Crowd-sourcing Advice from Multiple Experts in Reinforcement Learning

Rohit Sonker
Summer Intern
StARLinG Lab
University of Texas at Dallas

1 Problem Motivation

The final goal of any reinforcement learning or planning algorithm is to explore the environment and learn the required objective within a limited number of trails. However, with complex and dynamic environments and goals of the real world, simple reinforcement learning fails to inherently learn all the desired objectives. Any real world task, for example driving a car, requires multiple goals to be properly defined. Moreover, real world scenarios involve complex and dynamic environments, hence the agent is unable to learn the desired behaviour. Furthermore, since the states are only partially observable, the agent is unable to learn the small nuances of the desired behaviour, which maybe extremely crucial in applications such as self driving cars.

Incorporating human advice to a learning algorithm is one natural way to address this issue. These advice rules are taken from experts of the domain and can be used to speed up learning. More importantly, these advice rules can help the expert learn behaviour which may otherwise be impossible to learn due to the partially observable states.

Prior work to incorporate advice has been in the form of state and action preferences, Kunapali et al.[1], preferential labels in supervised learning, Odom and Natarajan [2], or by action preferences over a set of states, Maclin et al.[3].

In this work we consider the presence of multiple experts. Let us consider an example, suppose a self driving car considers advice by multiple drivers, some drivers may stop just before a "STOP" sign, others may gradually decelerate from a long way before, both of which may be beneficial to the algorithm at different instances. Or, consider a scenario in heavy traffic, different experts may have a different way of navigating through the traffic quickly. In single advice case, we always consider that we have an optimal expert. This may not always be true. Such cases require advice from multiple experts instead of a single one.

Hence, we need a method which can incorporate advice from the multiple experts and figure out the best advice. The algorithm should also be able to combine advice rules from the experts and the policy from the base algorithm to generate the best policy.

2 Problem Statement

The aim is to develop a method to incorporate advice from multiple experts in a reinforcement learning or task planning algorithm. The algorithm must be able to combine the advice rules with the policy from the base RL or planning algorithm to generate a better policy.

3 Proposed Model and Algorithm

The advice rules are taken as conditional statements which give us the preferences over actions in a certain state. These preferences can be used to create policy distributions.

We consider policy shaping as the approach to incorporate the advice taken from experts. The advice rules are first converted into policy distributions or policy factors, which can then be combined with the policy distribution from the reinforcement learning algorithm. This approach is similar to the bayesian approach followed by Griffith(2013)[4]. The proposed model is described below.

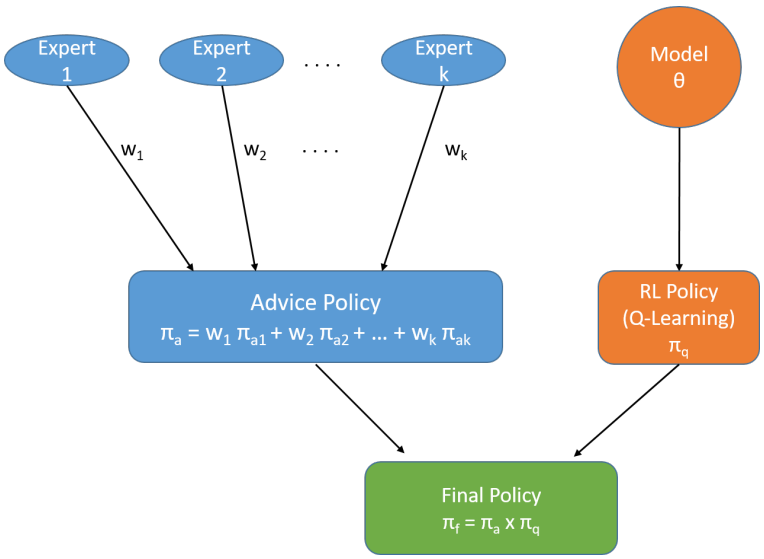

Figure 1: Policy Shaping by Incorporating Advice

It is assumed that we have k expert preferences which are used to generate k expert policy distributions π_{ak}. The advice is taken in the form of conditional statements which are only valid for certain states. Hence advice is only incorporated in the algorithm only on the states where it is applicable.

These multiple advice policies are weighted and then combined together to give the complete expert policy π_a . This expert policy is then combined with the policy given by the reinforcement learning algorithm, in this case we consider a Q-Learning algorithm, π_q to give the final policy π_f. The action taken at every step is according to this policy π_f.

The weights of the advice given by various experts and the Q-Learning updates are calculated using Expectation Maximization or soft alternating optimization. The weights of the experts are calculated in the E-step, and the reinforcement learning model is updated in the M-step. This process allows us to combine the advice rules instead of relying on the advice given by a single expert from the crowd of experts. This described in the algorithm below.

