Independent Study Report

Nirvan Singhania, Sri Keshav Kothapalli

Advised by Professor Praveen Paruchuri

May 18, 2020

1 Introduction

In Reinforcement Learning, a common data efficient approach is Transfer Learning. When an RL agent leverages TL, it uses knowledge acquired in one or more (source) tasks to speed up its learning in a more complex (target) task.The most prominent transfer unit between agents is action. This is also referred to as advice transferred between an expert known as the Teacher and a receiver known as the Student.

Thus Action Advising is a popular area of research for transfer learning between agents. We will be concentrating on the different ways of modelling the Teacher which primarily deals with two problems -

- Decide what to advice
- Decide when to advice using a limited advice budget

Thus our entire work has been divided into three stages

1. Modelling of a teacher agent using heuristics [5]. Here the teacher agent does not have a model and just employs an algorithm to come to a decision given the student information. This involves three types of advising.
    - Early Advising
    - Importance Sampling

  - Mistake Correcting

2. Modelling a teacher as an RL agent learning to teach , using **Q-Teaching** algorithm [3]. Another notable work not covered but related to this is by Zimmer et al. [7]

3. Using a **curiosity driven** model for an agent for self-supervised exploration [6]. Here we do not seek rewards from the environment and the agent is able to learn the task by acting as a teacher for himself.

2 Heuristic-Based Advising

The environment used here is Taxi-V3. Here the agent has to pickup a passenger from a location and drop him at his destination, where these locations and destinations are positions in a fixed dimension grid.

In this setting there is an agent which acts like a teacher and the teacher has a **budget**. The task of deciding when to advise a learner(student) gives rise to different possible algorithms which are detailed below.

For comparison purposes we used same number of training episodes (10000) for all three algorithms used in this section.

2.1 Early Advising

Early Advising - Here the main idea is that the teacher **uses up all of its advice budget in the initial learning stage** of the student.

Observations:

- Initially the student on following advice can perform as well as the teacher on the task.(hence the horizontal line)
- However once when the budget finishes, the student performance diminishes and it starts again, from a somewhat better position.
- The rest of the graph is symbolic of the student learning without the teacher.

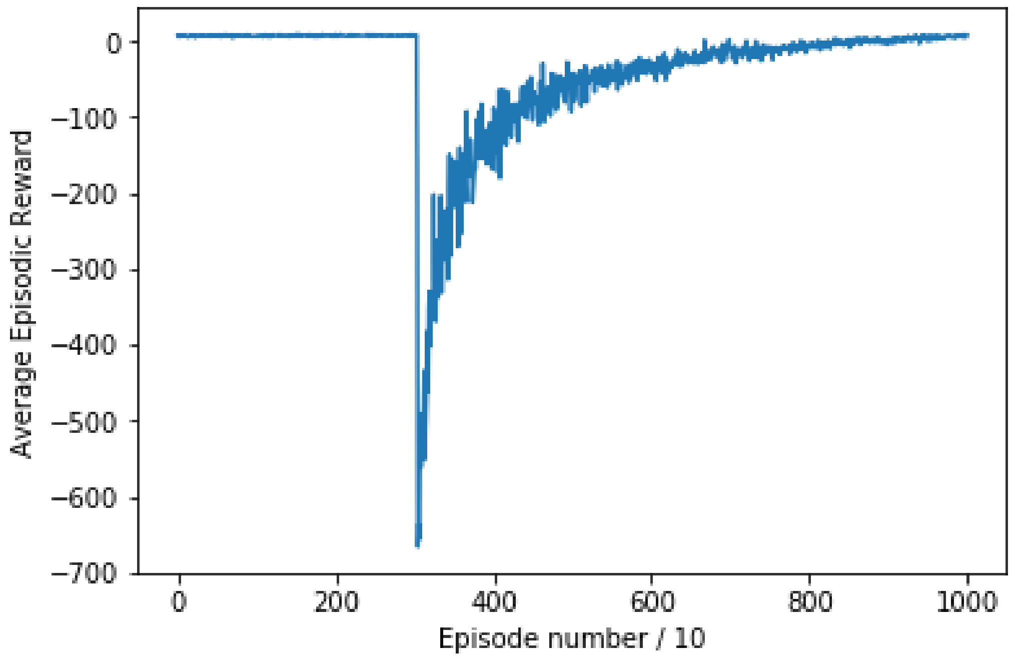


Figure 1: Early Advising plot of average episodic reward vs number of episodes

2.2 Importance Advising

Main Idea: Since the advice is limited, **give advice to the student only on situations when it is important**.

The importance for a state s is determined by:

$$I(s) = max_a Q(s,a) - min_a Q(s,a) \tag{1}$$

And the advice for that state is given only when $I(s) \geq t$, where t is a threshold value.

Observations:

- The learning is much smoother as compared to early advising)
- The agent does not suffer from drastic performance drops and moves to saturation effectively.

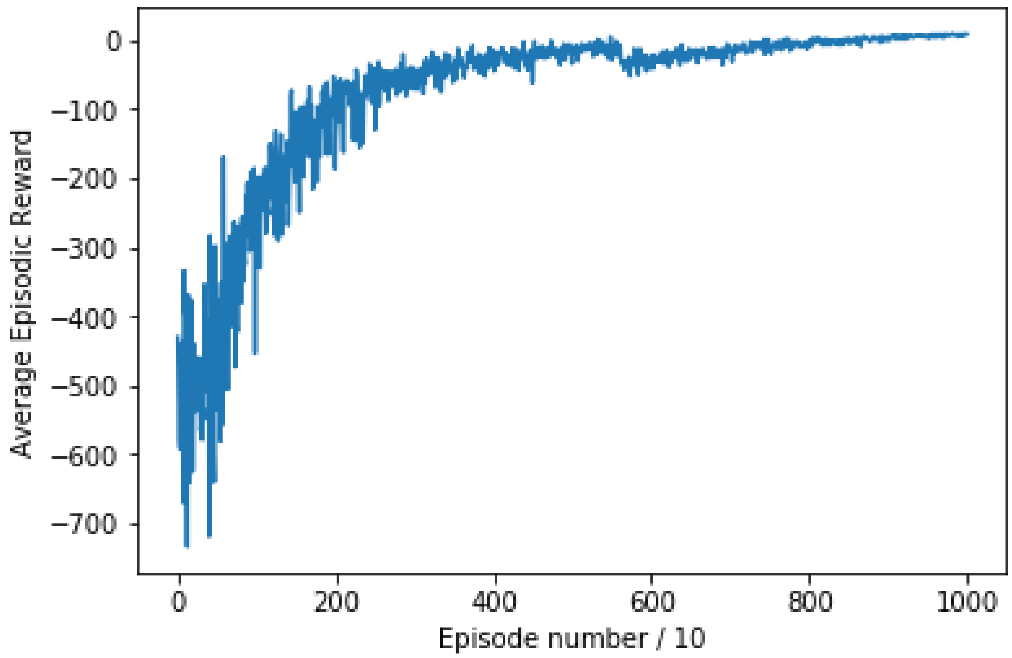


Figure 2: Importance Advising plot of average episodic reward vs number of episodes

- There is a small drop before episode 600, symbolizing the stage where budget has been used up. Thus we see that the agent is able to transition from an **advised** policy to a **non-advised** policy pretty smoothly.

