Journal of Machine Learning Research (EWRL Series) (2016) Submitted 9/16; Published 12/16

Value-Aware Loss Function for Model Learning in Reinforcement Learning

Amir-massoud Farahmand FARAHMAND@MERL.COM
Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA

André M.S. Barreto AMSB@LNCC.BR
National Laboratory for Scientific Computing (LNCC), Petrópolis, RJ, Brazil

Daniel N. Nikovski NIKOVSKI@MERL.COM
Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA

Editor: Gergely Neu, Vicenç Gómez, Csaba Szepesvári

Abstract

We consider the problem of estimating the transition probability kernel to be used by a model-based reinforcement learning (RL) algorithm. We argue that estimating a generative model that minimizes a probabilistic loss, such as the log-loss, might be an overkill because such a probabilistic loss does not take into account the underlying structure of the decision problem and the RL algorithm that intends to solve it. We introduce a loss function that takes the structure of the value function into account. We provide a finite-sample upper bound for the loss function showing the dependence of the error on model approximation error and the number of samples.

1. Introduction

Model-based reinforcement learning (RL) (Sutton and Barto, 1998; Szepesvári, 2010) is based on the idea that one might learn a good policy for an RL agent by first learning a good model of the environment and then using the learned model to find the policy (with the possibility of interleaving these two phases as in Dyna by Sutton 1990). More concretely, suppose that we are given a dataset $\mathcal{D}_n = \{(X_i, A_i, R_i, X'_i)\}_{i=1}^n$ with $X'_i \sim \mathcal{P}^*(\cdot|X_i, A_i)$, the true transition probability kernel of a Markov Decision Process (MDP), and $R_i \sim R(\cdot|X_i, A_i)$, the true reward kernel of the MDP. The model learning phase is to estimate $\mathcal{P}^*$ by $\hat{\mathcal{P}}$ and $r(x,a) = \mathbb{E}[R(\cdot|x,a)]$ by $\hat{r}$. The learned model is then used to generate new samples, see e.g., (Sutton et al., 2008; Farahmand et al., 2009b; Hester and Stone, 2013; Deisenroth et al., 2015). A standard RL/Planning algorithm can use these samples to find a close to optimal policy, possibly by first finding an approximation to the optimal (action-)value function. In the rest of this work, we only focus on learning $\mathcal{P}^*$.

How can we learn a good model $\hat{\mathcal{P}}$ of $\mathcal{P}^*$? Since $\mathcal{P}^*$ is a conditional probability distribution/density, one might be tempted to use one of the standard distribution estimation techniques. Some generic approaches are Maximum Likelihood Estimation (MLE), Maximum Entropy (MaxEnt) estimation, the Maximum A Posteriori (MAP) estimation, and Bayesian posterior inference.

Let us focus on the MLE. The ML estimator is the minimizer of the empirical negative-log loss, which in turn is an empirical approximation to the KL divergence between the true model and the estimate, i.e., $\hat{P} \leftarrow \operatorname{argmin}_{P \in \mathcal{M}} \mathsf{KL}(P_n || P) \equiv \operatorname{argmax}_{P \in \mathcal{M}} \frac{1}{n} \sum_{X_i \in \mathcal{D}_n} \log P(X_i)$ with $\mathcal{D}_n = \{X_i\}_{i=1}^n$ and $X_i \sim P^*$, the true distribution of data, and $\mathcal{M}$ being the probability model space (this formulation is for unconditional distribution; the conditional one is similar). If the true model P^* belongs to the model space $\mathcal{M}$, one might show that under certain conditions $\hat{P} \to P^*$ (in some well-defined sense, for example, in the total variation or the KL distance). This would be more than enough for the purpose of using the model for planning (cf. (2) in Section 2). This approach is what most model-based RL algorithms use to estimate the model $\hat{\mathcal{P}}$ of $\mathcal{P}^*$.

Nevertheless, requiring such a guarantee on the error behaviour of the learned model according to a probabilistic loss such as the KL-divergence might be an overkill for the purpose of planning. To intuitively understand why this might be the case, consider the following hypothetical situations.

