In *Department of Computer Science Technical Report, Texas Tech University*, Lubbock, Texas, USA, May 17 2011.

Bootstrap Learning and Augmented Reinforcement Learning for Robust Performance in Agent Domains

Mohan Sridharan and Mamatha Aerolla
Department of Computer Science
Texas Tech University
Lubbock, TX 79409
{mohan.sridharan, mamatha.aerolla}@ttu.edu

May 17, 2011

Abstract

In application domains characterized by dynamic changes and non-deterministic action outcomes, it is frequently difficult for agents or robots to function without any human supervision. Although human feedback can help an agent learn a rich representation of the task and domain, humans may not have the expertise or time to provide elaborate, accurate and real-time feedback in complex domains. Widespread deployment of intelligent agents hence requires that these agents operate autonomously using limited and unreliable high-level feedback from non-expert human observers. Towards this objective, we advocate the use of bootstrap learning in an online augmented reinforcement learning framework. In the absence of human feedback, the agent operates by sensing the environment, i.e., using environmental feedback. When a human participant provides high-level feedback on the agent's performance, the agent uses a learning scheme that enables the environmental feedback and human feedback to bootstrap off of each other to continuously revise their relative contributions to the agent's action choice policy. The algorithms are implemented and evaluated in two simulated domains: *Tetris* and *Keepaway soccer*.

1 Introduction

Intelligent agents or robots interacting with humans in dynamic domains need the ability to operate reliably, efficiently and autonomously [11, 23]. Existing approaches to human-robot or human-computer interaction (HRI/HCI) predominantly focus on enabling the agent to operate autonomously based on sensory inputs [6, 9, 16], or to learn from extensive manual training and domain knowledge [1, 4, 12, 17, 25]. In dynamic domains characterized by partial observability and non-determinism, it is typically difficult for an agent to operate without any human input [8, 24]. On the other hand,

although human feedback can help an agent learn a rich representation of the task and domain, humans frequently do not possess the expertise or time to provide elaborate, accurate and real-time feedback in complex domains.

Widespread deployment of learning agents in the real-world requires that these agents be accessible to non-expert users who can provide limited high-level feedback in response to the agent's observed performance, e.g., positive/negative reinforcement of the agent's action choices or a selection from multiple options posed by the agent. Recent research has focused on enabling a robot or agent to acquire human feedback when needed (or available) and merge it with the information extracted from sensory cues. However, these methods do not model the unreliability of human inputs and require elaborate knowledge of the domain, limiting their use to simple simulated domains or specific robot tasks [3, 15, 19]. This paper presents an augmented reinforcement learning (ARL) framework that incorporates bootstrap learning to enable an agent to merge limited and unreliable high-level human feedback with the reinforcement obtained by sensing the result of environmental interactions. The ARL framework enables the environmental and human feedbacks to bootstrap off of each other to continuously and incrementally revise their relative contributions to the agent's action choices. The proposed approach is illustrated and evaluated in two simulated domains: (a) *Tetris*, which consists of a single agent; and (b) *Keepaway soccer*, a multiagent domain.

The remainder of the paper is organized as follows. Section 2 describes related work and Section 3 describes the proposed scheme and test domains. Experimental results are presented in Section 4, followed by conclusions in Section 5.

2 Related Work

Sophisticated approaches have been developed for key human-computer and human-robot interaction (HCI and HRI) challenges such as autonomous operation, engagement, safety, acceptance and interaction protocol design [11, 23]. Many algorithms have been developed to enable autonomous operation in HCI/HRI, using a variety of sensory inputs (e.g., visual, verbal and range data) to model social and environmental cues [7]. Considerable work has been done on using embodied relational agents and virtual agents in applications such as health care [18]. However, the autonomous operation typically requires a significant amount of domain knowledge, limiting the application of these algorithms to specific applications.

Significant research has also been performed on enabling a robot or a simulated agent to learn from demonstrations provided by a human observer [4, 12, 2, 10]. Many of these methods focus on building sophisticated mathematical models using recent research findings in a wide range of related fields such as control theory, biology and psychology. Some approaches have been based on theories of social interactions among humans and an understanding of the human learning process. A key constraint of these schemes is that the associated feedback and information can only be provided by human participants who possess substantial knowledge of the domain and the agent's capabilities.

