Learning Policies for Markov Decision Processes from Data*

Manjesh K. Hanawal†, Hao Liu, Henghui Zhu,‡ and Ioannis Ch. Paschalidis§, *Fellow, IEEE*

***Abstract*—We consider the problem of learning a policy for a Markov decision process consistent with data captured on the state-actions pairs followed by the policy. We parameterize the policy using features associated with the state-action pairs. The features can be handcrafted or defined using kernel functions in a reproducing kernel Hilbert space. In either case, the set of features can be large and only a small, unknown subset may need to be used to fit a specific policy to the data. The parameters of such a policy are recovered using ℓ_1-regularized logistic regression. We establish bounds on the difference between the average reward of the estimated and the unknown original policy (regret) in terms of the generalization error and the ergodic coefficient of the underlying Markov chain. To that end, we combine sample complexity theory and sensitivity analysis of the stationary distribution of Markov chains. Our analysis suggests that to achieve regret within order $O(\sqrt{\epsilon})$, it suffices to use training sample size on the order of $\Omega(\log n \cdot poly(1/\epsilon))$, where n is the number of the features. We demonstrate the effectiveness of our method on a synthetic robot navigation example.**

***Index Terms*—Machine learning, Markov Decision Processes, reinforcement learning, regression.**

I. Introduction

MARKOV Decision Processes (MDPs) offer a framework for many dynamic optimization problems under uncertainty [1], [2]. When the state-action space is not large and transition probabilities for all state-action pairs are known, standard techniques such as policy iteration and value iteration can compute an optimal policy. More often than not, however, problem instances of realistic size have very large state-action spaces and it becomes impractical to know the transition probabilities everywhere and compute an optimal policy using these *off-line* methods.

For such large problems, one resorts to approximate methods, collectively referred to as *Approximate Dynamic Programming (ADP)* [3], [4]. ADP methods approximate either the value function and/or the policy and optimize with respect to the parameters, as for instance is done in *actor-critic* methods [5], [6], [7]. The optimization of approximation parameters requires the learner to have access to the system and be able to observe the effect of applied control actions.

In this paper, we adopt a different perspective and assume that the learner has no direct access to the system but only has samples of the actions applied in various states of the system. These actions are generated according to a policy that is fixed but unknown. We will call such a policy the target policy because it is the policy we wish to estimate. As an example, the actions could be followed by an expert player who plays an optimal policy. Our goal is to learn a policy consistent with the observed states-actions, which we will call *demonstrations*.

Learning from an expert is a problem that has been studied in the literature and referred to as apprenticeship learning [8], [9], imitation learning [10], learning from demonstrations [11], or learning parameterized skills [12]. While there are many settings where it could be useful, the main application driver has been robotics [13], [14], [15]. Additional interesting application examples include: learning from an experienced human pilot/driver to navigate vehicles autonomously, learning from animals to develop bio-inspired policies [16], and learning from expert players of a game to train a computer player. In all these examples, and given the size of the state-action space, we will not observe the actions of the expert in all states, or more broadly "scenarios" corresponding to parts of the state space leading to similar actions. Still, our goal is to learn a policy that generalizes well beyond the scenarios that have been observed and is able to select appropriate actions even at unobserved parts of the state space.

A plausible way to obtain such a policy is to learn a mapping of the states-actions to a lower dimensional space. Borrowing from ADP methods, we can obtain a lower-dimensional representation of the state-action space through the use of features that are functions of the state and the action taken at that state. In particular, we will consider policies that combine various features using a weight vector and reduce the problem of learning the target policy to learning this weight/parameter vector. Essentially, we will be learning a parsimonious parametrization of the target policy used by the expert.

The related work in the literature on learning through demonstrations can be broadly classified into *direct* and *indirect* methods [14]. In the direct methods, supervised learning techniques are used to obtain a best estimate of the expert's behavior; specifically, a best fit to the expert's behavior is

* Research partially supported by the NSF under grants DMS-1664644, CNS-1645681, CCF-1527292, and IIS-1237022, by the ARO under grant W911NF-12-1-0390, and by the ONR under grant MURI N00014-16-1-2832. M.K. Hanawal has been supported by an IIT Bombay IRCC SEED grant and an INSPIRE faculty fellowship (IFA-14/ENG-73) from DST, Government of India. Hao Liu has been supported by the Lin Guangzhao & the Hu Guozan Graduate Education International Exchange Fund.

† M. K. Hanawal is with Industrial Engineering and Operations Research, IIT-Bombay, Powai, MH 400076, India. E-mail: mhanawal@iitb.ac.in.

‡ H. Liu, H. Zhu are with the Center for Information and Systems Engineering, Boston University, Boston, MA 02215, USA. H. Liu is also with the College of Control Science and Engineering, Zhejiang University, Hangzhou, Zhejiang, 310027 China. E-mail: {haol, henghuiz}@bu.edu.

§ Corresponding author. I. Ch. Paschalidis is with the Department of Electrical and Computer Engineering and the Division of Systems Engineering, Boston University, Boston, MA 02215, USA. E-mail: yannisp@bu.edu, URL: http://sites.bu.edu/paschalidis/.

obtained by minimizing an appropriately defined loss function. A key limitation of this method is that the estimated policy is not well adapted to the parts of the state space not visited often by the expert, thus resulting in poor action selection if the system enters these states. Furthermore, the loss function can be non-convex in the policy, rendering the corresponding problem hard to solve.

Indirect methods, on the other hand, evaluate a policy by learning the full description of the MDP. In particular, one solves a so called *inverse reinforcement learning* problem which assumes that the dynamics of the environment are known but the one-step reward function is unknown. Then, one estimates the reward function that the expert is aiming to maximize through the demonstrations, and obtains the policy simply by solving the MDP. A policy obtained in this fashion tends to generalize better to the states visited infrequently by the expert, compared to policies obtained by direct methods. The main drawback of inverse reinforcement learning is that at each iteration it requires solving an MDP which is computationally expensive. In addition, the assumption that the dynamics of the environment are known for all states and actions is unrealistic for problems with very large state-action spaces.

In this work, we exploit the benefits of both direct and indirect methods by assuming that the expert is using a *Randomized Stationary Policy (RSP)*. As we alluded to earlier, an RSP is characterized in terms of a vector of features associated with state-action pairs and a parameter $\boldsymbol{\theta}$ weighing the various elements of the feature vector. We consider the case where we have many features, but only relatively few of them are sufficient to approximate the target policy well. However, we do not know in advance which features are relevant; learning them and the associated weights (elements of $\boldsymbol{\theta}$) is part of the learning problem.

We will use supervised learning to obtain the best estimate of the expert's policy. As in actor-critic methods, we use an RSP which is a parameterized "Boltzmann" policy and rely on an ℓ_1-regularized maximum likelihood estimator of the policy parameter vector. An ℓ_1-norm regularization induces sparse estimates and this is useful in obtaining an RSP which uses only the relevant features. In [17], it is shown that the sample complexity of ℓ_1-penalized logistic regression grows as $\mathcal{O}(\log n)$, where n is the number of features. As a result, we can learn the parameter vector $\boldsymbol{\theta}$ of the target RSP with relatively few samples, and the RSP we learn generalizes well across states that are not included in the demonstrations of the expert. Furthermore, ℓ_1-regularized logistic regression induces a convex optimization problem which can be solved efficiently.

The approach we presented requires that one specifies in advance the features used in defining an RSP, which gives rise to the question of how to select the appropriate features. This question of *feature engineering* is one that applies to all ADP methods. Many times, and depending on the specific application, one may have intuition on the type of functions that could be used as appropriate features, which allows for handcrafting specific features functions. Motivated by ideas in machine learning, also used in learning user behavior [18] and in earlier but different work on representing MDP policies [19], [20], we will also use kernel functions in a *Reproducing Kernel Hilbert Space (RKHS)* to define features. In the latter case, one only has to specify a kernel function and the logistic regression will determine an appropriate parameterization of the policy using feature functions in the RKHS.

A. Related Work

There is substantial work in the literature on learning MDP policies by observing experts; see [14] for a survey. We next discuss papers that are more closely related to our work.

In the context of indirect methods, [21] develops techniques for estimating a reward function from demonstrations under varying degrees of generality on the availability of an explicit model. [8] introduces an inverse reinforcement learning algorithm that obtains a policy from observed MDP trajectories followed by an expert. The policy is guaranteed to perform as well as that of the expert's policy, even though the algorithm does not necessarily recover the expert's reward function. In [9], the authors combine supervised learning and inverse reinforcement learning by fitting a policy to empirical estimates of the expert demonstrations over a space of policies that are optimal for a set of parameterized reward functions. [9] shows that the policy obtained generalizes well over the entire state space. [11] uses a supervised learning method to learn an optimal policy by leveraging the structure of the MDP, utilizing a kernel-based approach. A kernel-based policy representation is also used in [19], [20] but in a different way than in our setting. [12] proposes a way to learn policies corresponding to different tasks, where each task is being modeled as an MDP. Finally, [10] develops the DAGGER (Dataset Aggregation) algorithm that trains a deterministic policy which achieves good performance guarantees under its induced distribution of states.

Our work focuses on a parameterized set of policies rather than parameterized rewards. In this regard, our work is similar to approximate DP methods which parameterize the policy, e.g., expressing the policy as a linear functional in a lower dimensional parameter space. This lower-dimensional representation is critical in overcoming the well known "curse of dimensionality." Moreover, we establish explicit bounds on regret not available in earlier work that considers parameterizing policies, e.g., [12].

B. Contributions

We adopt the ℓ_1-regularized logistic regression to estimate a target RSP that generates a given collection of state-action samples. We evaluate the performance of the estimated policy and derive a bound on the difference between the average reward of the estimated RSP and the target RSP, typically referred to as *regret*. We show that a good estimate of the parameter of the target RSP implies an associated bound on the regret. To that end, we generalize a sample complexity result on the log-loss of the maximum likelihood estimates [17] from the case of two actions available at each state to the multi-action case. Using this result, we establish a sample complexity result on the regret.

Our analysis is based on the novel idea of separating the loss in average reward into two parts. The first part is due to the error in the policy estimation (training error) and the second part is due to the perturbation in the stationary distribution of the Markov chain caused by using the estimated policy instead of the true target policy (perturbation error). We bound the first part by relating the result on the log-loss error to the Kullback-Leibler divergence [22] between the estimated and the target RSPs. The second part is bounded using the ergodic coefficient of the induced Markov chain. Finally, we evaluate the performance of our method on a synthetic example.

A preliminary, conference version of the work in this paper has appeared in [23]. Compared to that paper which only considered the case where only two actions are available at each state, the present paper handles the general case with multiple potential actions at each state. Further, the present paper contains several proofs missing from [23], the kernel-based feature selection approach, and a larger case study.

The paper is organized as follows. In Section II, we introduce some of our notation and state the problem. In Section IV, we discuss how to define policy features. In Section III, we describe the supervised learning algorithm used to train the policy. In Section V, we establish a result on the (log-loss) error in policy estimation. In Section VI, we establish our main result, which is a bound on the regret of the estimated policy. In Section VIII, we introduce a robot navigation example and present our numerical results. We end with concluding remarks in Section IX.