Consider a visually-enabled robot that is supposed to learn how to navigate within a building. If we consider the camera image as a part of the state of the robot, trying to learn a transition probability kernel means that we have to learn how the camera image changes when the robot takes actions.[1] This is a very high-dimensional state space and trying to learn such a conditional distribution with a high accuracy, in the log-loss sense, is difficult. Nonetheless, modeling the probability distribution at that level of accuracy is not required to learn a policy that can navigate the robot in the building just fine. It is enough to have a crude model that describes the geometry of the building and possibly the objects. This "navigator" robot does not really need to know the detail of paintings on the walls, the texture of objects, and many other visual details of the building. On the other hand, if the goal is to have an interior "decorator" robot that suggests how to redecorate the building to make it visually appealing, all those visual information is likely required.

The difference between the navigator robot and the decorator one is not in the transition model of their environment, but is in the decision problem that they have to solve. The difference in the decision problem is reflected in the difference in the reward functions and as a result in the value functions. It is desirable to have a model learning formalism that takes the decision problem, or at least some aspects of it, into account.

Furthermore, the implicit assumption that model approximation error can be made zero, that is $\mathcal{P}^*$ belongs to $\mathcal{M}$ used for estimation, may not be correct for many estimators. When we have the model approximation error, the model learning method must make a compromise in the choice of the estimate: None of the models in $\mathcal{M}$ would be the same as $\mathcal{P}^*$ (e.g., in the almost sure sense), so the estimation method has to choose a model with a minimum error with respect to (w.r.t.) some loss function. The choice of the loss function becomes important. A loss function that is designed for a particular decision problem in hand provides a better approximation, for the task of solving the very same decision problem, than a probabilistic one that does not take the decision problem into account.

These arguments suggest that generic distribution estimation approaches such as MLE, which minimizes the KL-divergence w.r.t. the empirical distribution, might not be the best

1. The state should also include robot's internal variables such as its joint parameters. One may also argue that we have to include some aspects of the past images in the state too. For the simplicity of the discussion, we only consider the current visual observation as the state of the agent.

candidate for learning a model to be used within a model-based RL framework. Can we design a better "decision-aware" loss function that takes the decision problem into account?

In the rest of this paper, we describe an approach that incorporates some aspects of the underlying decision problem into model learning. We go beyond the probabilistic losses in model learning and define a loss function that considers the structure of the value function. We call the approach that minimizes such a loss function *Value-Aware Model Learning (VAML)*. We also report a finite-sample error upper bound that shows the effect of the model approximation error and the number of training samples. This ensures the soundness of the algorithm that minimizes the proposed loss. Due to space limitation, we skip some of the derivations, all the proofs and the empirical results. These will be reported later.

2. Value-Aware Model Learning

Let $\mathsf{Planner}$ be an algorithm that receives a model $\hat{\mathcal{P}}$ and outputs a policy π, i.e., $\pi \leftarrow \mathsf{Planner}(\hat{\mathcal{P}})$. Here we assume that the reward function is already known to $\mathsf{Planner}$, so we do not explicitly pass it as an argument. For a user-defined initial probability distribution $\rho \in \bar{\mathcal{M}}(\mathcal{X})$, with $\bar{\mathcal{M}}(\mathcal{X})$ being the space of probability distributions defined on the state space $\mathcal{X}$, we evaluate the performance of π by

$$J(\pi) = \int \mathrm{d}\rho(x) V^{\pi}(x).$$

The goal of a successful model learner can be defined as follows: Given a dataset $\mathcal{D}_n = \{(X_i, A_i, X_i')\}_{i=1}^n$ with $Z_i = (X_i, A_i) \sim \nu(\mathcal{X} \times \mathcal{A}) \in \bar{\mathcal{M}}(\mathcal{X} \times \mathcal{A})$, potentially different from ρ, and $X_i' \sim \mathcal{P}^*(\cdot|X_i, A_i)$, find $\hat{\mathcal{P}}$ such that $J(\pi)$ for $\pi \leftarrow \mathsf{Planner}(\hat{\mathcal{P}})$ is as large as possible.

This is a very generic goal. To make it more concrete, we have to make a few choices. First suppose that $\mathsf{Planner}$ uses the Bellman optimality operator defined based on $\hat{\mathcal{P}}$ to find a $\hat{Q}^*$, that is $\hat{T}^* : Q \mapsto r + \gamma \hat{\mathcal{P}} \max_a Q$, and then outputs $\pi = \hat{\pi}(\cdot; \hat{Q}^*)$, the greedy policy w.r.t. $\hat{Q}^*$. The use of the Bellman [optimality] operator is central to value-based approaches such as the class of (Approximate) Value Iteration (Ernst et al., 2005; Munos and Szepesvári, 2008; Farahmand et al., 2009a; Mnih et al., 2015) and (Approximate) Policy Iteration (Lagoudakis and Parr, 2003; Lazaric et al., 2012; Farahmand et al., 2016) algorithms.

