An Application of Reinforcement Learning to Supervised Autonomy

Gavin Taylor and **Kawika Barabin** and **Kent Sayre**
Computer Science Department
United States Naval Academy
572M Holloway Rd. Stop 9F
Annapolis, MD 21402-5002

Abstract

Reinforcement Learning is a field of Machine Learning and Artificial Intelligence which seeks to use past experiences to predict future success in autonomous domains. Reinforcement Learning carries the advantages of generality, in that it makes few assumptions, and limits itself only to what can be learned from the data. This same generality, however, often makes it a poor approach for generating autonomy in specific applications, where prior domain knowledge can be leveraged to ease computation and improve accuracy. In this paper, we describe a new, more realistic application of Reinforcement Learning to scenarios of *supervised* autonomy, in which the vast prior research in this field can be leveraged to provide useful and succinct information to a human supervisor tasked with command and control, as well as automatically identify negative anomalies of both the sudden and "low-and-slow" varieties. Additionally, we identify an exciting new field of research in which the error observed in Reinforcement Learning predictions may be used to identify inefficiencies and evaluate autonomous systems and human decision making.

1 Introduction

The military and commercial communities increasingly rely on autonomous systems to augment their capabilities. For example, power plants feature automatic monitoring and safety features, and the military increasingly employs unmanned systems in denied or politically sensitive theaters. However, it is rare for these systems to be truly autonomous; generally, fine-grained decisions are made by computer (such as an autopilot), while coarse-grained decisions (such as mission tasking or flight plans) are made by human operator. Thus, autonomy is not a binary on-off switch, but instead lies on a continuum. As technology progresses, it is expected that daily operations will shift along this continuum such that more and more decisions will be made by autonomous machine. The current point in the continuum is referred to as supervised autonomy, in which human supervisors are tasked with maintaining awareness and identifying and responding to negative surprises, potentially for multiple systems simultaneously. In this scenario, it is important that accurate measures of mission progress be communicated quickly and succinctly, so that supervisors can direct their attention properly, particularly when overtasked.

In this paper we make several points. First, we believe the field of Reinforcement Learning can provide the tools necessary to communicate this type of information. This is addressed in more detail in Section 3. Second, we believe automated feature selection for Reinforcement Learning provides an intuitive way to identify features of interest in autonomous systems, as is discussed in Section 4. Finally, we believe Reinforcement Learning may provide several additional benefits for negative anomaly detection and human decision-making analysis. These benefits are discussed in Section 5.

2 Reinforcement Learning

The purpose of this section is to describe the field of Reinforcement Learning so that the new applications can be better understood. We begin by describing the problem intuitively, before introducing the mathematical definition. Imagine, for some new unknown system, a human is presented with telemetry data entirely describing every aspect of the system (position, orientation, velocity, sensor readings, control surface position, etc.) for every second of a large number of missions. Given this data, it is possible to imagine the human learning what the system is capable of doing (for example, when learning about an unmanned aircraft, the human may learn that altitude gain can be achieved by setting particular control surfaces to certain settings, but there is a maximum climb rate which cannot be exceeded). Now imagine an expert assigns a "reward" to each observation; a state in which the mission is completed would receive a high reward, while a state in which the system is damaged would receive a negative reward. A logical question, then, is given some new state, what rewards can we expect to receive in the future? Are there certain actions we can take to maximize this expected future rewards? These questions are very important in a context with human supervision; if we expect to receive many high rewards, no intervention may be required. However, if we are in a state with low expected rewards, retasking may be necessary. Reinforcement Learning provides algorithms for performing this predictive analysis.

