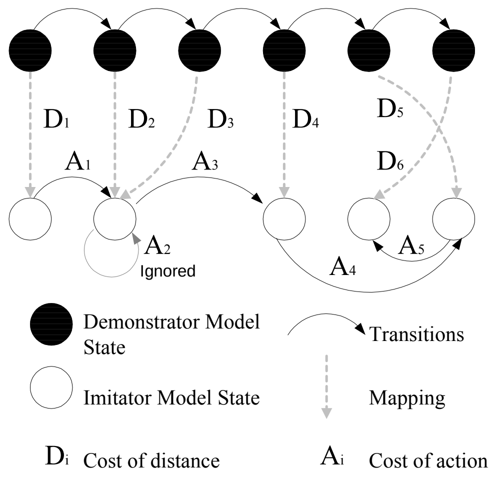


Figure 2: Cost-Based Model Mapping

state sequence. Both of these mapping steps are achieved here by optimizing a cost function C. This cost function consists of two components representing the cost of the actions selected to achieve the mapped transitions, C_a , and a cost, C_s , computed based on a distance metric between the observed and mapped states:

$$C = C_a + C_s$$

For the example in Figure 2 these cost factors are:

$C_a = A_1 + A_2 + A_3 + A_4 + A_5$

$C_s = D_1 + D_2 + D_3 + D_4 + D_5 + D_6$

where A_i is the cost of the action associated with the i^{th} transition and D_j is the distance metric between the j^{th} state mapping between the observed sequence and the matched internal state sequence. It is important to note here that the state and transition mapping between observed and internal model is generally not one-to-one and that therefore multiple distances can be associated with each state in these sequences. These cost factors can be defined in different ways by the user or an autonomous learning component, resulting in the potential for different types of imitation behavior. For example, by giving more weight to one feature of the internal state representation, the importance of exactly matching the parts of task related to this feature will be emphasized while features with lower weights might be ignored if their achievement introduces too high a cost. In this way, the choice of cost function can directly influence the resulting imitation policy, thus providing additional flexibility to this imitation approach.

A second choice in the construction of the matching state sequence is the one between establishing lowest cost matches locally across a short part of the model or doing so globally for the complete model. While establishing a minimum cost match globally would result in the best match according to the cost function used, the cost of such a procedure is very high. Moreover, establishing such a global match can only be accomplished if the entire demonstration is observed before the imitation strategy is formed and executed. Using a local matching procedure, on the other hand, can permit an imitating agent to start executing the first steps of the imitation policy before the demonstrator has finished the complete task.

The approach presented here forms a local solution by incrementally searching for state and transition matches for the observed sequence. This local solution could be used subsequently as a starting point for a global optimization procedure to improve the policy for future use.

2.2 State Mapping Using Heuristic Search

The approach presented here uses A* search with a limited search horizon to construct a policy mapping incrementally. To construct an admissible heuristic, the approach taken here assumes that at least one internal action is needed for one attribute change in the state and on this basis estimates the heuristic cost from the present state of the imitator model until the end of the observed state sequence. This heuristic function also assumes that once it guesses the cost for reaching the closest state of the remaining observed model, to reach every other state in the observed model thereafter will take at least one internal action. The total cost while performing the heuristic search is as follows:

$$C = C_A + C_H$$
$$C_H = C_{HC} + C_{HR}$$

Here C_A refers to the actual cost and C_H refers to the heuristic cost used by the A* search. The heuristic cost is again divided into two other costs, C_{HC} and C_{HR} where C_{HC} refers to the heuristic cost to reach the closest state in the observed model from the imitator's internal state and C_{HR} refers to the heuristic cost to reach the rest of the observed model from the closest observed state.

