Chapter 4

Experiments in HELP

The final case study explores another dimension of HELP where the modes of human interactions are varied. The study details two standard reinforcement learning algorithms and describes the different forms of inputs that humans can provide and how they can be used by the algorithm. We instantiate the approach in a car driving simulator.

4.1 Case Study 3 - Extending the Modes of Human Interaction

Imagine a situation where an instructor is teaching people how to drive a vehicle. He shows them how to start and control the car, how to switch gears, parallel park, obey traffic rules and so on; he praises safe driving, discourages dangerous manoeuvres and gives his trainees advice and motivation. We are now faced with the task of being the driving instructor and not teaching a set of human trainees but a set of artificial agents how to drive using the mentioned techniques. At this point, we ask ourselves -

- Is it possible to represent the form of teaching used by the driving instructor in a mathematical model?
- Is there a mathematical equivalent to the words “show”, “praise”, “discourage”, “advice” and “motivation”?
- How can the artificial agent understand and utilize the information provided by the instructor?
- Given that information, can the agent learn and respond similar to a human?

This case study is aimed at exploring the answers to some of these questions. We focus primarily on the different types of information that the human instructor can

provide and how they affect the agent's ability to learn. This problem is investigated in the realm of Reinforcement Learning (RL) where the trainees are considered to be RL agents. An RL agent is one that takes actions in an unknown environment and tries to maximize its overall reward. The RL framework is explained in Figure.4.1. Formally,

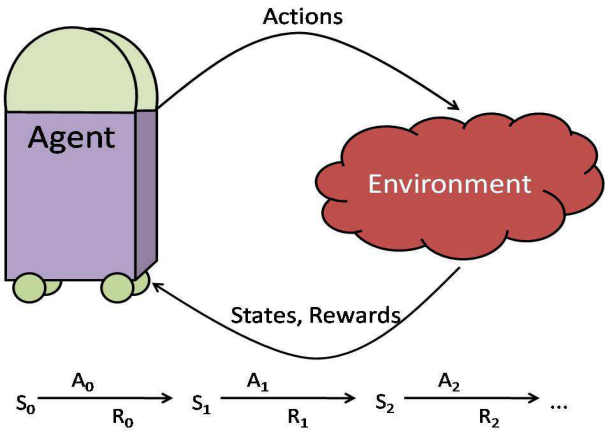

Figure 4.1: Traditional RL Framework.

the basic reinforcement learning model consists of: a set of environment states S; a set of actions A; and a set of scalar "rewards". At each time t, the agent perceives its state $s_t \in S$ and the set of possible actions $A(s_t)$. It chooses an action and receives from the environment the new state s_{t+1} and a reward r_t. The environment is formulated as a Markov Decision Process (MDP) where state transitions and rewards probabilities are typically stochastic. Given this setup, the reinforcement learning agent must develop a policy $\pi : S \rightarrow A$ (mapping of states to actions) that maximizes the cumulative reward received. In a traditional RL setting the agent learns autonomously. However in this case study, we introduce a human instructor in the loop and investigate the effects of his interactions on the RL agent.

4.2 Experimental Setup

This section describes the various components of the experiment. We begin by formally introducing the car driving domain and its characteristics, then briefly describing two standard reinforcement learning algorithms and then a note on the different types of

instructors that will be assisting the artificial agent.

4.2.1 The Highway Car Domain

The environment used in this study is a simulated version of the car domain similar to one described in [Abbeel and Ng (2004); Syed et al. (2008)]. A descriptive screen-shot of the simulator is shown in the Figure 4.9. The highway consists of 3 lanes with an additional off-road lane, one on either side (indicated by the green patch). The agent (blue square) is driving on the highway at a constant speed in the presence of oncoming cars (red triangles). Every state of the agent in the driving world is described by using

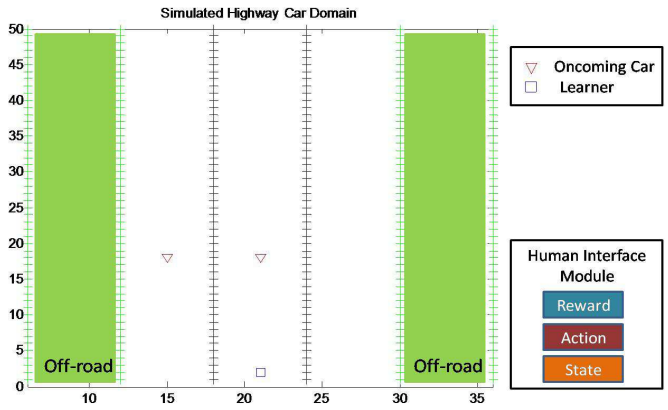

Figure 4.2: Screenshot of the Driving Simulator.