Researchers are increasingly focusing on using limited high-level human feedback in robot/agent domains based on need and availability. For instance, Rosenthal et

al. [19] developed a *CoBot* that associates each action with probability functions of success and failure, and seeks human help (on failure) to localize and navigate to desired locations. Knox and Stone [15] developed the *TAMER* (Training an Agent Manually vis Evaluative Reinforcement) framework to enable a human to train a learning agent, and used different linear functions to combine human and environmental feedback and maximize a reward function in simulated domains. The work described in this paper is also based on an RL framework and a scheme to combine human and environmental feedback. The key difference is that the two feedback mechanisms bootstrap off of each other to continuously revise their relative contributions to the agent's action choice policy, thereby making best use of the human feedback.

3 Problem Formulation

This section describes the augmented reinforcement learning framework, which combines bootstrap learning with reinforcement learning, followed by the specific formulations for the experimental domains.

3.1 The RL Framework and Bootstrap Learning

Reinforcement learning (RL) is a computational goal-oriented approach, where an agent repeatedly performs actions on the environment and receives a state estimate and a reward signal [21]. It is common to model an RL task as a Markov decision process (MDP). In this paper, the standard formulation is augmented to include the human feedback signal, resulting in the tuple $\langle S, A, T, R, H \rangle$:

- S is the set of states.
- A is the set of actions.
- $T : S \times A \times S' \rightarrow [0,1]$, is the state transition function.
- $R : S \times A \rightarrow \Re$ is the environmental reward function.
- H is the human reward signal.

At each step, the agent uses a policy to probabilistically select an action $a \in A$ in state $s \in S$:

$$\pi : S \times A \rightarrow [0,1] \tag{1}$$

The goal is to compute the policy that maximizes the expected future reward over a planning horizon. One of many different schemes such as policy iteration, value iteration and policy gradient algorithms can be used to compute this policy. The key difference with respect to the standard MDP formulation is the inclusion of high-level human feedback that (like environmental feedback) can be unreliable. In addition, though the environmental feedback is obtained instantaneously for a given state and action, the human feedback can be a complex function of current, past or future states and actions.

As shown in Figure 1, a bootstrap learning scheme is used to effectively merge the two feedback mechanisms, modeling the action choice policy as a function of the feedback signals, the current state and the current action:

$$a = \underset{a \in A}{\operatorname{argmax}}\, f(R, H) \tag{2}$$

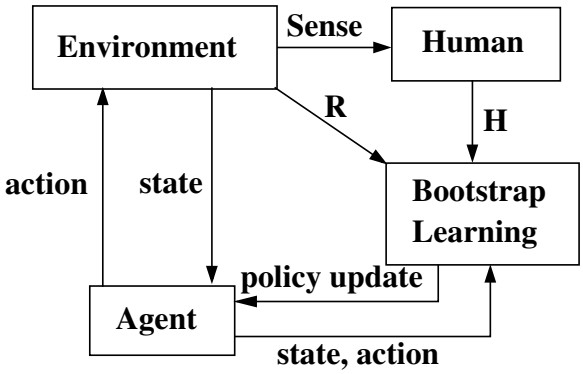

Figure 1: Augmented RL framework with bootstrap learning.

where R is the environmental feedback, H is the human feedback and a is the action choice that maximizes the function of R and H. In the experimental domains described below, the following linear function was evaluated:

$$a = \underset{a \in A}{\text{argmax}} \{w_r \cdot R + w_h \cdot H\} \tag{3}$$

where w_h is the "weight" assigned to the human feedback and w_r is the weight assigned to the environmental feedback. Since the weights are a measure of relative importance, we can set $w_r = 1$ and use w_h as the relative importance of human feedback. Choosing actions as a linear function of the feedback signals is similar to the scheme reported to have the best performance in the *Mountain Car* domain [15]. The novelty lies in the fact that our approach enables the two feedback mechanisms to bootstrap off of each other to update the weights and continuously revise the relative importance of each feedback mechanism. Another weighting scheme: $a = \text{argmax}_{a \in A} \{w_r \cdot R(1 + H^{w_h})\}$, referred to as the *exponential* scheme, was used to investigate a correlation between the feedback signals, but experimental results did not support such a correlation. The bootstrap learning scheme to update the weights proceeds as follows:

- The agent initially assumes the absence of human feedback and learns a policy through environmental interactions. The agent evaluates different policies generated by varying the parameters of the underlying RL algorithm (e.g., policy gradient), using a "performance measure" suitable for the domain—see Sections 3.2, 3.3 for examples.
- At any given time, the agent keeps track of the top N policies: $\pi_i, i \in [1,N]$, i.e., the policies that result in high values of the performance measure: $pm_i, i \in [1,N]$. Action choices are made based on one of these policies (with $w_r = 1$ in Equation 3), where the probability of a policy being chosen is proportional to the value of its performance measure relative to other policies.
- When human feedback is provided, the agent maintains a separate policy based on this feedback mechanism. The agent also keeps track of the degree of match between the action chosen by the current environmental feedback-based policy and action that would be chosen based on the human feedback-based policy. The degree of match can, for instance, be a numerical count of the number of times the two policies result

in the same action choice.

- If $m_i, i \in [1,N]$ represent the degree of match between the human feedback-based policy and the best environmental feedback-based policies, the weight associated with the human feedback can be estimated as:

$$w_h = \frac{\sum_i pm_i \times m_i}{\sum_i pm_i} \tag{4}$$

  where a high value of the weight represents a high degree of belief associated with the human feedback.
- A similar scheme can be used to evaluate the agent's performance with respect to one or more sources of human feedback if a substantial amount of human feedback is available. In addition, it is possible to update the weight/trust incrementally across different episodes. For instance, the weight w_h can be updated after episode k:

$$w_h^k = \frac{pm^{k-1} w_h^{k-1} + pm^k m^k}{pm^{k-1} + pm^k} \tag{5}$$

  based on the performance measure in this and the immediately previous episode.

The core component is the continuous and online update of the combined action policy, where the agent alternately assumes the policy (or policies) based on each feedback mechanism to be ground truth, in order to update the weight associated with the policy based on the other feedback mechanism. Since action choices are based on the combined policy (Equation 3), the agent quickly adapts to different humans, unreliability of environmental feedback, and dynamic changes over a period of time as the human observer gets tired or bored. Specific instances of this learning scheme are described below for the two simulated test domains.

3.2 Tetris Domain

Tetris is a game played on a $w \times h$ grid in which "tetrominoes" (i.e., shapes) of four blocks fall one at a time from the top of the grid, stacking up on the grid's base or any blocks below. If the blocks fall such that a row is completely filled with blocks, then that row is cleared. All the blocks in that row disappear and all the blocks in higher rows shift down by a row. When the blocks stack up to the top of the grid, the game ends. The goal of a Tetris player is to maneuver the falling blocks to clear as many lines as possible and maximize episode duration. The *performance measure* in this domain is hence the number of lines cleared per game (i.e., per episode). A screenshot of the domain is shown in Figure 2.

One challenge in this domain is the size of the state space—the 20×10 board shown in Figure 2 has a state space of $\approx 2^{200}$. The action space consists of four actions: *move left, move right, rotate clockwise, drop*, and the feedback signal from the environment (or human) is assumed to be instantaneous. Knox and Stone [15] developed a much smaller set of 21 features for this domain, e.g., column heights, maximum height of columns, number of holes and difference in heights of adjacent columns. An analysis of the significance of the feature set shows that it is in fact possible to obtain similar performance with just 12 features (maximum and average height of columns, number

Figure 2: The Tetris application domain

of holes and difference in column heights). However, for ease of comparison, we use the set of 21 features in our experiments. Two different RL algorithms were used: the policy gradient (PG) algorithm implemented in the libPG library [5] and the cross-entropy (CE) method [22] as used by Knox and Stone [15]. PG methods parametrize the policy function and use gradient descent to converge on a stochastic policy that optimizes the long-term reward—they are more robust to changes in policy parameters. The CE method learns weights for the feature vectors using sampling techniques in order to maximize the reward—it has been shown to outperform many RL methods in the Tetris domain.

3.3 Keepaway Soccer Domain

Keepaway is a subtask of robot soccer involving a small number of players. One team, the keepers, tries to maintain possession of a ball within a limited region, while another team, the takers, tries to gain possession. Whenever the takers take possession or the ball leaves the region, the episode ends and the players are reset for another episode with the keepers being given possession of the ball again. This domain is implemented within the RoboCup soccer simulator [20]. Parameters of the task include the size of the region, the number of keepers and the number of takers. Figure 3 shows a screen shot of an episode with 3 keepers and 2 takers (called $3vs.2$ or $3v2$ for short) playing in a $20m \times 20m$ region.