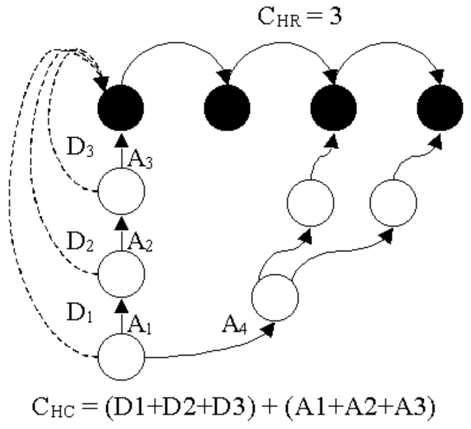


Figure 3: Calculation of Heuristic Cost

Figure 3 shows an example of the calculation of the heuristic cost of the imitator as the sum of C_{HC} and C_{HR}, where C_{HC} is calculated as shown in the figure after finding the closest state of the observed model (which in this example is the first state of the observed model). Then C_{HR} can be calculated as the number of remaining states from the closest observed state to the rest of the observed model (in this example three). While calculating C_{HC}, each time the attribute that contributes to the highest cost is changed and its cost is removed from the state difference and added to the heuristic until the state difference becomes zero. The action cost to reach the closest state is the number of times the state difference is decremented multiplied by the minimum action cost of the imitator's internal action. This heuristic is used in the search process which stops when the lowest cost mapping is found or the limit is reached.

3. Experiment

To illustrate the operation and results of the imitation approach presented here, a number of experiments have been performed using a simulated agent environment called Wumpus World which is based on an early computer game. In this environment, the agent explores a grid world to collect gold pieces (G). At the same time it has to avoid pits (P) and wumpi (W). The agent can remove the wumpi by shooting them. The objective of the game is to collect as many gold pieces as possible, return to the initial grid location, and exit the cave. The actions available to the imitating agent are Forward (GF), Turn left (L), Turn right (R), Shoot (S), and Grab (G). The Shoot operation is used to shoot a wumpus and Grab is used to collect the gold pieces. In the experiments presented here it is assumed that the imitator has full access to its state and that it observes the state of the demonstrator.

Here each observed state contains the information of the observable features of the demonstrator. The features of the agent are the current x and y coordinates, the orientation, and if the agent carries a piece of gold. The features of the world included in the state are the presence and location of any wumpi or pieces of gold.

In this experiment, the demonstrator starts from a start position and shoots a wumpus. Then it grabs the gold and returns to the start position and exits through the action Climb (C). But the imitator agent, who observes the same discrete number of states and transitions, does not repeat the task in the same way since it does not have the capability to shoot. Instead it tries to approximate the task as shown in the Figure 4.

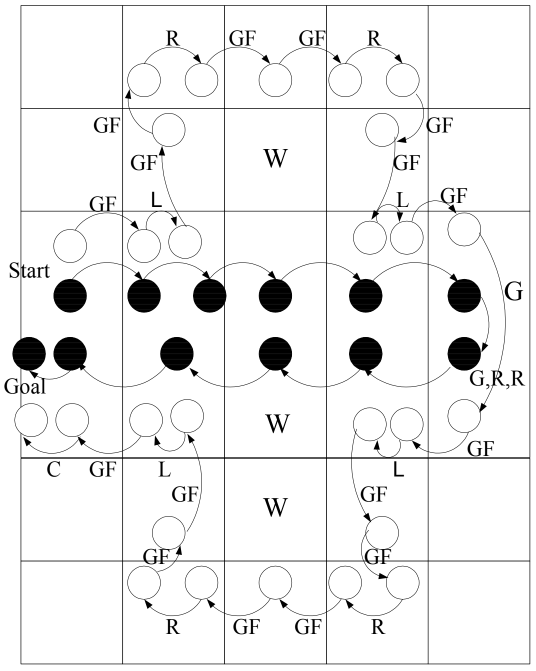


Figure 4: Imitation Example

This figure shows that the imitator model initially performs in the same way as the demonstrator. However, instead of shooting the wumpus it changes its orientation, moves up to avoid the three wumpi, and moves towards the nearest approximately matched state of the observed model. Then it grabs the gold and on its way back again encounters the risk of being killed by the wumpi. Hence it again acts differently from the demonstrator and ultimately reaches a state close to the observed state and exits in the same way as demonstrator.