a set of features. We use three features in particular, one indicating whether the agent has collided with an oncoming car, two indicating whether the agent has gone off-road and a feature describing which lane the agent is currently in. The goal of the agent is to drive safely on the highway by learning to avoid collisions with oncoming cars, to avoid going off-road and to maintain lane discipline. To achieve this goal, the agent is equipped with three actions (left, right and stay in place) and has access to a human driving instructor who is assumed to be optimal at the given task. The instructor communicates his knowledge to the agent through a keyboard interface (shown in the figure).

4.2.2 Learning Algorithms

Here we describe the algorithms used by the agent to learn the driving task. We first give a more formal definition of an MDP and then detail two algorithms, Q-learning and R-Max, that have been widely used to solve MDPs.

An MDP is a 5-tuple $\langle S, A, \mathcal{T}, R, \gamma \rangle$ with states S, actions A, transition functions $\mathcal{T} : S \times A \mapsto \Pr[S]$, reward function $R : S \times A \mapsto [R_{min}, R_{max}]$, and discount factor γ. The function we would like to maximize is $\sum_{t=0}^{\infty} \gamma^t R_{at}(s_t, s_{t+1})$, where the actions a_t are chosen according to policy π. We now detail the learning algorithms using the above definition as the foundation.

4.2.2.1 Q-learning

The Q-learning algorithm [Watkins (1989)] is a model free learning algorithm that has been widely used to solve infinite horizon MDPs. Model free indicates that the agent can learn to perform the task without ever having to build a model of the environment in which it is taking actions. Given the formal definition of an MDP, this algorithm focuses on learning which action is optimal for each state. The algorithm computes the "Quality" of a state-action combination: $Q : SXA \rightarrow \mathbb{R}$. The recursive step of the Q-learning algorithm is given by the equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{4.1}$$

Using the above equation, the Q-values are iteratively updated as the agent takes actions in different states and receives rewards. The optimal policy π^* is computed using the formula:

$$\pi^*(s_t) = \arg\max_a Q(s_t, a_t) \tag{4.2}$$

This algorithm has been proven [Watkins (1989)] to converge to optimal policy given the following set of criteria -

- The system must be a deterministic MDP.
- The reward values are bounded by some constant R_{max}.

- The agent visits every possible state-action pair infinite times.

The algorithm variables, α (learning rate) and γ (discount factor) have a significant influence of the rate of convergence of the algorithm. The learning rate determines to what extent the newly acquired information will override the old information and the discount factor determines the importance of future rewards. The algorithm has several advantages: it is easy to implement, is exploration insensitive, one of the most effective model-free algorithm for learning from delayed reinforcement. On the other hand, algorithm does not scale to large complex domains and it takes a very long time to converge.

4.2.2.2 R-Max

R-Max [Brafman and Tennenholtz (2002)] is a model-based reinforcement learning algorithm which is said to attain near-optimal average reward in polynomial time. In contrast to a Q-learner, the R-Max agent maintains a complete, possibly inaccurate model of its environment and acts based on the optimal policy derived from this model. The algorithm initializes the model optimistically implying that all actions in all states return the maximum possible reward. This acts as a driving force causing the agent to explore different parts of the state space by taking different actions and updating its model based on its observations (feedback from the environment).

To summarize the algorithm, it starts with an initial estimate for the model and assumes that all states and all actions return the maximum reward and with probability 1, transition to a fictitious state S_o. On the basis of the current model, an optimal policy is computed using one of several methods. The one used in this implementation is Value Iteration. The transitions and rewards received for every state-action pair are stored and once sufficient information has been acquired, the model is updated and the optimal policy is recomputed. This process repeats. The optimality and convergence proofs of R-Max can be found in [Brafman and Tennenholtz (2002)]. This algorithm has the advantage of actively encouraging exploration and does not require the MDP to be deterministic. A more descriptive form of algorithm is available below:

Algorithm 3 R-Max Algorithm

Input:

- the number of states S, the number of actions A, the upper bound on the reward function R_{max} and the error bound ϵ.