Mathematically, the problem is described as a Markov Decision Process (MDP). An MDP is a tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where $\mathcal{S}$ is the set of possible states the agent could inhabit, $\mathcal{A}$ is the set of actions the agent can take, $\mathcal{P}$ is a transition kernel describing the probability of transitioning between two states given the performance of an action ($p(s'|s,a)$, where $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$), $R : \mathcal{S} \mapsto \Re$

is the reward function, and $\gamma \in (0,1)$ is the discount factor, which is used to describe how much we prefer rewards in the short term to rewards in the long term. The expected future rewards from a given state when optimal actions are taken is defined as the *optimal value* of that state, where the optimal value function is defined by the Bellman equation

$$V^*(s) = R(s) + \gamma \max_a \int_{\mathcal{S}} p(s'|s,a)V^*(s')ds'. \quad (1)$$

This function defines the optimal value of a state as the expected, discounted sum of rewards from this state forward. In large or complex domains, it is impossible to calculate V^* exactly, forcing instead the construction of an approximation. A large number of approaches to perform this approximation exist, each with their own benefits and drawbacks.

Classically, an accurate value function is thought of as useful for creating autonomy; if choosing between two states, the decision-making agent should choose the one with higher value. This turns long-term planning into a series of greedy, short-term decisions. Because value function approximation techniques are usually general, and can often be applied to any MDP, advancements in theoretical Reinforcement Learning lead to better RL-based autonomy in any domain.

This same generality is a drawback in practical application, however. While it is theoretically pleasing to begin with no assumptions or prior knowledge about the agent being observed, this is not usually the case. For example, in the case of a new UAV, the known dynamics of the UAV can usually be used to generate a better autopilot than would be generated by Reinforcement Learning, which ignores this tremendously useful prior information.

In addition, physical systems carry an additional difficulty in that no matter what action is taken, the rate of change between states is gradual, particularly when states are observed with high frequency. Therefore, any mistake made on a given state can be quickly fixed in the next timestep. This means the difference in optimal value between different actions is likely minimal, and correctly choosing actions possibly requires more precision than an approximation architecture can reasonably supply.

In light of these challenges, it is worthwhile to consider other scenarios which may make more sense for Reinforcement Learning than policy generation for physical systems. Therefore, the remainder of this paper focuses not on the traditional applications of a value function for autonomy generation, but instead on new applications of this function in a supervised autonomy context.

3 Supervised Autonomy

In a supervised autonomy context, we assume the agent under most circumstances is capable of performing its task unassisted; the problem then becomes one of helping a possibly-overtasked human supervisor identify cases which do require human intervention. For example, consider the case of a human tasking multiple UAVs at a time to accomplish a variety of missions. Perhaps one of the systems being watched is making slower-than-expected progress, and so another system will have to be tasked to complete its mission before the opportunity expires. Or, more dramatically, perhaps a malfunction occurs and human intervention is required to land the UAV away from the runway with as little damage as possible, in a location easy for recovery. In these cases, the identification of states from which we expect to receive low rewards is an appropriate way to identify moments which are unlikely to result in mission success.

Intuitively, the value of a state can be thought of as a one-dimensional measure of predicted success. In scenarios where supervisors must keep situational awareness on several independent agents, the distillation of this information into something quickly digestible and understandable is essential. Because the value function is built based on what is expected to occur, given past behavior and an optimal autopilot, a small or negative value function may be an extremely useful indicator of possible mission failure.

We note that the level of accuracy necessary for providing this feedback to a supervisor or performing this analysis is far less than the level of accuracy an agent requires to actually choose actions based entirely on the value function.

4 Feature Identification and Selection

This viewpoint also may provide ancillary benefits. Mathematically, many approximation schemes take the form of a linear approximation, where $V(s) \approx \Phi(s)w$, for all $s \in \mathcal{S}$. In this approximation, $\Phi(s)$ is a set of feature values for state s, and w is a vector which linearly weights these features. The quality of the approximation depends in large part on the usefulness of these features. Therefore, techniques which can select the few features from a large dictionary which result in the highest quality approximation can provide a great benefit, and are an active area of study (Kolter and Ng 2009; Johns, Painter-Wakefield, and Parr 2010; Mahadevan and Liu 2012; Liu, Mahadevan, and Liu 2012). Interestingly for this domain, the identification of features which are highly correlated with future success and failure may provide an intuitive means of understanding the domain and in the construction of early-warning alarms.