This is still a very general case, so we focus on the more specified goal of finding a $\hat{\mathcal{P}}$ such that the difference between T^*Q and $\hat{T}^*Q$ is not large. We may write this as

$$\begin{aligned} c(\hat{\mathcal{P}}, \mathcal{P}^*; V)(x,a) &= \left| \left[\mathcal{P}^*(\cdot|x,a) - \hat{\mathcal{P}}(\cdot|x,a) \right] V(\cdot) \right| = \left| \left\langle \, \mathcal{P}^*(\cdot|x,a) - \hat{\mathcal{P}}(\cdot|x,a) \, , \, V \, \right\rangle \right| \\ &= \left| \int \left[\mathcal{P}^*(\mathrm{d}x'|x,a) - \hat{\mathcal{P}}(\mathrm{d}x'|x,a) \right] V(x') \right|, \end{aligned} \tag{1}$$

in which we substituted $\max_a Q(\cdot, a)$ with V to simplify the presentation. The quantity $c(\hat{\mathcal{P}}, \mathcal{P}^*; V)(x,a)$ is the pointwise error of applying the Bellman operator based on the learned model on V compared to the true Bellman operator. We may sometimes use $\mathcal{P}_z(\cdot)$ with $z = (x,a) \in \mathcal{Z} = \mathcal{X} \times \mathcal{A}$ to refer to $\mathcal{P}(\cdot|x,a)$, so $\mathcal{P}_z V = \int \mathcal{P}(\mathrm{d}y|x,a) V(\mathrm{d}y)$.

It might be argued that since

$$\left| \left\langle \, \mathcal{P}^*(\cdot|x,a) - \hat{\mathcal{P}}(\cdot|x,a) \, , \, V \, \right\rangle \right| \le \| \mathcal{P}^*(\cdot|x,a) - \hat{\mathcal{P}}(\cdot|x,a) \|_1 \, \| V \|_\infty \, , \tag{2}$$

learning $\hat{\mathcal{P}}$ such that the ℓ_1-norm of its difference with the true $\mathcal{P}^*$ is small would be enough for having a model that is good to approximate the Bellman operator. This can be achieved by minimizing the KL divergence because Pinsker's inequality shows that for two probability distributions P_1 and P_2, we have

$$\|P_1 - P_2\|_1 \leq \sqrt{2\mathsf{KL}(P_1||P_2)}. \tag{3}$$

These two upper bounds together justify the use of MLE for model learning for RL problems since MLE is the minimizer of the empirical approximation of the KL divergence, as shown in Section 1. This is the argument, sometimes implicit, behind most model-based RL algorithms that use a log-loss or a similar "probabilistic" loss to estimate the model.

Finding a minimizer for the KL divergence, Hellinger distance, ℓ_1 loss, or other losses that depend only on the probability distributions, however, ignores the underlying decision problem, which is specified through the reward/value function. As an extreme example, suppose that $r(x) = c$ for all $x \in \mathcal{X}$, so V^π is constant for all policies, and the optimal policy would not have any preference over any of the actions. So even if $\mathcal{X}$ is a very large space (e.g., a subset of $\mathbb{R}^d$ with a large d), and however complex $\mathcal{P}^*$ is (e.g., the dynamics is not very regular), learning a $\hat{\mathcal{P}}$ sufficient to find the optimal policy is indeed very easy: Any transition probability distribution suffices to find the optimal policy. In contrast, $\|\hat{\mathcal{P}} - \mathcal{P}^*\|_1$ goes to zero at a convergence rate that depends on dimension d and regularities of $\mathcal{P}^*$, and can be very slow, e.g., $O(n^{-1/2d})$. An estimator for $\mathcal{P}^*$ that ignores this extra information requires more samples in order to provide a guarantee that the error in the model-based planning is small.[2] Moreover, and maybe more importantly, if the true transition kernel $\mathcal{P}^*$ does not belong to the model space $\mathcal{M}$ from which $\hat{\mathcal{P}}$ is estimated, we can only hope to find the "closest" model within $\mathcal{M}$ to $\mathcal{P}^*$. The notion of closeness, however, depends on the distance measure. A distance measure that explicitly takes into account the decision problem and what really matters for $\mathsf{Planner}$ is superior to the one that does not.