Keepaway is a challenging domain because the state space is too large to explore exhaustively, each agent has partial state information, and all agents in the team have to learn simultaneously. Since it is based on the RoboCup simulator, agents receive (noisy) visual perceptions every 150msec indicating the relative distance and angle to visible objects. In each episode, agents choose from higher-level macro actions

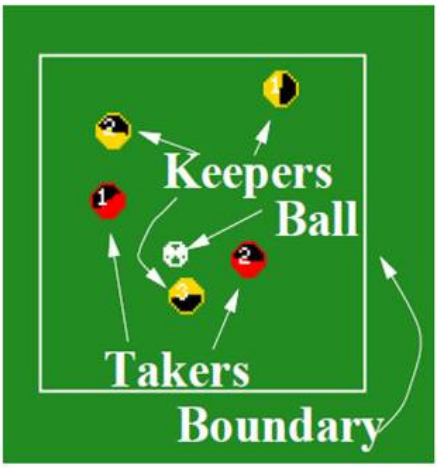

Figure 3: Keepaway (3v2) soccer domain.

based on the available skills (*HoldBall(), PassBall(k), GetOpen(), GoToBall(), BlockPass(k)*) instead of executing parametrized primitive actions (e.g., *Turn(angle)* and *Dash(power)*). The focus is on enabling the keepers to retain possession of the ball for as long as possible—the length of an episode is hence used as the *performance measure*. As in [20], the state space for the keepers is discretized to 13 variables that capture the distances and angles between the keepers, takers and the center of the region. The takers follow a default policy of moving to the ball. The semi-Markov decision process (SMDP) version of $Sarsa(\lambda)$ algorithm is used as the RL algorithm to modify the behavior of the keepers. Despite the reduction in state and action space, learning by exploring all possible states and actions is a challenge in this domain. Tile coding is hence used to make learning feasible, and the associated parameter values are chosen based on the values reported in [20].

Reinforcement signals in the keepaway domain are based on the performance of the team of keepers. Though environmental feedback can be encoded to occur instantaneously, the domain changes too quickly for a human to provide feedback for a specific state and action, i.e., human feedback cannot be modeled as being instantaneous. The human feedback is likely to be a complex function of prior time steps, i.e., a set of states and actions. Based on prior work [13, 14], the credit assignment of human feedback is modeled as a gamma distribution, as shown in Figure 4. The parameters of this function were estimated experimentally by conducting a study of the reaction times of a set of human participants in standard cognitive experiments. The resultant distribution is similar to that reported in [14]. If $p(x)$ is the gamma distribution-based probability density function (PDF), the credit assigned to a time interval as a result of a unit reinforcement is computed by integrating the PDF over the interval. In the experiments below, for a human feedback at time t, the mean of the gamma PDF is hence located at $\approx (t - 0.5)$ to assign credit to a set of states and actions. The PDF indicates

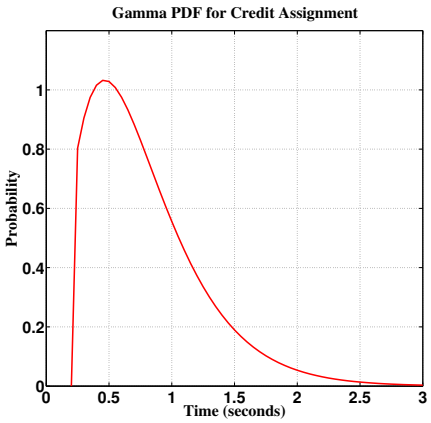

Figure 4: Probability density function $p(x)$ for credit assignment estimated as $gamma(2.5, 0.3)$. The x-axis denotes time before reinforcement (i.e., in the past).

that human input is delayed and that the credit drops off exponentially as we consider past states and actions.