One such technique for performing automated feature selection while calculating an approximate value function is L_1-Regularized Approximate Linear Programming (RALP) (Petrik et al. 2010). RALP is unique among other feature-selection approaches in that it has tightly bounded value function approximation error, even when using off-policy samples, and when the domain features a great deal of noise (Taylor and Parr 2012).

RALP works by solving the following convex optimization problem:

$$\begin{aligned} \min_w \quad & \rho^T \Phi w \\ \text{s.t.} \quad & \sigma^r + \gamma \Phi(\sigma^{s'})w \leq \Phi(\sigma^s)w \ \ \forall \sigma \in \Sigma \\ & \|w\|_{-1} \leq \psi. \end{aligned} \quad (2)$$

Φ is the feature matrix, Σ is the set of all samples, where σ is one such sample, in which the agent started at state σ^s, received reward σ^r, and transitioned to state $\sigma^{s'}$. ρ is a distribution, which we call the state-relevance weights, in keeping with the (unregularized) Approximate Linear Programming

(ALP) terminology of (de Farias and Van Roy 2003). $\|w\|_{-1}$ is the L_1 norm of the vector consisting of all weights excepting the one corresponding to the constant feature.

This final constraint, which contributes L_1 regularization, provides several benefits. First, regularization in general ensures the linear program is bounded, and produces a smoother value function. Second, L_1 regularization in particular produces a sparse solution, producing automated feature selection from an overcomplete feature set. Finally, the sparsity results in few of the constraints being active, speeding the search for a solution by a linear program solver, particularly if constraint generation is used.

Most relevantly to this problem, the selected features are those most useful in predicting the future receipt of rewards. Practically, these can be interpreted as the few features, from a potentially large candidate set, which carry the strongest signal predicting success or failure; while this definitely improves prediction accuracy, it may also be useful in related tasks in which an intuitive prediction of future autonomous performance will be useful, such as in scenarios of transfer learning.

5 Approximation Error for Alarming and Analysis

We also propose an exciting direction of research in using the error in approximation as a useful signal in flight analysis and alarming. Consider again the Bellman equation of Equation 1, and a single observation, consisting of a state, a received reward, and the next state (an observation can be denoted (s, r, s'), where $s, s' \in \mathcal{S}$, and $r = R(s)$). Also assume the existence of some approximation $\hat{V}$, such that $\hat{V}(s) \approx V^*(s)$ for all $s \in \mathcal{S}$. If the approximation is exactly correct, and everything goes as expected, then $\hat{V}(s) = r + \gamma\hat{V}(s')$ for all such observations. If this equality does not hold, then we have non-zero *empirical Bellman error*

$$BE(s) = r + \gamma\hat{V}(s') - \hat{V}(s). \quad (3)$$

When the Bellman error is non-zero, there are several possible explanations.

The first explanation is that the approximation to the value function was incorrect. This explanation has classically received the most attention, as researchers tried to tune their predictions to be as accurate as possible. However, even if the value function $\hat{V} = V^*$ were calculated exactly, in a non-deterministic environment there would still be non-zero Bellman error.

A second explanation is that the prediction would have been accurate, but the controller (autopilot or human) chose an action which was not optimal. We can expect this to happen consistently to some degree. This is because optimality is defined as choosing the action that maximizes the sum of rewards; however, existing autopilot and human decision making is performed using an approach independent of Reinforcement Learning, and ignorant of the designed reward function. Realistically, all approaches are targeting some approximation of optimal performance, and it is unreasonable to expect them to precisely agree on optimal decision making. However, we argue that decision making resulting from these differing approaches should not differ too much, particularly if the reward function is carefully tuned. Under this circumstance, unexpected large drops in the value function would be interesting in performing objective performance assessment, such as in human training and flight analysis.

