Generating Real-Time Crowd Advice to Improve Reinforcement Learning Agents

Gabriel V. de la Cruz Jr.
School of EECS
Washington State University
gabriel.delacruz@wsu.edu

Bei Peng
School of EECS
Washington State University
bei.peng@wsu.edu

Walter S. Lasecki
Computer Science Department
University of Rochester
wlasecki@cs.rochester.edu

Matthew E. Taylor
School of EECS
Washington State University
taylorm@eecs.wsu.edu

Abstract

Reinforcement learning is a powerful machine learning paradigm that allows agents to autonomously learn to maximize a scalar reward. However, it often suffers from poor initial performance and long learning times. This paper discusses how collecting on-line human feedback, both in real time and *post hoc*, can potentially improve the performance of such learning systems. We use the game Pac-Man to simulate a navigation setting and show that workers are able to accurately identify both when a sub-optimal action is executed, and what action should have been performed instead. Our results demonstrate that the crowd is capable of generating helpful input. We conclude with a discussion the types of errors that occur most commonly when engaging human workers for this task, and a discussion of how such data could be used to improve learning. Our work serves as a critical first step in designing systems that use real-time human feedback to improve the learning performance of automated systems on-the-fly.

Introduction

Reinforcement learning (Sutton and Barto 1998) is a very flexible, robust approach to solving problems such as backgammon (Tesauro 1995), helicopter control (Ng et al. 2004), and simulated robot soccer (Stone et al. 2006). However, early in the training process much of the problem space is unexplored, often resulting in poor performance because reasonable policies are only discovered after a considerable amount of trial-and-error. In this paper, we propose the idea of using on-demand human intelligence, available via crowdsourcing platforms such as Amazon Mechanical Turk, in order to provide immediate feedback to reinforcement learning systems based on the intuition and experience of the human observer.

To test whether crowd workers are able to accurately provide such advice, we perform a set of experiments that measure the crowd's ability to generate just-in-time warnings to an agent playing Pac-Man. First, we establish that the crowd can collectively identify the correct point at which an error occurs with over 91% accuracy. Second, we demonstrate that not only can this mistake identification be done in real time with a mean latency of just 0.39 seconds, but also that workers are able to identify what the optimal move *would* have been been. Third, we compare the crowd's performance in this real-time setting with an offline "review" setting where game playback can be controlled and replayed. In this setting, mistakes can be better estimated, with a mean distance from the correct position of just 0.15 seconds.

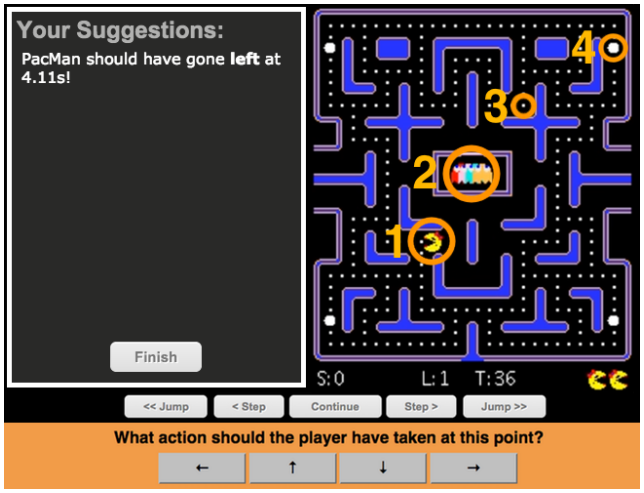


Figure 1: This screenshot shows the web interface of the user study with game layout, and components of the Pac-Man game: 1) Pac-Man, 2) 4 Ghosts, 3) Pills, and 4) Power Pills. Crowd workers watch a video of the game in progress and mark mistakes made by a reinforcement learning agent.