In the following algorithm,
π_q denotes the policy from the Q-learning / RL algorithm
π_a denotes the advice policy
π_f denotes the final combined policy
θ is the model/weights of the RL algorithm
w_k denotes the weight defined to the policy of an expert k
α is the learning rate of the RL algorithm
γ is the discount factor
$R(s, a)$ is the reward function

Algorithm 1

Initialize θ
for each iteration i **do**
  $s \leftarrow$ current state
  $A \leftarrow$ set of possible actions for state s
  for each action $a \in A$ **do**
    use θ^i to evaluate $\pi_q^i(s,a)$
    for each expert k **do**
      $\pi_{f,k}^i(s,a) = \pi_{a,k}(s,a) \times \pi_q^i(s,a)$
      let $c^{k,i} = \exp(E_{\pi_{f,k}}(\sum_{t'=t}^{\infty} R(s_{t'}, a_{t'})))$
      {In our Case, the Q Value under policy $\pi_{f,k}$} $c^{k,i} = \exp(Q_{\pi_{f,k}}(s_t, a_t))$
      let $w^{k,i} = \frac{c^{k,i}}{\sum_k c^{k,i}}$
    end for
    $\pi_a^i = \sum_k w^{k,i} \pi_{a,k}(s,a)$
    $\pi_f^i = \pi_a^i \times \pi_q^i$
  end for
  $action = \arg\max_a \pi_f^i(s,a)$
  {Now we update the model θ }
  $\theta^{i+1} = \theta^i - \alpha \nabla loss\_func(\theta^i)$
  {In our case for Q-Learning Algorithm}
  $Q(s,a) = (1-\alpha)Q(s,a) + \alpha(R(s,a,s') + \gamma \max_{a'} Q_{\pi_f}(s',a'))$
end for

4 Domain & Experiment Setup

The domain selected for this problem is the Pacman Project by UC Berkeley. The experiments were conducted on the 'Medium Grid' layout as described below. Q-Learning is used as the base reinforcement learning algorithm.

Figure 2: Pacman Medium Grid Layout

The main advantage of Pacman that makes it attractive for reinforcement learning methods is that both the state space and the action space are discrete and defining the game as an MDP is a straightforward process:

- A state $s \in S$ consists of the map layout and the current game score as well as the positions of Pacman, all ghosts and all remaining pac-dots.
- There are a maximum of five actions $a \in A$ available in each state: moving one step in one of the four cardinal directions as long as the path is not obstructed by a wall or standing still.

- The transition model $T(s, a, s')$ depends on the behavior of the ghosts. While there is no noise when applying an action, i.e. if the action a is going eastwards Pacman's position in the next state s' will always be one step further to the east than in s, the ghosts behave in a stochastic way.

- The reward function $R(s, a, s')$ is defined as the difference in game score that occurs when performing the action a in s that leads to s'. If Pacman eats a pac-dot he gains 10 points, if he wins the game by eating the final pac-dot on the map he gains 500 points and if he loses by colliding with a ghost he is punished by receiving -500 points. Additionally, in each time step the score is reduced by one to encourage fast play. In the implementation that was used eating a power pellet is not rewarded with any points but eating a ghost afterwards leads to 200 extra points.

The algorithm was run for 50000 episodes with an exploration probability of 0.10 to take random action. All the advice is given to the algorithm before the training starts. The advice considered for this experiment is to turn away when a ghost is the current direction of movement x steps away. Results are compared for three cases -

- **No Advice :** Learning works only according to the baseline Q-Learning algorithm.

- **Single Expert :** A single advice rule is given to the learning algorithm. The advice given here is to turn away when a ghost is 1 step ahead in the moving direction.

- **Multiple Experts :** Multiple advises are considered from different experts. The following different advice rules are applied -

    1. Expert 1 : Turn away when a ghost is the current direction of movement at most 9 steps away

    2. Expert 2 : Turn away when a ghost is the current direction of movement at most 5 steps away

    3. Expert 3 : Turn away when a ghost is the current direction of movement at most 3 steps away

    4. Expert 4 : Turn away when a ghost is the current direction of movement at most 1 steps away

    5. Expert 5 : Turn away when a ghost is the current direction of movement at most 7 steps away

The advice is incorporated by decreasing the policy factor in direction of the ghost to 0.1. All the other actions are given the factor of 1. This factor distribution is combined with the policy distribution from Q-Learning to give the final policy distribution. This final distribution is used to calculate the best action to take.