Notational conventions. Bold letters are used to denote vectors and matrices; typically vectors are lower case and matrices upper case. Vectors are column vectors, unless explicitly stated otherwise. Prime denotes transpose. For the column vector $\mathbf{x} \in \mathbb{R}^n$ we write $\mathbf{x} = (x_1, \ldots, x_n)$ for economy of space, while $\|\mathbf{x}\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$ denotes its p-norm. Vectors or matrices with all zeroes are written as $\mathbf{0}$, the identity matrix as $\mathbf{I}$, and $\mathbf{e}$ is the vector with all entries set to 1. For any set $\mathcal{S}$, $|\mathcal{S}|$ denotes its cardinality. We use $\log$ to denote the natural logarithm and a subscript to denote different bases, e.g., $\log_2$ denotes logarithm with base 2.

II. Problem Formulation

Let $(\mathcal{X}, \mathcal{A}, \boldsymbol{P}, R)$ denote a Markov Decision Process (MDP) with a finite set of states $\mathcal{X}$ and a finite set of actions $\mathcal{A}$. For a state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$, let $P(y|x, a)$ denote the probability of transitioning to state $y \in \mathcal{X}$ after taking action a in state x. The function $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ denotes the one-step reward function.

A *Randomized Stationary Policy (RSP)* is a mapping from states to a discrete probability distribution; it specifies a probability distribution over the actions for each state. We consider a set of RSPs parameterized by vectors $\boldsymbol{\theta} \in \mathbb{R}^n$ and denote it as $\{\mu_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \mathbb{R}^n\}$. For a given parameter $\boldsymbol{\theta}$ and a state $x \in \mathcal{X}$, $\mu_{\boldsymbol{\theta}}(a|x)$ denotes the probability of taking action a at state x. Specifically, we consider the Boltzmann-type RSPs of the form

$$\mu_{\boldsymbol{\theta}}(a|x) = \frac{\exp\{\boldsymbol{\theta}' \boldsymbol{\phi}(x, a)\}}{\sum_{b \in \mathcal{A}} \exp\{\boldsymbol{\theta}' \boldsymbol{\phi}(x, b)\}}, \tag{1}$$

where $\boldsymbol{\phi} : \mathcal{X} \times \mathcal{A} \to [0,\, 1]^n$ is a vector of features associated with each state-action pair (x, a). The features are normalized to take values in $[0, 1]$. We can interpret the vector $-\boldsymbol{\phi}(x, a)$ as an energy function associated with action a in state x and view $\mu_{\boldsymbol{\theta}}(a|x)$ as assigning higher probability to low energy actions. Such policies are widely used to analyze MDPs with a large state-action space and where the transition probabilities are possibly unknown [24]. Henceforth, we identify an RSP by its parameter $\boldsymbol{\theta}$. We assume that the policy is *sparse*, that is, the vector $\boldsymbol{\theta}$ has only $r < n$ non-zero components and each is bounded by K, i.e., $|\theta_i| < K$ for all i. Given an RSP $\boldsymbol{\theta}$, the resulting transition probability matrix of the induced Markov chain is denoted by $\boldsymbol{P}_{\boldsymbol{\theta}}$, whose (x, y) element is $P_{\boldsymbol{\theta}}(y|x) = \sum_{a \in \mathcal{A}} \mu_{\boldsymbol{\theta}}(a|x) P(y|x, a)$ for all state pairs (x, y).

Notice that for any RSP $\boldsymbol{\theta}$, the sequence of states $\{X_k\}$ and the sequence of state-action pairs $\{X_k, A_k\}$ form a Markov chain with state space $\mathcal{X}$ and $\mathcal{X} \times \mathcal{A}$, respectively. We assume that for every $\boldsymbol{\theta}$, the Markov chains $\{X_k\}$ and $\{X_k, A_k\}$ are irreducible and aperiodic with stationary probabilities $\pi_{\boldsymbol{\theta}}(x)$ and $\eta_{\boldsymbol{\theta}}(x, a) = \pi_{\boldsymbol{\theta}}(x) \mu_{\boldsymbol{\theta}}(a|x)$, respectively.

The average reward function associated with an RSP $\boldsymbol{\theta}$ is a function $\overline{R} : \mathbb{R}^n \to \mathbb{R}$ defined as

$$\overline{R}(\boldsymbol{\theta}) = \sum_{(a,x)} \eta_{\boldsymbol{\theta}}(x, a) R(x, a). \tag{2}$$

Let now fix a target RSP $\boldsymbol{\theta}^*$. As we assumed above, $\boldsymbol{\theta}^*$ is sparse having at most r non-zero components θ_i^*, each satisfying $|\theta_i^*| \leq K$. This is simply the policy used by an expert (not necessarily optimal) which we wish to learn. We let $\mathcal{S}(\boldsymbol{\theta}^*) = \{(x_i, a_i) : i = 1, 2, \ldots, m\}$ denote a set of state-action samples generated by playing policy $\boldsymbol{\theta}^*$. The state samples $\{x_i : i = 1, 2, \ldots, m\}$ are independent and identically distributed (i.i.d.) drawn from the stationary distribution $\boldsymbol{\pi}_{\boldsymbol{\theta}^*}$ and a_i is the action taken in state x_i according to the policy $\mu_{\boldsymbol{\theta}^*}$. It follows that the samples in $\mathcal{S}(\boldsymbol{\theta}^*)$ are i.i.d. according to the distribution $\mathcal{D} \sim \eta_{\boldsymbol{\theta}^*}(x, a)$.

The i.i.d. assumption is needed to characterize the number of samples required to learn the target policy (cf. Sec. V) and is pervasive in the related machine learning literature, e.g., [17]. If under the RSP $\boldsymbol{\theta}^*$ the induced Markov chain converges fast to its stationary probability distribution (fast mixing), one can simply obtain nearly i.i.d. samples from the same trajectory by allowing some lag between consecutive samples. Alternatively, it is possible to sample from different trajectories starting from a different initial state.

We assume we have access only to the demonstrations $\mathcal{S}(\boldsymbol{\theta}^*)$ while the transition probability matrix $\boldsymbol{P}_{\boldsymbol{\theta}^*}$ and the target RSP $\boldsymbol{\theta}^*$ are unknown. The goal is to learn the target policy $\boldsymbol{\theta}^*$ and characterize the average reward obtained when the learned RSP is applied to the Markov process. In particular, we are interested in estimating the target parameter $\boldsymbol{\theta}^*$ from the samples efficiently and evaluate the performance of the estimated RSP with respect to the target RSP, i.e., bound the regret defined as

$$\mathrm{Reg}(\mathcal{S}(\boldsymbol{\theta}^*)) = \overline{R}(\boldsymbol{\theta}^*) - \overline{R}(\hat{\boldsymbol{\theta}}),$$

where $\hat{\boldsymbol{\theta}}$ is the estimated RSP from the samples $\mathcal{S}(\boldsymbol{\theta}^*)$.

III. Estimating the policy

Next we discuss how to estimate the target RSP $\boldsymbol{\theta}^*$ from the m i.i.d. state-action training samples in $\mathcal{S}(\boldsymbol{\theta}^*)$. Given the Boltzmann structure of the RSP we have assumed, we fit a logistic regression function using a regularized maximum likelihood estimator as follows:

$$\begin{array}{rl} \max_{\boldsymbol{\theta}\in\mathbb{R}^n} & \sum_{i=1}^m \log \mu_{\boldsymbol{\theta}}(a_i|x_i) \\ \text{s.t.} & \|\boldsymbol{\theta}\|_1 \leq B, \end{array} \tag{3}$$

where B is a parameter that adjusts the trade-off between fitting the training data "well" and obtaining a sparse RSP that generalizes well on a test sample.

We can evaluate how well the maximum likelihood function fits the samples in the logistic function using a log-loss metric, defined as the expected negative of the log-likelihood over the (random) test data. Formally, for any parameter $\boldsymbol{\theta}$, log-loss is given by

$$\epsilon(\boldsymbol{\theta}) = \mathbb{E}_{(x,a)\sim\mathcal{D}}[-\log \mu_{\boldsymbol{\theta}}(a|x)], \tag{4}$$

where the expectation is taken over state-action pairs (x, y) drawn from the distribution $\mathcal{D}$; recall that we defined $\mathcal{D}$ to be the stationary distribution $\eta_{\boldsymbol{\theta}^*}(x, a)$ of the state-action pairs induced by the policy $\boldsymbol{\theta}^*$. Since the expectation is taken with respect to new data not included in the training set $\mathcal{S}(\boldsymbol{\theta}^*)$, we can think of $\epsilon(\boldsymbol{\theta})$ as an *out-of-sample* metric of how well the RSP $\boldsymbol{\theta}$ approximates the actions taken by the target RSP $\boldsymbol{\theta}^*$.

We also define a sample-average version of log-loss: given any set $\mathcal{S} = \{(x_i, a_i) : i = 1, 2, \ldots, m\}$ of state-action pairs, define

$$\hat{\epsilon}_{\mathcal{S}}(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}(-\log \mu_{\boldsymbol{\theta}}(a_i|x_i)). \tag{5}$$

We will use the term *empirical* log-loss to refer to log-loss over the training set $\mathcal{S}(\boldsymbol{\theta}^*)$ and use the notation

$$\hat{\epsilon}(\boldsymbol{\theta}) \stackrel{\triangle}{=} \hat{\epsilon}_{\mathcal{S}(\boldsymbol{\theta}^*)}(\boldsymbol{\theta}). \tag{6}$$

To estimate an RSP, say $\hat{\boldsymbol{\theta}}$, from the training set, we adopt the logistic regression with an ℓ_1-norm regularization introduced in [17]. The specific steps are shown in Algorithm 1.

Algorithm 1 Training algorithm to estimate the target RSP $\boldsymbol{\theta}^*$ from the samples $\mathcal{S}(\boldsymbol{\theta}^*)$.

Initialization: Fix $0 < \gamma < 1$ and $C \geq rK$.
Split the training set $\mathcal{S}(\boldsymbol{\theta}^*)$ into two sets $\mathcal{S}_1$ and $\mathcal{S}_2$ of size $(1-\gamma)m$ and γm respectively. $\mathcal{S}_1$ is used for training and $\mathcal{S}_2$ for cross-validation.
Training:
for $B = 0, 1, 2, 4, \ldots, C$ **do**
  Solve the optimization problem (3) for each B on the set $\mathcal{S}_1$, and let $\boldsymbol{\theta}_B$ denote the optimal solution.
end for
Validation: Among the $\boldsymbol{\theta}_B$'s from the training step, select the one with the lowest "hold-out" error on $\mathcal{S}_2$, i.e, $\hat{B} = \arg\min_{B\in\{0,1,2,4\ldots,C\}} \hat{\epsilon}_{\mathcal{S}_2}(\boldsymbol{\theta}_B)$ and set $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_{\hat{B}}$.

IV. Selecting features

As we noted in the Introduction, [24] and most of the related literature take the feature function $\boldsymbol{\phi}$ as given. Features are typically set on a case-by-case basis, depending on the specific MDP setting and representing aspects of the state and action that are considered important in selecting an appropriate action for each state.

The potential obvious drawback of a fixed $\boldsymbol{\phi}$ is that it may introduce a large approximation error in representing the target policy. One possibility to address this issue is to use techniques as in [25] to jointly tune the features and the parameter vector $\boldsymbol{\theta}$ after we obtain an initial estimated RSP $\hat{\boldsymbol{\theta}}$.

