Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Improving Reinforcement Learning with Human Input

Matthew E. Taylor
Borealis AI, Edmonton, AB
matthew.taylor@borealisai.com

Abstract

Reinforcement learning (RL) has had many successes when learning autonomously. This paper and accompanying talk consider how to make use of a non-technical human participant, when available. In particular, we consider the case where a human could 1) provide demonstrations of good behavior, 2) provide online evaluative feedback, or 3) define a curriculum of tasks for the agent to learn on. In all cases, our work has shown such information can be effectively leveraged. After giving a high-level overview of this work, we will highlight a set of open questions and suggest where future work could be usefully focused.

1 Introduction

Reinforcement learning [Sutton and Barto, 1998] (RL) has had many successes in both virtual and physical settings. Unfortunately, in many cases, significant amounts of data and/or computation is required to reach reasonable performance. If working in a simulator, such initial poor performance may be acceptable. When working in the real world, however, poor performance has a real cost. More troubling, there may be cases where a reward function is not well defined. For instance, if a person purchases a robot and wishes it to learn a new task in her house, few consumers will be able to fully define a reward function that the robot can learn to optimize.

This line of work, therefore, asks the following question:

If there is a non-technical, sub-optimal, human who can help an agent learn, how can an agent best leverage their knowledge to learn (near-) optimal behavior?

The following paper discusses three instances where a non-technical human can provide information to a learning agent via demonstration, feedback, or curriculum design.

2 Background: Reinforcement Learning

RL is a very general framework for learning to maximize a real-valued reward, $R(S, A) \mapsto \Re$. Unlike supervised learning, there are no positive or negative examples. Instead, an agent must learn to take an action, $a \in A$, in each state, $s \in S$. This is nontrivial, particularly because the transition function that describes how the agent's actions affect the state, $T(S, A) \mapsto S$, is typically unknown. A mapping from states to actions is called a policy, $\pi(S) \mapsto A$. One common way to learn a policy is to first learn an action-value function, $Q(S, A) \mapsto \Re$, which estimates the long-term discounted reward for a given state-action pair. Given an optimal action-value function, the optimal policy would be realized by always executing the action with the highest value in the current state.

3 Leveraging Human Demonstrations

One way of learning faster is to re-frame the problem to that of mimicking a demonstrator. In this case, the agent typically solves a supervised learning problem so that it can perform similarly to a demonstrator [Argall *et al.*, 2009]. LfD methods typically assume that the demonstrator is near-optimal—even if there is an environmental reward signal, LfD methods typically do not use it to improve the policy.

3.1 Human-Agent Transfer

Human-agent transfer [Taylor *et al.*, 2011] (HAT) reimagines demonstration-based learning from the perspective of transfer learning [Taylor and Stone, 2009], where knowledge gained from a prior task is used as a bias to bootstrap RL in a related but different task. HAT views the demonstration itself as initial bias (coming from some *source* agent) that informs the RL's action choices, but gradually weans RL off this bias as the RL's own value function improves over time. In particular, HAT follows the following steps (based on the earlier idea of Rule Transfer [Taylor and Stone, 2007]):

1. A human or trained agent demonstrates a learned policy in the task. Record $\langle s, a \rangle$ samples.
2. Use a supervised learning method to summarize this data. Mapping states to actions represents a policy that is the best guess the agent has at what the human or trained agent would do.
3. Use the learned classifier to bootstrap learning in the new task. For instance, our previous work [Taylor and Stone, 2007] provided three ways of leveraging this knowledge. One such method is Probabilistic Policy Reuse. The standard ϵ-greedy action selection is changed so that with

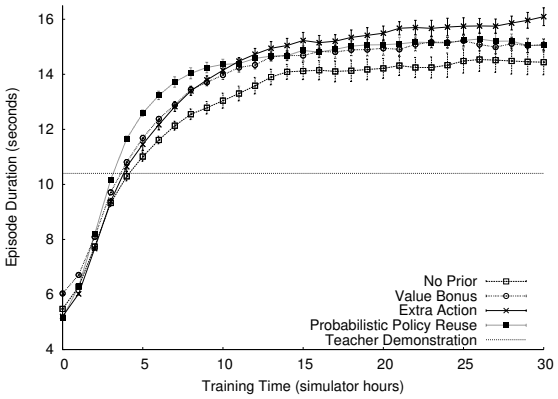

Figure 1: This graph summarizes the performance of Sarsa learning in Keepaway using four different algorithms. One demonstration of 20 episodes (roughly 3 minutes of wall-clock time) was used for all three HAT learners. Error bars show the standard error.

probability ϵ the agent takes a random action, with probability Φ the agent takes the action suggested by the classifier, and with probability $1-\epsilon-\Phi$ the agent exploits its action-value function. Φ begins near 1 and is gradually decayed over time.