2.3 Mistake Correcting

Main Idea: To use the budget better, **give advice to the student on important situations provided he is not taking an optimal action**. This is thus an advanced variation of Importance Advising.

Advice for a state is given only when $I(s) \geq t$, and $a' \notin armax_a Q(s,a)$ where t is a threshold value, a' is the student action.

Observations:

- The graph is still smooth just like importance advising)
- However there is no longer a small performance drop in later stages

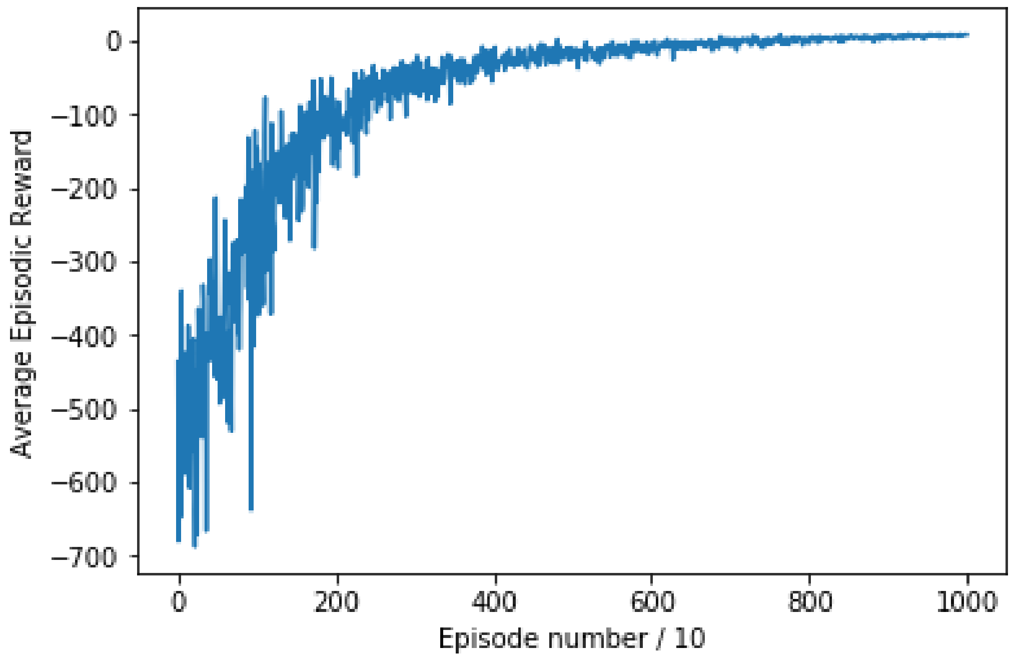


Figure 3: Mistake Correcting plot of average episodic reward vs number of episodes

and the graph moves smoothly towards saturation. This is because a stage where all budget has finished was not reached.

The main advantage of this approach is efficient use of the budget. The budget for all three algorithms was 40000

Algorithm	Budget Remaining at end
Early Advising	0
Importance Advising	0
Mistake Correcting	10442

Table 1: Here are the values of budget remaining after training for each of the given algorithms.

Note in 1 mistake correcting approach still has about a fourth of the budget remaining while importance advising completely exhausted the budget.

2.4 Comparison of Different Approaches

Below in 4 we compare the graphs of all the above algorithms with that of a normal training graph.

Observations:

- Early Advising performs the worst while both Mistake-correcting and Importance advising are the best.
- As compared to the normal training the advising methods based on importance of state, yield higher reward and also reach saturation much early
- Thus we can claim that they are trained faster and better using advice in a budgeted form as compared to a simple training without advice.

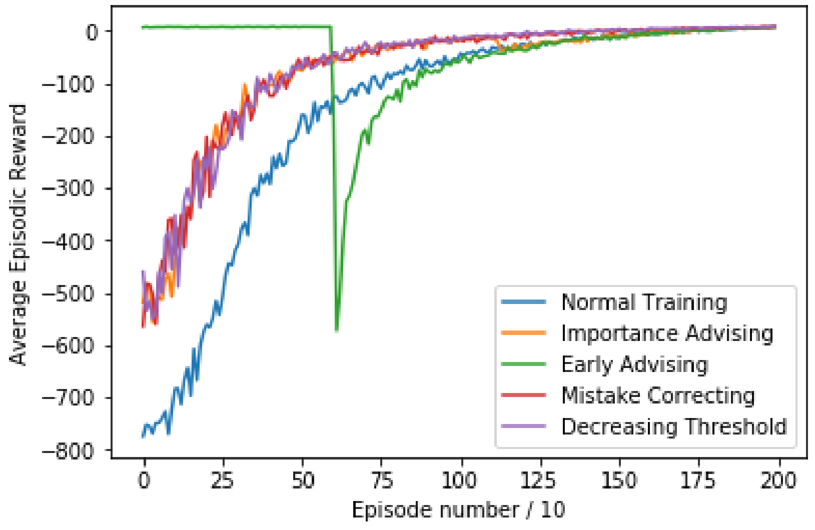


Figure 4: Comparison

2.5 Experiment with student algorithm

We used the same heuristic (Importance Advising) on student using different algorithms namely, Q-Learning, SARSA and Expected SARSA to see their affect.

Thus this means in each setting the trained teacher giving advice was the same but the student was using a different algorithm to train.

Observations:

- Q-Learning performs best while SARSA performs the worst.
- Expected SARSA has more drastic drops than Q-Learning, which symbolizes that Q-Learning is able to generalize more smoothly after exhaustion of budget.