An appealing alternative is to use feature functions defined in a *Reproducing Kernel Hilbert Space (RKHS)*. A similar strategy was used in [20], where features of just the state were represented in an RKHS. Here, we use features of both the state and the action. Let $\mathbf{y}(x, a) \in \mathcal{F} \subseteq \mathbb{R}^N$ be some representation (or encoding) of a state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$. As an example, if we represent the state x with some real vector $\mathbf{x} \in \mathbb{R}^N$, we could define $\mathbf{y}(x, a) = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}, a)$. (This is in fact the encoding we will use in the example we present in Section VIII.) Consider now a symmetric positive semidefinite kernel function $K : \mathcal{F}\times\mathcal{F} \to \mathbb{R}$, that is, satisfying

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j K(\mathbf{y}_i, \mathbf{y}_j) \geq 0,$$

for any choice of n, $\mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{R}^n$, and $\mathbf{y}_i \in \mathcal{F}$. Examples of kernels over $\mathbb{R}^N$ include the *linear* kernel $K(\mathbf{x}, \mathbf{y}) \equiv \mathbf{x}'\mathbf{y}$, the *polynomial* kernel $K(\mathbf{x}, \mathbf{y}) \equiv (c + \mathbf{x}'\mathbf{y})^d$ for some choice of $c \geq 0$ and $d \in \mathbb{N}$, and the *Gaussian* kernel $K(\mathbf{x}, \mathbf{y}) \equiv \exp(-c\|\mathbf{x} - \mathbf{y}\|_2^2)$ for some choice of $c > 0$.

Let now $K_{\mathbf{x}}(\cdot) \equiv K(\mathbf{x}, \cdot)$ denote the function of one variable obtained by fixing the first argument of K to $\mathbf{x}$ for any $\mathbf{x} \in \mathcal{F}$. Define $\mathcal{H}_0$ to be the vector space of scalar valued functions that can be written as finite linear combinations of elements $K_{\mathbf{x}}$ for some $\mathbf{x} \in \mathcal{F}$, i.e.,

$$\mathcal{H}_0 = \left\{ \sum_{j=1}^{n} \alpha_j K_{\mathbf{x}_j} : \mathbf{x}_j \in \mathcal{F}, n \in \mathbb{N}, \alpha_j \in \mathbb{R}, j = 1, \ldots, n \right\}.$$

For any $f, g \in \mathcal{H}_0$ such that $f = \sum_{j=1}^n \alpha_j K_{\mathbf{x}_j}$ and $g = \sum_{i=1}^n \beta_i K_{\mathbf{x}_i}$, we can define an inner product $\langle f, g\rangle_{\mathcal{H}_0} \equiv \sum_{i=1}^n \sum_{j=1}^n \alpha_i\beta_j K(\mathbf{x}_i, \mathbf{x}_j)$ and a norm $\|f\|_{\mathcal{H}_0} \equiv \sqrt{\langle f, f\rangle_{\mathcal{H}_0}}$. The closure $\mathcal{H}$ of $\mathcal{H}_0$ is called an RKHS and any $f \in \mathcal{H}$ that admits a finite representation in terms of kernel functions satisfies the *reproducing property*

$$\langle K_{\mathbf{x}}, f\rangle_{\mathcal{H}} = \sum_{j=1}^{n} \alpha_j K(\mathbf{x}, \mathbf{x}_j) = f(\mathbf{x}).$$

Using some kernel function $K(\cdot, \cdot)$, for some representative collection of state-action pairs (x_i, a_i) encoded as $\mathbf{y}(x_i, a_i)$, $i = 1, \ldots, n$, we can now define corresponding feature functions $\phi_i(x, a) = K(\mathbf{y}(x_i, a_i), \mathbf{y}(x, a))$ and a feature vector $\boldsymbol{\phi}(x, a) = (\phi_1(x, a), \ldots, \phi_n(x, a))$. This approach offers a way of generating a large number of features simply by selecting a kernel function and a collection of state-action pairs.

V. LOG-LOSS PERFORMANCE

In this section we establish a sample complexity result indicating that relatively few training samples (logarithmic in the dimension n of the RSP parameter $\boldsymbol{\theta}$) are needed to guarantee that the estimated RSP $\hat{\boldsymbol{\theta}}$ has out-of-sample log-loss close to the target RSP $\boldsymbol{\theta}^*$. We will use the line of analysis in [17] but generalize the result from the case where only two actions are available for every state to the general multi-action setting.

We start by relating the difference of the log-loss function associated with RSP $\boldsymbol{\theta}$ and its estimate $\hat{\boldsymbol{\theta}}$ to the relative entropy, or Kullback-Leibler (KL) divergence, between the corresponding RSPs. For a given $x \in \mathcal{X}$, we denote the KL-divergence between RSPs, $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ as $D(\boldsymbol{\mu}_{\boldsymbol{\theta}_1}(\cdot|x)\|\boldsymbol{\mu}_{\boldsymbol{\theta}_2}(\cdot|x))$, where $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\cdot|x)$ denotes the probability distribution induced by RSP $\boldsymbol{\theta}$ on $\mathcal{A}$ in state x, and is given as follows:

$$D(\boldsymbol{\mu}_{\boldsymbol{\theta}_1}(\cdot|x)\|\boldsymbol{\mu}_{\boldsymbol{\theta}_2}(\cdot|x)) = \sum_{a\in\mathcal{A}} \mu_{\boldsymbol{\theta}_1}(a|x) \log \frac{\mu_{\boldsymbol{\theta}_1}(a|x)}{\mu_{\boldsymbol{\theta}_2}(a|x)}.$$

We also define the average KL-divergence, denoted by $D_{\boldsymbol{\theta}}(\boldsymbol{\mu}_{\boldsymbol{\theta}_1}\|\boldsymbol{\mu}_{\boldsymbol{\theta}_2})$, as the average of $D(\boldsymbol{\mu}_{\boldsymbol{\theta}_1}(\cdot|x)\|\boldsymbol{\mu}_{\boldsymbol{\theta}_2}(\cdot|x))$ over states visited according to the stationary distribution $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ of the Markov chain induced by policy $\boldsymbol{\theta}$. Specifically, we define

$$D_{\boldsymbol{\theta}}(\boldsymbol{\mu}_{\boldsymbol{\theta}_1}\|\boldsymbol{\mu}_{\boldsymbol{\theta}_2}) = \sum_{x\in\mathcal{X}} \pi_{\boldsymbol{\theta}}(x) D(\boldsymbol{\mu}_{\boldsymbol{\theta}_1}(\cdot|x)\|\boldsymbol{\mu}_{\boldsymbol{\theta}_2}(\cdot|x)). \quad (7)$$

Lemma V.1 *Let $\hat{\boldsymbol{\theta}}$ be an estimate of RSP $\boldsymbol{\theta}$. Then,*

$$\epsilon(\hat{\boldsymbol{\theta}}) - \epsilon(\boldsymbol{\theta}) = D_{\boldsymbol{\theta}}(\boldsymbol{\mu}_{\boldsymbol{\theta}}\|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}).$$

Proof:

$$\begin{aligned}
&\epsilon(\hat{\boldsymbol{\theta}}) - \epsilon(\boldsymbol{\theta}) \\
&= \sum_{x\in\mathcal{X}, a\in\mathcal{A}} \eta_{\boldsymbol{\theta}}(x,a)[\log \mu_{\boldsymbol{\theta}}(a|x) - \log \mu_{\hat{\boldsymbol{\theta}}}(a|x)] \\
&= \sum_{x\in\mathcal{X}} \pi_{\boldsymbol{\theta}}(x) \sum_{a\in\mathcal{A}} \mu_{\boldsymbol{\theta}}(a|x) \log \frac{\mu_{\boldsymbol{\theta}}(a|x)}{\mu_{\hat{\boldsymbol{\theta}}}(a|x)} \\
&= \sum_{x\in\mathcal{X}} \pi_{\boldsymbol{\theta}}(x) D(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\cdot|x)\|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}(\cdot|x)) \\
&= D_{\boldsymbol{\theta}}(\boldsymbol{\mu}_{\boldsymbol{\theta}}\|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}).
\end{aligned}$$

■

For the case of a binary action at every state, [17] showed the following result. To state the theorem, let $poly(\cdot)$ denote a function which is polynomial in its arguments and recall that m is the number of state-action pairs in the training set $\mathcal{S}(\boldsymbol{\theta}^*)$ used to learn $\hat{\boldsymbol{\theta}}$.

Theorem V.2 ([17], Thm. 3.1) *Suppose action set $\mathcal{A}$ contains only two actions, i.e., $\mathcal{A} = \{0, 1\}$. Let $\epsilon > 0$ and $\delta > 0$. In order to guarantee that, with probability at least $1 - \delta$, $\hat{\boldsymbol{\theta}}$ produced by the algorithm performs as well as $\boldsymbol{\theta}^*$, i.e.,*

$$D_{\boldsymbol{\theta}^*}(\boldsymbol{\mu}_{\boldsymbol{\theta}^*}\|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}) \leq \epsilon,$$

it suffices that $m = \Omega((\log n) \cdot poly(r, K, C, \log(1/\delta), 1/\epsilon))$.

We will generalize the result to the case when more than two actions are available at each state. We assume that $|\mathcal{A}| = H$, that is, at most H actions are available at each state. By introducing in (1) features that get activated at specific states, it is possible to accommodate MDPs where some of the actions are not available at these states. We establish the following key theorem, whose proof is provided in the Appendix.

Theorem V.3 *Let $\varepsilon > 0$ and $\delta > 0$. In order to guarantee that, with probability at least $1-\delta$, $\hat{\boldsymbol{\theta}}$ produced by Algorithm 1 performs as well as $\boldsymbol{\theta}^*$, i.e.,*

$$|\epsilon(\hat{\boldsymbol{\theta}}) - \epsilon(\boldsymbol{\theta}^*)| < \varepsilon, \quad (8)$$

it suffices that

$$m = \Omega\Big((\log n) \cdot poly(r,\, K,\, C,\, H,\, \log(1/\delta),\, 1/\varepsilon)\Big).$$

Furthermore, in terms of only H, $m = \Omega(H^3)$.

VI. BOUNDS ON REGRET

Theorem V.3 provides a sufficient condition on the number of samples required to learn a policy whose log-loss performance is close to the target policy. In this section we study the regret of the estimated policy, defined in Sec. II as the difference between the average reward of the target policy and the estimated policy. Given that we use a number of samples in the training set proportional to the expression provided in Theorem V.3, we establish explicit bounds on the regret.

We will bound the regret of the estimated policy by separating the effect of the error in estimating the policy function (which is characterized by Theorem V.3) and the effect the estimated policy function introduces in the stationary distribution of the Markov chain governing how states are visited. To bound the regret due to the perturbation of the stationary distribution, we will use results from the sensitivity analysis of Markov chains. In Section VI-A we provide some standard definitions for Markov chains and state our result on the regret, while in Section VII we provide a proof of this result.

A. Main Result

We start by defining the fundamental matrix of a Markov chain.

