Nonparametric Stochastic Compositional Gradient Descent for Q-Learning in Continuous Markov Decision Problems

Ekaterina Tolstaya*, Alec Koppel†, Ethan Stump†, Alejandro Ribeiro*

***Abstract*— We consider Markov Decision Problems defined over continuous state and action spaces, where an autonomous agent seeks to learn a map from its states to actions so as to maximize its long-term discounted accumulation of rewards. We address this problem by considering Bellman's optimality equation defined over action-value functions, which we reformulate into a nested non-convex stochastic optimization problem defined over a Reproducing Kernel Hilbert Space (RKHS). We develop a functional generalization of stochastic quasi-gradient method to solve it, which, owing to the structure of the RKHS, admits a parameterization in terms of scalar weights and past state-action pairs which grows proportionately with the algorithm iteration index. To ameliorate this complexity explosion, we apply Kernel Orthogonal Matching Pursuit to the sequence of kernel weights and dictionaries, which yields a controllable error in the descent direction of the underlying optimization method. We prove that the resulting algorithm, called KQ Learning, converges with probability 1 to a stationary point of this problem, yielding a fixed point of the Bellman optimality operator under the hypothesis that it belongs to the RKHS. Numerical evaluation on the continuous Mountain Car task yields convergent parsimonious learned action-value functions and policies that are competitive with the state of the art.**

I. Introduction

Markov Decision Problems offer a flexible framework to address sequential decision making tasks under uncertainty [2], and have gained broad interest in robotics [3], control [4], and artificial intelligence [5]. Despite this surge of interest, few works in reinforcement learning address the computational difficulties associated with continuous state and action spaces in a principled way that guarantees convergence. The goal of this work is to develop new reinforcement learning tools for continuous problems which are provably stable and whose complexity is at-worst moderate.

In the development of stochastic methods for reinforcement learning, one may attempt to estimate the transition density of the Markov Decision Process (MDP) (model-based [6]), perform gradient descent on the value function with respect to the policy (direct policy search [7]), and pursue value function based (model-free [8], [9]) methods which exploit structural properties of the setting to derive fixed point problems called *Bellman equations*. We adopt the latter approach in this work , motivated by the fact that an action-value function tells us both how to find a policy and how to evaluate it in terms of the performance metric we have defined, and that a value function encapsulates structural properties of the relationship between states, actions, and rewards.

Consider the fixed point problem defined by Bellman's optimality equation [10]. When the state and action spaces are finite and small enough that expectations are computable, fixed point iterations may be used. When this fails to hold, stochastic fixed point methods, namely, Q-learning [9], may be used, whose convergence may be addressed with asynchronous stochastic approximation theory [11]. This approach is only valid when the action-value (or Q) function may be represented as a matrix. However, when the state and action spaces are infinite, this is no longer true, and the Q-function instead belongs to a generic function space.

In particular, to solve the fixed point problem defined by Bellman's optimality equation when spaces are continuous, one must surmount the fact that it is defined for infinitely many unknowns, one example of Bellman's curse of dimensionality [10]. Efforts to sidestep this issue assume that the Q-function admits a finite parameterization, such as a linear [12] or nonlinear [13] basis expansion, is defined by a neural network [14], or that it belongs to a reproducing kernel Hilbert Space (RKHS) [15], [16]. In this work, we adopt the latter nonparametric approach, motivated by the fact that combining fixed point iterations with different parameterizations may cause divergence [17], [18], and in general the Q-function parameterization must be tied to the stochastic update to ensure the convergence of both the function sequence and its parameterization [19].

Our main result is a memory-efficient, nonparametric, stochastic method that converges to a fixed point of the Bellman optimality operator almost surely when it belongs to a RKHS. We obtain this result by reformulating the Bellman optimality equation as a nested stochastic program, a topic investigated in

This work is supported by grants NSF DGE-1321851 and ARL DCIST CRA W911NF-17-2-0181. All proofs are given in [1].

* Department of ESE, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104. Email: {eig, aribeiro}@seas.upenn.edu.

† Computational and Information Sciences Directorate, U.S. Army Research Laboratory, Adelphi, MD, 20783. Email: {alec.e.koppel.civ, ethan.a.stump2.civ}@mail.mil.

operations research [20] These problems have been addressed in finite settings with stochastic *quasi-gradient* (SQG) methods[21] which use two time-scale stochastic approximation to mitigate the fact that the objective's stochastic gradient not available due to its dependence on a *second expectation*, referred to as the double sampling problem in [12].

Here, we use a non-parametric generalization of SQG for Q-learning in infinite MDPs (Section III), motivated by its success for policy evaluation in finite [12] and infinite MDPs [22]. However, a function in a RKHS has comparable complexity to the number of training samples processed, which is in general infinite, an issue is often ignored in kernel methods for Markov decision problems [23]. We address this memory bottleneck (the curse of kernelization) by requiring memory efficiency in both the function sample path and in its limit through the use of sparse projections which are constructed greedily via matching pursuit [24] , akin to [25], [22]. Greedy compression here is appropriate since (a) kernel matrices induced by arbitrary data streams will likely become ill-conditioned and hence violate assumptions required by convex methods [26], and (b) parsimony is more important than exact recovery as the SQG iterates are not the target signal but rather a noisy stepping stone to Bellman fixed point. Rather than unsupervised forgetting [27], we tie the projection-induced error to guarantee stochastic descent [25], only keeping those dictionary points needed for convergence.