Included in this paper is a discussion on how we plan to leverage crowd advice in RL. We have identified three integration strategies that includes: 1) automatically accepting the crowd's advice, 2) using reward shaping to update the agent's policy, and 3) treating the crowd as human demonstrator. In addition, we also look into three strategies for collecting the crowd's advice: 1) worker provides suggestion while the RL agent is learning in real-time, 2) a cyclic system where an RL agent records a video clip of a portion in the game, uploads it to a crowdsourcing platform, retrieves

the crowd's advice and integrate it, and 3) RL agent decides when and what to ask the crowd that can be incorporated to the the first two strategies.

This work is the first research to establish the crowd's ability to react to mistakes made by an intelligent agent in real time, and provide accurate guidance on a preferred alternative action. Our work informs the design of future systems that use human intelligence to guide untrained systems through the learning process, without limiting systems to only learn from their mistakes far after they make them.

The contributions of this paper are to:

- present the idea that on-line crowds can provide assistance to learning agents in real-time, as the need arises.
- demonstrate that crowd workers can respond quickly and accurately enough to provide just-in-time feedback.
- show that workers can also improve their accuracy in *post hoc* review settings for use in future situations.
- discuss strategies on how we plan to integrate crowd advice to improve learning.

Background and Related Work

Reinforcement learning has a history of succeeding on difficult problems with little information. This paper leverages the on-policy learning algorithm Sarsa (Sutton and Barto 1998). A Sarsa agent learns to estimate Q-values, representing the estimated total reward the agent would receive in a state s, execute action a, and then follows the current policy until the episode is terminated. Over time, this type of temporal difference learning allows the agent to learn a near-optimal policy that collects as much reward as possible (in expectation).

While autonomous methods like Sarsa have many empirical successes and sound theoretical underpinnings, if a human is available to provide useful information, the agent is often able to learn much faster. For example, if a human understands the domain, she may be able to provide high-level advice to an agent (Maclin, Shavlik, and Kaelbling 1996). Another approach is to allow a user see trajectories and label individual actions as good or bad (Judah et al. 2010). A more active approach involves providing human rewards to an agent, which the agent could either interpret numerically (Thomaz and Breazeal 2006; Knox and Stone 2010) or categorically (Loftin et al. 2014). Finally, when a human can temporarily control the agent to perform the correct behavior, learning from demonstration techniques (Argall et al. 2009) can be used to mimic the human, or potentially improve upon the human's demonstrated behavior (Taylor, Suay, and Chernova 2011). In all of these examples, the approaches differ in 1) the level of expertise required of the human, 2) the type of interaction, and 3) how the advice is represented and integrated into the agent's control policy.

Among the most closely-related existing work to this paper is that investigating methods for crowdsourcing control and recognition tasks. Human control of robots has been explored in the context of a robot Ouija board and navigation setting (Goldberg et al. 2000). The Robot Management System (Toris, Kent, and Chernova 2014) (RMS) also uses on-line contributors to crowdsource human-robot interaction studies. RMS used groups of participants working from their home computers to practice controlling a robot using camera views and a web-based control interface.

Legion (Lasecki et al. 2011) explored using crowd workers to collaboratively control a robot in real time. This was the first work to show that on-demand human intelligence could be used to control a robot when an automatic system is unable to proceed. Legion:AR (Lasecki et al. 2013) showed that an active learning approach could be used in an activity recognition setting to call on crowd support for an action-labeling task only when needed. In both systems, low-latency responses were achieved by keeping the crowd continuously engaged with a task for a period of time. However, complete crowd control does not allow the system effectively evaluate its own policy. In this work, we explore if and how we can use real-time crowds in an advisory role, without needing the crowd to directly control the Pac-Man avatar, while still maintaining exceptional response speeds.

Experimental Design

Our Pac-Man agent (see Figure 1) used Sarsa to learn a near-optimal policy to win the game while earning as many points as possible using an existing open learning implementation (Taylor et al. 2014). Due to the large state space, the agent uses seven high-level features for function approximation to learn a continuous Q-value function.