Definition 1
The fundamental matrix of a Markov chain with state transition probability matrix $\boldsymbol{P}_{\boldsymbol{\theta}}$ induced by RSP $\boldsymbol{\theta}$ is

$$\boldsymbol{Z}_{\boldsymbol{\theta}} = (\boldsymbol{A}_{\boldsymbol{\theta}} + \boldsymbol{e}\boldsymbol{\pi}'_{\boldsymbol{\theta}})^{-1},$$

where $\boldsymbol{e}$ denotes the vector of all 1's, $\boldsymbol{A}_{\boldsymbol{\theta}} = \boldsymbol{I} - \boldsymbol{P}_{\boldsymbol{\theta}}$ and $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ denotes the stationary distribution associated with $\boldsymbol{P}_{\boldsymbol{\theta}}$. Also, the group inverse of $\boldsymbol{A}_{\boldsymbol{\theta}}$ denoted as $\mathbf{A}_{\boldsymbol{\theta}}^{\#}$ is the unique matrix satisfying

$$\boldsymbol{A}_{\boldsymbol{\theta}}\boldsymbol{A}_{\boldsymbol{\theta}}^{\#}\boldsymbol{A}_{\boldsymbol{\theta}} = \boldsymbol{A}_{\boldsymbol{\theta}},\ \boldsymbol{A}_{\boldsymbol{\theta}}^{\#}\boldsymbol{A}_{\boldsymbol{\theta}}\boldsymbol{A}_{\boldsymbol{\theta}}^{\#} = \boldsymbol{A}_{\boldsymbol{\theta}},\ \boldsymbol{A}_{\boldsymbol{\theta}}\boldsymbol{A}_{\boldsymbol{\theta}}^{\#} = \boldsymbol{A}_{\boldsymbol{\theta}}^{\#}\boldsymbol{A}_{\boldsymbol{\theta}}.$$

Most of the properties of a Markov chain can be expressed in terms of the fundamental matrix $\boldsymbol{Z}$. For example, if $\boldsymbol{\pi}_1$ is the stationary probability distribution associated with a Markov

chain whose transition probability matrix is $\boldsymbol{P}_1$ and if $\boldsymbol{P}_2 = \boldsymbol{P}_1 - \boldsymbol{E}$ for some perturbation matrix $\boldsymbol{E}$, then the stationary probability distribution $\boldsymbol{\pi}_2$ of the Markov chain with transition probability matrix $\boldsymbol{P}_2$ satisfies the relation

$$\boldsymbol{\pi}_1' - \boldsymbol{\pi}_2' = \boldsymbol{\pi}_2' \boldsymbol{E} \boldsymbol{Z}_{\boldsymbol{1}}, \tag{9}$$

where $\boldsymbol{Z}_1$ is the fundamental matrix associated with $\boldsymbol{P}_1$.

Definition 2
The ergodic coefficient of a matrix $\boldsymbol{B}$ with equal row sums is

$$\tau(\boldsymbol{B}) = \sup_{\mathbf{v}'\mathbf{e}=0; \|\mathbf{v}\|_1=1} \|\mathbf{v}'\boldsymbol{B}\|_1 = \frac{1}{2} \max_{i,j} \sum_s |b_{is} - b_{js}|. \tag{10}$$

The ergodic coefficient of a Markov chain indicates sensitivity of its stationary distribution. For any stochastic matrix $\boldsymbol{P}$, $0 \leq \tau(\boldsymbol{P}) \leq 1$.

We now have all the ingredients to state our main result bounding regret.

Theorem VI.1 *Given $\epsilon > 0$ and $\delta > 0$, suppose $m = \Omega((\log n) \cdot poly(r, K, C, H, \log(1/\delta), 1/\epsilon))$ i.i.d. samples are used by Algorithm 1 to produce an estimate $\hat{\boldsymbol{\theta}}$ the unknown target RSP policy parameter $\boldsymbol{\theta}^*$. Then with probability at least $1 - \delta$, we have*

$$|\overline{R}(\boldsymbol{\theta}^*) - \overline{R}(\hat{\boldsymbol{\theta}})| \leq \sqrt{2\epsilon \log 2} R_{\max}(1 + \kappa).$$

where

$$R_{\max} = \max_{(x,a) \in \mathcal{X} \times \mathcal{A}} |R(x, a)|$$

and κ is a constant that depends on the RSP $\hat{\boldsymbol{\theta}}$ and can be any of the following:

- $\kappa = \|\boldsymbol{Z}_{\hat{\boldsymbol{\theta}}}\|_\infty$,
- $\kappa = \|\boldsymbol{A}^{\#}_{\hat{\boldsymbol{\theta}}}\|_\infty$,
- $\kappa = 1/(1 - \tau(\boldsymbol{P}_{\hat{\boldsymbol{\theta}}}))$,
- $\kappa = \tau(\boldsymbol{Z}_{\hat{\boldsymbol{\theta}}}) = \tau(\boldsymbol{A}^{\#}_{\hat{\boldsymbol{\theta}}})$.

The constant κ is referred to as *condition number*. The regret is thus governed by the condition number of the estimated RSP; the smaller the condition number of the trained policy, the smaller is the regret.

Among different values of κ, setting $\kappa = \tau(\boldsymbol{Z}_{\hat{\boldsymbol{\theta}}})$ gives the tightest upper bound [26][Lemma 4.1]. Further, when the state space is finite, the value of $\tau(\boldsymbol{Z}_{\hat{\boldsymbol{\theta}}})$ can be bounded as follows

$$\frac{1}{\min_i |1 - \lambda_i|} \leq \tau(\boldsymbol{Z}_{\hat{\boldsymbol{\theta}}}) \leq \frac{\text{card}(\mathcal{X})}{\min_i |1 - \lambda_i|},$$

where $\text{card}(\mathcal{X})$ denotes the cardinality of the set $\mathcal{X}$ and λ_i, $i = 1, \ldots, \text{card}(\mathcal{X})$, are the eigenvalues of the transition probability matrix induced by the RSP $\hat{\boldsymbol{\theta}}$. Thus, when the eigenvalues of the transition probability matrix of the estimated RSP are not close to 1, the Markov chain is well-conditioned and the regret is small.

VII. Proof of the Main Result

In this section we analyze the average reward obtained by the MDP when we apply the estimated RSP $\hat{\boldsymbol{\theta}}$, and prove the regret bound in Theorem VI.1. First, we bound the regret as the sum of two parts.

$$\begin{aligned}
&\text{Reg}(\mathcal{S}(\boldsymbol{\theta}^*)) = \overline{R}(\boldsymbol{\theta}^*) - \overline{R}(\hat{\boldsymbol{\theta}}) \\
&= \sum_x \sum_a [\eta_{\boldsymbol{\theta}^*}(x, a) - \eta_{\hat{\boldsymbol{\theta}}}(x, a)] R(x, a) \\
&= \sum_x \pi_{\boldsymbol{\theta}^*}(x) \sum_a \mu_{\boldsymbol{\theta}^*}(a|x) R(x, a) \\
&\quad - \sum_x \pi_{\hat{\boldsymbol{\theta}}}(x) \sum_a \mu_{\hat{\boldsymbol{\theta}}}(a|x) R(x, a) \\
&= \sum_x \pi_{\boldsymbol{\theta}^*}(x) \sum_a [\mu_{\boldsymbol{\theta}^*}(a|x) - \mu_{\hat{\boldsymbol{\theta}}}(a|x)] R(x, a) \\
&\quad - \sum_x [\pi_{\hat{\boldsymbol{\theta}}}(x) - \pi_{\boldsymbol{\theta}^*}(x)] \sum_a \mu_{\hat{\boldsymbol{\theta}}}(a|x) R(x, a) \\
&\leq \left| \sum_x \pi_{\boldsymbol{\theta}^*}(x) \sum_a [\mu_{\boldsymbol{\theta}^*}(a|x) - \mu_{\hat{\boldsymbol{\theta}}}(a|x)] R(x, a) \right| \\
&\quad + \left| \sum_x [\pi_{\hat{\boldsymbol{\theta}}}(x) - \pi_{\boldsymbol{\theta}^*}(x)] \sum_a \mu_{\hat{\boldsymbol{\theta}}}(a|x) R(x, a) \right|. \qquad (11)
\end{aligned}$$

Note that the first absolute sum above has terms $\sum_a [\mu_{\boldsymbol{\theta}^*}(a|x) - \mu_{\hat{\boldsymbol{\theta}}}(a|x)]$ for all x that are related to the estimation error from fitting the RSP policy $\hat{\boldsymbol{\theta}}$ to $\boldsymbol{\theta}^*$. The second part has terms $\sum_x |\pi_{\hat{\boldsymbol{\theta}}}(x) - \pi_{\boldsymbol{\theta}^*}(x)|$ that are related to the perturbation of the stationary distribution of the Markov chain by applying the fitted RSP $\hat{\boldsymbol{\theta}}$ instead of the original $\boldsymbol{\theta}^*$. In the following, we bound each term separately. We begin with the first term.

We have:

$$\begin{aligned}
&\left| \sum_x \pi_{\boldsymbol{\theta}^*}(x) \sum_a [\mu_{\boldsymbol{\theta}^*}(a|x) - \mu_{\hat{\boldsymbol{\theta}}}(a|x)] R(x, a) \right| \\
&\leq \sum_x \pi_{\boldsymbol{\theta}^*}(x) \sum_a |\mu_{\boldsymbol{\theta}^*}(a|x) - \mu_{\hat{\boldsymbol{\theta}}}(a|x)| \cdot |R(x, a)| \\
&\leq R_{\max} \sum_x \pi_{\boldsymbol{\theta}^*}(x) \|\boldsymbol{\mu}_{\boldsymbol{\theta}^*}(\cdot|x) - \boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}(\cdot|x)\|_1, \qquad (12)
\end{aligned}$$

where $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\cdot|x)$ denotes the probability distribution (a vector) on the action space $\mathcal{A}$ induced by the RSP $\boldsymbol{\theta}$ at state x.

The bound in (12) is related to the difference in the log-loss between the RSPs $\boldsymbol{\theta}^*$ and $\hat{\boldsymbol{\theta}}$. To see this, we need the following result that connects the ℓ_1 distance between two distributions with their KL-divergence. Let $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ denote two probability vectors on $\mathcal{A}$. From the variation distance characterization of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ we have the following lemma.

Lemma VII.1 *([22, Lemma 11.6.1])*

$$D(\boldsymbol{p}_1 \| \boldsymbol{p}_2) \geq \frac{1}{2 \log 2} \|\boldsymbol{p}_1 - \boldsymbol{p}_2\|_1^2. \tag{13}$$

Continuing the chain of inequalities from (12), we obtain

$$\begin{aligned}
&\left|\sum_x \pi_{\boldsymbol{\theta}^*}(x)\sum_a[\mu_{\boldsymbol{\theta}^*}(a|x)-\mu_{\hat{\boldsymbol{\theta}}}(a|x)]R(x,a)\right| \\
&\quad\leq\ R_{\max}\sum_x \pi_{\boldsymbol{\theta}^*}(x)\sqrt{2\log 2 D\left(\boldsymbol{\mu}_{\boldsymbol{\theta}^*}(\cdot|x)\|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}(\cdot|x)\right)} \\
&\quad\leq\ \sqrt{2\log 2}R_{\max}\cdot \\
&\qquad\sqrt{\sum_x \pi_{\boldsymbol{\theta}^*}(x) D\left(\boldsymbol{\mu}_{\boldsymbol{\theta}^*}(\cdot|x)\|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}(\cdot|x)\right)} \\
&\quad=\ \sqrt{2\log 2}R_{\max}\sqrt{D_{\boldsymbol{\theta}^*}(\boldsymbol{\mu}_{\boldsymbol{\theta}^*}\|\boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}})}. \qquad (14)
\end{aligned}$$

In the first inequality, we applied Lemma VII.1 by setting $\boldsymbol{p}_1 = \boldsymbol{\mu}_{\boldsymbol{\theta}^*}(\cdot|x)$ and $\boldsymbol{p}_2 = \boldsymbol{\mu}_{\hat{\boldsymbol{\theta}}}(\cdot|x)$ for each x. In second inequality, we applied Jensen's inequality. We can now use Theorem V.3 to bound (14).