As a result, we conduct functional SQG descent via sparse projections of the SQG. This maintains a moderate-complexity sample path exactly towards Q^*, which may be made arbitrarily close to a Bellman fixed point by decreasing the regularizer. In contrast to the convex structure in [22], the Bellman optimality equation induces a non-convex cost functional, which requires us to generalize the relationship between SQG for non-convex objectives and coupled supermartingales in [28] to RKHSs. In doing so, we establish that the sparse projected SQG sequence converges almost surely to the Bellman fixed point with decreasing learning rates (Section IV). Moreover, on Continuous Mountain Car [29], we observe that our learned action-value function attains a favorable trade-off between memory efficiency and Bellman error, which then yields a policy whose performance is competitive with the state of the art.

II. Markov Decision Processes

We model an autonomous agent in a continuous space as a Markov Decision Process (MDP) with continuous states $\mathbf{s} \in \mathcal{S} \subset \mathbb{R}^p$ and actions $\mathbf{a} \in \mathcal{A} \subset \mathbb{R}^q$. When in state $\mathbf{s}$ and taking action $\mathbf{a}$, a random transition to state s' occurs according to the conditional probability density $\mathbb{P}(\mathbf{s}'|\mathbf{s},\mathbf{a})$. After the agent transitions to a particular $\mathbf{s}'$ from $\mathbf{s}$, the MDP assigns an instantaneous reward $r(\mathbf{s},\mathbf{a},\mathbf{s}')$, where the reward function is a map $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$.

In Markov Decision problems, the goal is to find the action sequence $\{\mathbf{a}_t\}_{t=0}^{\infty}$ so as to maximize the infinite horizon accumulation of rewards, i.e., the value function: $V(\mathbf{s},\{\mathbf{a}_t\}_{t=0}^{\infty}) := \mathbb{E}_{\mathbf{s}'}[\sum_{t=0}^{\infty}\gamma^t r(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_t') \mid \mathbf{s}_0 = \mathbf{s},\{\mathbf{a}_t\}_{t=0}^{\infty}]$. The action-value function $Q(\mathbf{s},\mathbf{a})$ is the conditional mean of the value function given the initial action $\mathbf{a}_0 = \mathbf{a}$:

$$Q(\mathbf{s},\mathbf{a},\{\mathbf{a}_t\}_{t=1}^{\infty}) := \mathbb{E}_{\mathbf{s}'}\left[\sum_{t=0}^{\infty}\gamma^t r(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_t') \,|\, \mathbf{s}_0=\mathbf{s},\mathbf{a}_0=\mathbf{a},\{\mathbf{a}_t\}_{t=1}^{\infty}\right]. \tag{1}$$

We consider the case where actions $\mathbf{a}_t$ are chosen according to a stationary stochastic policy, where a policy is a mapping from states to actions: π: $\mathcal{S} \to \mathcal{A}$. We define $Q^*(\mathbf{s},\mathbf{a})$ as the maximum of (1) with respect to the action sequence. The reason for defining action-value functions is that the optimal Q^* may be used to compute the optimal policy π^* as

$$\pi^*(\mathbf{s}) = \operatorname*{argmax}_{\mathbf{a}} Q^*(\mathbf{s},\mathbf{a}) \,. \tag{2}$$

Thus, finding Q^* solves the MDP. Value-function based approaches to MDPs reformulate (2) by shifting the index of the summand in (1) by one, as well as exploiting the time invariance of the Markov transition kernel and the homogeneity of the summand, to derive the Bellman optimality equation:

$$Q^*(\mathbf{s},\mathbf{a}) = \mathbb{E}_{\mathbf{s}'}\Big[r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'} Q^*(\mathbf{s}',\mathbf{a}') \,\Big|\, \mathbf{s},\mathbf{a}\Big]. \tag{3}$$

where the expectation is taken with respect to the conditional distribution $\mathbb{P}(d\mathbf{s}' \mid \mathbf{s},\mathbf{a})]$ of the state $\mathbf{s}'$ given the state action pair $(\mathbf{s},\mathbf{a})$. The right-hand side of Equation (3) defines the Bellman optimality operator $\mathscr{B}^*$: $\mathcal{B}(\mathcal{S}\times\mathcal{A}) \to \mathcal{B}(\mathcal{S}\times\mathcal{A})$ over $\mathcal{B}(\mathcal{S}\times\mathcal{A})$, the space of bounded continuous action-value functions Q: $\mathcal{B}(\mathcal{S}\times\mathcal{A}) \to \mathbb{R}$:

$$(\mathscr{B}^*Q)(\mathbf{s},\mathbf{a}) := \mathbb{E}_{\mathbf{s}'}[r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}')]. \tag{4}$$

[4] [Proposition 5.2] establishes that the fixed point of (4) is the optimal action-value function Q^*. Thus, to solve the MDP, we seek to compute the fixed point of (4) for all $(\mathbf{s},\mathbf{a}) \in \mathcal{S}\times\mathcal{A}$.