Figure 1 shows the results of using HAT in the 3 vs. 2 Keepaway [Stone *et al.*, 2006] domain. HAT can sometimes improve the jumpstart (i.e., the initial performance, relative to learning without a prior) and consistently improves both the performance at the end of 30 simulated hours of training and the total reward (i.e., the area under the curve). Another performance metric is the time-to-threshold. For instance, consider the number of simulated hours needed to reach a performance of 14 sections — learning from scratch requires roughly 14 hours, while learning with probabilistic policy reuse requires roughly half that time. Saving 7 hours of training time by using only 3 minutes of human time is often a favorable tradeoff.[1]

3.2 Confidence-Based HAT

Even though the demonstration may cover a small part of the state space, the classifier produced will be able to suggest actions for any state. In some cases, the suggested action may be very different from what the original demonstrator would have provided. Confidence-based HAT [Wang and Taylor, 2017] (CHAT) builds upon HAT by modifying steps 2 and 3. In particular, when learning a classifier, it also learns a confidence measure that can be used. In step 3, the action suggested by the classifier is only executed if the confidence in the predicted action is above some threshold confidence. Decision trees, neural networks, and Gaussian processes have all been shown to work well with CHAT — a confidence threshold parameter can significantly improve the performance of an agent, relative to learning from scratch or

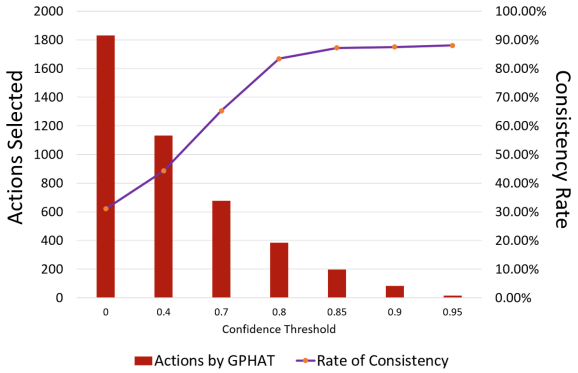

Figure 2: This figure shows how different amounts of advice will be used with different CHAT confidence thresholds.

learning with HAT, in both 3 vs. 2 Keepaway and in the game of Mario [Karakovskiy and Togelius, 2012].

For an example of how the confidence threshold affects performance, we first let a trained agent play Mario using its fixed policy to generate 20 demonstration episodes. Second, we train a Gaussian process. Third, we compare the actions suggested by the classifier with the actual actions made by the fixed-policy agent to see how often they are the same. Figure 2 shows how the CHAT agent acts with different confidence thresholds. For each confidence threshold, we show the number of actions made by CHAT and the rate of consistency with respect to the fixed-policy agent. When the confidence threshold is very high, the actions are consistent, but very few actions will be selected (and performance will be similar to that of learning from scratch because little advice is used). When the confidence threshold is very low, actions made by GPHAT are less likely to be the same as the source task agent's actions (and the performance will be similar to that of using HAT, which has no confidence threshold).

3.3 Shaping with Inverse Reinforcement Learning

Rather than using collected demonstrations to learn a classifier and then change the action-selection method, consider instead how the actual reward could be changed. In particular, Inverse Reinforcement Learning (IRL) is the problem of learning a reward function using a set of observations from expert demonstrations. The method introduced in [Suay *et al.*, 2016] uses IRL on the demonstrations to learn a set of weights[2], w over a set of reward features f: $R_{IRL}(s) = \sum_{i=1}^{n} w_i f_i(s)$. $R_{IRL}(s)$ is now a reward function that can be used as a shaping reward, such that the new reward R' is provided to the agent as $R(s,a,s') = R(s,a,s') + F(s,a,s') = R(s,a,s') + \gamma R_{IRL}(s') - R_{IRL}(s)$. Because R_{IRL} is used as a potential-based shaping reward, the optimal policy is guaranteed to be unchanged from using only R [Ng *et al.*, 1999].