To generate the videos used in the user study, we recorded Pac-Man being controlled by a human who intentionally made different types of mistakes. Then, we picked 10–14 seconds which contained one (and only one) mistake. Q-values for the agent's trajectory were also recorded, confirming that the human-created mistakes had lower Q-values than the "correct" action. We create four videos where each contained a mistake: Video 1) moving so that Pac-Man is trapped by one or more ghosts, Video 2) not moving towards an edible ghost after eating a power pill, Video 3) taking an empty path instead of going for pills when they are no risk, and Video 4) not going for all edible ghosts that are within close range.

To study the hypothesis that crowd workers can provide information useful to reinforcement learning agents, we consider four settings. First, a video of Pac-Man is played only once (*real-time*) or the worker can view it multiple times (*review*). Second, the worker may be asked to identify the time at which the mistake is made (*Mistake Identification*), or asked to identify both the mistake time as well as suggest the optimal action (*Action Suggestion*).

We want to measure the performance of users in identifying the point at which a mistake is made and suggesting optimal action Pac-Man should have executed. To evaluate worker actions, we can compare to recorded Q-values.

User Studies

Workers were first shown instructions describing the task, as well as the rules of Pac-Man. During a preliminary test of the web interface, we found that workers would sometimes identify mistakes *before* the sub-optimal action was

executed, anticipating the mistake. We provided explicit instructions to workers to encourage them to identify the exact time at which a mistake was made. Workers were then directed to a tutorial which asked them to complete an example task using the marking interface. After the tutorial, workers were given a new video and asked to identify the mistake. Finally, workers were able to provide general comments before submitting.

We recruited crowd workers from Amazon Mechanical Turk (AMT) for our experiments. While AMT provides immediately, programmatic access to crowds, it also poses a number of challenges, including that workers: 1) are unlikely to be experts, 2) may not take the task seriously and not read the instructions, and 3) may intentionally select incorrect times/actions. Our methods need to be robust to these challenges, unlike in Learning from Demonstration, where demonstrations are typically assumed to be optimal.

16 Human Intelligence Tasks (HITs) on AMT encompassed our four different types of experiments. Each experiment was tested with 4 different videos. We collected data from 30 unique workers per HIT and every worker was paid 25 cents. Trials are denoted as AMT 1–16 in Table 1.

Result Analysis

This section presents the results of our study in three parts. First, we establish that the crowd can identify the mistake with high accuracy. Second, we demonstrate that not only can Mistake Identification be done in real-time but that workers can also successfully identify what the optimal "correct" move would have been. Third, we compare the crowd's performance in the real-time setting with offline "review" setting and show that if additional time is available, even more accurate performance can be achieved.

Mistake Identification

Our performance measure is based on how many workers can correctly identify and suggest a time that is close to the correct mistake time. Histograms provide a visual representation of the accuracy of workers in different settings. The mistake times are reported as game move numbers, which are 986, 1809, 1116 and 334, for Videos 1–4, respectively. These video clips are 10 to 14 seconds long, corresponding to 250–350 total game moves, and the mistakes located roughly three quarters of the way through the clip. However, because Pac-Man moves continually, it is difficult for workers to identify the exact frame when the mistake was executed.

To quantify how accurate the workers were, we calculated the difference between the actual mistake time and the identified mistake time, where zero corresponds to a perfect answer. We selected a threshold of 30 actions, roughly 1 second, so that any answer within ± 1 second will be counted as correctly identifying the mistake. Figure 2(a) shows the distribution of workers' answers for AMT 1 and 2. Responses within the 956–1016 moves range are considered to be correct, showing only two errant responses.

To compute the mean difference between the time reported by a worker and the real error time μ_{diff}, we use:

$$\mu_{diff}(AMT_k) = \frac{\sum_{i=1}^{n} |t_{w_i} - t_m|}{n} \quad (1)$$

where k is the group number, n is the total number of workers per group that are within the threshold, t_{w_i} is the ith worker's suggested time, and t_m is the correct mistake time. The standard deviation is also computed using:

$$\sigma_{diff}(AMT_k) = \sqrt{\frac{\sum_{i=1}^{n} (\mu_{diff}(AMT_k) - |t_{w_i} - t_m|)^2}{n}} \quad (2)$$

where a low value indicates suggestions are tightly clustered.