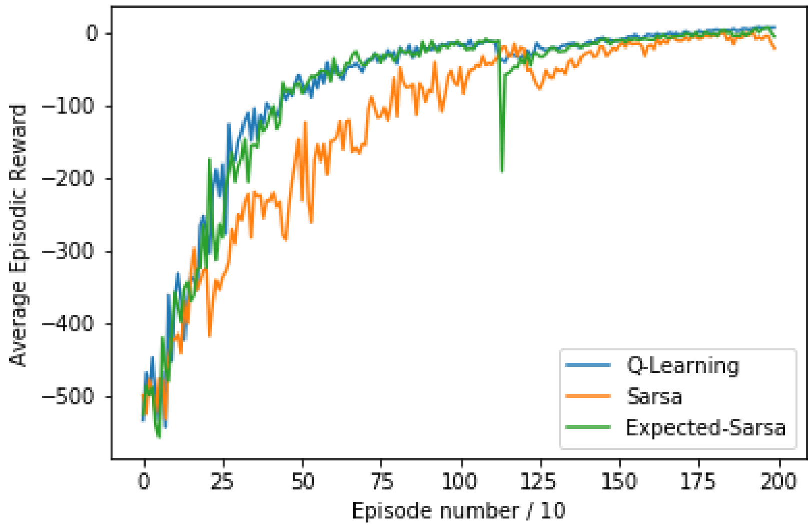


Figure 5: Student Algorithm Comparison

3 Q-Teaching (Theory)

3.1 Why move to RL for teacher

The main idea in heuristic based approaches was that we had an agent trained on a task. We called this agent the Teacher. The Teacher employed different algorithms to distribute advice to students(untrained agents) under a limited budget.

One drawback of these methods is that since they are based solely on the teacher's q-values they are not able to handle non-stationarity in the students learning task, and also have to be given a threshold of q-value differences, above which a state is considered important. This parameter needs to be manually tuned for each student.

Another important observation is as follows

The advice given in these approaches is independent of the budget and is only dependent on whether the budget is positive or not.

Hence the Teacher in this case has no idea how to manipulate advice based on different amounts of remaining budget. In other words if remaining advice is 10 or 10000 will not affect the Teacher's decision in any way.

3.2 Definitions

Student - A student agent is an agent acting in an environment and capable of accepting advice from another agent

Teacher - A teacher agent is an agent capable to execute a teaching policy and provide action advice to a student agent learning a specific task.

Acting Task - The acting task is the task for which the teacher gives advice and can be defined as an MDP of the form $M =< S, A, T, R, \gamma >$ on an environment E.

Teaching Task The teaching task is the task of providing action advice to a student agent to assist him in learning faster or learning better the acting task. Any teaching task is accompanied by a finite advice budget, B.

Teaching Action Space Given the action space A of the acting task, the action space of the teacher in timestep t depends on budget b_t

$$A_T = \begin{cases} \{a, None\} & \text{if } b_t \geq 0 \\ \{None\}, & \text{otherwise} \end{cases}$$

where $a \in A$

Teaching State Space The teacher agent state space in timestep t has the following form: $S_T =< s_t, a_t, b_t >$, where s_t is the student state, a_t is the student action and b_t is the current budget remaining.

Teaching Policy A teaching policy π_T is a deterministic policy of the form $\pi_T(S_T) = A_T$

3.3 Goal

The goal is to come up with a Teacher which can solve the task of teaching or in other words find the optimal teaching policy π_T for a specific task.

Thus there are two phases:

1. Acting Task Learning Phase: Here the Teacher first has to itself learn the task and arrive at an optimal **task policy** π_E. This can be done using any known standard algorithm

2. Teaching Task Learning Phase: The Teacher already trained on the Acting Task,now has to learn an optimal **teaching policy** π_T

3.4 Similarity with Exploration-Exploitation

The decision of teacher whether to advice or not can be viewed as the famous exploration-exploitation problem in RL. When the teacher gives an advice, an obedient student follows it thus becoming a deterministic actuator for the teacher viewed as an acting agent here. The teacher thus by advising its best action acts greedily and thus exploits.

On the other hand, when not giving advice the teacher entrusts the student to autonomously control its environment. Hence from the teacher's point of view, it is letting the student explore.

Thus teaching policy is similar to an exploration-exploitation policy.

3.5 Value Regret

In a convergence horizon T, the value regret, R^V of an exploration-exploitation policy (i.e., teaching policy) with respect to both an acting policy π^* obtained after the T period and an acting policy (i.e., student's policy), π_t, in time step t is:

$$R^V = \sum[\max_a Q^*(s_t, a) - Q^*(s_t, \pi_t(s_t))]$$

where Q^* denotes the corresponding value function of π^*

The intuition behind this definition of regret in this context (where the acting agent is the student) is that **the best teacher for any specific student would ideally be the student himself, when it would have reached convergence or its near-optimal policy.**. And at such a stage the value regret will be equal to 0.

Hence the teaching policy must **minimize** the value regret.

Cited from the paper, **The important thing to note here is that because a student agent receives a finite amount of advice it cannot improve its asymptotic performance, consequently the evaluation of a teaching policy should ideally be based on the student's optimal policy and not to that of some probably very different teacher, because that is its sustainable optimality.**

Its sustainable optimality is defined as to what is optimal given its simplistic internal representation. A teaching policy should be ideally evaluated on how much it speed up the student converging to its own optimal policy.

3.6 The algorithm

We need to train the teacher to make it learn the teaching policy. Following are some common terms:

We already know the teaching state space and teaching action space.

State transition for teacher Consider it is advising a student which currently is in state s, is about to take action a and budget for teacher left is b. State of teacher becomes $< s, a, b >$.

Let the advice be p, following which, the student moves on to the next task state $s^{'}$. For this state student action is $a^{'}$. Also if $a^{'} == None$ budget remains b, else it reduces to $b-1$.

Thus next state is either $< s^{'}, a^{'}, b >$ or $< s^{'}, a^{'}, b-1 >$

Teaching Episode/ Session This refers to one episode for training the teacher. A teaching episode will continue until the budget reduces to 0 or the student reaches its convergence horizon. **Thus a teaching episode may include multiple student episodes.**

Reward for teacher This will be basically the value regret. In this case we assume that we already have the student's action declared hence there is no need for predicting student's action. At any step, we have reward as,

$$r_T = \begin{cases} Q_{\sum}(s, a^{\star}) - Q_{\sum}(s, a^{'}) & \text{if } a_T = advice \\ 0, & \text{otherwise} \end{cases}$$

where $Q_{\sum}$ is the state-value function for acting policy, $a^{*} = \text{argmax}_a Q_{\sum}(s, a)$ and $a^{'}$ is the predicted student action.