$R_{IRL}(s)$ works well for shaping over states. But in some cases, shaping over state-action pairs may provide more improvement. In order to keep using a potential-based reward,

[1] Although out of scope for the current paper, HAT [Taylor *et al.*, 2011] also showed that combining demonstrations by different demonstrators of different qualities also improved performance.

[2] Although there are different solutions for the IRL problem, many of them require the transition function to be known. We focus on RE-IRL [Boularias *et al.*, 2011], a model-free IRL method.

we use the "trick" from Harutyunyan et al. [2015] that uses an additional action-value function to turn an arbitrary reward function into a potential-based shaping reward function. We found that using shaping over states and state-action pairs both worked to improve learning in a simple maze domain and in Mario.

4 Leveraging Human Feedback

In this section of the paper, we consider a complementary scenario. First, we now do not assume there is an environmental reward. Second, rather than teleoperating an agent (i.e., providing demonstrations), the human *trainer* will provide positive and negative feedback on the policy of the agent.

4.1 SABL

On each time step, the agent performs an action. The trainer then evaluates the agent's action, comparing it to the desired policy. Then, the trainer can provide positive or negative feedback to the agent, or provide no feedback at all. Note that the final case becomes more likely as the agent moves faster or the human becomes tired with giving feedback.

The SABL algorithm [Loftin *et al.*, 2015] is one such method for learning from a human trainer. The idea of learning from human feedback is not novel [Thomaz and Breazeal, 2006; Knox and Stone, 2009]. However, the primary difference between such existing methods and our method is that our algorithm allows for different *training strategies* by treating human feedback as categorical, rather than numeric, potentially allowing us to learn in the no feedback condition via *implicit communication*. The hypothesis that treating a feedback signal as numeric (e.g., +1 for positive feedback, 0 for no feedback, and -1 for negative feedback) loses information is borne out in improved results.

First, we posited that different trainers could have different feedback strategies. To test this hypothesis, we ran over 200 human-subjects studies via Amazon's Mechanical Turk. These Turkers trained an agent using SABL in a contextual bandit setting. Of the 227 participants, we found that the majority (125) used a "reward-focused strategy." When the agent performed the correct action, the participant most often gave positive feedback, but when the agent performed the incorrect action, the participant most often provided no feedback. We also found that 93 of the participants used a "balanced strategy" where they gave positive feedback when the agent performed the correct action and gave negative feedback when the agent performed the incorrect action.[3] This study demonstrates that neutral or no feedback can have meaning.

In order to leverage this insight, we frame the trainer as having three important parameters. First, the trainer makes a mistake with probability ϵ. Second, if the agent's action is consistent with the trainer's desired policy, the trainer will give explicit reward with probability $1-\mu^+$. Third, if the agent's action is inconsistent with the trainer's desired policy, the trainer will explicitly punish with probability $1-\mu^-$.

The μ parameters represent the trainer's training strategy — for example, $\mu^+ = 0.1$, $\mu^- = 0.1$ correspond to a balanced feedback strategy where nearly every action receives explicit feedback. Combining these elements, for time step t (each time step corresponds to an episode with the agent observing the world, choosing an action and receiving feedback), we have a distribution over the feedback f_t conditioned on the observation o_t, action a_t, and the trainer's target policy λ^*,

$$p(f_t = f^+ | o_t, a_t, \lambda^*) = \begin{cases} (1-\epsilon)(1-\mu^+), & \lambda^*(o_t) = a_t \\ \epsilon(1-\mu^+), & \lambda^*(o_t) \neq a_t, \end{cases} \tag{1}$$

$$p(f_t = f^- | o_t, a_t, \lambda^*) = \begin{cases} \epsilon(1-\mu^-), & \lambda^*(o_t) = a_t \\ (1-\epsilon)(1-\mu^-), & \lambda^*(o_t) \neq a_t, \end{cases} \tag{2}$$

$$p(f_t = f^0 | o_t, a_t, \lambda^*) = \begin{cases} (1-\epsilon)\mu^+ + \epsilon\mu^-, & \lambda^*(o_t) = a_t \\ \epsilon\mu^+ + (1-\epsilon)\mu^-, & \lambda^*(o_t) \neq a_t. \end{cases} \tag{3}$$

where f^+ is an explicit positive feedback, f^- is an explicit negative feedback, and f^0 represents a lack of feedback. These equations are then used in the Strategy-Aware Bayesian Learning (SABL) algorithm (Algorithm 1).