To establish that workers can correctly identify where and when mistakes occur in our game, we count the number of people who correctly identified the mistake. Video 1's review setting collective percentage of correct events has the highest over all four videos with 98.3%. This is followed by Videos 4 and 2, with 88.0% and 86.6% respectively. And Video 3 has the lowest accuracy with 68.4%. This observed high percentage of correct events from the three videos suggests that the crowd can identify a mistake in many cases.

It is also important to point out that there are instances in which the mistakes are more subtle, making it harder to identify. Video 3 has the lowest accuracy, and the Mistake Identification experiment has the least percentage of correct answers at 56.7%. However, the sparsity of the data as shown in Figure 2(b) suggests that the mistake was harder to find.

In summary, these results established that, in most cases, workers can identify a mistake in the Pac-Man game with an overall accuracy for review Mistake Identification at 80% and an accuracy of 91% for review Action Suggestion.

Optimal Action Identification

Given that workers can correctly identify mistakes, we next consider whether they can also accurately provide the action that should have been taken. To do this, we first have to verify that majority of the workers within the threshold suggested the same action, and second, the suggested action has the maximum Q-value in the recorded video's game state.

Figure 2(c) shows that all workers' suggested actions that are within the 30-move threshold, in both the real-time and review cases, meaning that a majority of workers do suggest similar actions.

Knowing that the crowd reaches consensus on a single action, we can now compare the crowd's advice to the recorded Q-values of the game to verify if it is the correct (near-optimal) action. The maximum Q-value of the 4 possible Pac-Man actions determines what action Pac-Man should perform. In Video 1, a step before 986 moves should suggest the Q-values for the next move. At move 985, the Q-values are: *up* $= 1729$, *right* $= 1621$, *down* $= 1768$, and *left* $= 1621$. In the human-controlled game in Video 1, Pac-Man went right at this time when it should have gone down (the maximum Q-value). And as shown in Figure 2(c), workers did suggest for Pac-Man should move downward in Video 1. Similar in the other three videos demonstrate that workers can identify that a mistake has been made but as well as provide an advice that is useful and near-optimal.

Video #	HIT Group	Video Category	Experiment Type	Mean Difference μ_{diff}	Std Dev σ_{diff}	# of Suggestions <= 30 moves	% Correct Events	% Correct Suggestions
1	AMT 1	real-time review	Mistake Identification	11.250	5.992	28	96.6%	-
	AMT 2			2.733	3.483	30	100.0%	-
	AMT 3	real-time review	Action Suggestion	10.379	7.660	29	100.0%	93.1%
	AMT 4			1.679	2.970	28	96.6%	100.0%
2	AMT 5	real-time review	Mistake Identification	9.529	7.698	17	56.7%	-
	AMT 6			7.375	8.712	24	80.0%	-
	AMT 7	real-time review	Action Suggestion	8.588	6.236	17	56.7%	82.4%
	AMT 8			11.333	9.418	27	93.1%	55.6%
3	AMT 9	real-time review	Mistake Identification	8.750	5.365	8	26.7%	-
	AMT 10			2.882	4.820	17	56.7%	-
	AMT 11	real-time review	Action Suggestion	8.444	8.053	18	60.0%	77.8%
	AMT 12			4.833	6.281	24	80.0%	66.7%
4	AMT 13	real-time review	Mistake Identification	8.889	6.710	27	93.1%	-
	AMT 14			2.042	4.704	24	82.8%	-
	AMT 15	real-time review	Action Suggestion	6.882	6.499	17	56.7%	94.1%
	AMT 16			3.074	5.737	27	93.1%	92.6%

Table 1: Summary results for our 16 Amazon Mechanical Turk HITs. Here, we only consider suggestions within 30 moves (1.25 seconds) of the true mistake time as correct.