This reward has a high value when the value of the greedy action is significantly higher than the value of the action that the student would take. This means that the **teacher is encouraged to advise when the advised action is significantly better than the action the student would take.**

Q-Teaching rewards all **no advice actions** with 0. The advantages of such a scheme is that the teacher's cumulative reward is based only on the value gain produced when advising and **a teaching episode can finish when the budget finishes**

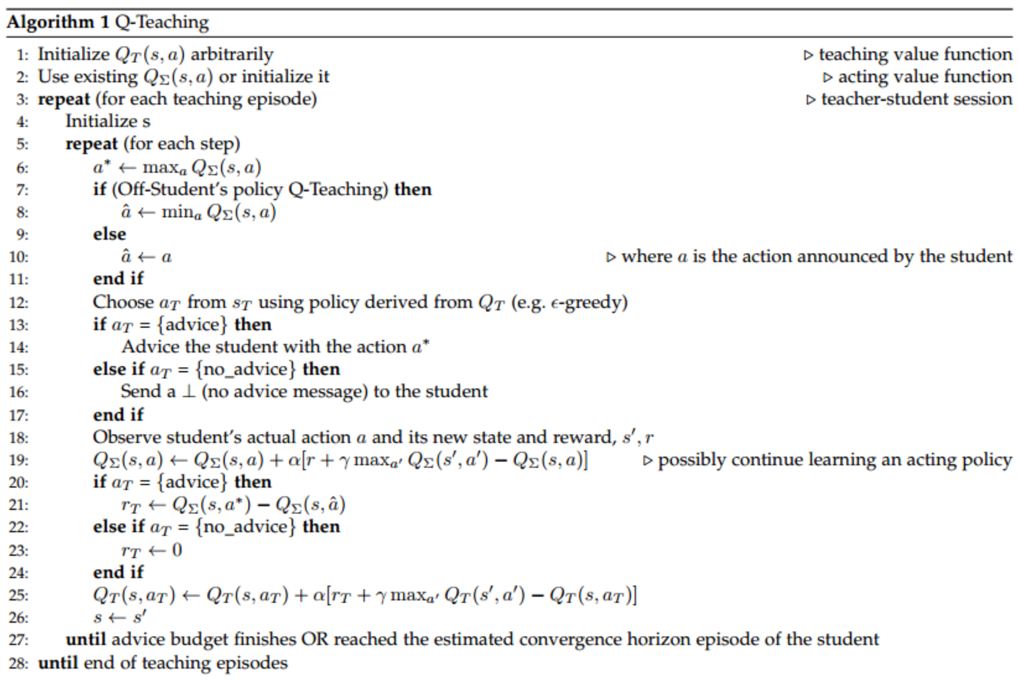
Off-Policy and On-Policy Q-Teaching These two are born depending on how the teacher predicts student action . The On-Policy method uses the action produced by the student itself and thus uses its value to compute the reward.

On the other hand , in Off-Policy uses the principle of assuming the worst and hence assumes that the student will be selecting the worst actions. Thus it computes a reward based on difference of maximum and minimum $Q_{\sum}$ values for a state.

Update Step for Teacher

Denote Q_T as the state-value function for teaching policy which is to be learnt. At every step let the teacher reward be r_T. current state be s_T, action(advice) be a_T, then we have, the update similar to Q-Learning based on TD-Error.

$$Q_T(s_T, a_T) = Q_T(s_T, a_T) + \alpha[r_T + \gamma max_{a_T^{'}} Q_T(s_T^{'}, a_T^{'}) - Q_T(s_T, a_T)]$$



Algorithm 1 Q-Teaching

```
1: Initialize $Q_T(s,a)$ arbitrarily                                   ▷ teaching value function
2: Use existing $Q_\Sigma(s,a)$ or initialize it                        ▷ acting value function
3: repeat (for each teaching episode)                                   ▷ teacher-student session
4:     Initialize s
5:     repeat (for each step)
6:         $a^* \leftarrow \max_a Q_\Sigma(s,a)$
7:         if (Off-Student's policy Q-Teaching) then
8:             $\hat{a} \leftarrow \min_a Q_\Sigma(s,a)$
9:         else
10:            $\hat{a} \leftarrow a$                    ▷ where $a$ is the action announced by the student
11:        end if
12:        Choose $a_T$ from $s_T$ using policy derived from $Q_T$ (e.g. $\epsilon$-greedy)
13:        if $a_T$ = {advice} then
14:            Advice the student with the action $a^*$
15:        else if $a_T$ = {no_advice} then
16:            Send a $\perp$ (no advice message) to the student
17:        end if
18:        Observe student's actual action $a$ and its new state and reward, $s', r$
19:        $Q_\Sigma(s,a) \leftarrow Q_\Sigma(s,a) + \alpha[r + \gamma \max_{a'} Q_\Sigma(s',a') - Q_\Sigma(s,a)]$   ▷ possibly continue learning an acting policy
20:        if $a_T$ = {advice} then
21:            $r_T \leftarrow Q_\Sigma(s,a^*) - Q_\Sigma(s,\hat{a})$
22:        else if $a_T$ = {no_advice} then
23:            $r_T \leftarrow 0$
24:        end if
25:        $Q_T(s,a_T) \leftarrow Q_T(s,a_T) + \alpha[r_T + \gamma \max_{a'} Q_T(s',a') - Q_T(s,a_T)]$
26:        $s \leftarrow s'$
27:    until advice budget finishes OR reached the estimated convergence horizon episode of the student
28: until end of teaching episodes
```

Figure 6: The algorithm stated in the paper

4 Q-Teaching (Results)

4.1 Teacher Training

Goal Here we will be training the teacher to make it learn a teaching policy. This is to understand the trends in total rewards received by a teacher in every **teaching episode**

Setting The teacher is an agent with full knowledge of task(with optimal acting policy) and we have a student using Q-Learning to learn the task.At every step of a teaching episode, the teacher gets a reward r_T .For every teaching episode , we have with us the **total teacher reward obtained for that episode**. The rewards are averaged over 10 teaching episodes .Thus we plot the Average Teacher Episodic Reward vs Teaching Episode Number.