A third explanation is that the approximation was accurate, but something unexpected happened during the state transition. A positive example would be if a tailwind pushed the aircraft closer to a target location than would normally be expected; in this case, the value of the next state would be larger than expected. More interestingly for supervised autonomy, however, would be in identifying unexpected drops in value, or negative anomalies. It is perhaps obvious this would be useful in identifying sudden, dramatic drops, as might result from an event such as mechanical failure. More interesting, however, is the potential for identifying prolonged periods of underperformance which might indicate a "low-and-slow" anomaly which might otherwise be difficult to detect. These types of anomalies occur when a small error impacts an output over a long period of time. These errors are difficult to detect as they typically do not cross any predefined error boundaries.

To our knowledge, the Bellman error has never been used to perform this type of analysis.

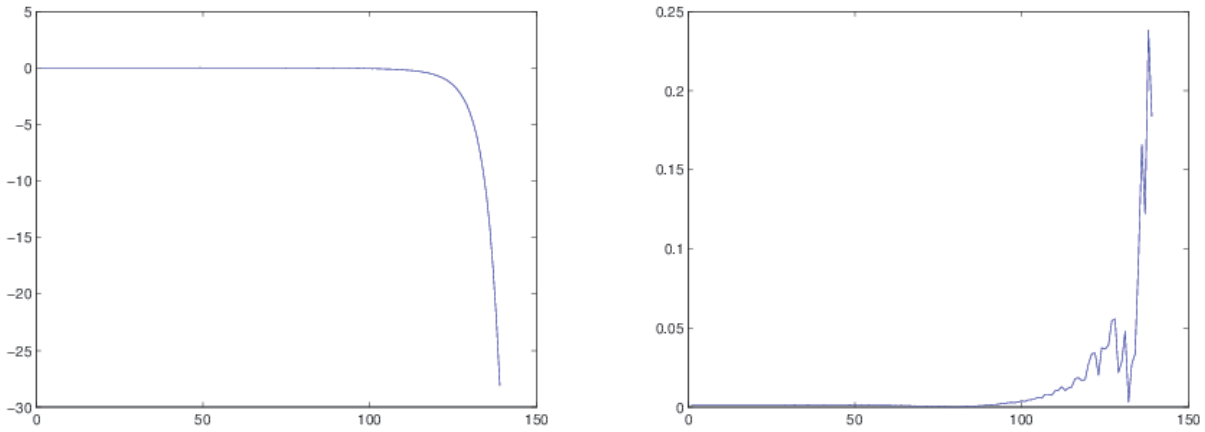
6 Illustration

To illustrate the insights of Sections 3 and 5, we use data from two different domains. First, we use a simulated domain often used for benchmarking value function approximation algorithms. This domain provides a fairly simple environment to demonstrate the application of Reinforcement Learning in an easily-visualized way. Second, we use real world data collected by the in-flight computers of landing TigerShark UAVs to demonstrate the technique's efficacy even in less-predictable, real-world scenarios.

Synthetic Domain

We use a common Reinforcement Learning benchmark domain, in which an autonomous agent learns to ride a bicycle (Randløv and Alstrøm 1998; Lagoudakis and Parr 2003). In this domain, the bicycle begins facing north, and must reach a goal state one kilometer to the east. To be precise, a state in the space consists of the bicycle's angle from perfectly upright, the angular velocity and angular acceleration with respect to the bicycle and the ground, the handlebar angle and angular velocity of the handlebar, and the angle of the goal state from the current direction of travel. Actions include leaning left or right, turning the handlebars left or right, or doing nothing at all. The reward function is a shaping function based on progress made towards the goal.

Samples to learn from are drawn by beginning in an upright position, and taking random actions until failure. Given a large enough set of such samples, the agent can learn a value function, which it then uses to balance and successfully navigate through the space on a high percentage of trials. To do this, the approximate value function was constructed using RALP, using features composed of a large number of monomials defined on dimensions of the state space.