Real-time vs. Review

We expected the real-time setting to be considerably harder than review setting. This assumption can be verified by considering the mean difference for each setting — the average mean difference for real-time setting is 9.1 moves (≈ 0.36 seconds) while review case is of 4.5 moves (≈ 0.18 seconds), as shown in Table 1. The lower mean difference in review experiments shows that the crowd's performance in real-time setting with an offline "review" setting where game playback can be controlled and replayed, show that if additional time is available, even closer estimates of the point of the mistake can be gathered.

We performed a 4×2 Between Subjects Factorial ANOVA test of all Action Suggestion experiments shows that the difference of suggested mistake time by workers between subjects real-time and review setting was statistically significant ($F = 5.10, p < .05, \eta^2 = .023$). This difference between real-time and review setting in all Mistake Identification experiments is also significant ($F = 5.02, p < .05, \eta^2 = .022$). This indicates that the different mistakes in each video can also affect worker's ability to identify them.

Interestingly, Video 2 has two sets of workers that suggest different actions in Action Suggestion, and there is only a small difference between the mean difference of real-time and review setting in Mistake Identification for Video 2. This indicates that the mistake in Video 2 was harder for workers to identify than the other three videos.

It is notable here that the average of mean differences in the real-time setting for Mistake Identification results to 9.8 moves (≈ 0.39 seconds), and with Action Suggestion at 8.8 moves (≈ 0.35 seconds), which are both very close to the human response for tasks with no high-level reasoning needed (e.g., clicking a button in response to a visual stimulus). This suggests that crowd advice for tasks, such as navigation, can be collected nearly as fast as people can physically respond. This quickly-available input can, in turn, be used to improve real-time learning of virtual and physical agents.

Paradoxically, workers were able to give more accurate answers when asked to suggest a move compared to when they are only asked to identify when a mistake is made. One possible explanation for this is that setting up the problem so that workers are expecting to act as if they are making a move in the game itself is a better means of priming workers to react quickly. As a result, no slowdown of response is seen, even though they are providing more information.

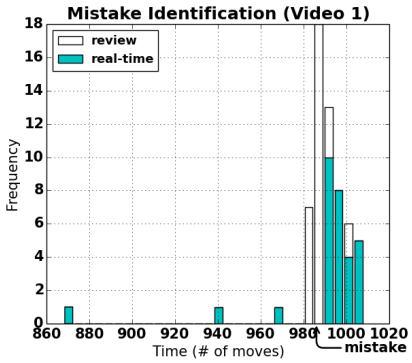
Discussion and Future Work

Future work will focus on developing learning algorithms that to leverage the unique strengths of human input on-the-fly without being detrimentally affected by incorrect advice. Although others (Taylor, Suay, and Chernova 2011) have incorporated advice from multiple demonstrators in past work, errors from crowdsourced workers are a unique challenges and opportunities to scale these systems. Furthermore, we plan to continue to improve our interfaces to further reduce the latency of worker responses. One potential method to do this is to leverage workers' ability to predict when mistakes might be made to collectively decrease latency below the best after-the-fact response speed possible. We are also interested in studying how the number of examples during the tutorial affects participants' accuracy. Finally, we are interested in eliciting a confidence measure from workers, potentially allowing us to weight different pieces of advice.

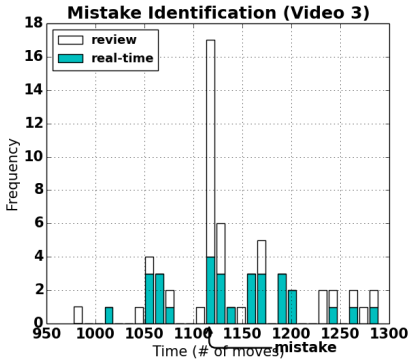
Leveraging Crowd Advice in RL

Our study has shown that the crowd can identify that a mistake that has been made as well as suggest a near-optimal action. This section focuses on the next stage of our study, which is leveraging the crowd's advice into the learning process of an RL agent. We will discuss three strategies for integrating the crowd's advice into the learning process.