Initialize:

- Construct a model M consisting of $S + 1$ states and A actions. Here the additional state S_o is considered to be a fictitious state.
- Every state and action in the model is initialized with R_{max} (optimistic) and transitions are initialized such that $T_M(S_i, a, S_o) = 1$ for all states S_i and for all actions a.
- Initialize a boolean value *known/unknown* to all state-action pairs indicative of how well the algorithm can predict the outcome given a state-action pair.

Repeat:

- Given a model compute the optimal policy (ϵ close) for the current state and execute it until a new state-action pair becomes known.
- Observe and record the number of times a state-action pair has been visited and the possible outcomes during each visit. Once it crosses a threshold, toggle the boolean value from *unknown* to *known*.
- Repeat.

4.2.3 Forms of Human Input

The role of the instructor in the driving domain is to transfer his knowledge of the task to the artificial agent using natural methods of interaction. When teaching humans, the instructor uses demonstration, motivation, advice, and other such social methods to train the humans. However we have seen from the previous sections that agents understand their world only in terms of states, actions and rewards. The question is then how can a human teach an agent to drive using natural means of interaction and still provide the agent with signals that it can comprehend. This dilemma is illustrated in Figure 4.3. To overcome the barrier, we introduce three types of human instructors:

1. The Motivational Instructor
2. The Directive Instructor
3. The Associative Instructor

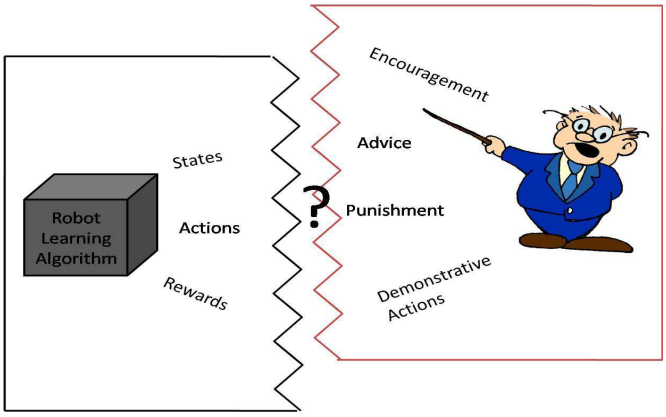

Figure 4.3: Bridging the communication gap between the instructor and artificial agent.

In order to understand the mapping of human interactions to agent-understandable language, let us analyze the contributions of each of the instructors:

- The first instructor (motivational) is one who commends the agent when it performs a desirable action and at the same time punishes the agent for performing sub-optimally. The inputs from this instructor are translated to the learner in the form of *Rewards*.

- The second instructor (directive) is one who directly commands the agent by telling it what to do. He manoeuvres the agent through the highway by controlling its moves until the agent has learned to perform autonomously. The inputs from this instructor are translated to the learner in the form of *Actions*.

- The third instructor (associative) is one who teaches the agent by relating the components of driving that require similar behavior. By doing this he allows the agent to link the different aspects of driving that require the same or associated responses. The inputs from this instructor are translated to the learner in the form of *States*.

Given that we now have a way to bridge the gap between the teacher and the agent, we explore the performance of the agent with each of these styles of instruction.

4.3 Driving Instructor and the Various Modes of Interaction

We have now setup the experiment by describing the car driving domain, the artificial agents and the types of human instructors. In this section, we execute the different styles of teaching on the two simulated agents (Q-learner and R-Max learner) and evaluate the learned policy of the agents. During each trial, we study the effects on the interaction on the two learners as well as on the instructor.

4.3.1 The Motivational Instructor

It is well known that right from the formative years a child is nurtured by its parents using a combination of reinforcement, praise, punishment and anger. Using this interaction, the child learns to do things that will produce positive reactions from its parents. We extend this form of interaction and learning, to the driving domain experiment. The instructor uses a combination of positive and negative rewards to indicate to the agent whether it is driving well or not. In the long run, the agent should learn to take actions that maximize the total received reward. This approach to machine learning has been successfully used by Thomaz et al. (2006) and Knox and Stone (2009). In Thomaz et al. (2006), the human observer uses social modes of interaction like rewards, guidance and advice to communicate the task to the agent. It was successfully applied to a simulated version of Sophie's Kitchen and a real world robot learning experiment. Knox and Stone (2009) describe a framework called TAMER, which stands for Training an Agent Manually via Evaluative Reinforcement. The human observer provides rewards to the agent depending on his assessment of the agent's policy. The underlying algorithm models the human's reward function using a function-approximator and extends this reinforcement to unseen states using a regression algorithm. This framework was successfully applied to a game of Tetris and the Mountain Car domain.