Algorithm 1 The SABL algorithm. The feedback distribution $p(f_t | o_t, a_t, \lambda^*(o_t) = a')$ is described by Equations 1, 2 and 3. $takeAction(a_t)$ does not return until the episode finishes.

$\forall o \in O, a \in A$: $P[o,a] \leftarrow \frac{1}{|A|}$
$t \leftarrow 0$
while user has not terminated learning **do**
  $o_t \leftarrow observeWorld()$
  $a_t \leftarrow \text{argmax}_{a' \in A} P[o_t, a']$
  $takeAction(a_t)$
  $f_t \leftarrow receiveFeedback()$
  for all $a' \in A$ **do**
    $P[o_t, a'] \leftarrow p(f_t | o_t, a_t, \lambda^*(o_t) = a') P[o_t, a']$
  $P[o_t, \cdots] \leftarrow normalize(P[o_t, \cdots])$
  $t \leftarrow t + 1$

Results in our contextual bandit domain showed that treating rewards as categorical worked better than treating them as numeric. A further extension in this work [Loftin *et al.*, 2015] is to introduce Inferring-SABL, or I-SABL, that uses a Bayesian approach to learn a trainer's μ^+ and μ^- parameters. Once we learn the correct action for some observations, we can use I-SABL to infer the trainer's strategy in real time and therefore improve the agent's policy even when no explicit feedback is provided.

4.2 Implicit Communication and Sequential Tasks

While the previous section focused on applying SABL and I-SABL to a contextual bandit, other work [Peng *et al.*, 2016] considered a sequential decision task. In particular, we assume that the agent has the transition function (e.g., a map of the environment) and access to a planner. Thus, the problem is reduced to discovering which possible goal state the trainer desires. For example, Figure 3 shows one example Sokeban-like environment with its paired English command.

[3] The remaining participants were either "punishment focused" (6) or "inactive" (3).

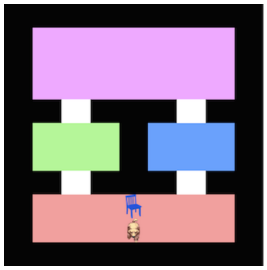

Figure 3: One of the test environments with the command "Move the blue chair to the purple room."

To enable language learning from agents trained with reward and punishment, we use a probabilistic model [MacGlashan *et al.*, 2014] that connects the IBM Model 2 (IBM2) language model [Brown *et al.*, 1990] with a factored generative model of tasks, and the goal-directed SABL algorithm [Loftin *et al.*, 2015] for learning from human feedback.

In addition to showing that sequential decision tasks can be learned via positive and negative feedback from human-subjects studies, the Learning Agents Modeling Belief-Dependent Action Speeds (LAMBDAS) approach also considers how a learner can adapt its action execution speed to learn more efficiently from human trainers. One problem researchers have observed [Li *et al.*, 2016] is that trainers often reduce the amount of explicit feedback over time. LAMBDAS is designed to implicitly motivate trainers to provide use explicit feedback in states where it would be most useful. We demonstrate that the agent's action execution speed can be successfully modulated to strategically/implicitly encourage more explicit feedback from a human trainer in parts of the state space where the learner has more uncertainty about how to act. Our results show that an adaptive speed agent dominates two fixed-speed agents according to multiple metrics, while making better use of limited human feedback.

4.3 Leveraging Human Curricula

Humans acquire knowledge efficiently by starting from simple concepts, and then gradually generalizing to more complex ones using previously learned information. Recent work [Taylor *et al.*, 2007; Bengio *et al.*, 2009; Kumar *et al.*, 2010; Lee and Grauman, 2011] has shown that machine-learning algorithms can benefit from a similar training strategy, called *curriculum learning*. Rather than considering all training examples at once, the training data can be introduced in a meaningful order such that the learner can build up a complex model — the agent can learn faster on more difficult examples after it has mastered simpler examples. In most existing work, the curriculum is generated either automatically [Kumar *et al.*, 2010; Lee and Grauman, 2011; Narvekar *et al.*, 2016; Svetlik *et al.*, 2016], by iteratively selecting examples with increasing difficulty tailored to the current ability of the learner, or manually by the algorithm designer, who typically has specialized knowledge of the problem domain or algorithm [Stanley *et al.*, 2005; Taylor *et al.*, 2007; Karpathy and Van De Panne, 2012]. How *non-expert humans* design curricula is a relatively neglected topic.

