

Cost-Based Policy Mapping for Imitation

Srichandan V. Gudla and Manfred Huber

Department of Computer Science Engineering
University of Texas at Arlington
Arlington, TX 76019-0015
{gudla,huber}@cse.uta.edu

Abstract

Imitation represents a powerful approach for programming and autonomous learning in robot and computer systems. An important aspect of imitation is the mapping of observations to an executable control strategy. This is particularly important if the behavioral capabilities of the observed and imitating agent differ significantly. This paper presents an approach that addresses this problem by locally optimizing a cost function representing the deviation from the observed state sequence and the cost of the actions required to perform the imitation. The result are imitation strategies that can be performed by the imitating agent and that as closely as possible resemble the observations of the demonstrating agent. The performance of this approach is illustrated within the context of a simulated multi-agent environment.

Introduction

As computer and robot systems move into more complex real-world environments, adaptive capabilities and ease of programming become increasingly important. This is particularly important for systems that have to interact with humans and that are under the control of a person who is not a skilled computer programmer. In such situations it becomes essential that other means of programming or autonomous learning capabilities are available to achieve task performance. Imitation, or learning from demonstration is a technique that takes an intermediate stance between fully autonomous learning and direct programming. In this paradigm the system acquires new behaviors by observing other agents, be they human or artificial, operating in its environment. Instead of learning in solitude, the agent is now able to benefit from the experience of others. This framework can be seen either as a learning paradigm that provides the learning robot with richer information about its environment, or alternatively as a simpler approach to programming that allows the programmer to communicate new behaviors to the system by demonstrating them.

Robot Imitation and learning from demonstration have received significant interest in recent years (Atkeson and Schaal, 1997; Kang and Ikeuchi, 1993; Mataric, 1994). Within the field of robotics most of this has focused on

imitation in humanoid systems. Most approaches in this domain address imitation initially 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 approaches have been devised to adapt the observed sequences on-line to address this problem. A second limitation of these techniques is that imitation at such a low level often limits its application to relative small task domains with short task sequences 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 (Demiris, 1999; Peterson and Cook, 1998). While it permits to address larger, more extended tasks, most of these approaches require that the imitating and the demonstrating agent have identical representations and behavioral repertoires and that the imitating agent can observe the action 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 approach to imitation presented here is aimed at imitation from observations of the demonstrating agent at a functional level. Functional 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 here 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 imitating and demonstrating agent are dissimilar. To achieve the mapping between observation and imitation, the approach presented here uses a distance metric to establish the control strategy that most closely reproduces the effects of the observed sequence. This paper will focus on the mapping from an observed state sequence to an executable control strategy and largely ignore the perceptual challenges involved in translating sensory, and in particular visual input into representations

of the state of the environment. The techniques introduced here will be illustrated using the WISE simulation environment (Holder and Cook, 2001) which is based on the Wumpus world computer game.

Imitation Using Cost-Based Policy Mapping

Imitation takes place when an agent learns a task from observing the execution of the same task 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 here can be substantially different and that the imitating might not be capable of performing the precise action sequence of the demonstrating agent. Moreover, it might not be capable of fully performing all aspects of the task in the same fashion due to missing capabilities. For example, if a mobile robot with a simple front 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 to dust due to its limited size. Similarly, it will not be able to perform other 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, at the behavior level, differs substantially from the one observed. The approach presented here addresses this challenge by means of establishing a lowest cost approximation to the observed sequence. The result is an agent that approximately repeats the observed task.

Underlying the approach presented here 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:

1. Mapping perceptual observations to a model of the observed task execution
2. Mapping the observed model onto the internal behavior model of the imitating agent
3. Execution of the resulting policy

The first step here 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 model where states represent the observed state of the demonstrator and the environment and transition occur whenever a significant 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 directly observed or that can not be extracted by the perceptual routines will also not be represented in the model of the observed task.

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 of the demonstrator, 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 by the agent. 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 process is then to find the policy, i.e. a mapping from states to actions in the internal model that produces the state sequence that most closely matches the sequence represented in the model of the observed task 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, thus imitating the agent. If the internal model is an accurate representation of the actual behavioral capabilities of the imitating agent the execution of the policy 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 of the observed model and states 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 built on the same state representation and thus that there is a one-to-one correspondence between observed and internal states.

However, since the behavioral capabilities of the demonstrator and the imitating agent are generally not identical, the mapping process might 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 matching policy, the approach presented here searches for the best match using a cost function defined on the state and action space of the internal model. Figure 1 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 a distance metric between the states and the cost of the actions selected in the internal model.

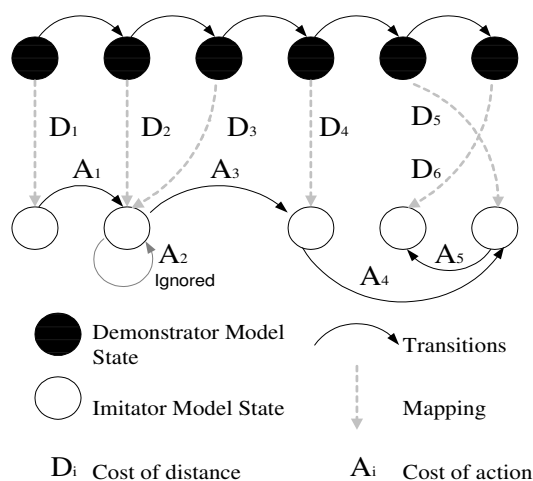


Figure 1: Cost-based model mapping

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:

1. Mapping the start state of the observed model to a corresponding start state in the imitator's.
2. Mapping each transition in the observed model onto transitions in the imitator's internal model such as to produce the closest matching state sequence.