Let us now understand how this instructor teaches the agent to drive. The screenshot of the simulator, shown previously in Figure 4.9, shows a Human Interface Module through which the human provides his input to learner. When the experiment starts, the agent is performing randomly as it has no prior on the task to be learnt. The teacher

observes the agent act and when he feels that agent has performed a suboptimal action (causing it to collide or go off-road), he activates the interface module. At this point, the simulated driving world comes to standstill and awaits the assessment of the current environment configuration. The human uses the keyboard and provides a scalar value that could be positive indicating praise or negative indicating punishment. The agent receives and processes this reward using its reinforcement learning algorithm. For all states where the human does not provide reinforcement, the agent receives a default reward of 0. Figure 4.4 illustrates the various states of the driving world and types of reinforcement given by the optimal teacher.

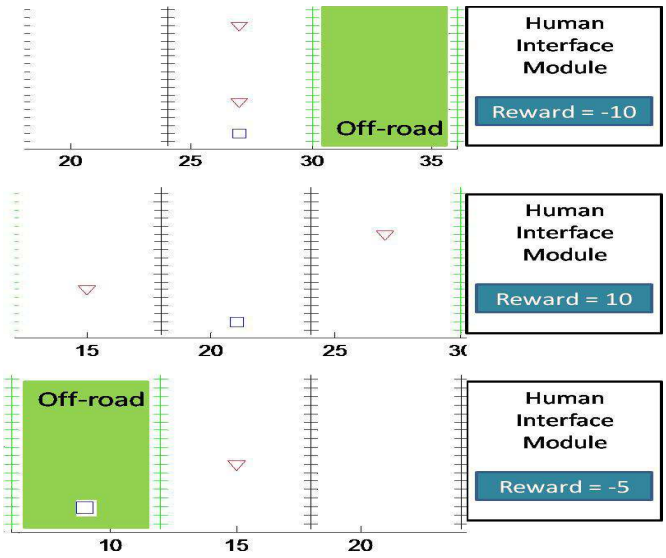

Figure 4.4: Different Configurations of the Driving Domain and the Reinforcement given by the Instructor.

4.3.1.1 Effect on the Learners

Q-learning

In the standard Q-learning algorithm, the agent has access to a reward function. In this trial, the motivational instructor assumes the role of the reward function and provides the agent with appropriate rewards. The Q-learning update now becomes:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[HumanReward_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (4.3)$$

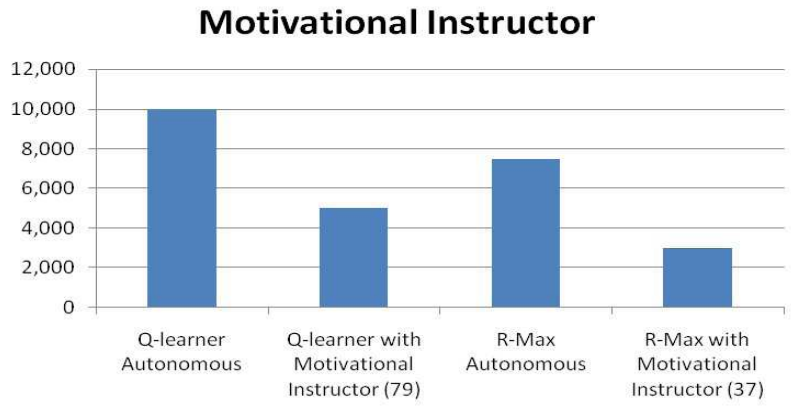

Figure 4.5: Number of steps taken by the learners to reach the optimal policy (with Motivational Instructor).

Every state in the environment has a set of features associated with it. Each reward provided by the human is linked to a state and in turn to a feature configuration set. This association allows the Q-learner to learn the quality of human reward expected for a particular feature set. Thus when the agent encounters an unseen state that has the similar feature values as a state previously encountered, it will act as though they are the same state. This action is then followed by positive or negative reward depending on the human's assessment of the agent's last action. The performance was further tested by modifying the variables of the algorithm.

R-Max

R-Max as described previously is an efficient model based algorithm. The model is initialized optimistically with high reward in all states, for all actions. As the agent acts, it receives reward from the environment and updates its list of state-action pairs visited. In this trial, the reward is no longer given by the environment but given by the instructor. Essentially the change made to the algorithm can be represented as $Environment_{reward} \Rightarrow Human_{reward}$. The agent associates the reward obtained with the feature configuration of concerned state. This information is extrapolated to unseen states and is executed by the agent the next time it updates its model. The advantage with this algorithm over Q-learning is that the agent actively explores and does not

depend on the human to explore. This way learning is much faster. The optimality of the obtained policy was tested by modifying the variables of the algorithm.