4. Learning to Imitate Using Reinforcement

In the imitation process described above, the final imitation strategy heavily depends on the structure of the cost function. As a consequence, this cost function provides a means to modify the future behavior of the imitation system. If an optimal cost function can be found, the quality of the imitation strategies constructed in response to future demonstrations can be increased. This is achieved here by including a learning mechanism that tries to acquire a modified cost function such that subsequent imitation strategies can be further improved. In this work the state distance function is computed as a weighted sum:

$$D_j = \Sigma_i w_i f(a_i)$$

where w_i represents the weight and $f(a_i)$ represents the square difference of each state attribute a_i. The weight vector, w, is learned here over time such as to result in the highest reward possible for the imitating agent.

4.1 Reinforcement Learning for Imitation

Since there is no feedback or communication assumed from the demonstrator to the imitator, the learning mechanism of the imitator interacts with the environment to receive feedback such that the learning mechanism can update its knowledge to adapt to the environment and to improve the imitation process as shown in the Figure 5.

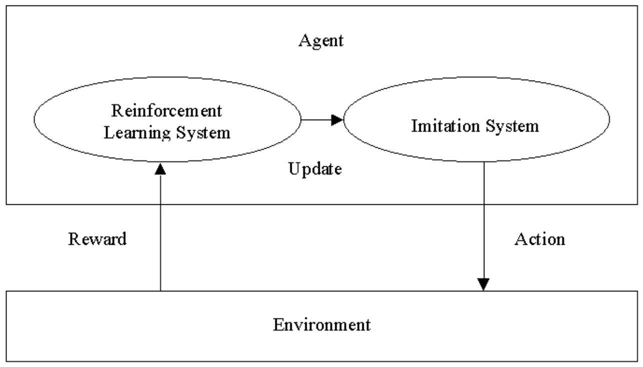


Figure 5: Reinforcement Learning for Imitation

For this purpose, a reinforcement learning algorithm is chosen where the learning system interacts in a closed loop with the environment. During each attempt of execution of a sequence of actions by the imitator, the environment provides an evaluation or reinforcement and the learning system has to learn from this how to improve imitation.

Another factor that needs to be considered in the construction of a reinforcement learning algorithm is that it has to deal with continuous rewards and outputs. In the work presented here, this led to the development of an algorithm that is closely related to the SRV (Stochastic Real Valued) Algorithm [3]. This algorithm computes its output as a function of a random activation generated using the Gaussian distribution. The activation, which here corresponds to a weight vector used in the cost function of the imitation approach, depends on a mean and the standard deviation. These, in turn, depend on the inputs to the learning system in the form of reinforcement received from the environment. The algorithm adjusts the parameters to increase the probability of producing an optimal value and hence of finding an optimal solution.

The following are the update equations used to learn the weight vector and the expected reinforcement:

$$m_i = m_i + \alpha (R - \tilde{R}) (w_i - m_i)$$

$$\sigma_i = \gamma \sigma_i$$

$$\tilde{R} = \tilde{R} + \beta (R - \tilde{R})$$

Here m_i is the mean and σ_i is the standard deviation of the estimated weight vector of each state attribute i and the combination of these represent the Gaussian distribution for each attribute weight. R represents the actual reinforcement received for one trial after executing a particular imitation policy, and $\tilde{R}$ is the expected reinforcement for the weight vector distribution. The symbols α and β are the learning rates for the mean and the expected reinforcement equations, whereas γ is the rate at which the standard deviation monotonically decreases.

This learning system operates by randomly picking weight vectors from the Gaussian distribution. As the learning system is exposed to more trials, it receives more feedback which in turn changes the mean value of the Gaussian distribution such that higher reward can be expected from the environment in the future. The random activations generated from the distribution are the weights used in conjunction with the state attributes to calculate the state differences. Hence different imitation strategies are generated and evaluated until an optimal weight vector is reached that generates an optimal imitation strategy. As a result, the learned weight vector identifies the important attributes within the state representation. In the approach taken here weights are assumed to be independent

One limitation of this approach of imitation with reinforcement learning is that the learning mechanism can optimize the cost function only for tasks whose objectives can be expressed in terms of the available state attributes.