(a) Value as a function of time

(b) Empirical Bellman error as a function of time

Figure 1: Bicycle Domain

As our illustration, we consider one run, in which the bicyclist fails to remain upright despite access to a good previously-calculated value function. As the agent moves through the state space, the value function of its current state changes as a function of time. Figure 1a graphs this value until failure. Notice that as the bicycle begins to oscillate and eventually collapse, the expectation begins to drop, as it becomes less and less likely the agent will be able to recover and receive the rewards associated with achieving the task. It is easy to imagine an alarm system set off in the beginning stages of this process, alerting a supervisor of the coming failure.

Given the same run, we also plot the absolute value of the empirical Bellman error of Equation 3 as a function of time in Figure 1b. Notice the predictions were extremely accurate, and Bellman error very low, until shortly before the bicycle began to fall. In analyzing the performance of the agent, this information provides insight into when the initial mistakes were made which would, several time steps in the future, result in a failed mission; in the analysis of the agent's performance, this is crucial information.

If, however, the interest is not in analyzing past performance, but is in identifying anomalies which may require human intervention in real time, the Bellman error illustrated in Figure 1b would again be extremely useful. It is plainly visually obvious when unexpected behavior began, and it is equally clear it correlates with the beginnings of the task failure. In particular, we would like to emphasize the anomalies begin to appear well before the bicycle collapses, allowing it to serve as an early warning system.

Real-World Domain

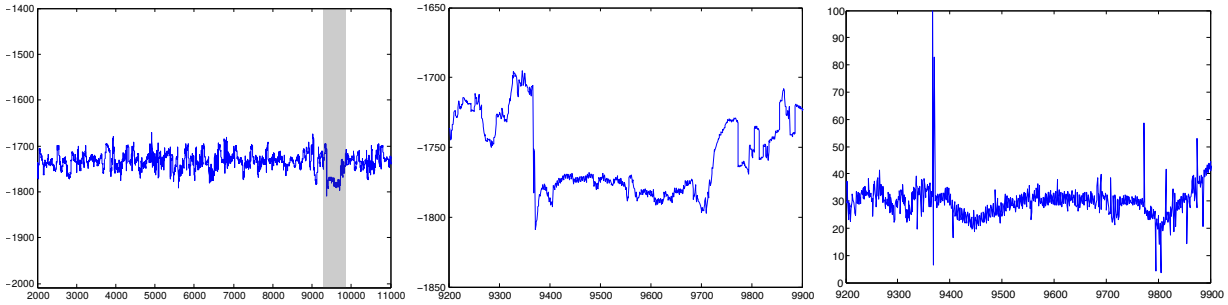
The TigerShark Unmanned Aerial Vehicle is a remotely-operated surveillance aircraft with a 22-foot wingspan and weighing 260 pounds, currently in use in theater operations. It can perform autonomous waypoint navigation, or be controlled manually. For this domain, we focus on autonomous alarming for TigerShark landings.

Telemetry data from 1,267 landing attempts were collected, for a total of approximately 6.2 million individual samples, which were collected every 1000 milliseconds. Landings were performed both by human pilots and the autopilot. The samples themselves are made up of 192 different sensor readings from the UAV control and sensor surfaces, as well as sensor data taken on the ground. Aside from flight status information like altitude, latitude, longitude, ground speed, etc., the data includes information about the controls coming from the pilot or autopilot, and the connection between the ground station and the UAV.

During landing, pilots have instructions to not allow their roll to extend past a certain angle, to follow a prescribed glide slope, and to keep their aircraft as close to the extended center line of the runway as possible. Therefore, in addition to the sensor readings, features also included the glide slope, the distance from the desired glide slope squared, the distance from the center line squared, the roll squared, and the reward awarded to the state.