5 Results and Discussion

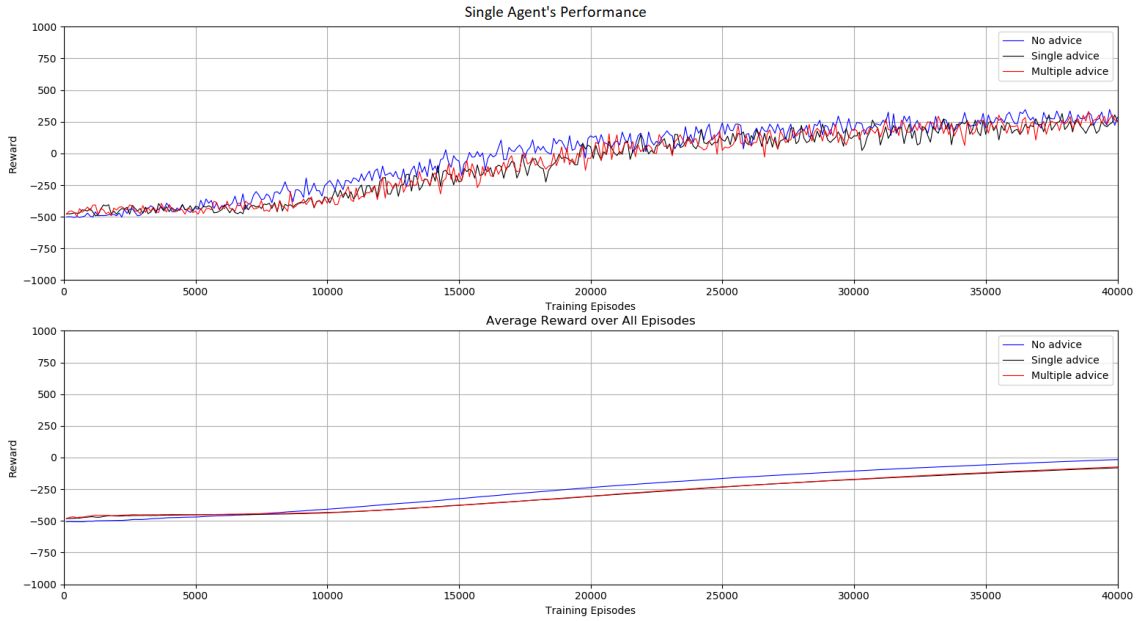

Figure 3: Rewards accumulated over 40000 episodes

As can be observed from the plot, all three methods converge at nearly the same time. Also the optimal reward value with or without advice is the same. This happens because Q-learning is able to learn the best way to avoid ghosts given enough number of episodes. However, if we consider the initial 1000 episodes, we are able to see the significant effects of giving advice.

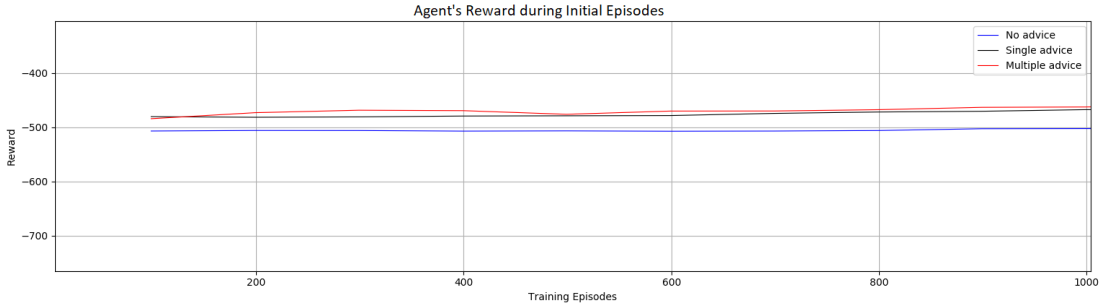

Figure 4: Rewards accumulated over first 1000 episodes

During the initial episodes, having advice is easily able to outperform the cases of having no advice. Now, single advice given in this case is to turn away when the ghost is just 1 step away. This is actually a good advice for the game. Incorporating the advice's of all experts in this case makes Pacman's behaviour more 'safe', as it avoids the ghosts before they come too close. This is beneficial in some cases whereas in some it is not optimal. Hence, single advice and multiple advice's produce nearly the same reward outcome over time. Multiple advice would have performed better, had the single advice not been so beneficial.

As we progress through the episodes, we reach a point where the simple Q-Learning algorithm performs at par and sometimes better than the advice algorithms. This happens because Q-learning is able to learn the optimal method to avoid ghosts and thereby accumulate maximum reward. The

performance of advice algorithm also depends on the advice being given. The ideal advice would be something that the algorithm is unable to learn on its own, or would take a long time to do so. In such a case the advice driven algorithms would converge to the optimal reward values significantly faster.

6 Future Work

The future work would be to build a better framework for incorporating more general advice from different experts. This would allow more robust testing of the given algorithm. Also, multiple advice rules can be combined by other methods such as Noisy Max to compare with the current EM based policy combining algorithm.

7 References

[1]Kunapuli, G., Odom, P., Shavlik, J., & Natarajan, S., Guiding Autonomous Agents to Better Behaviors through Human Advice, *IEEE International Conference on Data Mining (ICDM)*, 2013.

[2]Odom, P., Natarajan, S., Actively Interacting with Experts: A Probabilistic Logic Approach,*European Conference on Machine Learning and Principles of Knowledge Discovery in Databases*, 2016.

[3]R. Maclin, J. Shavlik, L. Torrey, T. Walker, and E. Wild. Giving Advice about Preferred Actions to Reinforcement Learners Via Knowledge-Based Kernel Regression.*Proceedings of the 20th National Conference on Artificial Intelligence*, 2005.

[4] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. *Advances in Neural Information Processing Systems*, 2013.