5. Experiments

To illustrate the operation and the results of the learning system, additional experiments have been performed using the Wumpus World. As described previously the attributes of the state are the x and y coordinates, the orientation, the number of pieces of gold the agent carries the number of arrows. These attributes are associated with weights which are initially set to uniform values, making all attributes equally important. The learning system's task is to change these weights such that the expected reward increases.

Figure 6 shows a sample environment which the agent uses for learning. In this world, the observed model starts at the initial position, shoots the wumpus on its way to the position where one of the two pieces of gold is lying, grabs the gold and returns to the start position to exit the world. These changes in the environment are assumed to be permanent and once the demonstrator has completed its task, the wumpus is already dead and only the second piece of gold lying in the world one square diagonal to the

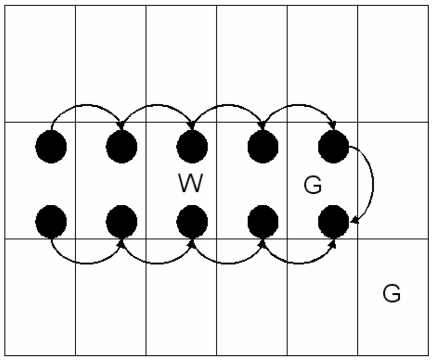


Figure 6: Example World Used for Learning

location of the gold acquired by the demonstrator remains as shown in the figure. Here, the imitating agent who observes the same discrete number of states and transitions cannot repeat the same task because the wumpus and the first piece of gold no longer exist in the environment

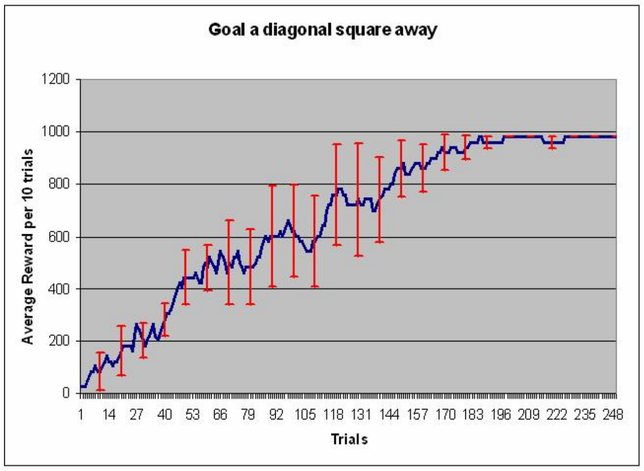
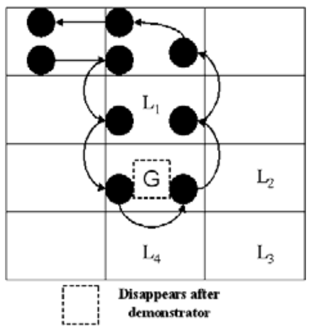


Figure 7: Learning Curve for Sample World

As shown in the Figure 7, the imitating agent is able to learn to improve its performance starting from the first trial until some optimal point is reached. This figure shows a running average over ten trials representing the average reward over five separate learning experiments. In addition, the standard deviations over ten experiments are shown as error bars for every ten trials.

Once the imitating agent reaches the optimum value, the standard deviation becomes almost. This implies that the imitating agent is able to grab the gold even if the gold is out of place from where the demonstrator grabbed it. This is the case because the imitation strategy produced now tries to imitate the demonstrator in light of the new weight vector determined through learning. In this example the imitator learns that gold is relatively more important than other attributes. Hence the imitating agent is able to grab the gold in another square in order to reduce overall cost of the imitation with respect to the observed model.