The reward function was a shaping function based around the error from the desired glide slope, the distance from the center line of the runway, and the absolute value of the roll. In addition, the reward function contained additional penalties if the aircraft left the extended center third of the runway, or performed too large of a roll. By rule, either of the latter two cases should result in a decision to abort the landing.

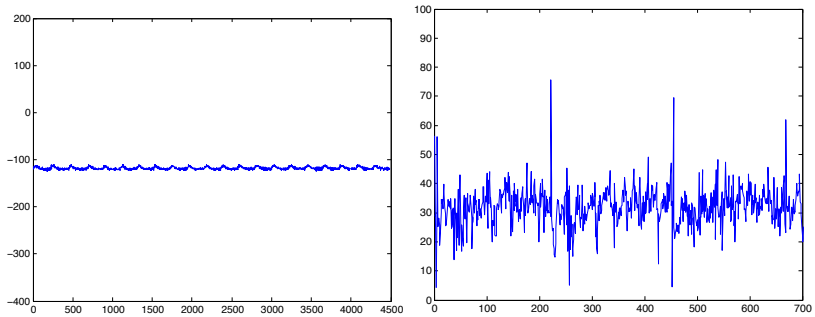
This reward function differs from the one in the bicycle domain in a very significant way. In the synthetic domain, the actions taken by the autonomous agent were taken with the purpose of maximizing the rewards received. In the real-world domain, the actions taken are simply those of a pilot attempting to land an aircraft. As such, we can expect a much larger Bellman error in most states, as the value function is making a prediction on the erroneous assumption the



(a) Value as a function of time on unsuccessful flight

(b) Zoomed value as a function of time on unsuccessful flight

(c) Zoomed empirical Bellman error as a function of time on unsuccessful flight



(d) Value as a function of time on successful flight

(e) Empirical Bellman error as a function of time on successful flight

Figure 2: TigerShark Domain

pilot is aware of, and is trying to maximize, the reward function. Nevertheless, we will see the Bellman error is nonetheless helpful.

From the set of all landing samples, 2000 were randomly selected, and a value function computed using RALP. We then applied this value function to two separate flights. In the first flight, though the pilot eventually succeeded in landing the plane, the path was erratic and the landing should have been aborted due to an inability to stay within the allowed center third of the runway, and inability to maintain the roll within allowed parameters.

In Figure 2a we see the value function of this landing. The shaded area is an area of special interest, as it is quite low, and produces the lowest value of the landing. We zoom on this area for both the value function (Figure 2b) and the Bellman error (Figure 2c). First, we note that both the value function and the Bellman error are much less smooth than in the bicycle domain. As discussed in Section 5, this is expected to some degree, as the pilot is not making decisions based on the reward function. Nevertheless, there is still a great deal of information in these two graphs. In particular, we note that the Bellman error spikes greatly in the timestep before the value function begins to drop. This drop results from the aircraft swinging so wide as to cross the boundary for allowed flight, outside the center third of the runway, and trying to recover by banking steeply. At this point, the landing should have been aborted. The spike in Bellman error preceding this drop illustrates the potential for a Bellman error-based alarm.

In contrast, Figures 2d and 2e are the result of a flight without any such negative occurrences (Figure 2e is zoomed to provide the same scale as Figure 2c). Note that the two value functions 2a and 2d are scaled identically on the y axis, though the bad value function is much, much lower, as you would expect from a flight struggling to stay on course. Second, we note the value function for the good flight is much less erratic. The Bellman error, meanwhile, remains generally low, with no significant spikes to set off an alarm.

In a scenario of human performance analysis, it is clear from the magnitude of the value function that the pilot for this flight avoided the problems of the first flight; the small magnitude implies the second pilot performed much closer to the way an optimal pilot would be expected to. Additionally, the small oscillations imply a steadier flight.

These illustrations also demonstrate the usefulness of the Bellman error in alarming an autonomous mission; when things went wrong, the Bellman error spiked, providing a concise channel of communication of the error to the operator.