We next bound the second term in (11) using techniques from perturbation analysis of eigenvalues of a matrix. We have

$$\begin{aligned}
&\left|\sum_x(\pi_{\hat{\boldsymbol{\theta}}}(x)-\pi_{\boldsymbol{\theta}^*}(x))\sum_a \mu_{\hat{\boldsymbol{\theta}}}(a|x)R(x,a)\right| \\
&\quad\leq\ \sum_x|\pi_{\hat{\boldsymbol{\theta}}}(x)-\pi_{\boldsymbol{\theta}^*}(x)|\sum_a|\mu_{\hat{\boldsymbol{\theta}}}(a|x)R(x,a)| \\
&\quad\leq\ R_{\max}\sum_x|\pi_{\hat{\boldsymbol{\theta}}}(x)-\pi_{\boldsymbol{\theta}^*}(x)|\sum_a\mu_{\hat{\boldsymbol{\theta}}}(a|x) \\
&\quad=\ R_{\max}\sum_x|\pi_{\hat{\boldsymbol{\theta}}}(x)-\pi_{\boldsymbol{\theta}^*}(x)|, \qquad (15)
\end{aligned}$$

where we used the definition of $R_{\max}$ in the second inequality and the last equality follows by noting that $\sum_a \mu_{\hat{\boldsymbol{\theta}}}(a|x)=1$ for all x. To further bound the difference in stationary distributions in (15), we use a relation between the perturbation of the stationary distribution and the condition number of the Markov chain. We recall the following result that is useful to bound the difference between the stationary distribution induced by the optimal RSP and that induced by the estimated RSP in terms of the condition number of the Markov chain.

Lemma VII.2 ([27], [26]) *Let $\boldsymbol{\pi}_{\boldsymbol{\theta}_1}$ and $\boldsymbol{\pi}_{\boldsymbol{\theta}_2}$ be the unique stationary distributions of the stochastic matrices $\boldsymbol{P}_{\boldsymbol{\theta}_1}$ and $\boldsymbol{P}_{\boldsymbol{\theta}_2}$, respectively. Let $\boldsymbol{E}=\boldsymbol{P}_{\boldsymbol{\theta}_1}-\boldsymbol{P}_{\boldsymbol{\theta}_2}$. Then,*

$$\|\boldsymbol{\pi}_{\boldsymbol{\theta}_1}-\boldsymbol{\pi}_{\boldsymbol{\theta}_2}\|_1 \leq \kappa\|\boldsymbol{\pi}'_{\boldsymbol{\theta}_1}\boldsymbol{E}\|_1, \qquad (16)$$

where κ is a constant that can take the following values

- $\kappa = \|\boldsymbol{Z}_{\boldsymbol{\theta}_2}\|_\infty$,
- $\kappa = \|\boldsymbol{A}^{\#}_{\boldsymbol{\theta}_2}\|_\infty$,
- $\kappa = 1/(1-\tau(\boldsymbol{P}_{\boldsymbol{\theta}_2}))$,
- $\kappa = \tau(\boldsymbol{Z}_{\boldsymbol{\theta}_2}) = \tau(\boldsymbol{A}^{\#}_{\boldsymbol{\theta}_2})$.

Proof: The proof follows by setting $\boldsymbol{\pi}_1 = \boldsymbol{\pi}_{\boldsymbol{\theta}_2}$ and $\boldsymbol{\pi}_2 = \boldsymbol{\pi}_{\boldsymbol{\theta}_1}$ in (9) and using the relation

$$\|\boldsymbol{\pi}_{\boldsymbol{\theta}_1}-\boldsymbol{\pi}_{\boldsymbol{\theta}_2}\|_1 = \|\boldsymbol{\pi}'_{\boldsymbol{\theta}_1}\boldsymbol{E}\boldsymbol{Z}_{\boldsymbol{\theta}_2}\|_1 \leq \|\boldsymbol{\pi}'_{\boldsymbol{\theta}_1}\mathbf{E}\|_1\|\boldsymbol{Z}_{\boldsymbol{\theta}_2}\|_1.$$

The other relations follow similarly from Sec. 3 of [26]. ∎

Continuing the chain of inequalities in (15) and applying Lemma VII.2 by setting $\pi_{\boldsymbol{\theta}_1}=\pi_{\boldsymbol{\theta}^*}$ and $\pi_{\boldsymbol{\theta}_2}=\pi_{\hat{\boldsymbol{\theta}}}$, we obtain

$$\begin{aligned}
&\left|\sum_x(\pi_{\hat{\boldsymbol{\theta}}}(x)-\pi_{\boldsymbol{\theta}^*}(x))\sum_a\mu_{\hat{\boldsymbol{\theta}}}(a|x)R(x,a)\right| \\
&\qquad\leq R_{\max}\kappa\|\boldsymbol{\pi}'_{\boldsymbol{\theta}^*}(\boldsymbol{P}_{\boldsymbol{\theta}^*}-\boldsymbol{P}_{\hat{\boldsymbol{\theta}}})\|_1, \qquad (17)
\end{aligned}$$

where, with some overloading of notation, κ is now as specified in the expressions provided in the statement of Theorem VI.1.

The ith component of the vector $\boldsymbol{\pi}'_{\boldsymbol{\theta}^*}(\boldsymbol{P}_{\boldsymbol{\theta}^*}-\boldsymbol{P}_{\hat{\boldsymbol{\theta}}})$ is given by

$$\begin{aligned}
&[\boldsymbol{\pi}'_{\boldsymbol{\theta}^*}(\boldsymbol{P}_{\boldsymbol{\theta}^*}-\boldsymbol{P}_{\hat{\boldsymbol{\theta}}})]_i \\
&\quad=\ \sum_x \pi_{\boldsymbol{\theta}^*}(x)[\boldsymbol{P}_{\boldsymbol{\theta}^*}(i|x)-\boldsymbol{P}_{\hat{\boldsymbol{\theta}}}(i|x)] \\
&\quad=\ \sum_x \pi_{\boldsymbol{\theta}^*}(x)\sum_a[\boldsymbol{P}(i|x,a)(\mu_{\boldsymbol{\theta}^*}(a|x)-\mu_{\hat{\boldsymbol{\theta}}}(a|x))],
\end{aligned}$$

where, in the last equality, we applied the definition of the transition probability $\boldsymbol{P}_{\boldsymbol{\theta}}$ associated with RSP $\boldsymbol{\theta}$.

It follows

$$\begin{aligned}
&\|\boldsymbol{\pi}'_{\boldsymbol{\theta}^*}(\boldsymbol{P}_{\boldsymbol{\theta}^*}-\boldsymbol{P}_{\hat{\boldsymbol{\theta}}})\|_1 \\
&=\sum_y\left|\sum_x\pi_{\boldsymbol{\theta}^*}(x)\sum_a[\boldsymbol{P}(y|x,a)(\mu_{\boldsymbol{\theta}^*}(a|x)-\mu_{\hat{\boldsymbol{\theta}}}(a|x))]\right| \\
&\leq\sum_x\pi_{\boldsymbol{\theta}^*}(x)\sum_y\sum_a\left|[\boldsymbol{P}(y|x,a)(\mu_{\boldsymbol{\theta}^*}(a|x)-\mu_{\hat{\boldsymbol{\theta}}}(a|x))]\right| \\
&\leq\sum_x\pi_{\boldsymbol{\theta}^*}(x)\sum_a\left|\mu_{\boldsymbol{\theta}^*}(a|x)-\mu_{\hat{\boldsymbol{\theta}}}(a|x)\right|, \qquad (18)
\end{aligned}$$

where the last inequality follows by noting that $\sum_y \boldsymbol{P}(y|x,a)=1$ for all (x,a).

Now, using (18), (15), similar steps as in (14), and Theorem V.3, we can bound the second term in (11) as

$$\begin{aligned}
&\left|\sum_x(\pi_{\hat{\boldsymbol{\theta}}}(x)-\pi_{\boldsymbol{\theta}^*}(x))\sum_a\mu_{\hat{\boldsymbol{\theta}}}(a|x)R(x,a)\right| \\
&\qquad\leq\sqrt{2\epsilon\log 2}\ \kappa\ R_{\max}. \qquad (19)
\end{aligned}$$

Finally, combining (14) and (19) and applying Theorem V.3, the result in Theorem VI.1 follows.

VIII. A ROBOT NAVIGATION EXAMPLE

In this section, we discuss the experimental setup to simulate an MDP and validate the effectiveness of our proposed learning algorithm. We consider the problem of learning the policy used by an agent (a robot) as it moves on a 2-dimensional grid. After simulating the movement of the robot and recording its actions in various states, we use these states-action samples to learn the policy of the robot and evaluate its performance.

A. Environment and Agent Settings

Consider an agent moving in a 21×21 grid, shown in Fig. 1. The agent's position is specified by a two-tuple *state* $\mathbf{x}=(x_1,x_2)$, representing its coordinates in the grid. We assume $(0,0)$ is at the southwest corner of the grid and we make the

convention that coordinates $\mathbf{x} = (x_1, x_2)$ on the grid identify the square defined by the four points (x_1, x_2), $(x_1 + 1, x_2)$, $(x_1, x_2 + 1)$, and $(x_1 + 1, x_2 + 1)$. For example, when we say that the agent is at $\mathbf{x} = (x_1, x_2)$ we mean that the agent can be anywhere in the above square.

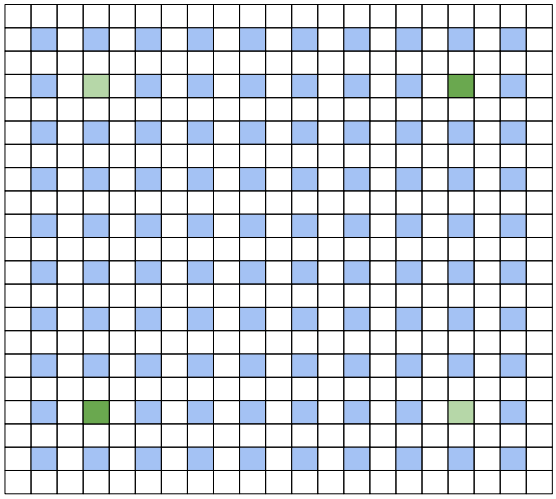

Fig. 1. The environment of the MDP is a 21×21 grid. Colored (or shaded) grid squares correspond to waypoints (identified by the southwest vertex of the square) for defining features that are used by the estimated policy. The four squares colored green correspond to waypoints on the grid with an associated reward. The one step reward function of the MDP is a weighted sum of Gaussian functions specified by these reward points on the grid.

At each time instance, the agent can take 4 actions: `North`, `East`, `West`, and `South`. Without loss of generality, we assume that the agent can only move to a neighboring grid point. The destination of the agent is based on its current position and the action. We assume that the agent movements are subject to uncertainty, which can cause the agent's intended next position to shift to a point adjacent to that position. For example, the agent in state (x_1, x_2) taking action `North` will enter state $(x_1, x_2 + 1)$ with probability 0.8 (the intended next position), but can enter either of the states $(x_1 - 1, x_2)$, $(x_1 + 1, x_2)$, $(x_1, x_2 - 1)$ and (x_1, x_2) with probability 0.05, respectively. At the boundary points of the grid, the agent bounces against the "wall" in the opposite direction with its position unchanged.