Returning to (1), there are three hurdles that should be addressed. The first is that $c(\hat{\mathcal{P}}, \mathcal{P}^*; V)(x, a)$ is defined as a pointwise measure of error, but we would like to learn a model that is valid for the whole state-action space $\mathcal{X} \times \mathcal{A}$. The second is that $\mathcal{P}^*$, which is the main object of interest, is not known after all. Instead we have $\mathcal{D}_n = \{(X_i, A_i, X_i')\}_{i=1}^n$ and as a result the empirical conditional distribution $\mathcal{P}_n(\cdot|x, a) = \frac{1}{n}\sum_{i=1}^n \delta_{X_i'|X_i,A_i}(\cdot|x, a)$. The third is that the loss is defined for a particular V but the model is going to be applied to a sequences of V that is not known a priori.

We can easily address the first concern by defining the cost functional (i.e., loss) as the expected squared pointwise loss w.r.t. a probability distribution $\nu \in \bar{\mathcal{M}}(\mathcal{X} \times \mathcal{A})$. To address the second concern, we might follow the usual recipe in machine learning and statistics, the empirical risk minimization, by replacing the true state transition kernel $\mathcal{P}^*$ with the observed empirical distribution $\mathcal{P}_n$ and $\nu \in \bar{\mathcal{M}}(\mathcal{X} \times \mathcal{A})$ with the empirical measure $\nu_n(\cdot) = \frac{1}{n}\sum_{i=1}^n \delta_{(X_i,A_i)}(\cdot)$.

To handle the problem of not knowing V, we suggest two approaches, the first one only briefly. One solution is to interleave the estimation of V and $\hat{\mathcal{P}}$: First choose $V_0 = r$, which

2. The relationship between the probabilistic loss and the LHS of (2) is a bit more subtle than what we portrayed here, but this should be enough for our current discussion.

is assumed to be known (or can be estimated). Given $\hat{V}_k$, estimate $\hat{\mathcal{P}}_{k+1}$ by minimizing

$$\hat{\mathcal{P}}_{k+1} \leftarrow \operatorname*{argmin}_{\hat{\mathcal{P}} \in \mathcal{M}} \sum_{(X_i, A_i) \in \mathcal{D}_n} \left| \hat{V}_k(X_i') - \int \hat{\mathcal{P}}(\mathrm{d}x' | X_i, A_i) \hat{V}_k(x') \right|^2 .$$

After finding $\hat{\mathcal{P}}_{k+1}$, we use a value-based Planner with the model $\hat{\mathcal{P}}_{k+1}$ to find a new $\hat{V}_{k+1}$, i.e., $\hat{V}_{k+1} \leftarrow \mathsf{Planner}(\hat{\mathcal{P}}_{k+1})$. This procedure is repeated. The idea is that the estimated $\hat{\mathcal{P}}_{k+1}$ would be enough to generate samples required for finding a good $\hat{V}_{k+1}$ if Planner is an approximate value iteration that applies $\hat{T}^*$ (based on $\hat{\mathcal{P}}_{k+1}$) only once. We do not study this approach any further in this work.

Another solution is to take a robust approach w.r.t. the choice of value function. We define the loss function to reflect the fact that our goal is to find a $\hat{\mathcal{P}}$ that is suitable for all V in a given value function space $\mathcal{F}$. Therefore, we define

$$c_{2,\nu}^2(\hat{\mathcal{P}}, \mathcal{P}^*) = \int \mathrm{d}\nu(x, a) \sup_{V \in \mathcal{F}} \left| \int \left[\mathcal{P}^*(\mathrm{d}x' | x, a) - \hat{\mathcal{P}}(\mathrm{d}x' | x, a) \right] V(x') \right|^2 . \tag{4}$$

To understand this loss better, let us focus on a single state-action pair (x, a) and study the pointwise loss. Note that even though

$$\sup_{V \in \mathcal{F}} \left| [\mathcal{P}^*(\cdot | x, a) - \hat{\mathcal{P}}(\cdot | x, a)] V(\cdot) \right| \le \left\| \hat{\mathcal{P}}(\cdot | x, a) - \mathcal{P}^*(\cdot | x, a) \right\|_1 \sup_{V \in \mathcal{F}} \|V\|_\infty , \tag{5}$$

the LHS is often much smaller than its upper bound for many choices of $\mathcal{F}$. They would only become equal when $\mathcal{F}$ is the space of bounded measurable functions, which is much larger than the usual function spaces that we often deal with, e.g., defined based on a finite set of basis functions.[3]

As the goal is to minimize the LHS of (5), and because its RHS can be a loose upper bound for most choices of $\mathcal{F}$, directly optimizing the LHS can lead to better models compared to minimizing the ℓ_1 loss or the KL distance (minimized by MLE), which itself is yet another level of upper bounding according to (3).