4 Experimental Setup and Results

This section reports the results of experiments performed in the Tetris and Keepaway domains. The goal was to evaluate two hypotheses: (a) performance improves significantly when the human and environmental feedbacks are combined, in comparison to when the feedback mechanisms are used individually; and (b) using the bootstrap learning scheme improves performance in comparison when it is not used with the underlying reinforcement learning algorithms. In the remainder of this paper, *ARL* refers to the use of bootstrap learning in the modified RL framework. All results are statistically significant (at 95% level) unless otherwise stated.

Human Participants: Four non-expert humans participated in the experiments. These participants were provided a high-level description of the test domains, including available action choices and the performance measures to be maximized by the agent(s). However, these participants had no knowledge of the states and the underlying algorithms, and very little knowledge of these domains prior to these experiments.

Experiments were conducted in the Tetris domain using cross entropy as the underlying RL algorithm. The ARL approach and the linear combination function of Equation 3 was used to merge human feedback with the environmental feedback. For comparison, we implemented the linear combination scheme that iteratively annealed the weight assigned to the human feedback at the end of each episode—this scheme provided the best performance in the *Mountain Car* domain [15]. Figure 5 shows experimental results, with each data point obtained by averaging the results over a set of 20 trials. Human input was provided at infrequent intervals—no more than 5 times in any episode and on average 2 times per episode. The participants were provided breaks

between trials.

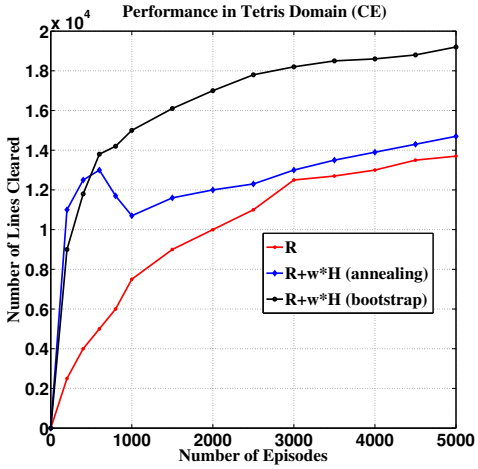

Figure 5: Performance in the Tetris domain using cross entropy as the default RL algorithm. The ARL approach performs significantly better than CE and the scheme that anneals the weight factor for human feedback.

The results show that the ARL approach improves performance (i.e., larger number of lines are cleared) in comparison to the default CE approach, and results in better performance than the combination scheme that anneals the weight assigned to human feedback between episodes [15]. The performance improvement is due to the ARL approach's ability to adapt to the unreliability of feedback signals, thereby exploiting their complementary properties. The performance of the ARL approach is also much better than that obtained with just the human feedback—these results are not included in Figure 5 because it is infeasible to provide human feedback over different states and actions for a large number of episodes.

Next the performance of the ARL approach was evaluated using policy gradient (PG) as the underlying RL algorithm. The results of these experiments are shown in Figure 6. As expected, the default PG algorithm does not result in as many lines being cleared as with the CE algorithm. However, the ARL approach for merging the feedback mechanisms still performs significantly better, i.e., clears a much larger number of lines than the default PG algorithm.

Finally, experiments were conducted in the Keepaway soccer domain, where the performance of the keepers was measured in terms of their ability to retain possession of the ball for as long as possible. The underlying RL algorithm was the SMDP version of *Sarsa*(λ). The ARL approach was used to incrementally determine the best combination of human feedback and environmental feedback to be used to determine the action choice policy. Here, both the linear combination scheme (Equation 3) and the exponential combination scheme (Section 3.1) were evaluated. As described in Section 3.3, it may not be possible to provide instantaneous human feedback in this

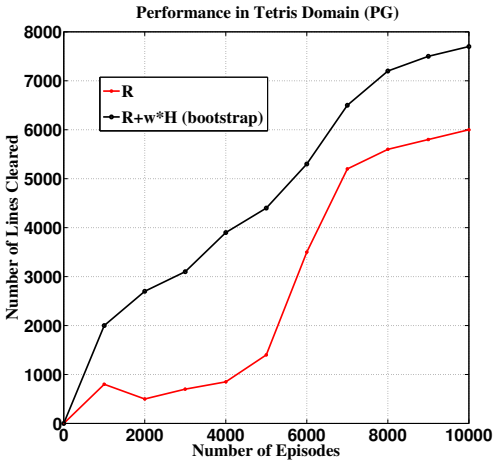

Figure 6: Performance in the Tetris domain using policy gradient as the default algorithm. The ARL approach significantly improves the performance of the PG algorithm.

domain. In order to measure the suitability of the gamma PDF shown in Figure 4 for credit assignment when human feedback is provided, performance was measured with and without the use of this function. Figure 7 summarizes the results, with each data point (as in the Tetris domain) representing the average over 20 trials. Given that episode times can vary (as seen in Figure 7) the human participants provided feedback infrequently—no more than two times in any episode. The human participants also provided (intentionally) incorrect feedback approximately once every ten feedback attempts.