Figure 4.5 compares the performance of the two learning algorithms, that learns from a motivational instructor, the task of safely driving a car on a highway with two oncoming cars. The performance is calculated as a measure of the number steps taken before the agent is optimal and the number human interactions (shown in the bracket) that were required to reach an optimal policy.

4.3.1.2 Effect on the Instructor

This experiment was run using 2 human subjects. Their role as the motivational instructor was explained to them and they were given the task of teaching the agents to learn to drive safely. Figure 4.6 shows the performance of the agent across 5 trials with human subjects. The graph shows how the instructor was able to learn *when* and *how* to give the motivation in order to enhance the agent's learning.

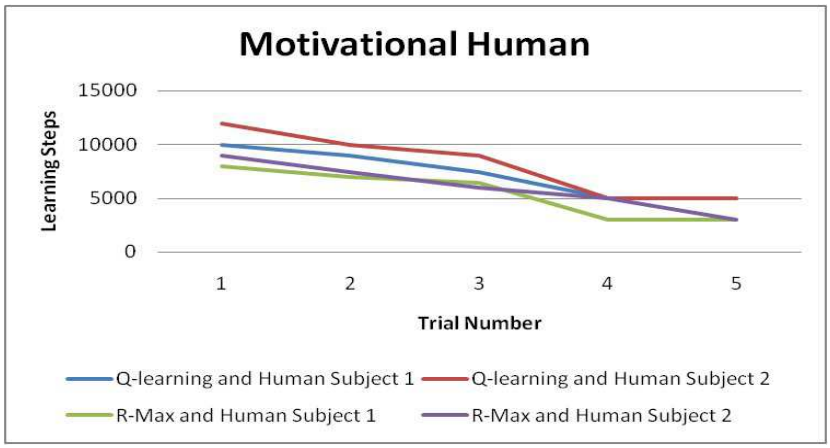

Figure 4.6: Performance of the agent across 5 trials with 2 Motivational human subjects.

Here are a few observations describing the effects of the learning interface on the human:

- The Q-learning agent is slow learner and takes a long time to converge. This can be attributed to that fact that it is exploration insensitive. The human observes that the agent is not exploring, so he modifies his method of providing rewards.

Humans alter their behavior, taking into account the agent's capabilities, and in a manner that makes the agent explores it world.

- Humans tend to give reward even before agent has performed the required action. The human is in effect protecting the agent from imminent danger. Although the human might think that he is correcting the behavior, the Q-learner, since it doesn't have a model, is unable to associate received reward with future states.

- Humans like to see immediate effects of their rewards. If this is not seen, there is a tendency to punish the agent (negative reward) for the mistake. A mistake that the Q-learning agent is still to recognize.

4.3.2 The Directive Instructor

A Directive Instructor is one who teaches the agent by explicitly telling it what set of actions to take given the current configuration of the world. The instructor does not provide a measure of reinforcement but instead a direct action that should be taken. This type of learning is analogous to way the teachers help students solve mathematical problems. They help the children by telling them how to solve the first few set of problems until the students are able to solve them on their own. Behavioral Cloning [Bain and Sammut (1996)] is another similar technique. In this method, learning agents first watch either a human or agent expert execute the required behavior and then try to replicate the patterns of behavior observed (clones). The authors of [Bain and Sammut (1996)] point out some disadvantages to the method of Behavioral Cloning. The first is that the clones were brittle and difficult to understand. Cobot, A Social Reinforcement Learning agent [Jr. et al. (2001)] is an agent that lives in an online community called LambdaMOO [Jr. et al. (2001)]. Within the scope of the online community, it learns in a manner similar to driving agent. Cobot watches other people take actions, watches them interact and forms associations with this knowledge. Using the model it has built, Cobot takes actions in the community, interacts with other people and further refines its model based on the feedback it receives from other users.

The interface of the directive instructor is similar to the previous one where the

communication takes place through the human interface module. Once the driving simulator is started, the agent begins by performing randomly. The instructor, observing the agent, steps in by clicking on the simulator window. The world stops and awaits an directive action as input from the human. The human now has one of three choices, corresponding to the three actions of the agent, stay in place, left and right. The human selects the correct action to be taken and the agent follows that directive. For all states where there has been no human intervention, the agent takes actions autonomously as defined by its policy. This process continues until the agent has learned to perform the task autonomously. Figure 4.7 illustrates the various states of the driving world and types of directive actions given by the optimal teacher.