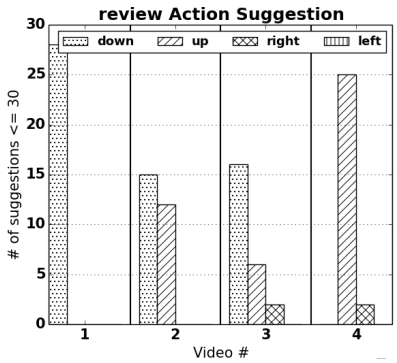
One strategy that provides an easy integration of the crowd's advice is to automatically take the crowd's suggested action. This integration strategy can be implemented in both review and real-time setting. However, in a real-time



(a) A histogram of the distribution of workers' suggestions in AMT 1 and 2 using Video 1 (mistake time: 986).



(b) A histogram of the distribution of workers' suggestions in AMT 9 and 10 using Video 3 (mistake time: 1116).



(c) Number of times each action was suggested by workers (Optimal: 1–down, 2–down, 3–down, 4–up).

Figure 2: Selected exemplar results from our 16 Amazon Mechanical Turk experiments.

system, we will have to consider the possibility of a time lag based on two factors: 1) number of workers and 2) diversity of suggestions. It implies that a mistake is present when a high number of workers provide suggestions, but it can be computationally expensive to filter the data if all suggestions are considered. Also, a diverse data requires more comparison to make a decision for integration. Although important, these added steps can delay integration in a real-time system. If a time lag does exist, we can still implement a real-time system by having the workers provide their suggestion a few moves before the moment of interest. We can also consider a *quasi* real-time system where we can pause the game at the moment of interest to collect the workers' suggestion. Forcing the Pac-Man to execute a near-optimal action using the crowd's advice brings the RL agent closer to the optimal policy because the agent is forced to explore in a direction consistent with a high-performing policy.

Our study has shown that there are mistakes that are harder to identify — there is a need to quantify the confidence of the crowd's advice. We can approach this by allowing the crowd to provide a confidence rate on their suggestion. Such confidence can be use to simply determine whether to integrate it or not, but it can also be use to weigh its effect during integration like in the case on our second strategy where we consider changing the Q-values. Another approach is to determine a confidence rate for each worker. As we collect more data from workers, we can also determine the worker's confidence rating. The confidence rating can represent if a worker normally agrees with others, and if their suggestions are good.

The second integration strategy is using reward shaping, a method used in reinforcement learning that consists of supplying additional training rewards to a learning agent to guide its learning process (Ng, Harada, and Russell 1999). The framework of *Markov decision processes* (MDPs) is widely adopted to study sequential reinforcement learning. In a traditional reinforcement learning algorithm, we seek to learn an optimal policy for some MDP $M = (S, A, T, \gamma, R)$. With giving additional "shaping" rewards, we aim to learn the optimal policy faster for *transformed* MDP $M = (S, A, T, \gamma, R^{'})$, where $R^{'} = R + F$ is the new reward function, and F is the *shaping reward function*. If the former MDP M received reward $R(s, a, s^{'})$ based on transitioning to $s^{'}$ with taking an action a in state s, the *transformed* MDP would receive reward $R(s, a, s^{'}) + F(s, a, s^{'})$ on the same event.

In our Pac-Man domain, in Mistake Identification experiments, we can set the shaping reward function $F(s, a_p, s^{'}) = n$ when the crowd gives correct advice (Pac-Man is in state s and takes a wrong action a_p), where n is a negative value, and $F(s, a_p, s^{'}) = 0$ otherwise. In Action Suggestion setting, except shaping reward function for Pac-Man's wrong action, we should also set $F(s, a_c, s^{'}) = p$ based on crowd's correct action suggestion (Pac-Man is in state s and the crowd suggests the optimal action a_c), where p is a positive value. Considering our current learning algorithm, each action taken by Pac-Man in the game will receive a reward associated with the score. Then, the additional training reward will directly change the reward function and indirectly change the Q-value of taking that action, making the Q-value of the wrong demonstrated action smaller and the suggested action larger.