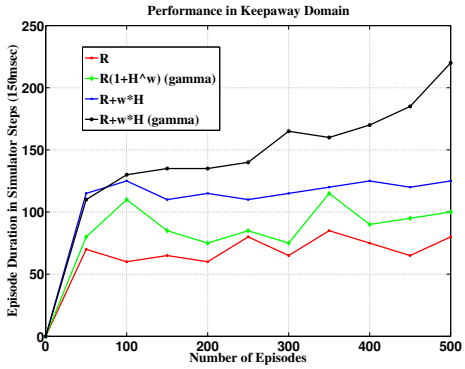

Figure 7: Performance in the Keepaway soccer domain using policy gradient. The ARL approach performs better than the default *Sarsa*(λ) algorithm. Using the learned gamma distribution for credit assignment significantly boosts performance.

The plots in Figure 7 indicate that all feedback combination mechanisms that use the ARL approach perform better than the default RL algorithm without any human feedback, despite the added unreliability in the human feedback. The linear combination of the two feedback mechanisms results in significantly better performance than the policy gradient RL algorithm. The best performance is achieved when the gamma PDF is used for credit assignment along with the linear combination of the feedback signals. The exponential combination scheme (with the gamma PDF) does not improve performance substantially, which may indicate that this combination scheme does not reflect the true relationship between the feedback signals. The experimental results in the two test domains indicate that using the gamma PDF-based credit assignment, bootstrap learning and linear combination scheme in a modified RL framework is a promising option for further investigation in other domains.

Threats to Validity: When human input is used in intelligent agent (or autonomous robot) domains, the performance may depend on the capabilities of the human participants involved in the study. The experiments reported in this paper used feedback provided by four human participants at infrequent intervals. Though the performance of the participants (when considered individually) were consistent across the two different domains, additional trials may be required in other domains to further substantiate the results reported here. Specifically, future experiments will consider a larger number of human participants, larger number of episodes (especially for Keepaway), varying amounts of added noise, other combination functions, other appropriate application domains and a thorough analysis of the corresponding experimental results.

5 Conclusions and Future Work

Human participants can enable agents or robots to learn a rich representation of the task, thereby operating reliably and efficiently in dynamic domains. However, it may be infeasible for a human to possess the time and expertise to provide elaborate, accurate and real-time feedback to agents in complex domains. This paper described an approach that uses bootstrap learning in an augmented reinforcement learning framework to enable an agent to effectively merge the limited and unreliable high-level feedback from a human with the reinforcement signals obtained through interactions with the environment. The agent incrementally and continuously revises weights that determine the relative contribution of each feedback mechanism to its action choice policy. The agent is hence able to make best use of the available information. Experimental results in the Tetris and Keepaway domains indicate that the approach described in this paper outperforms the individual feedback mechanisms and the established schemes to combine these feedback signals.

The main aim of the research reported in this paper is to find reliable ways of combining human and environmental feedbacks. One direction of further research is to use the proposed scheme along with an underlying probabilistic belief representation to enable an agent (or a robot) to operate in partially observable domains and automatically acquire relevant human input when needed. Future research will also focus on other multiagent and multirobot domains collaborating towards a shared objective in the real-world.

Acknowledgment

This work was supported in part by the ONR Science of Autonomy award N00014-09-1-0658.

References

[1] Brenna Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[2] A. Billard and K. Dautenhahn. Experiments in Social Robotics: Grounding and Use of Communication in Autonomous Agents. *Adaptive Behavior*, 7(3-4):415–438, 1999.

[3] Joydeep Biswas and Manuela Veloso. WiFi Localization and Navigation for Autonomous Indoor Mobile Robots. In *International Conference on Robotics and Automation (ICRA)*, Anchorage, USA, May 3-8 2010.