Specifications	Value
Algorithm	Off-Policy Q-Teaching
Budget for every Teaching Episode	100
Alpha	0.2
Gamma (Discount Factor)	0.98
Number of Teacher Training Episodes	4000
Discretizing Factor for Budget	10
Number of episodes to average over	10

Table 2: Training-related Parameters

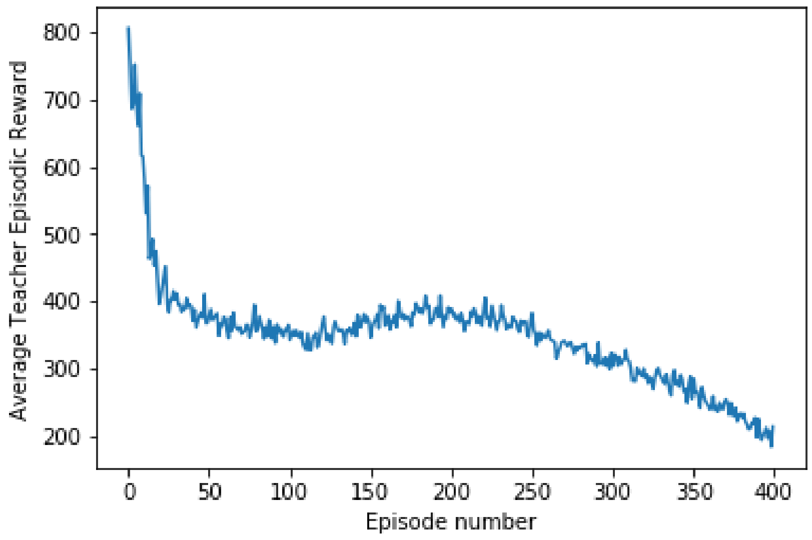


Figure 7: Rewards of the Q-Teaching RL teacher agent during training phase

The teacher's episodic reward depends on the difference of Q-values of best and worst states. Following is an analysis of the training:

- Initially the student does not know the right decision to take for important states that is states with very high difference in their Q-values. Hence initially in the beginning rewards are higher and start decreasing as the student learns about the most important states.
- With more time the states concentrated on have a relatively smaller difference and the teacher learns in this case to spend budget more

cautiously, since previously a single use of budget meant a great increase in reward. But this is not the case here. Hence the shape of the curve becomes like a upward plateau.

- Slowly after some episodes of training, when the student has been trained the rewards seem to decrease since teacher is now less required for advising.

Thus there are three stages of student and the training reward of teacher shows the rewards the teacher get for each of these student stages namely

- Untrained Student
- Semi-trained Student
- Trained Student

4.2 Monitoring Student Progress during Teacher Training

Goal In order to make sure that the teacher is not adversely affecting student's performance, after every interval of 30 training (teaching) episodes, we take the student and run it for 10 student episodes where it does not learn anything and just interacts with the environment without any advice from the teacher.

Setting The average student episodic reward for this is obtained and then plotted against the interval number. This gives us information on how well the student was able to perform at different stages of the teacher training. **If there is a downward trend that means the teacher's advice steered the student into the wrong direction or impaired its learning instead of helping it.**

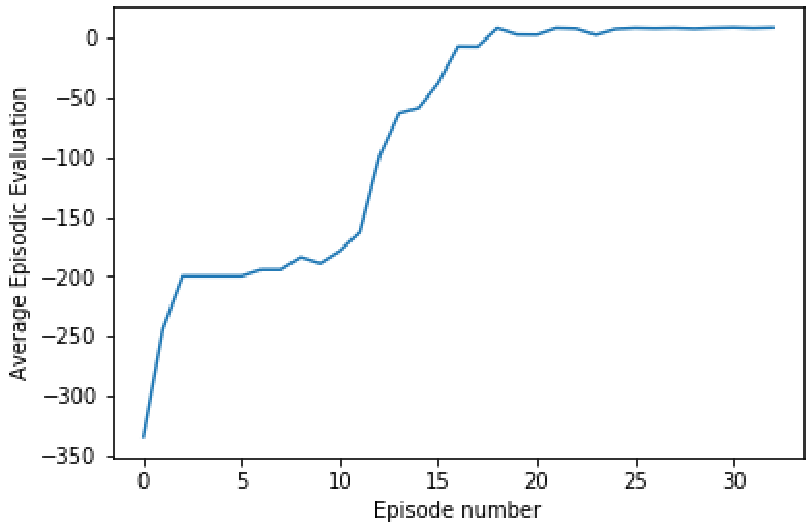


Figure 8: Average episodic reward of student vs Interval number During teacher training

Observations:

- Notice that the graph obtained is a little different from the one obtained when a student is trained without advice. This is because it is representing progress of a Student under a teacher training session not during a task learning episode.
- We can see that even in this case the asymptotic behaviour of the student remains unchanged.

4.3 Testing of trained Teacher

Goal Now that we have a trained Teacher(it has an optimal task policy as well as optimal teaching policy). we use it on an untrained agent (with no task-level knowledge) to see how it reaches its goal with the help of teacher's advice.

Setting In this case thus teacher has a budget. The Student starts learning with its own algorithm (Q-Learning) and at every step asks teacher for advice. Based on the student's state, intended action and budget the teacher gives the advice. Note that this training comprises only student episodes (= 50000)

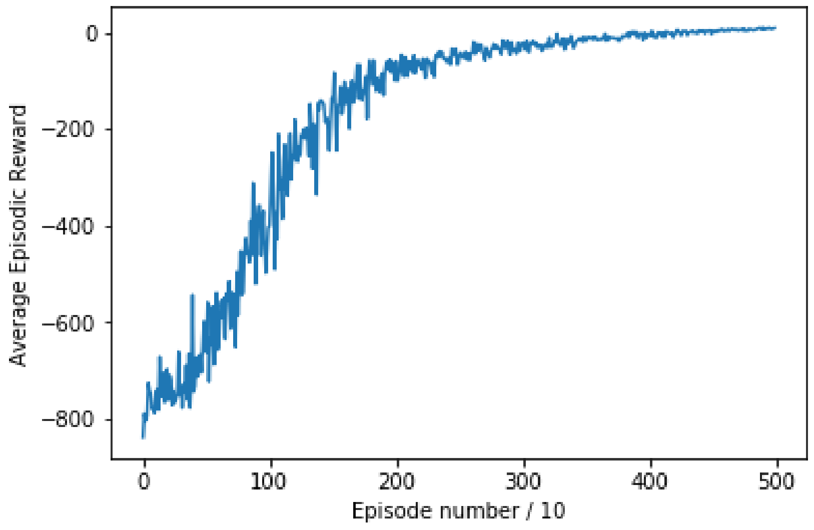


Figure 9: Student average episodic reward vs episodes

4.4 Comparison with other teaching methods

Goal In this section we compare the approaches for Q-Teaching with the best heuristic-based algorithm(mistake-correcting) and normal training method without teacher. We know that the ultimate measure of how good a teaching policy comes down to how well it has been able to help a student converge to its optimal task-level policy.