The third integration strategy we have considered is Learning from Demonstration (LfD), which uses the *examples* learned from demonstration teacher to derive a policy that executes demonstrated behaviors (Argall et al. 2009), in contrast to learning from *experience* as in Reinforcement Learning (Sutton and Barto 1998). The LfD learning problem can be divided into two steps: *gathering* the examples, and *deriving* a policy from gathered examples. In our Pac-Man domain, crowd workers can be treated as human demonstrators, and identified mistake time and/or suggested actions are examples learned from them. Considering the Action Suggestion setting, if the worker provides an action suggestion to Pac-Man, the current state of Pac-Man game and suggested action will be recorded as demonstrated data. The demonstration technique used here can be considered as *interactive approach*, which allows the policy to be updated incrementally as demonstration data become

available. Demonstrations here can prevent Pac-Man from suffering long periods of blind exploration when no reward feedback is given.

The limitations of Learning from Demonstration is that it is heavily limited by the quality of demonstrated information, the sequence of state-action pairs executed by the demonstration teacher. It suffers from the state spaces that have not been demonstrated and the limited expertise of demonstrators who are not able to perform optimal policy. Therefore, integrating crowd advice into reinforcement learning algorithm also suffers from these limitations.

Strategies for Collecting the Crowd's Advice

Having already pointed out different strategies for integrating the crowd advice, we want to build effective strategies for collecting and harnessing the crowd's advice. We have identified three strategies that we will discussed further below.

Our first strategy is to collect the crowd's advice through a real-time system. We will use an interface similar to that used in this study where workers can provide an advice by simply pressing a button to suggest a mistake, and/or additional buttons for suggesting the correct action. Multiple workers will watch the same RL agent as it learns how to navigate through its environment. This can be deployed on the web or it can also be done locally through several computers. The integration will depend whether there are enough workers to provide suggestions, and whether a consensus is reached. It also depends on what type of advice is given by the crowd, whether a worker is suggesting an action or just simply identifying a mistake. In any case, the crowd's advice can be integrated on-the-fly by having the RL agent take the suggested action via the strategies suggestion in the previous subsection.

The second strategy will work more towards a review setting. Instead of manually recording videos of a human-controlled game and uploading it to Mechanical Turk to collect inputs, we will collect worker inputs using a cyclic system that involves having the RL agent itself record video clips of the game during the learning process, and automatically uploads them to our current crowd worker interface that collects advice. This strategy would be most useful in the early-stages of the learning process and we assume that the RL agent has not converged to the optimal policy — video clips can be recorded with a uniform random distribution and there will be some videos that have no mistakes. Besides using the Mechanical Turk to collect data, we will also look into using an on-site set of workers for faster data collection.

Our third strategy can be an added functionality to the first two strategies. The RL agent decides when and what to ask the crowd. The cue for the agent to decide when to ask the crowd can be 1) when the agent has no sufficient prior experience of its current state, and 2) when the agent does not have strong confidence on what action to take if the differences of the Q-values are insignificant. For the first cue, we can use reasoning like in R-MAX (Brafman and Tennenholtz 2003) to decide if a state has been visited "often enough" to assume some amount of certainty. Another option is to use confidence intervals for different action (e.g., using UCT (Kocsis and Szepesvri 2006)) to determine when to ask for help. Since theoretically there can be an implicit cost associated with asking for crowd's advice that by using the key idea of *active learning* (Settles 2010), an RL agent should ask for fewer number of crowd advice in an episode but should still yield a greater reward or a better policy.