The loss function (4) reflects the influence of the value function on the model-learning objective. If we happen to know that V has certain regularities, e.g., it belongs to the Sobolev space $\mathbb{W}^k(\mathbb{R}^d)$ or a reproducing kernel Hilbert space, this loss function lets us focus on learning a model that can discriminate between such value functions, and not more.

After substituting the distributions with their empirical counterparts, we obtain

$$\begin{aligned} c_{2,n}^2(\hat{\mathcal{P}}) = c_{2,\mathrm{d}\nu_n}^2(\hat{\mathcal{P}}, \mathcal{P}_n) &= \frac{1}{n} \sum_{(X_i, A_i) \in \mathcal{D}_n} \sup_{V \in \mathcal{F}} \left| \int \left[\mathcal{P}_n(\mathrm{d}x' | X_i, A_i) - \hat{\mathcal{P}}(\mathrm{d}x' | X_i, A_i) \right] V(x') \right|^2 \\ &= \frac{1}{n} \sum_{(X_i, A_i) \in \mathcal{D}_n} \sup_{V \in \mathcal{F}} \left| V(X_i') - \int \hat{\mathcal{P}}(\mathrm{d}x' | X_i, A_i) V(x') \right|^2 . \end{aligned}$$

3. One might argue that it would be better to define the loss function as $c_{2,\nu}^2(\hat{\mathcal{P}}, \mathcal{P}^*) = \sup_{V \in \mathcal{F}} c_\nu(\hat{\mathcal{P}}, \mathcal{P}^*; V) = \sup_{V \in \mathcal{F}} \int \mathrm{d}\nu(x, a) \left| \int \left[\mathcal{P}^*(\mathrm{d}x' | x, a) - \hat{\mathcal{P}}(\mathrm{d}x' | x, a) \right] V(x') \right|^2$, that is, to have supremum over V outside the integral over state-actions. We do not pursue this path as it is not as computationally appealing as the current formulation. Investigating this alternative formulation is an interesting topic for future research.

The output of VAML is

$$\hat{\mathcal{P}} \leftarrow \operatorname*{argmin}_{\mathcal{P}\in\mathcal{M}} c_{2,n}^2(\mathcal{P}). \tag{6}$$

To completely specify the algorithm, we have to select $\mathcal{F}$ and $\mathcal{M}$. There are several possible choices, which in general should be informed by our prior knowledge about the underlying problem and guided by a model selection procedure. Due to space limitation, we only mention that one can obtain a gradient descent-based algorithm if we use linear function approximator for $\mathcal{F} = \mathcal{F}_B = \left\{ V_\theta(x) = \phi^\top(x)\theta : \theta \in \mathbb{R}^p, \|\theta\|_2 \le B \right\}$ with $\phi : \mathcal{X} \to \mathbb{R}^p$ being the value features, and the exponential families $\mathcal{M} = \left\{ \hat{P}_w : w \in \mathbb{R}^{p'} \right\}$ in which $\hat{P}_w$ is defined by model features $\phi' : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}^{p'}$ and the weight vector $w \in \mathbb{R}^{p'}$, i.e., $\hat{\mathcal{P}}_w(\mathrm{d}x'|x,a) = \frac{\exp\left(\phi'^\top(x'|x,a)w\right)}{\int \exp\left(\phi'^\top(x''|x,a)w\right)\mathrm{d}x''}\mathrm{d}x'$. Both of these choices are general and flexible. The use of linear function approximators is a common practice in the RL literature. The exponential families are also very general and can represent a wide variety of distributions (Canu and Smola, 2006; Sriperumbudur et al., 2014)

3. Statistical Analysis of VAML

We report a (simplified) finite error upper bound that shows that VAML is indeed a sound algorithm in the sense that it finds a $\hat{\mathcal{P}}$ that has a small error (4), given enough data points n and under standard capacity conditions on $\mathcal{M}$.