Setting For each of the algorithms a trained teacher is taken. (No training of teacher for heuristic-based algorithms.) Now for each of them we have an untrained student which will begin its learning in the presence of advice of these teachers. Student corresponding to Q-Teaching is trained for only 5000 episodes while that under the advice of others is trained for 10000 episodes. The progress of the student under the influence of the teachers is plotted.

Observations:

- Both Off-policy and On-policy Q-teaching have shown much better results as compared to other algorithms.
- They have outperformed not only the normal training algorithm but also heuristic based algorithm as visible in the graph.
- Within about half of the episodes used by mistake-correcting the Q-Teaching algo has reached the saturation level reward. Hence we have faster convergence and higher rewards.

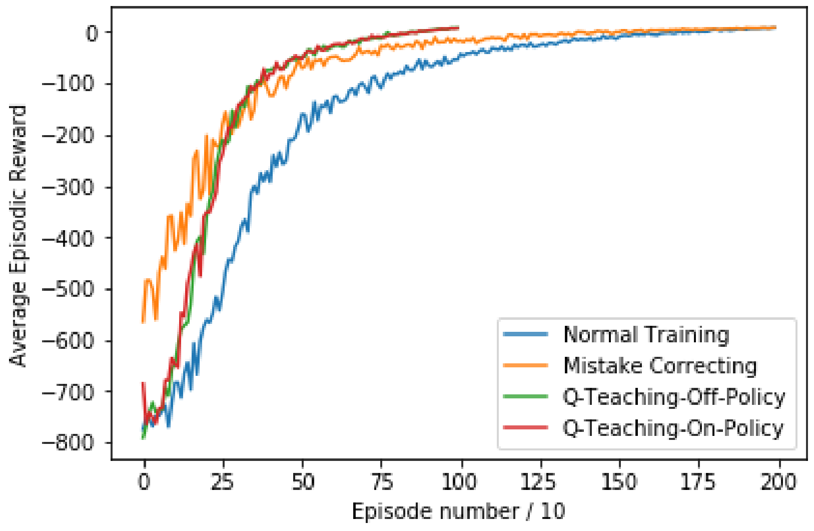


Figure 10: Average Episodic Reward vs Number of Episodes

5 Curiosity based Exploration (Theory)

5.1 Why Curiosity

Two main problems of Reinforcement Learning:

1. **Sparse rewards or non-existing rewards problem** Most rewards do not contain information, and hence are set to zero. However, as rewards act as feedback for RL agents, if they don't receive any, their knowledge of which action is appropriate (or not) cannot change.

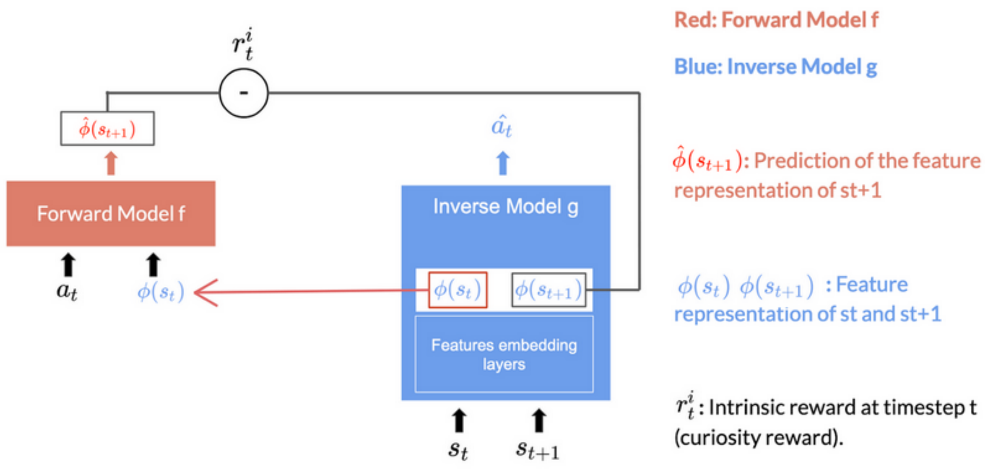
2. **Extrinsic reward function is handmade** In each environment, a human has to implement a reward function, which is hard to scale in big and complex environments.

The idea of Curiosity-Driven learning, is to build a reward function that is intrinsic to the agent (generated by the agent itself). It means that **the agent will be a self-learner since he will be the student but also the feedback master.**

Curiosity is an intrinsic reward that is equal to the error of our agent **to predict the next state given current state and action taken.** This encourages our agent to perform actions that reduces the uncertainty in the agent's ability to predict the consequence of its own action.

Measuring this curiosity value thus requires building a model which predicts next-state.

5.2 Intrinsic Curiosity Module



r_t^i
-
$\hat{\phi}(s_{t+1})$
$\hat{a}_t$
Forward Model f
Inverse Model g
a_t
$\phi(s_t)$
$\phi(s_t)$
$\phi(s_{t+1})$
Features embedding layers
s_t
s_{t+1}
Red: Forward Model f
Blue: Inverse Model g
$\hat{\phi}(s_{t+1})$: Prediction of the feature representation of st+1
$\phi(s_t)$ $\phi(s_{t+1})$: Feature representation of st and st+1
r_t^i: Intrinsic reward at timestep t (curiosity reward).

Figure 11: The Intrinsic Curiosity Module

The Inverse Model:

Inverse model helps in generating feature representation of state. The state representation has to have good features in order to reduce the complexity of next-state prediction. A good feature space must have 3 features:

1. Needs to model things that can be controlled by the agent
2. Also needs to model things that cannot be controlled by the agent but can affect it
3. Should not model things not in agent's control and also not affecting it.