This work instead presents users with a set of tasks (see Figure 4) that the user can pick from, along with accompanying English commands. After generating a curriculum, the participant watches an automated trainer teach the agent to accomplish each task in the curriculum, along with a final target task (not directly represented in the 16 tasks).

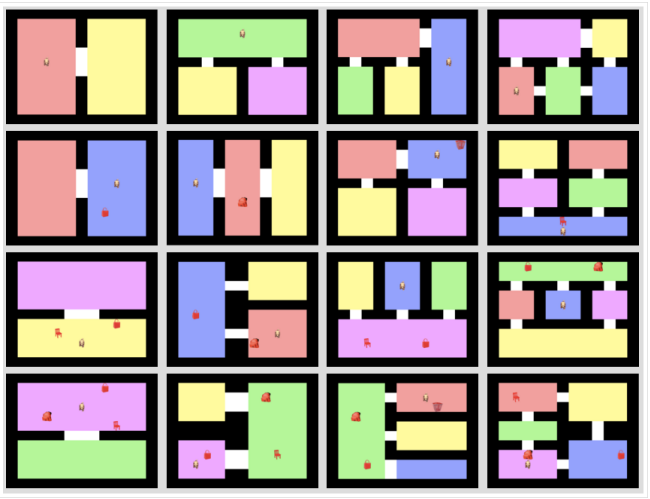

Figure 4: The library of 16 environments is organized by the number of rooms and objects.

After collecting 160 human-generated curricula, we were pleased to see that the curricula all improved learning on the target task. Much to our dismay, however, we found that these human-generated curricula did not always outperform randomly generated curricula. While this result confirms the power of curricula when tasks are chosen to be simpler than the final, target task, we had expected human intuition to outperform random selection!

After further analysis of the results, we found that participants were biased towards gradually introducing complexity to the curriculum, as expected. We also found that participants tended to select concepts (i.e., room color or object type) that were related to the final target task. We therefore updated our curriculum learning algorithm to be biased in learning the target task towards concepts that were most frequently seen in the curriculum. With this small bias, we then found that the human-designed curricula not only helped the trainer teach the target concept more quickly but that the human-designed curricula dominated random curricula.

5 Open Questions and Conclusion

While the positive results we have presented are encouraging, there are many remaining open questions. For instance, rather than only providing demonstrations before learning, it would be interesting to allow a participant to remain to give demonstrations on-demand, either because the demonstrator noticed sub-optimal behavior, or because the agent proactively asks the demonstrator for help.

It is not clear when, or why, one learning from demonstration method out performs another. What is it about different domains that causes one method to be superior to another?

Rather than limiting the human participant to a single type of advice, what possibilities exist for multi-modal advice?

What about other methods of advice, could be incorporated into these paradigms (e.g., natural language advice, designating goal or sub-goal states, providing the agent a similarity function [Rosenfeld *et al.*, 2017], etc.)?

Our results on implicit communication suggest that if a participant is better able to understand an agent, she is better able to help that agent. What other techniques can be used to better help non-technical participants understand the internal workings of agents? Does this indeed allow for better advice, and does this help the agent more than if the participant treats the agent as a complete black box?

We hope the reader will agree that there are many interesting unanswered questions in this area and that additional effort by our community in this area is justified.

References

[Argall *et al.*, 2009] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57(5):469–483, May 2009.

[Bengio *et al.*, 2009] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pages 41–48, 2009.

[Boularias *et al.*, 2011] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *AI Stats*, pages 182–189, 2011.

[Brown *et al.*, 1990] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85, 1990.

[Harutyunyan *et al.*, 2015] Anna Harutyunyan, Sam Devlin, Peter Vrancx, and Ann Nowe. Expressing arbitrary reward functions as potential-based advice. In *AAAI*, pages 2652–2658, 2015.

[Karakovskiy and Togelius, 2012] S. Karakovskiy and J. Togelius. The Mario AI benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):55–67, March 2012.

[Karpathy and Van De Panne, 2012] Andrej Karpathy and Michiel Van De Panne. Curriculum learning for motor skills. *Advances in AI*, pages 1015–1023, 2012.

[Knox and Stone, 2009] W. Bradley Knox and Peter Stone. Interactively Shaping Agents via Human Reinforcement: The TAMER Framework. In *KCAP*, pages 9–16, 2009.