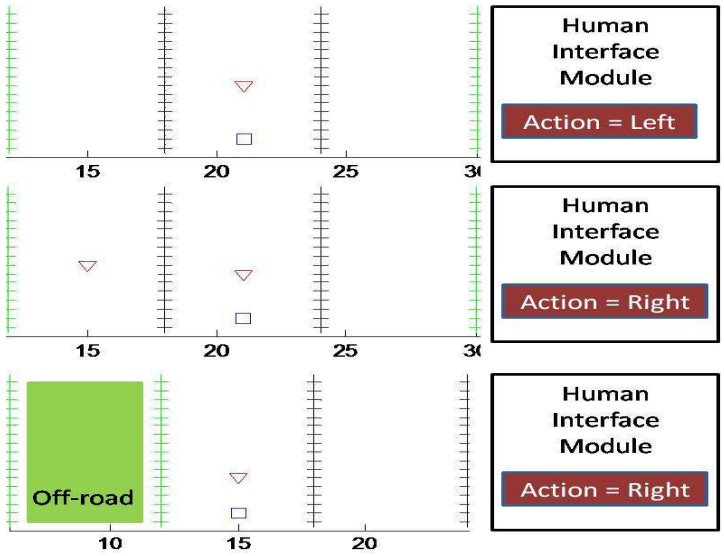

Figure 4.7: Different Configurations of the Driving Domain and the Directive Actions given by the Instructor.

4.3.2.1 Effect on the Learners

Q-learning

The Q-learning agent takes actions by following its policy, represented as $\pi^{(s_t)} = \arg\max_a Q(s_t, a_t)$. It chooses the actions that is said to provide the maximum utility. The Directive Instructor has the ability to overrule the action suggested by the agent's

policy, forcing it to take the teacher's choice of action. The algorithm then uses the reward acquired from taking the new action to update its Q-value. It is important to note here that, the reinforcement now comes from a standard reward function. The human only provides directive actions. In Q-learning, the agent does not actively explore, however since the teacher has direct control of the agent, the agent can be comfortably steered to parts of the driving world where there is relevant information to be learned. Learning tends to be quicker when comparing the Directive Expert with the Motivational Expert. This is attributed to the fact that the directive teacher explicitly controls exploration while the motivator controls exploration implicitly via the cumulative reward received.

R-Max

The R-Max Learner acquires the actions to be performed by computing the optimal policy. As explained previously, this step is completed using Value Interaction. It produces Q-values for every state-action pair and the agent acts by choosing the action that has the maximum value/utility. The instructor has the ability to override the action suggested by the policy and provide a more appropriate action for the current state configuration. The agent uses the reward obtained from taking that action and updates its list of known and visited states. During the next iteration, the recomputed model will encode the new information given by the instructor. We know that the R-Max learner initially builds an optimistic model and starts by actively exploring it. However, when interacting with the Directive Instructor, this tends to be counter-productive as agents sometimes explore parts of the state space where there is nothing relevant to be learned. This in turn results in the instructor having to step in more often just to correct the unnecessary exploration of R-Max learner.

Figure 4.8 compares the performance of the two learning algorithms, that learns from a directional instructor, the task of safely driving a car on a highway with two oncoming cars. The performance is calculated as a measure of the number steps taken before the agent is optimal and the number human interactions (shown in the bracket) that were required to reach an optimal policy.

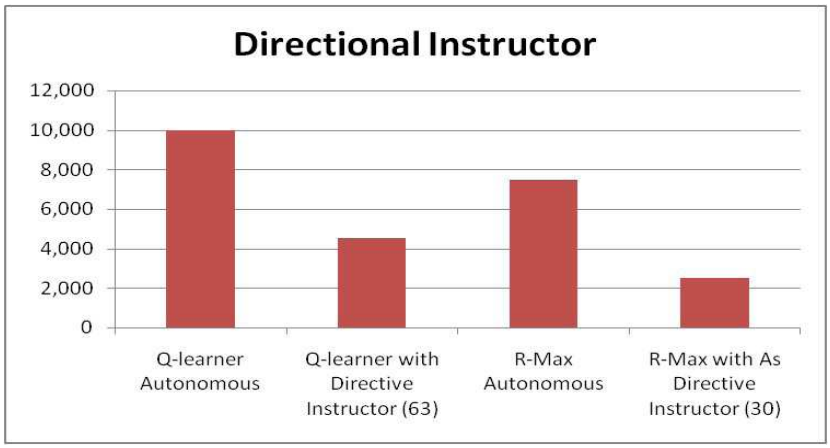

Figure 4.8: Number of steps taken by the learners to reach the optimal policy (with Directional Instructor).