The Inverse Model (g), aims at predicting the action $a_(t)$, and in doing so, it learns an internal feature representation of the state and next state, denoted by $\phi(s(t))$ and $\phi(s(t+1))$

$$a_t = g(s_t, s_{t+1}; \theta_I)$$

where a_t is the Predicted Action, g is the learning function for inverse model, s_t the current state, s_{t+1} the next state and θ_I represents the inverse model parameters.

Since this neural network is only used to predict the action, it has no incentive to represent within its feature embedding space the factors of variation in an environment that does not affect the agent itself or the agent does not control. which is exactly what is required

The inverse model loss is given by:

$$L_I = H(a_t, a_t)$$

where H represents the cross-entropy function and a_t is the real action, a_t the predicted action.

The Forward Model

We use the above feature space to train a forward dynamics model that predicts the future representation of the next state $\phi(s_{t+1})$, given the feature representation of the current state $\phi(s_t)$ and the action a_t

$$\phi\ (s_{t+1}) = f(\phi(s_t), a_t; \theta_F))$$

where $\phi\ (s_{t+1})$ feature representation of next state, $\phi(s_t)$ feature representation of current state and a_t current real action taken, f the forward model learning function, θ_F the forward model parameters.

The forward loss function is given by:

$$L_F(\phi(s_t), \phi\ (s_{t+1})) = \frac{1}{2}||\phi\ (s_{t+1}) - \phi(s_t)||_2^2$$

The intrinsic reward is given by

$$r_t^i = \frac{\eta}{2}||\phi\ (s_{t+1}) - \phi(s_t)||_2^2$$

where η is a positive scaling factor.

Final minimization problem reduces to:

$$\min_{\theta_P, \theta_I, \theta_F} \left[-\lambda\, \mathbb{E}_{\pi(s_t;\theta_P)}[\sum_t r_t] + (1-\beta)L_I + \beta L_F \right]$$

where θ_P represents the policy parameters, λ is a scalar that weighs the importance of policy gradient against intrinsic reward, β weighs inverse model loss against forward model loss.

Finally the intrinsic reward calculated is supplied to the agent as curiosity.

6 Results

6.1 Experiment

We have run the algorithm on the environment "Cartpole-V1". State space is a vector of length 4 of continuous values. Action space is discrete with 2 values. For more information here is the environment specification

The episode is run for a maximum of 200 steps and reward is 1 for every step taken. The environment is considered solved when the average reward for 50 trials comes out to be grater than 195.

6.2 Performance of ICM

We train the agent using the models described above. We do not take into consideration **any of the rewards obtained from the environment**

Setting The goal here is to see how well the agent learns the task. In the first plot, we use the score obtained from the environment for the graph. That is the agent is trained on the basis of the intrinsic reward, but in order to evaluate how well it is getting trained, we calculate the total reward it received from the environment in each episode and plotted it.

Below are the parameters used for training:

Specifications	**Value**
Algorithm	ICM+A2C
Environment	CartPole-V1
Maximum number of episodes	1000
Beta (β)	0.2
lambda(λ)	0.1
eta(η)	100
Discount Factor	0.99
Learning Rate (Actor)	0.001
Learning Rate (Critic)	0.005
Learning Rate (ICM)	0.001

Table 3: Parameters and their values

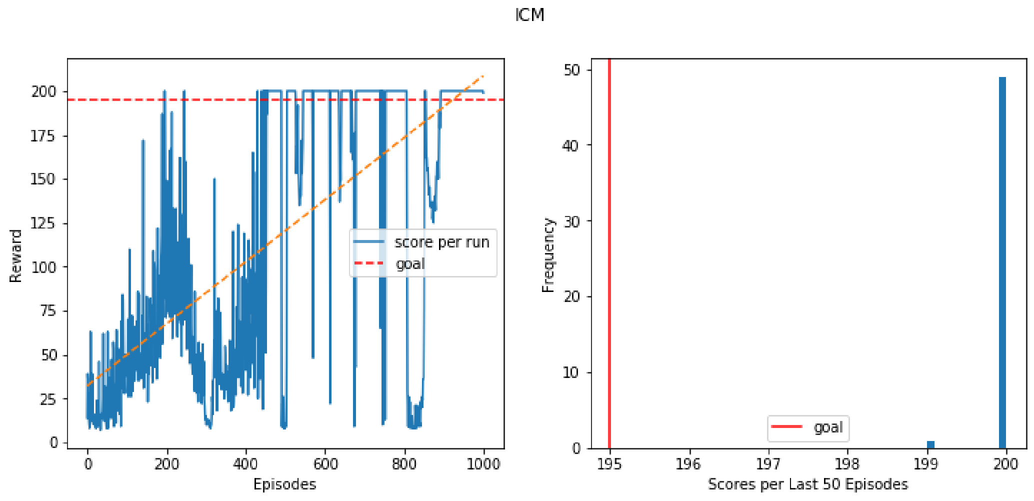


Figure 12: ICM Performance

Observations:

- The red line symbolizes the goal for the reward (200). The task is complete the average reward ≥ 195 for 50 trials. As visible that agent has been able to learn the behaviour.
- The right plot shows the frequency histogram for the last 50 episodes and it is clear that all values are greater than the goal.
- Below we also plot the moving average where we see that the average is always more than 195. This is a proof that the problem is solved.

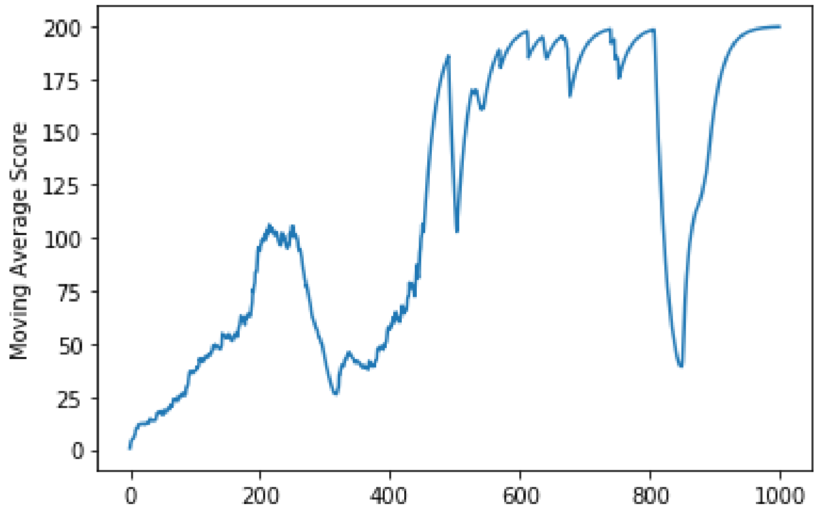


Figure 13: Moving average of the environment episodic reward vs Episodes

Intrinsic Reward Now we also plot the intrinsic reward vs the number of episodes.