[Kumar *et al.*, 2010] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Proc. of NIPS*, pages 1189–1197, 2010.

[Lee and Grauman, 2011] Yong Jae Lee and Kristen Grauman. Learning the easy things first: Self-paced visual category discovery. In *CVPR*, pages 1721–1728, 2011.

[Li *et al.*, 2016] Guangliang Li, Shimon Whiteson, W. Bradley Knox, and Hayley Hung. Using informative behavior to increase engagement while learning from human reward. *Autonomous Agents and Multi-Agent Systems*, 30(5):826–848, 2016.

[Loftin *et al.*, 2015] Robert Loftin, Bei Peng, James MacGlashan, Michael L Littman, Matthew E Taylor, Jeff Huang, and David L Roberts. Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous Agents and Multi-Agent Systems*, 30(1):30–59, 2015.

[MacGlashan *et al.*, 2014] J. MacGlashan, M. L. Littman, R. Loftin, B. Peng, D. L. Roberts, and M. E. Taylor. Training an agent to ground commands with reward and punishment. In *Proceedings of the AAAI Machine Learning for Interactive Systems Workshop*, pages 6–12, 2014.

[Narvekar *et al.*, 2016] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. Source task creation for curriculum learning. In *AAMAS*, pages 566-574, 2016.

[Ng *et al.*, 1999] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, pages 278-287, 1999.

[Peng *et al.*, 2016] Bei Peng, James MacGlashan, Robert Loftin, Michael L. Littman, David L. Roberts, and Matthew E. Taylor. A Need for Speed: Adapting Agent Action Speed to Improve Task Learning from Non-Expert Humans. In *Proc. of AAMAS*, pages 957–965, 2016.

[Rosenfeld *et al.*, 2017] Ariel Rosenfeld, Matthew E. Taylor, and Sarit Kraus. Leveraging Human Knowledge in Tabular Reinforcement Learning: A Study of Human Subjects. In *Proc. of IJCAI*, pages 3823–3830, 2017.

[Stanley *et al.*, 2005] Kenneth O. Stanley, Bobby D. Bryant, and Risto Miikkulainen. Evolving neural network agents in the NERO video game. In *CIG*, pages 182–189, 2005.

[Stone *et al.*, 2006] Peter Stone, Gregory Kuhlmann, Matthew E. Taylor, and Yaxin Liu. Keepaway Soccer: From Machine Learning Testbed to Benchmark. In *RoboCup-2005: Robot Soccer World Cup IX*, Pages 93–105, 2006.

[Suay *et al.*, 2016] Halit Bener Suay, Tim Brys, Matthew E. Taylor, and Sonia Chernova. Learning from Demonstration for Shaping through Inverse Reinforcement Learning. In *Proc. of AAMAS*, pages 429–437, 2016.

[Sutton and Barto, 1998] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT Press, 1998.

[Svetlik *et al.*, 2016] M Svetlik, M Leonetti, J Sinapov, R Shah, N Walker, and P Stone. Automatic curriculum graph generation for reinforcement learning agents. In *Proc. of AAAI*, pages 2590–2596, 2016.

[Taylor and Stone, 2007] Matthew E. Taylor and Peter Stone. Cross-Domain Transfer for Reinforcement Learning. In *Proc. of ICML*, pages 879-886, 2007.

[Taylor and Stone, 2009] Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(1):1633–1685, 2009.

[Taylor *et al.*, 2007] Matthew E. Taylor, Peter Stone, and Yaxin Liu. Transfer Learning via Inter-Task Mappings for Temporal Difference Learning. *Journal of Machine Learning Research*, 8(1):2125–2167, 2007.

[Taylor *et al.*, 2011] Matthew E. Taylor, Halit Bener Suay, and Sonia Chernova. Integrating Reinforcement Learning with Human Demonstrations of Varying Ability. In *Proc. of AAMAS*, pages 617–624, 2011.

[Thomaz and Breazeal, 2006] Andrea Lockerd Thomaz and Cynthia Breazeal. Reinforcement Learning with Human Teachers: Evidence of Feedback and Guidance with Implications for Learning Performance. In *AAAI*, pages 1000–1005, 2006.

[Wang and Taylor, 2017] Zhaodong Wang and Matthew E. Taylor. Improving Reinforcement Learning with Confidence-Based Demonstrations. In *IJCAI*, pages 3027–3033, 2017.