The environment contains points with associated rewards. Specifically, as shown in Fig. 1, squares (waypoints) colored dark green at northeast and southwest have associated reward equal to 1 and squares (waypoints) colored light green at northwest and southeast have associated reward equal to 20. The reward $r_{\mathbf{x}}$ of a waypoint $\mathbf{x}$ "spreads" on the grid according to a Gaussian function. Specifically, the immediate reward at point $\mathbf{y}$ due to the reward associated with waypoint $\mathbf{x}$ is given by

$$r_{\mathbf{x}} \frac{1}{\sqrt{2\pi}} e^{-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2}}$$

and is constant for all actions. Summing over all reward waypoints $\mathbf{x}$, the immediate reward at some point $\mathbf{y}$ is given by

$$f(\mathbf{y}) = \sum_{\mathbf{x}} r_{\mathbf{x}} \frac{1}{\sqrt{2\pi}} e^{-\frac{\|\mathbf{x}-\mathbf{y}\|_2^2}{2}}, \tag{20}$$

where we assume that a point $\mathbf{x}$ with no associated reward satisfies $r_{\mathbf{x}} = 0$. The one step reward function induced by the four reward waypoints of Fig. 1 is shown in Fig. 2.

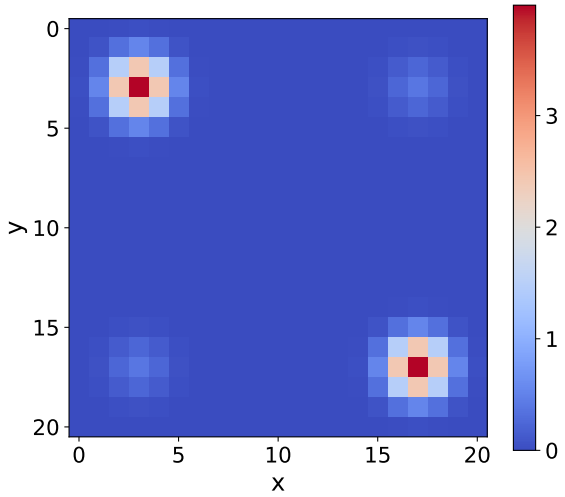

Fig. 2. The one step reward function induced by the reward waypoints shown in Fig. 1.

The agent (robot) entering some state (position) $\mathbf{y}$ collects reward equal to the immediate reward $f(\mathbf{y})$. The objective of the agent is to navigate on the grid so as to maximize the long-term average reward collected.

The key step of efficient learning is to define appropriate features we will use to represent the agent's policy. The way of selecting features in this paper is inspired by spline interpolation [28]. We define 36 waypoints on the grid, shown in Fig. 1 (blue and green grid squares). For a state-action pair $(\mathbf{x}, a)$, we set $\mathbf{y}$ to be the intended next state and define the features

$$\phi_i(\mathbf{x}, a) = f_i(\mathbf{y}), \quad i = 1, \ldots, 36,$$

where $f_i(\cdot)$ is defined as

$$f_i(\mathbf{y}) = r_{\mathbf{x}_i} \frac{1}{\sqrt{2\pi}} e^{-\frac{\|\mathbf{x}_i-\mathbf{y}\|_2^2}{6.215}}, \tag{21}$$

and $\mathbf{x}_i$ is the location of the ith waypoint.

Notice that we are using the kernel-based feature selection method discussed in Sec. IV. Specifically, for each state-action pair $(\mathbf{x}, a)$ we use the encoding $\mathbf{y}(\mathbf{x}, a) = \arg\max_{\mathbf{z}} P(\mathbf{z}|\mathbf{x}, a)$, that is, the most likely next state. We select a collection of state-action pairs $(\mathbf{x}_i, a_i)$, where $\mathbf{x}_i$ are our waypoints. We use a different encoding for these state-action pairs, namely, $\hat{\mathbf{y}}(\mathbf{x}_i, a_i) = \mathbf{x}_i$, thus slightly generalizing the feature selection scheme presented in Sec. IV. We then define feature functions $\phi_i(\mathbf{x}, a) = K_i(\hat{\mathbf{y}}(\mathbf{x}_i, a_i), \mathbf{y}(\mathbf{x}, a))$ using a radial basis function kernel given by the expression in (21), where we allow the magnitude to depend on i (equal to $r_{\mathbf{x}_i}$).

B. Simulation and Learning Performance

For the MDP we have introduced, we use value iteration [2] to find the best policy. We generate independent state-action samples according to this policy. Then, we estimate the policy

using the above samples according to the logistic regression algorithm discussed in Section III. We largely follow the style of [17] in presenting our numerical results. We compare average rewards from 3 different policies with respect to the number of features and the number of samples as follows:

1) *Target policy*: We choose the policy obtained by the value iteration algorithm [2] as the target policy. It is used to generate samples for learning.
2) *ℓ_1-regularized policy*: The RSP trained using the Algorithm in Section III.
3) *Unregularized policy*: The RSP trained using logistic regression in Section III, but without the ℓ_1 constraint on the parameter vector $\boldsymbol{\theta}$ (cf. problem (3)).
4) *Greedy policy*: The agent takes the action with the largest expected next step reward, i.e., the local reward feature is the only consideration. This policy is used as a baseline.

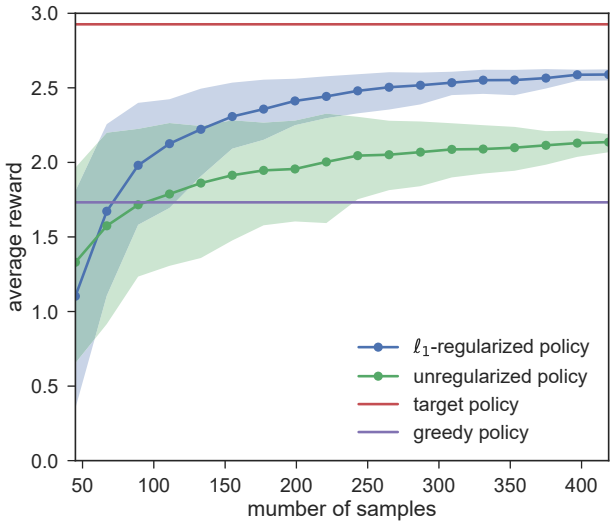

Fig. 3. Average rewards (over 100 runs) of different policies as a function of the number of samples. The shaded regions of the plot are the upper and lower 10% percentile regions.

We randomly sample the state-action pairs generated by the optimal policy and use these samples to form a training set to be used in learning an RSP (as in Section III). This process is repeated 100 times. The average rewards of all policies we considered as a function of the number of samples used for training are displayed in Fig. 3. As shown in Fig. 3, both the ℓ_1-regularized and the unregularized policy benefit from learning and monotonically increase their performance as the number of training samples grows large. In fact, the regularized policy approaches the performance of the target policy for sufficiently large training samples. For most runs, the policies we learn (both regularized and unregularized) perform better than the greedy policy. Furthermore, the regularized policy performs better than the unregularized, demonstrating that the former generalizes better out-of-sample.

To better compare the regularized with the unregularized policy, we also consider the average (Kullback-Leibler) KL-divergence of each of these policies with the target policy

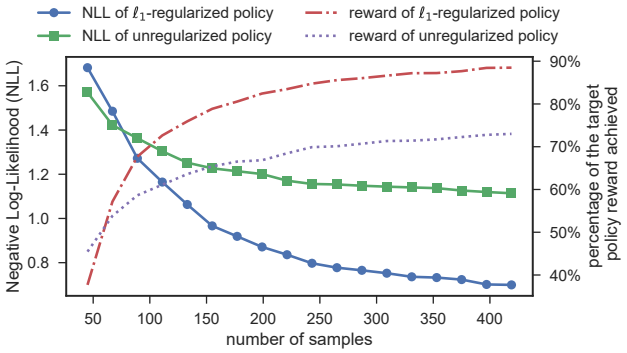

Fig. 4. Left-side y-axis: average (over 100 runs) negative log-likelihood (NLL) of the two regressed policies as a function of the number of samples. Right-side y-axis: percentage of the target policy reward achieved by the two regressed policies as a function of the number of samples.

(cf. the quantity defined in (7)). Since the target policy is deterministic, the average KL-divergence becomes the negative log-likelihood. Fig. 4 (left-side y-axis) shows the average negative log-likelihood of the two regressed policies. The same figure also plots (right-side y-axis) the percentage of the target policy reward achieved by each of the two policies as we vary the number of samples. It can be seen that the regularized policy is closer to the target policy compared with the unregularized one.

We can also observe that, as expected, the regularized policy increases the sparsity of the estimated parameter vector $\hat{\boldsymbol{\theta}}$. In particular, let us define a sparsity metric for $\hat{\boldsymbol{\theta}}$ as

$$\rho(\hat{\boldsymbol{\theta}}) = \frac{|\{\hat{\theta}_i | \hat{\theta}_i < 0.1 \max_j(\hat{\theta}_j)\}|}{n}.$$

Then, taking the average of the above metric over 1400 different runs consisting of different training sets and number of samples used from each training set, we obtain that average $\rho(\hat{\boldsymbol{\theta}}) = 0.64$ for the regularized policy compared to average $\rho(\hat{\boldsymbol{\theta}}) = 0.28$ for the unregularized policy. However, this larger than a factor of two decrease in the average sparsity metric comes at some computation cost. The average running time to compute the regularized policy is 78.68 seconds, which is not excessive but much larger than the 1.07 seconds, on average, needed to compute the unregularized policy. [1]

IX. Conclusions

We considered the problem of learning a policy in a Markov decision process using the state-action samples associated with the policy. We focused on a Boltzmann-type policy that is characterized by feature vectors associated with each state-action pair and a parameter that is sparse.

To learn the policy, we used ℓ_1-regularized logistic regression and showed that a good generalization error bound also guarantees a good bound on the regret, defined as the difference between the average reward of the estimated policy and the target policy. Our results suggest that one can estimate an effective policy using a training set of size proportional to

[1] The code was run on a 2×8-core 2.7 GHz Intel Xeon E5-2680 CPU with 64Gb of RAM and used only 4 threads and 8Gb of RAM.

the logarithm of the number of features used to represent the policy.

APPENDIX A
PROOF OF THEOREM V.3

The proof is similar to the proof of Theorem V.2 (Theorem 3.1 in [17]) but with key differences to accommodate the multiple actions per state. We start by introducing some notations and stating necessary lemmata.

Denote by $\mathcal{F}$ a class of functions over some domain $\mathcal{D}_{\mathcal{F}}$ and range $[-M,M] \subset \mathbb{R}$. Let $\mathcal{F}|_{\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)}} = \{(f(\mathbf{z}^{(1)}),\dots,f(\mathbf{z}^{(m)})) \mid f \in \mathcal{F}\} \subset [-M,M]^m$, which is the valuation of the class of functions at a certain collection of points $\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)} \in \mathcal{D}_{\mathcal{F}}$. It is said that a set of vectors $\{\mathbf{v}^{(1)},\dots,\mathbf{v}^{(k)}\}$ in $\mathbb{R}^m$ ε-covers $\mathcal{F}|_{\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)}}$ in the p-norm, if for every point $\mathbf{v} \in \mathcal{F}|_{\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)}}$ there exists some $\mathbf{v}^{(i)}$, $i=1,\dots,k$, such that $\|\mathbf{v}-\mathbf{v}^{(i)}\|_p \le m^{1/p}\varepsilon$. Let also denote $\mathcal{N}_p(\mathcal{F},\varepsilon,(\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)}))$ the size of the smallest set that ε-covers $\mathcal{F}|_{\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)}}$ in the p-norm. Finally, let $\mathcal{N}_p(\mathcal{F},\varepsilon,m) = \sup_{\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)}} \mathcal{N}_p(\mathcal{F},\varepsilon,(\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)}))$.