4.3.2.2 Effect on the Instructor

This experiment was run using myself as the human instructor. I found the interaction with the Q-learner to be more engaging as the agent seemed like a student waiting to receive orders from the teacher. The R-Max learner on the other hand tends to explore actively and therefore required numerous interactions with me in order to correct its behavior. Clearly the value of exploration is dependent on the type of information given by the human. Another interesting observation was that when the expert is teaching the agent, it is important for the agent to be a shown a negative reward. The reason for this is that if the agent is acting optimally right from the start, it will always receive high reward. It will think that all its actions are good because it doesn't have a notion of what is bad. Once the instructor purposefully shows a negative reward, it allows the agent to differentiate states with good reward and those with low reward. The policy computed by Q-learning or R-Max will then be able to better characterize the task.

4.3.3 The Associative Instructor

The Associative Instructor uses a novel approach to interact with an artificial agent. In this form of interaction, the human has the task of linking/clustering states that have similar properties, for example, states that require the same action to be performed.

This method although subtle is often seen when humans interact and teach one another. Consider the task of teaching a child to solve basic math problems. Let the number of problems indicates the number of states and the 4 arithmetic operators indicate the number of actions. During the course of interaction, one can imagine the teacher telling the student "Use the same technique that you used to solve Problem 1". The teacher has now become an associative instructor. The teacher has indicated that the current problem (state n) can be solved by the same arithmetic operation (action) that was used to solve an earlier problem (state 1). The student now begins to forms links across the problems in an attempt to understand their implicit similarities. Figure 4.9 illustrates the same analogy in the car driving domain. It shows three different states of the world, all of which have one thing in the common: the agent can avoid a collision if it goes right. For that reason, the instructor is mapping all the three simulator states to a single cluster.

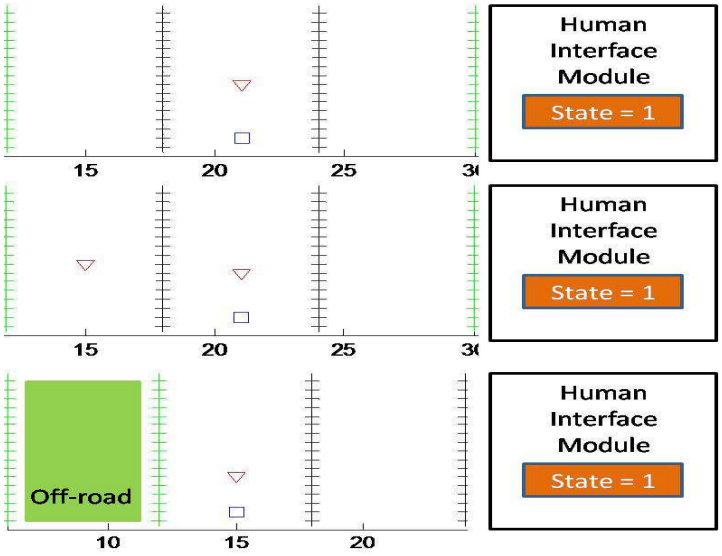

Figure 4.9: Different Configurations of the Driving Domain and the State Mapping given by the Instructor.

Using this method of instruction, it would be possible to cluster large state spaces to a smaller subset characterized by some common features. The human gives the input using the interface module. A question that we would like to answer at this point: "is it necessary for the expert to label every state to a cluster?". The answer is no.

We overcome this by introducing a notion of continuous clustering. To understand this better, consider the following state transition - $S_1, S_2, S_5, S_9, S_{12}$ and S_{28}. At state S_1, the instructor activates the interface module and maps that state to cluster 1. After having done this, all following states are mapped onto cluster 1 until the instructor accesses the interface module and indicates otherwise. In this manner, states S_2, S_5, S_9, S_{12} and S_{28} are sequentially linked to cluster 1 until the teacher activates the interface. In the driving domain, a cluster is defined by the action to be taken and it virtually contains a set of states in which the same action has to be performed. Using this method a large number of states can be clustered with very little interaction from the associative instructor. Intuitively, the instructor has converted the state space of the driving domain to small subset of N clusters. We formally define π^* as the policy of Associative Instructor. In our case, π^* consists of N cluster states each of which is associated with a single action.