Theorem 1 *Given a dataset of i.i.d. samples $\mathcal{D}_n = \{(X_i, A_i, X_i')\}_{i=1}^n$ with $(X_i, A_i) \sim \nu$ and $X_i' \sim \mathcal{P}^*(\cdot|X_i, A_i)$, let $\hat{\mathcal{P}}$ be the solution of (6). Assume that the metric entropy of $\mathcal{M}$ w.r.t. the empirical L_2-norm satisfies $\log \mathcal{N}(u, \mathcal{M}, L_2(P_{z_{1:n}})) \le O(u^{-2\alpha})$. Furthermore, let $\mathcal{F}$ be a finite dimensional linear function space with bounded features. For any fixed $\delta > 0$, with probability at least $1-\delta$, we have*

$$\mathbb{E}\left[\sup_{V\in\mathcal{F}} \left|(\hat{\mathcal{P}}_Z - \mathcal{P}_Z^*)V\right|^2\right] \le \inf_{\mathcal{P}\in\mathcal{M}} \mathbb{E}\left[\sup_{V\in\mathcal{F}} |(\mathcal{P}_Z - \mathcal{P}_Z^*)V|^2\right] + O\left(\sqrt{\frac{\log(1/\delta)}{n}}\right).$$

This upper bound shows the usual model (or function) approximation error (first term) and the estimation error (second term). The estimation error of $O(n^{-1/2})$ is the usual behaviour of the supremum of the empirical processes for models that are not very large.

Maybe more interesting is the effect of the model approximation error. The bound shows that the error of the estimated $\hat{\mathcal{P}}$ is comparable to the error of the best choice in the model class $\mathcal{M}$, i.e., $\inf_{\mathcal{P}\in\mathcal{M}} \mathbb{E}\left[\sup_{V\in\mathcal{F}} |(\mathcal{P}_Z - \mathcal{P}_Z^*)V|^2\right]$. This is reassuring since VAML was motivated by the fact that the important property of an estimated model $\hat{\mathcal{P}}$ should be that $|(\hat{\mathcal{P}}_z - \mathcal{P}_z^*)V|$ is small only for $V \in \mathcal{F}$ that might be encountered by the algorithm.

An important question is how the model learning error affects the performance loss of the outcome policy $\pi \leftarrow \mathsf{Planner}(\hat{\mathcal{P}})$. We see that the terms appearing in the performance loss upper bound of a result such as Theorem 7 of Ávila Pires and Szepesvári (2016) is in a form closely related to the pointwise loss (1), whose L_2-norm is controlled by our theorem. The aforementioned result is for the supremum norm. Developing the L_p-norm type of result for the setup considered in this work is an interesting future research direction.

References

Bernardo Ávila Pires and Csaba Szepesvári. Policy error bounds for model-based reinforcement learning with factored linear models. In *Conference on Learning Theory (COLT)*, 2016.

Stephane Canu and Alex J. Smola. Kernel methods and the exponential family. *Neurocomputing*, 69(7-9):714–720, 2006.

Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussan. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, 2015.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 6:503–556, 2005.

Amir-massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. Regularized fitted Q-iteration for planning in continuous-space Markovian Decision Problems. In *Proceedings of American Control Conference (ACC)*, pages 725–730, June 2009a.

Amir-massoud Farahmand, Azad Shademan, Martin Jägersand, and Csaba Szepesvári. Model-based and model-free reinforcement learning for visual servoing. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2917–2924, May 2009b.

Amir-massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. Regularized policy iteration with nonparametric function spaces. *Journal of Machine Learning Research (JMLR)*, 17(139):1–66, 2016.

Todd Hester and Peter Stone. TEXPLORE: Real-time sample-efficient reinforcement learning for robots. *Machine Learning*, 90(3), 2013.

Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research (JMLR)*, 4:1107–1149, 2003.

Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Finite-sample analysis of least-squares policy iteration. *Journal of Machine Learning Research (JMLR)*, 13: 3041–3074, October 2012.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015.

Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research (JMLR)*, 9:815–857, 2008.

Bharath K. Sriperumbudur, Kenji Fukumizu, Revant Kumar, Arthur Gretton, and Aapo Hyvärinen. Density estimation in infinite dimensional exponential families. *arXiv:1312.3516v3*, 2014.

Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning (ICML)*, 1990.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Richard S. Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, 2008.

Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan Claypool Publishers, 2010.