7 Conclusion

This document demonstrates exciting directions of research for both those interested in furthering capabilities in scenarios of supervised autonomy and theoreticians interested in Reinforcement Learning. First, we believe value function approximation will prove to be a useful tool in communi-

cating mission health to human supervisors of autonomous systems. Second, we describe feature selection techniques which are of use in any task which is benefited from a performance prediction. Finally, we propose the use of the Bellman error for anomaly detection and analysis of human decision-making.

Significant future work is required. First, it is important that single-channel anomaly detection techniques be developed and applied to the Bellman error in order to discover how best to alarm a system with this information. In this document, we have merely demonstrated the strong correlation between large Bellman error and poor mission performance, but have yet to construct an alarm for a system. This step will require an automated understanding of "normal" versus "abnormal" Bellman error variations.

Second, the illustrated approach did not use recent advances in parameter tuning or sampling for RALP (Taylor, Geer, and Piekut 2014). These advances demonstrated that heavier sampling or stronger weighting of the parameter ρ in equation 2 on states which have low value from a Lyapunov function defined on the state space result in a more accurate value function; a better value function would result in a more helpful picture. This could be leveraged in the TigerShark landing scenario.

Finally, our reward function in the real-world domain was created independently from the data, and somewhat haphazardly; the pilots in the collected data were not aware of, or trying to optimize, our value function. It may be extremely useful to apply inverse Reinforcement Learning on the data collected for training in order to create a reward function which better represents the goals of the pilots. We believe this would result in a much more informative Bellman Error than is illustrated in Figure 2.

Acknowledgement

The authors appreciate support from the Office of Naval Research Grant N0001414WX20507 and the Naval Research Laboratory Information Management and Decision Architectures Branch (Code 5580).

References

de Farias, D. P., and Van Roy, B. 2003. The Linear Programming Approach to Approximate Dynamic Programming. *Operations Research*.

Johns, J.; Painter-Wakefield, C.; and Parr, R. 2010. Linear complementarity for regularized policy evaluation and improvement. In Lafferty, J.; Williams, C. K. I.; Shawe-Taylor, J.; Zemel, R.; and Culotta, A., eds., *Advances in Neural Information Processing Systems 23*, 1009–1017.

Kolter, J. Z., and Ng, A. 2009. Regularization and Feature Selection in Least-Squares Temporal Difference Learning. In Bottou, L., and Littman, M., eds., *Proceedings of the 26th International Conference on Machine Learning*, 521–528. Montreal, Canada: Omnipress.

Lagoudakis, M. G., and Parr, R. 2003. Reinforcement Learning as Classification: Leveraging Modern Classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning*, volume 20, 424–431.

Liu, B.; Mahadevan, S.; and Liu, J. 2012. Regularized Off-Policy TD-Learning. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*.

Mahadevan, S., and Liu, B. 2012. Sparse Q-Learning With Mirror Descent. In *Conference on Uncertainty in Artificial Intelligence*.

Petrik, M.; Taylor, G.; Parr, R.; and Zilberstein, S. 2010. Feature selection using regularization in approximate linear programs for markov decision processes. In *Proceedings of the 27th International Conference on Machine Learning*.

Randløv, J., and Alstrøm, P. 1998. Learning to Drive a Bicycle using Reinforcement Learning and Shaping. In *Proceedings of the 15th International Conference on Machine Learning*, 463–471.

Taylor, G., and Parr, R. 2012. Value function approximation in noisy environments using locally smoothed regularized approximate linear programs. In de Freitas, N., and Murphy, K., eds., *Conference on Uncertainty in Artificial Intelligence*, 835–842.

Taylor, G.; Geer, C.; and Piekut, D. 2014. An Analysis of State-Relevance Weights and Sampling Distributions on L1-Regularized Approximate Linear Programming Approximation Accuracy. In *Proceedings of the 31st International Conference on Machine Learning*.