4.3.3.1 Effect on the Learners

Q-learning

The Q-learner uses its standard representation to learn the task. When the human steps in, he enters a cluster number (say 2) to which the current state is to be virtually linked. The underlying algorithm now makes note of the link as well as the action taken (say right) in that state. Henceforth, cluster 2 is characterized by the right action and all states in which the right action is to be performed is referenced to cluster 2. Algorithmically, this is achieved by replacing the Q-values of the agents policy, π with Q-values of the instructors's policy π^*. Initially the instructor's policy will be a random matrix, but it gets refined as the agent continues to take actions in the world. There exists a two way communication between the two policies thats lasts long enough until the agent can perform autonomously. As we have noted before, the Q-learner does not actively explore and this has a direct effect on the different states visited. The learner when interacting with associative instructor was able to perform well on small state spaces, but as the driving world grew bigger with more oncoming cars, the Q-learning with its poor exploration strategy was unable to help the instructor.

R-Max

R-Max learner as expected assigns clusters similar to the Q-learner and explores the state space in much more efficient manner. Large state spaces can be easily covered in no time as the algorithm drives visits to new states and the instructor assigns clusters to blocks of these states. Every move initiated by the associative instructors opens a two way channel between the agent's policy and that of the teacher. The states are appropriately linked to clusters and the Q-values are modified in both policies. Value Iteration is performed on this new information and the model produced encodes the state assignments of the instructors implicitly in the form of Q-values. The R-Max learner was found to converge quickly to the optimal policy.

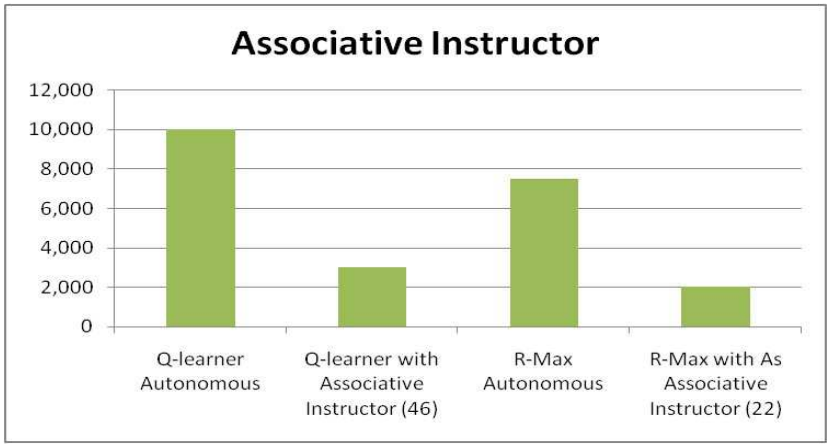

Figure 4.10: Number of steps taken by the learners to reach the optimal policy (Associative Instructor).

Figure 4.10 compares the performance of the two learning algorithms, that learns from a directive instructor, the task of safely driving a car on a highway. The performance is calculated as a measure of the number of collisions encountered before the agent is optimal and the number human interactions that were required to reach an optimal policy.

4.3.3.2 Effect on the Instructor

This experiment was carried out with two human subjects both of whom were unaware of this form of interaction. After providing the preliminary details, the humans played associative instructors to the learning agents for about 3 to 4 mins. During this time, they reported that it was very difficult to assign clusters to states by assigning numerical values. They felt that it was better to express the same thoughts using verbal commands, in a more socially engaging manner. They also reported that the R-Max agent was able to utilize the information they provided better than the Q-learner.

4.3.4 Which Agent?, Which Instructor?

After having described three different modes of interaction, we now ask what type of algorithm along with what type of instructor would be perfect to teach an RL agent to drive a car on a highway. From the plots shown in the previous sections, the R-Max learner seems to the better of the two learners and the Optimal Associative Instructor seems to be the best form of teaching. The drawbacks with respect to the R-Max learner is that it explores every part of the state space even if there is nothing important and the associative instruction method seems to be a very less natural mode of interaction as compared to rewards and actions. We now make a tradeoff between the best learner and the most natural form of interaction - the Motivational Instructor with the R-Max Learner.

In this study, we have extended the standard forms of human interaction to other modes and instantiated it in a car driving simulator. We were able to relate the inputs of a learning algorithm to natural modes of communication that humans are familiar with. This completes the last aspect of HELP in this Thesis.