To simplify the representation of log-loss of the general logistic function in (1), we use the following notations. For each $x \in \mathcal{X}$, $\boldsymbol{\phi}(x) = (\phi_1(x),\dots,\phi_H(x)) = (\boldsymbol{\phi}(x,1),\dots,\boldsymbol{\phi}(x,H)) \in [0,1]^{nH}$ denotes feature vectors associated with each action. For any $\boldsymbol{\theta} \in \mathbb{R}^n$, define the log-likelihood function $l : [0,1]^{nH} \times \{1,\dots,H\} \to \mathbb{R}$ as

$$l(\boldsymbol{\phi}(x),i) = -\log\left(\frac{\exp(\boldsymbol{\theta}'\boldsymbol{\phi}_i(x))}{\sum_{j=1}^{H}\exp(\boldsymbol{\theta}'\boldsymbol{\phi}_j(x))}\right).$$

Note $\mu_{\boldsymbol{\theta}}(a|x) = \exp\{-l(\boldsymbol{\phi}(x),a)\}$. Further, let $g : [0,1]^n \to \mathbb{R}$ be the class of functions $g(\mathbf{x}) = \boldsymbol{\theta}'\mathbf{x}$. We can then rewrite l using g as $l(\boldsymbol{\phi}(x),y) = l(g(\boldsymbol{\phi}_1(x)),\dots,g(\boldsymbol{\phi}_H(x)),y)$.

Lemma A.1 ([17], [29]) *Let there be some distribution D over $\mathcal{D}_{\mathcal{F}}$ and suppose $\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)}$ are drawn from D i.i.d. Then,*

$$\mathbb{P}\left[\exists f \in \mathcal{F} : \left|\frac{1}{m}\sum_{i=1}^{m} f(\mathbf{z}^{(i)}) - \mathbb{E}_{\mathbf{z}\sim D}[f(\mathbf{z})]\right| \ge \varepsilon\right] \le 8\mathbb{E}\big[\mathcal{N}_1(\mathcal{F},\varepsilon/8,(\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)}))\big]\exp\left(\frac{-m\varepsilon^2}{512M^2}\right).$$

Lemma A.2 ([30]) *Suppose $\mathcal{G} = \{g : g(\mathbf{x}) = \boldsymbol{\theta}'\mathbf{x},\ \mathbf{x} \in \mathbb{R}^n,\ \|\boldsymbol{\theta}\|_q \le B\}$ and the input $\mathbf{x} \in \mathbb{R}^n$ has a norm-bound such that $\|\mathbf{x}\|_p \le \zeta$, where $1/p + 1/q = 1$. Then*

$$\log_2 \mathcal{N}_2(\mathcal{G},\varepsilon,m) \le \frac{B^2\zeta^2}{\varepsilon^2}\log_2(2n+1). \tag{22}$$

Lemma A.3 ([17], [31]) *If $|f(\boldsymbol{\theta}) - \hat{f}(\boldsymbol{\theta})| \le \varepsilon$ for all $\boldsymbol{\theta} \in \Theta$, then*

$$f\Big(\arg\min_{\boldsymbol{\theta}\in\Theta}\hat{f}(\boldsymbol{\theta})\Big) \le \min_{\boldsymbol{\theta}\in\Theta} f(\boldsymbol{\theta}) + 2\varepsilon.$$

Lemma A.4 *Let $\mathcal{G}$ be a class of functions from $\mathbb{R}^n$ to some bounded set in $\mathbb{R}$. Consider $\mathcal{F}$ as a class of functions from $\mathbb{R}^H \times \mathcal{A}$ to some bounded set in $\mathbb{R}$, with the following form:*

$$\mathcal{F} = \big\{f_g(\boldsymbol{\phi}(x),y) = l(g(\boldsymbol{\phi}_1(x)),\dots,g(\boldsymbol{\phi}_H(x)),y), \quad g \in \mathcal{G},\ y \in \mathcal{A}\big\}. \tag{23}$$

If $l(\cdot,y)$ is Lipschitz with Lipschitz constant L in the ℓ_1-norm for every $y \in \mathcal{A}$, then we have

$$\mathcal{N}_1(\mathcal{F},\varepsilon,m) \le \big[\mathcal{N}_1(\mathcal{G},\varepsilon/(LH),m)\big]^H.$$

Proof: Let $\Gamma = \mathcal{N}_1(\mathcal{G},\varepsilon/(LH),m)$. It is sufficient to show for every m inputs

$$\mathbf{z}^{(1)} = (\boldsymbol{\phi}(x^{(1)}),y^{(1)}),\dots,\mathbf{z}^{(m)} = (\boldsymbol{\phi}(x^{(m)}),y^{(m)})$$

we can find Γ^H points in $\mathbb{R}^m$ that ε-cover $\mathcal{F}|_{\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)}}$. Fix some set of points $\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)} \subset \mathbb{R}^{nH+1}$. From the definition of $\mathcal{N}_1(\mathcal{G},\varepsilon/(LH),m)$, for each $j \in \{1,\dots,H\}$,

$$\mathcal{N}_1(\mathcal{G},\varepsilon/(LH),(\boldsymbol{\phi}_j(x^{(1)}),\dots,\boldsymbol{\phi}_j(x^{(m)}))) \le \Gamma.$$

Let [2] $\{\mathbf{v}_j^{(1)},\dots,\mathbf{v}_j^{(\Gamma)}\}$ be a set of Γ points in $\mathbb{R}^m$ that $\varepsilon/(LH)$-covers $\mathcal{G}|_{\boldsymbol{\phi}_j(x^{(1)}),\dots,\boldsymbol{\phi}_j(x^{(m)})}$. We use notation $v_{j,i}^k$ to denote the ith element of vector $\mathbf{v}_j^k$. Then, for any $g \in \mathcal{G}$ and $j \in \{1,\dots,H\}$, there exists a $k(j) \in \{1,\dots,\Gamma\}$ such that

$$\begin{aligned}\|(g(\boldsymbol{\phi}_j(x^{(1)})),\dots,g(\boldsymbol{\phi}_j(x^{(m)})) - \mathbf{v}_j^{k(j)}\|_1 &= \sum_{i=1}^{m}|g(\boldsymbol{\phi}_j(x^{(i)})) - v_{j,i}^{k(j)}| \\ &\le m\frac{\varepsilon}{LH}.\end{aligned} \tag{24}$$

Now consider Γ^H points with the following form

$$l(v_{1,1}^{(j_1)},v_{2,1}^{(j_2)},\dots,v_{H,1}^{(j_H)},y^{(1)}),\dots, l(v_{1,m}^{(j_1)},v_{2,m}^{(j_2)},\dots,v_{H,m}^{(j_H)},y^{(m)}),$$

where $j_1,\dots,j_H \in \{1,\dots,\Gamma\}$.

Given a $g \in \mathcal{G}$ and $f_g(\cdot) \in \mathcal{F}$, let $k(1),\dots k(H) \in \{1,\dots,\Gamma\}$ be as defined above and consider

$$\begin{aligned}&\Big\|\big(f_g(\mathbf{z}^{(1)}),\dots,f_g(\mathbf{z}^{(m)})\big) - \\ &\qquad\big(l(v_{1,1}^{k(1)},v_{2,1}^{k(2)},\dots,v_{H,1}^{k(H)},y^{(1)}),\dots, \\ &\qquad l(v_{1,m}^{k(1)},v_{2,m}^{k(2)},\dots,v_{H,m}^{k(H)},y^{(m)})\big)\Big\|_1 \\ &\le L\Big\|\Big(\sum_{h=1}^{H}|v_{h,1}^{k(h)} - g(\boldsymbol{\phi}_h(x^{(1)}))|,\dots, \\ &\qquad\sum_{h=1}^{H}|v_{h,m}^{k(h)} - g(\boldsymbol{\phi}_h(x^{(m)}))|\Big)\Big\|_1 \\ &= L\sum_{h=1}^{H}\sum_{i=1}^{m}|g(\boldsymbol{\phi}_h(x^{(i)})) - v_{h,i}^{k(h)}| \le m\varepsilon,\end{aligned}$$

where we used the Lipschitz property of the $l(\cdot)$ function in the first inequality and the last inequality follows from (24).

[2] One may find less than Γ points, but we consider the worst case scenario.

Thus, the Γ^H points ϵ-cover $\mathcal{F}|_{\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)}}$ in ℓ_1-norm. Finally, notice that the set of m points $\{\mathbf{z}^{(1)},\dots,\mathbf{z}^{(m)}\}$ is arbitrary, which concludes the proof. ∎

We now continue with the proof of Theorem V.3. Recall Algorithm 1 and let $\hat{B}$ be the smallest integer in $\{0,1,2,4,\dots\}$ that is greater or equal to rK. Notice that in Algorithm 1 one can use a larger C but we select the smallest possible to obtain a tighter bound. For such a $\hat{B}$, it follows that $rK \le \hat{B} \le 2rK$. Define a class of functions $\mathcal{G}$ with domain $[0,1]^n$ as

$$\mathcal{G} = \Big\{ g : [0,1]^n \to \mathbb{R} \;\Big|\; g(\mathbf{x}) = \boldsymbol{\theta}'\mathbf{x},\ \|\boldsymbol{\theta}\|_1 \le \hat{B} \Big\}.$$

By Lemma A.2 and Eq. (22),

$$\log_2 \mathcal{N}_2(\mathcal{G}, \varepsilon/H, m) \le \frac{\hat{B}^2 H^2}{\varepsilon^2} \log_2(2n+1).$$

The partial derivatives of the log-loss function are

$$\frac{\partial}{\partial x_i} l(x_1,\dots,x_H,k) = \begin{cases} -1 + \frac{\exp(x_i)}{\sum_{j=1}^H \exp(x_j)}, & k = i, \\ \frac{\exp(x_i)}{\sum_{j=1}^H \exp(x_j)}, & k \ne i, \end{cases}$$

and it can be seen that

$$\left| \frac{\partial}{\partial x_i} l(x_1,\dots,x_H,k) \right| \le 1.$$

Hence, the Lipschitz constant for $l(\cdot, y)$ is 1 for any $y = 1,\dots,H$.