The first part here involves determining the internal state that most closely matches the start state of the demonstrating agent and potentially finding a policy that moves the imitator to this state. The second part then corresponds to matching the observed transition sequence to an executable transition sequence for the imitator. Both of these mapping steps are achieved here by optimizing a cost function C . The cost function used here 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 1 these cost factors can be computed as:

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

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

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 approach.

A second choice in the construction of the matching state sequence in the internal model 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 in 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.

In the course of locally mapping the sequence of observed states to the internal model it is possible that states can either be matched exactly or that they have to be approximated because the behavioral capabilities of the imitating agent are not sufficient to achieve the exact outcome produced by the demonstrator.

Exact State Mapping

If the exact outcome state of a transition in the model of the observed task can be achieved by the imitating agent, the local mapping approach taken here still searches for the lowest cost policy to achieve this state. This however might require different actions as the one used by the demonstrator and a transition in the observed model might thus be mapped to a transition sequence of variable length in the imitator's internal model. In addition, multiple action sequences can potentially achieve the same outcome, requiring to select one of them. Here, the cost function is used as the selection criterion. Since the start and end states of each observed transition exactly match the ones selected in the internal model, the match is based largely on the cost of the selected actions. To find the lowest cost action sequence that accomplishes the desired overall transition, a search algorithm is used here. During search, the cost function serves as a heuristic and pruning criterion. Figure 2 shows an example result of this search process.

This figure shows the alternatives encountered during the search process and the associated action costs. The policy chosen by the imitating agent is the bottom one which has the lowest cost.

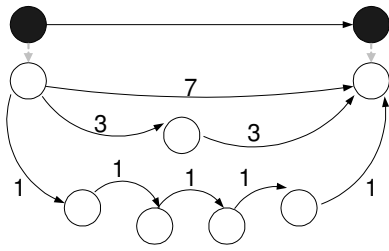


Figure 2: Search for optimal cost action sequence to perform an observed state transition

A side effect of using the cost function for model mapping is that demonstrator actions that do not change the observable state will be ignored as shown in Figure 3.

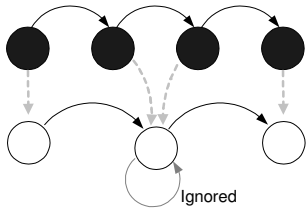


Figure 3: Removal of self-loops during mapping

This search process sequentially maps each state transition in the observed model to a sequence of transitions in the internal model of the imitating agent if they can be achieved. However, if the behavioral capabilities of the imitator are substantially different from the ones of the demonstrating agent, the outcome of a transition might not be achievable and the imitator has to find a policy that most closely approximates the outcome.

Approximate State Mapping

If the end state of an observed transition is not reachable, the algorithm incrementally increases its search horizon to find action sequences that skip states in the observed sequence. It does so in the forward as well as the backward direction (i.e. by backtracking along the already mapped policy segments) to find the lowest cost approximation.

Figure 4 shows an example where neither the first nor the second transitions are achievable by the imitating agent. In this figure, two possible transition strategies that can replace the initial observed transitions are indicated. Each of these skips one of the states in the observed sequence. If no action sequence for these cases can be found, the search horizon is extended further to include hypothetical observation sequences where one more state of the observed sequence is ignored. To limit the time spent in this process, the algorithm used here limits the maximum search horizon and if no action sequence is found that achieves the later state of the observed model exactly, reverts to the action sequence that throughout the search process led to the state most closely matching a state in the observed sequence.

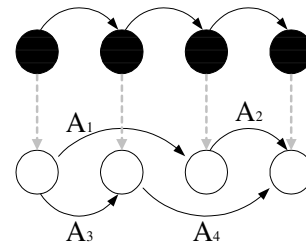


Figure 4: Search for lowest cost action sequence for observation models with unachievable transitions

Experiments

To illustrate the operation and results of the approach presented here, a number of experiments have been performed using a simulated agent environment related to the Wumpus World computer game. In this environment, the agent moves through a grid world to collect gold pieces (G). At the same time it has to avoid pits and wumpi (W), fierce creatures that can kill the agent. Also the agent can kill the wumpi in some particular conditions. 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 wumpi which will be killed if it is in the way the agent faces and Grab operation is used to collect the gold pieces. In these experiments, both demonstrator and imitator agent have been implemented. The imitating agent observes the state of the demonstrator and uses this to derive an imitation strategy.

Here each observed state from the file contains the information of the observable features of the demonstrator including the change in the world state. Including only the change in state decreases the amount of memory used as compared to storing the full instance of the world state. The features of the agent used here 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 the world. Since the wumpi can potentially kill an agent, a high negative reward is associated with the agent encountering a wumpus in order to avoid imitation in situations where the demonstrator is not affected by the wumpus. The weights for this reward can be changed by the user to make the imitator act differently.

For test case 1, the observed model starts from an initial position, shoot a wumpus on its way to a position where gold lies. Then it grabs the gold and returns to the start position to exit using the action Climb (C). However, the imitator agent, who observes the same discrete number of states and transitions, does not repeat the task in the same way since it is not capable of shooting. Instead it tries to approximate the task as shown in the Figure 5.

This figure shows that the imitator model initially performs in the same way as the demonstrator. However, instead of shooting the wumpus as the demonstrator, it changes its orientation by turning left, moves up through