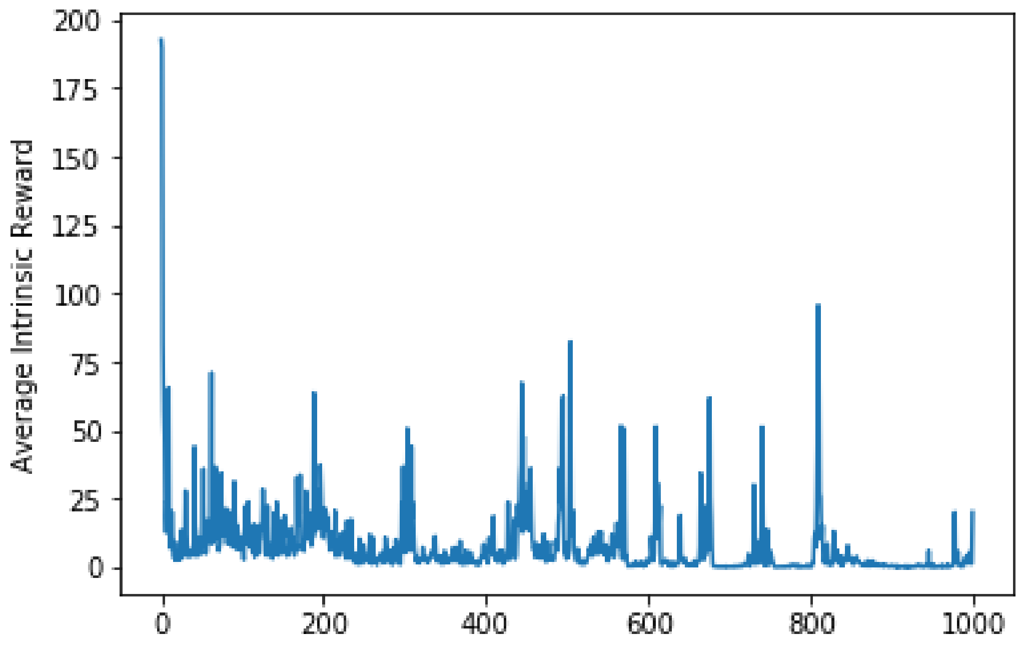


Figure 14: Intrinsic Episodic Reward vs Number of Episodes

Observations

- This is very different from a regular reward vs episodes graph
- Initially there is a very high reward because curiosity is high when there are unexplored parts
- There are spikes in rewards obtained as the agent ventures into uncertain spaces
- Towards the end we see the reward diminishing to 0 since the agent is quite certain now of the consequences of its actions.

6.3 Comparing with DQN results

The same environment was trained with the standard Double-DQN with experience replay. To demonstrate its performance, the corresponding graphs were plotted.

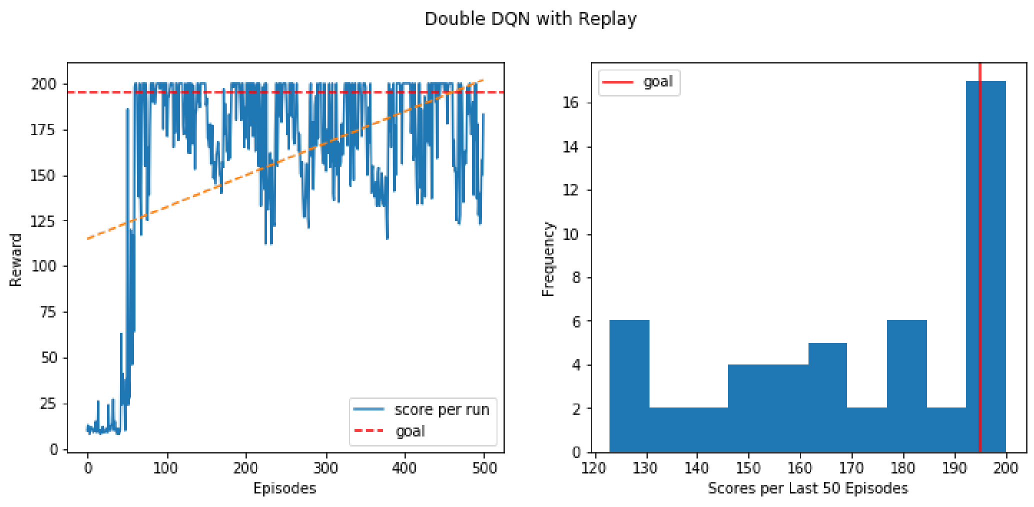


Figure 15: Performance with Double DQN with replay

Observations:

- As visible the ICM performance is fairly comparable to that of DQN.
- Even without having access to reward for a given state and action, ICM ends up accumulating higher rewards than DQN.
- Although DQN is faster in learning the task but ICM + A2C gave better results on the task.

6.4 ICM Advancements

ICM has some concerns regarding stochastic elements in the environment. This is famous as the **noisy TV problem** This is attributed by using **Random Network Distillation** methods [2], [1]

There have been attempts to use these random networks for multi agent action advising. [4]

7 Conclusion

Summarizing the work, we were able to demonstrate the role of action advising of a Teacher in three different ways

- Heuristic Based
- Reward-Dependent RL Agent Based
- Curiosity Based Model

Each method has its own pros and cons. In order for use in continuous spaces and multi-agent systems, the requirement of using deep networks in reinforcement learning is also paramount. Each of the methods share some common ideas which can be seen manifested in the definitions of Importance Advising, Value Regret and Curiosity.

References

[1] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning, 2018.

[2] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation, 2018.

[3] Anestis Fachantidis, Matthew Taylor, and Ioannis Vlahavas. Learning to teach reinforcement learning agents. *Machine Learning and Knowledge Extraction*, 2017.

[4] Ercument Ilhan, Jeremy Gow, and Diego Perez Liebana. Teaching on a budget in multi-agent deep reinforcement learning. pages 1–8, 08 2019.

[5] Matthew E. Taylor Lisa Torrey. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, 2013.

[6] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 2017.

[7] Matthieu Zimmer, Paolo Viappiani, and Paul Weng. Teacher-student framework: A reinforcement learning approach. 05 2014.