By Lemma A.4, we have

$$\log_2 \mathcal{N}_1(\mathcal{F}, \varepsilon, m) \le H \log_2 \mathcal{N}_1(\mathcal{G}, \varepsilon/H, m).$$

Using the relation $\mathcal{N}_1 \le \mathcal{N}_2$ ([17], [30], [31]), we obtain

$$\log_2 \mathcal{N}_1(\mathcal{F}, \varepsilon, m) \le \frac{\hat{B}^2 H^3}{\varepsilon^2} \log_2(2n+1). \tag{25}$$

We next find the range of class $\mathcal{F}$. To begin with, the range of class $\mathcal{G}$ is

$$|g(\mathbf{x})| = |\boldsymbol{\theta}'\mathbf{x}| \le \|\boldsymbol{\theta}\|_1 \|\mathbf{x}\|_\infty \le \hat{B}.$$

Since $l(\cdot, i)$ is Lipschitz in ℓ_1-norm with Lipschitz constant 1 and $|f(0,\dots,0,y)| = \log(H) < H$ (by the fact $H \ge 2$), then

$$|f_g(\boldsymbol{\phi}(x), y) - f(\mathbf{0}, y)| \le \sum_{h=1}^H |\boldsymbol{\theta}'\boldsymbol{\phi}_h(x)| \le H\hat{B},$$

which implies

$$|f_g(\boldsymbol{\phi}(x), y)| \le H\hat{B} + H. \tag{26}$$

Finally, let $m_1 = (1-\gamma)m$, which is the size of the training set in Algorithm 1. From Lemma A.1, Eq. (26) and Eq. (25), we have

$$\mathbb{P}\left[\exists f \in \mathcal{F} :\ \left| \frac{1}{m_1} \sum_{i=1}^{m_1} f(\boldsymbol{\phi}(x^{(i)}), y^{(i)}) - \mathbb{E}_{\mathbf{z}\sim D}[f(\mathbf{z})] \right| \ge \varepsilon \right]$$
$$\le 8 \cdot 2^{\frac{256 r^2 K^2 H^3}{\varepsilon^2}} (2n+1) \exp\left(\frac{-m_1 \varepsilon^2}{512 (2rK+1)^2 H^2} \right). \tag{27}$$

Treat $(1-\gamma)$ as a constant. To upper bound the right hand side of the above equation by δ, it suffices to have

$$m = \Omega\Big((\log n) \cdot \mathrm{poly}(r,\ K,\ H,\ \log(1/\delta),\ 1/\varepsilon) \Big). \tag{28}$$

The rest of the proof follows closely the proof of Theorem 3.1 in [17]. We outline the key steps for the sake of completeness. Suppose m satisfies (28); then, with probability at least $1-\delta$, for all $f \in \mathcal{F}$

$$\left| \frac{1}{m_1} \sum_{i=1}^{m_1} f(\boldsymbol{\phi}(x^{(i)}), y^{(i)}) - \mathbb{E}_{\mathbf{z}\sim D}[f(\mathbf{z})] \right| \le \varepsilon.$$

Thus, using our definition of $\mathcal{F}$ in (23), for any $\boldsymbol{\theta}$ with $\|\boldsymbol{\theta}\|_1 \le \hat{B}$ and with probability at least $1-\delta$, we have

$$\left| \frac{1}{m_1} \sum_{i=1}^{m_1} \big(-\log \mu_{\boldsymbol{\theta}}(y^{(i)}|x^{(i)}) \big) - E_{(x,y)\sim D}\big[-\log \mu_{\boldsymbol{\theta}}(y|x) \big] \right| \le \varepsilon.$$

Therefore, for all $\boldsymbol{\theta}$ with $\|\boldsymbol{\theta}\|_1 \le \hat{B}$ and with probability at least $1-\delta$, it holds

$$|\hat{\epsilon}_{\mathcal{S}_1}(\boldsymbol{\theta}) - \epsilon(\boldsymbol{\theta})| \le \varepsilon,$$

where $\mathcal{S}_1$ is the training set from Algorithm 1.

Essentially, we have shown that for m large enough the empirical log-loss function $\hat{\epsilon}_{\mathcal{S}_1}(\cdot)$ is a good estimate of the log-loss function $\epsilon(\cdot)$. According to Step 2 of Algorithm 1, $\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}:\|\boldsymbol{\theta}\|_1 \le \hat{B}} \hat{\epsilon}_{\mathcal{S}_1}(\theta)$. By Lemma A.3, we have

$$\begin{aligned} \epsilon(\hat{\boldsymbol{\theta}}) &\le \min_{\boldsymbol{\theta}:\|\boldsymbol{\theta}\|_1 \le \hat{B}} \epsilon(\boldsymbol{\theta}) + 2\varepsilon \\ &\le \epsilon(\boldsymbol{\theta}^*) + 2\varepsilon, \end{aligned} \tag{29}$$

where $\boldsymbol{\theta}^*$ is the target policy and the last inequality follows simply from the fact that $\|\boldsymbol{\theta}^*\|_1 \le rK \le \hat{B}$.

Eq. (29) indicates that the training step of Algorithm 1, finds at least one parameter vector $\hat{\boldsymbol{\theta}}$ whose performance is nearly as good as that of $\boldsymbol{\theta}^*$. At the validation step, we select one of the $\boldsymbol{\theta}_B$ found during training. It can be shown ([31]) that with a validation set of the same order of magnitude as the training set (and independent of n), we can ensure that with probability at least $1-\delta$, the selected parameter vector will have performance at most 2ε worse than that of the best performing vector discovered in the training step. Hence, with probability at least $1-2\delta$, the output $\hat{\boldsymbol{\theta}}$ of our algorithm satisfies

$$\epsilon(\hat{\boldsymbol{\theta}}) \le \epsilon(\boldsymbol{\theta}^*) + 4\varepsilon. \tag{30}$$

Finally, replacing δ with $\delta/2$ and ε with $\varepsilon/4$ everywhere in the proof, establishes Theorem V.3.

REFERENCES

[1] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc. New York, NY, USA, 1994.

[2] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 1995, vol. I and II.

[3] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[4] W. Powell, *Approximate Dynamic Programming*. John Wiley and Sons, 2007.

[5] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," *SIAM Journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[6] J. Wang and I. C. Paschalidis, "An actor-critic algorithm with second-order actor and critic," *IEEE Trans. Automat. Contr.*, vol. 62, no. 6, pp. 2689–2703, 2017.

[7] R. Moazzez-Estanjini, K. Li, and I. C. Paschalidis, "A least squares temporal difference actor-critic algorithm with applications to warehouse management," *Naval Research Logistics*, vol. 59, no. 3, pp. 197–211, 2012.

[8] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement leaning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

[9] G. Neu and S. Szepesvari, "Apprenticeship learning using inverse reinforcement learning and gradient methods," in *Proceedings of the Uncertainty in Artificial Intelligence (UAI)*, 2007.

[10] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Artificial Intelligence and Statistics (AISTATS)*, 2011.

[11] F. S. Melo and M. Lopes, "Learning from demonstration using MDP induced metrics," in *Proceedings of the European Conference on Machine Learning (ECML)*, 2010.

[12] B. Da Silva, G. Konidaris, and A. Barto, "Learning parameterized skills," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, Edinburgh, UK, 2012, arXiv preprint arXiv:1206.6398.

[13] S. Schaal, "Is imitation learning the route to humanoid robots?" in *In Trends in Cognitive Sciences*, 1999.

[14] B. Argall, S. Chernova, and M. Veloso, "A survey of robot learning from demonstrations," *Robotics and Autonomous System*, vol. 57, no. 5, 2009.

[15] W. Erlhagen, A. Mukovskiy, E. Bicho, G. Panin, C. Kiss, A. Knoll, H. Van Schie, and H. Bekkering, "Goal-directed imitation for robots: A bio-inspired approach to action understanding and skill learning," *Robotics and autonomous systems*, vol. 54, no. 5, pp. 353–360, 2006.

[16] I. C. Paschalidis and Y. Lin, "Animal-inspired optimal foraging via a distributed actor-critic algorithm," in *Proceedings of the 20th Mediterranean Conference on Control and Automation (MED 12)*, Barcelona, Spain, July 3–6 2012, pp. 1223–1228.

[17] A. Y. Ng, "Feature selection, L_1 vs. L_2 regularization, and rotational invariance," in *Proceedings of the International Conference on Machine Learning (ICML)*, June 2004.

[18] D. Bertsimas, V. Gupta, and I. C. Paschalidis, "Data-driven estimation in equilibrium using inverse optimization," *Mathematical Programming, Series A*, vol. 153, no. 2, pp. 595–633, 2015.

[19] J. A. Bagnell, "Learning decisions: Robustness, uncertainty, and approximation," Ph.D. dissertation, Carnegie Mellon University, 2004.

[20] G. Lever and R. Stafford, "Modelling Policies in MDPs in Reproducing Kernel Hilbert Space," in *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, San Diego, CA, 2015.

[21] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2000.

[22] T. A. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 2006.

[23] M. K. Hanawal, H. Liu, H. Zhu, and I. C. Paschalidis, "Learning parameterized policies for Markov decision processes through demonstrations," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 7087–7092.

[24] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM Journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[25] K. Prabuchandran, S. Bhatnagar, and V. S. Borkar, "An actor critic algorithm based on Grassmanian search," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 3597–3602.

[26] G. E. Cho and C. Meyer, "Comparison of perturbation bounds for the stationary distributions of a Markov chain," *Elsevier journal of Linear Algebra and Its Application*, vol. 335, pp. 137–150, 2001.

[27] E. Seneta, "Perturbation of the stationary distribution measured by ergodicity coefficients," *Advances in Applied Probability*, vol. 20, no. 1, 1998.

[28] C. De Boor, *A practical guide to splines*. Springer, 1978, vol. 27.

[29] D. Pollard, *Convergence of stochastic processes*. Springer-Verlag, 1984.

[30] T. Zhang, "Covering Number Bounds of Certain Regularized Linear Function Classes," *Journal of Machine Learning Research*, vol. 2, pp. 527–550, 2002.

[31] M. Anthony and P. L. Bartlett, *Neural network learning: Theoretical foundations*. Cambridge University Press, 2009.

Manjesh Kumar Hanawal received the B.E. degree in ECE from the National Institute of Technology, Bhopal, India, in 2004, the M.S. degree in ECE from the Indian Institute of Science, Bangalore, India, in 2009, and the Ph.D. degree from INRIA, Sophia Antipolis, France, and the University of Avignon, Avignon, France, in 2013. After spending two years as a postdoctoral associate at Boston University, he is now an Assistant Professor in Industrial Engineering and Operations Research at the Indian Institute of Technology Bombay, Mumbai, India. His research interests include communication networks, machine learning and network economics.

Hao Liu received the B.E. degree in Automation from Nankai University, Tianjin, China, in 2012. He is currently working toward the Ph.D. degree in the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China. He is a member of the Networked Sensing and Control group (NeSC). From 3/2015 to 3/2016, he has been a visiting student in the Center for Information and Systems Engineering, Boston University. His research interests include distributed computation and optimization in multi-agent networks, differential privacy in distributed computation, and machine learning.

Henghui Zhu received the B.Sc. degree in mathematics from Wuhan University, Wuhan, China, in 2012, and the M.Sc. degree in systems theory from the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China, in 2015. He is a Ph.D. candidate in the Division of Systems Engineering, Boston University, Boston, USA. His research interests include optimization, reinforcement learning, and deep neural networks.

Ioannis Ch. Paschalidis (M'96–SM'06–F'14) received the Diploma in ECE from the National Technical University of Athens, Athens, Greece, in 1991, and the M.S. and Ph.D. degrees, both in EECS, from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 1993 and 1996, respectively.

In September 1996 he joined Boston University where he has been ever since. He is a Professor and Data Science Fellow at Boston University with appointments in the Department of Electrical and Computer Engineering, the Division of Systems Engineering, and the Department of Biomedical Engineering. He is the Director of the Center for Information and Systems Engineering (CISE). He has held visiting appointments with MIT and Columbia University, New York, NY, USA. His current research interests lie in the fields of systems and control, networks, applied probability, optimization, operations research, computational biology, and medical informatics.

Dr. Paschalidis is a recipient of the NSF CAREER award (2000), several best paper and best algorithmic performance awards, and a 2014 IBM/IEEE Smarter Planet Challenge Award. He was an invited participant at the 2002 Frontiers of Engineering Symposium, organized by the U.S. National Academy of Engineering and the 2014 U.S. National Academies Keck Futures Initiative (NAFKI) Conference. He is the founding Editor-in-Chief of the IEEE Transactions on Control of Network Systems.