This same learning agent that was trained in the world where the gold is one diagonal square away from the gold acquired by the demonstrator is now tried in different scenarios by altering the demonstration and placing the gold in other locations, L1 to L4 , as shown on the left in Figure 8. Here the demonstrator starts from the start location moves to one of the gold pieces, grabs it, and moves back to the start position to exit the game. This is different task in the sense that not only the x coordinate is varied but also the y coordinate. Here the previously learned imitating agent that was trained on the sample world based on the task specified in Figure 6 is compared against the initial agent with uniform weights. The right table in Figure 8 shows the reward obtained by both agents if the second gold piece is placed in each of the locations.



Location	Without Learning	With Learning
L_1	987	987
L_2	-13	986
L_3	-13	984
L_4	-13	986

Figure 8: World for Testing of Learned Cost Function (left) and Rewards Obtained (right)

This table demonstrates that the agent that was trained on the first environment can outperforms the initial agent on new tasks within the same task domain without any additional learning on the particular task. This illustrates the benefit of using learning to modify the imitation mechanism rather than to optimize a particular, task-specific policy. Learning in this approach does not modify a policy directly but rather is aimed at identifying the functional attributes that are important for successful imitation. As a result, the learned information transfers readily to new tasks within the same task domain.

6. Conclusions

This paper presented an approach to imitation that constructs an imitation strategy by mapping an observed state sequence onto the internal model of the agent. This mapping uses a cost function, permitting it to be applied in situations where the behavioral capabilities of the demonstrating and imitating agent differ. The experiments presented show that the imitator is capable of imitating the demonstrator even under these circumstances by addressing the same task differently using its own action set. In this process it sometimes deviates from the observed state sequence, finding the closest state match that is achievable. This permits this approach to be used even if the demonstrator and imitator are different agent types.

A reinforcement learning approach is combined with the imitation to learn an optimal cost function and thus to improve the imitation process. Each time a sequence of actions is executed by the imitator, the learning system uses feedback provided by the environment to learn a cost function that increases the expected reward obtained on subsequent imitation attempts. This incrementally increases the quality of imitation such that the trained agent will imitate better than the imitating agent that has no knowledge of the environment. The results presented here show that the system is able to learn which aspects of the observations are important for imitation and that the learned cost function extends beyond the training tasks to other tasks within the same task domain.

7. Acknowledgements

This work was supported in part by NSF ITR-0121297.

References

[1] C. Atkeson and S. Schaal, Robot Learning From Demonstration. *Proceedings of the 14th Int. Conf. on Machine Learning*, San Francisco, CA, 1997, 12-20.

[2] J. Demiris, Active and passive routes to imitation. In *Proceedings of the AISB'99 Symposium on Imitation in Animals and Artifacts*, Edinburgh, Scotland, 1999.

[3] V. Gullapalli, Associative Reinforcement Learning of Real-valued Functions. *Technical Report 90-129, University of Massachusetts*, Amherst, MA, 01003, 1990.

[4] O.C. Jenkins, M.J. Mataric, and S. Weber, Primitive-Based Movement Classification for Humanoid Imitation. In Proceedings, *First IEEE-RAS International Conference on Humanoid Robotics*, Cambridge, MA, MIT, 2000.

[5] G. Peterson and D.J. Cook, DFA learning of opponent strategies. In *Proceedings of the Florida AI Research Symposium*, 1998, 367–371.

[6] L. Holder and D.J. Cook, An Integrated Tool for Enhancement of Artificial Intelligence Curriculum. *Journal of Computing in Higher Education 12*(2), 2001.

[7] L.P. Kaelbling, M.L. Littman, and A.W. Moore, Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, *4*, 1994, 237-285.

[8] S. Kang, and K. Ikeuchi, Toward automatic robot instruction from perception: Recognizing a grasp from observation. *IEEE Journal Robotics Automat.*, *9*(4), 1993.

[9] M.J. Mataric, Learning motor skills by imitation. In *Proceedings, AAAI Spring Symposium Toward Physical Interaction and Manipulation*, Stanford University, 1994.