[4] Cynthia Breazeal and Andrea Thomaz. Learning from Human Teachers with Socially Guided Exploration. In *International Conference on Robotics and Automation*, pages 3539–3544, 2008.

[5] O. Buffet and D. Aberdeen. The Factored Policy-Gradient Planner. *Artificial Intelligence*, 173(5-6):722–747, 2009.

[6] Maya Cakmak, Nick DePalma, Rosa Arriaga, and Andrea Thomaz. Exploiting Social Partners in Robot Learning. *Autonomous Robots*, 29:309–329, 2010.

[7] Juan Fasola and Maja Mataric. Robot Motivator: Increasing User Enjoyment and Performance on a Physical/Cognitive Task. In *International Conference on Development and Learning*, Ann Arbor, USA, August 2010.

[8] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A Survey of Socially Interactive Robots. *Robotics and Autonomous Systems*, 42(3-4):143–166, 2003.

[9] J. Forlizzi and C. DiSalvo. Service Robots in the Domestic Environment: A Study of the Roomba Vacuum in the Home. In *International Conference on Human-Robot Interaction, HRI-06*, pages 258–266, Salt Lake City, USA, March 2-4 2006.

[10] D. W. Franklin, T. W. Milner, and M. Kawato. Single Trial Learning of External Dynamics: What can the Brain Teach Us about Learning Mechanisms in Brain Inspired IT. *International Conference on Brain-Inspired Information Technology*, pages 67–70, 2007.

[11] Michael A. Goodrich and Alan C. Schultz. Human-Robot Interaction: A Survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.

[12] Daniel Grollman. *Teaching Old Dogs New Tricks: Incremental Multimap Regression for Interactive Robot Learning from Demonstration*. PhD thesis, Department of Computer Science, Brown University, 2010.

[13] W. E. Hockley. Analysis of Response Time Distributions in the Study of Cognitive Processes. *Journal of Experimental Psychology. Learning, Memory and Cognition*, 10, 1984.

[14] William Knox, Ian Fasel, and Peter Stone. Design Principles for Creating Human-Shapable Agents. In *AAAI Spring Symposium on Agents that Learn from Human Teachers*, 2009.

[15] William Knox and Peter Stone. Combining Manual Feedback with Subsequent MDP Reward Signals for Reinforcement Learning. In *International Conference on Autonomous Agents and Multiagent Systems*, May 2010.

[16] H. Lee, H. Kim, and C. Kim. Autonomous Behavior Design for Robotics Appliances. In *International Conference on Human-Robot Interaction*, pages 201–208, Washington D.C., USA, March 10-12 2007.

[17] Daniel Perzanowski, Alan Schultz, William Adams, Elaine Marsh, and Magda Bugajska. Building a Multimodal Human-Robot Interface. *IEEE Intelligent Systems*, 16(1):16–21, January-February 2001.

[18] Albert Rizzo, Thomas Parsons, Galen Buckwalter, and Patrick Kenny. A New Generation of Intelligent Virtual Patients for Clinical Training. In *The IEEE Virtual Reality Conference*, Waltham, USA, March 21 2010.

[19] Stephanie Rosenthal, Joydeep Biswas, and Manuela Veloso. An Effective Personal Mobile Robot Agent Through Symbiotic Human-Robot Interaction. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Toronto, Canada, May 2010.

[20] Peter Stone, Richard Sutton, and Gregory Kuhlmann. Reinforcement Learning for RoboCup Soccer Keepaway. *Adaptive Behavior*, 13:165–188, 2005.

[21] R. L. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.

[22] Istvan Szita and Andras Lorincz. Learning Tetris using Noisy Cross Entropy Method. *Neural Computation*, 18:2936–2941, 2006.

[23] Adriana Tapus, Maja Mataric, and Brian Scassellati. The Grand Challenges in Socially Assistive Robotics. *Robotics and Automation Magazine, Special Issue on Grand Challenges in Robotics*, 14(1):35–42, March 2007.

[24] Sebastian Thrun. Toward a Framework for Human-robot Interaction. *Human-Computer Interaction*, 19:2004, 2004.

[25] Peng Zang, Arya Irani, Peng Zhou, Charles Isbell, and Andrea Thomaz. Using Training regimens to Teach Expanding Function Approximators. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 341–348, 2010.