Conclusion

Reinforcement learning algorithms often suffer from poor early-stage performance since agents have to experience considerable amount of trial-and-error before learning an effective policy. Our approach uses real-time crowds to provide immediate assistance to the learning agent to help improve its performance. We ran a set of user studies to show that crowd workers from Amazon Mechanical Turk can respond quickly and accurately enough to provide just-in-time feedback to an agent playing Pac-Man. We show that workers can correctly identify the point at which a mistake is made by Pac-Man and the optimal action Pac-Man should have executed. We also showed that higher performance could be achieved by workers in *post hoc* review settings.

Our results demonstrated that 1) crowd workers are able to accurately choose the mistake time in real-time with a mean latency of just 0.39s, and 2) latency does not increase if workers must also suggest an action. By leveraging the crowd, we present an effective, scalable means of providing during-task assistance to learning agents. Next steps include leveraging crowd advice to improve an RL agent's learning performance.

Acknowledgements

This work was supported in part by NSF IIS-1319412.

References

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robot. Auton. Syst.* 57(5):469–483.

Brafman, R. I., and Tennenholtz, M. 2003. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research* 3:213–231.

Goldberg, K.; Chen, B.; Solomon, R.; Bui, S.; Farzin, B.; Heitler, J.; Poon, D.; and Smith, G. 2000. Collaborative teleoperation via the internet. In *In IEEE International Conference on Robotics and Automation (ICRA*, 2019–2024.

Judah, K.; Roy, S.; Fern, A.; and Dietterich, T. G. 2010. Reinforcement learning via practice and critique advice. In *AAAI*.

Knox, W. B., and Stone, P. 2010. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '10, 5–12. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Kocsis, L., and Szepesvri, C. 2006. Bandit based monte-carlo planning. In *In: ECML-06. Number 4212 in LNCS*, 282–293. Springer.

Lasecki, W. S.; Murray, K. I.; White, S.; Miller, R. C.; and Bigham, J. P. 2011. Real-time crowd control of existing interfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, 23–32. New York, NY, USA: ACM.

Lasecki, W. S.; Song, Y. C.; Kautz, H.; and Bigham, J. P. 2013. Real-time crowd labeling for deployable activity recognition. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, 1203–1212. New York, NY, USA: ACM.

Loftin, R.; Peng, B.; MacGlashan, J.; Littman, M. L.; Taylor, M. E.; Huang, J.; and Roberts, D. L. 2014. A strategy-aware technique for learning behaviors from discrete human feedback. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*.

Maclin, R.; Shavlik, J. W.; and Kaelbling, P. 1996. Creating advice-taking reinforcement learners. In *Machine Learning*, 251–281.

Ng, A. Y.; Kim, H. J.; Jordan, M. I.; and Sastry, S. 2004. Inverted autonomous helicopter flight via reinforcement learning. In *In International Symposium on Experimental Robotics*. MIT Press.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 278–287.

Settles, B. 2010. Active learning literature survey. *University of Wisconsin, Madison* 52:55–66.

Stone, P.; Kuhlmann, G.; Taylor, M. E.; and Liu, Y. 2006. Keepaway soccer: From machine learning testbed to benchmark. In Noda, I.; Jacoff, A.; Bredenfeld, A.; and Takahashi, Y., eds., *RoboCup-2005: Robot Soccer World Cup IX*, volume 4020. Berlin: Springer-Verlag. 93–105.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*, volume 28. MIT press.

Taylor, M. E.; Carboni, N.; Fachantidis, A.; Vlahavas, I.; and Torrey, L. 2014. Reinforcement learning agents providing advice in complex video games. *Connection Science* 26(1):45–63.

Taylor, M. E.; Suay, H. B.; and Chernova, S. 2011. Integrating reinforcement learning with human demonstrations of varying ability. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Tesauro, G. 1995. Temporal difference learning and TD-Gammon. *Commun. ACM* 38(3):58–68.

Thomaz, A. L., and Breazeal, C. 2006. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, 1000. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Toris, R.; Kent, D.; and Chernova, S. 2014. The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing. *Journal of Human-Robot Interaction* 3(2):25–49.