Compositional Stochastic Optimization. The functional fixed point equation in (3) has to be simultaneously satisfied for all state action pairs $(\mathbf{s},\mathbf{a})$. Alternatively, we can integrate (3) over an arbitrary distribution that is dense around any pair $(\mathbf{s},\mathbf{a})$ to write a nested stochastic optimization problem [28], [25], [22]. To do so, begin by defining the function

$$f(Q;\mathbf{s},\mathbf{a}) = \mathbb{E}_{\mathbf{s}'}\Big[r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}',\mathbf{a}') - Q(\mathbf{s},\mathbf{a}) \,\Big|\, \mathbf{s},\mathbf{a}\Big], \tag{5}$$

and consider an arbitrary everywhere dense distribution $\mathbb{P}(d\mathbf{s}, d\mathbf{a})]$ over pairs $(\mathbf{s}, \mathbf{a})$ to define the functional

$$L(Q) = \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a}}\Big[f^2(Q;\mathbf{s},\mathbf{a})\Big]. \tag{6}$$

Comparing (5) with (3) permits concluding that $\mathbf{Q}^*$ is the unique function that makes $f(Q;\mathbf{s},\mathbf{a}) = 0$ for all $(\mathbf{s},\mathbf{a})$. It then follows that $\mathbf{Q}^*$ is the only function that makes the functional in (6) take the value $L(Q) = 0$. Since this functional is also nonnegative, it follows that we can write the optimal Q function as

$$Q^* = \underset{Q\in\mathcal{B}(\mathcal{S}\times\mathcal{A})}{\operatorname{argmin}} \; L(Q)\,. \tag{7}$$

Computation of the optimal policy is thus equivalent to solving the optimization problem in (7). This requires a difficult search over all bounded continuous functions $\mathcal{B}(\mathcal{S}\times\mathcal{A})$. We reduce this difficulty through a hypothesis on the function class that we discuss next.

Reproducing Kernel Hilbert Spaces We propose restricting $\mathcal{B}(\mathcal{S}\times\mathcal{A})$ to be a Hilbert space $\mathcal{H}$ equipped with a unique reproducing kernel, an inner product-like map $\kappa : (\mathcal{S}\times\mathcal{A})\times(\mathcal{S}\times\mathcal{A}) \to \mathbb{R}$ such that

$$\text{(i) } \langle f, \kappa((\mathbf{s},\mathbf{a}),\cdot)\rangle_{\mathcal{H}} = f(\mathbf{s},\mathbf{a}), \text{ (ii) } \mathcal{H} = \overline{\operatorname{span}\{\kappa((\mathbf{s},\mathbf{a}),\cdot)\}} \tag{8}$$

In (8), property (i) is called the reproducing property. Replacing f by $\kappa((\mathbf{s}',\mathbf{a}'),\cdot)$ in (8) (i) yields the expression $\langle \kappa((\mathbf{s}',\mathbf{a}'),\cdot), \kappa((\mathbf{s},\mathbf{a}),\cdot)\rangle_{\mathcal{H}} = \kappa((\mathbf{s}',\mathbf{a}'),(\mathbf{s},\mathbf{a}))$, the origin of the term "reproducing kernel." Moreover, property (8) (ii) states that functions $f \in \mathcal{H}$ admit a basis expansion in terms of kernel evaluations (9). Function spaces of this type are referred to as reproducing kernel Hilbert spaces (RKHSs). We may apply the Representer Theorem to transform the functional problem into a parametric one [30]. In the Reproducing Kernel Hilbert Space (RKHS), the optimal Q function takes the following form

$$Q(\mathbf{s},\mathbf{a}) = \sum_{n=1}^{N} w_n \kappa((\mathbf{s}_n,\mathbf{a}_n),(\mathbf{s},\mathbf{a})) \tag{9}$$

where (s_n, a_n) is a realization of the random variables in $\mathcal{S}\times\mathcal{A}$. $Q \in \mathcal{H}$ is an expansion of kernel evaluations only at the training samples.

One complication of the restriction $\mathcal{B}(\mathcal{S}\times\mathcal{A})$ to the RKHS $\mathcal{H}$ is that this setting requires the cost to be differentiable with Lipschitz gradients, but the definition of $L(Q)$ [cf. (6)] defined by Bellman's equation (4) is non-differentiable due to the presence of the maximization over the Q function. We approximate the non-smooth cost by a smooth one by replacing the $\max_{\mathbf{a}} Q(\mathbf{s},\mathbf{a}')$ term in (6) by the softmax over continuous range $\mathcal{A}$, and, subsequently, we restrict focus to smoothed cost $L(Q)$. See [1] for details.

In this work, we restrict the kernel used to be in the family of universal kernels, such as a Gaussian Gaussian Radial Basis Function(RBF) kernel with constant diagonal covariance Σ, i.e.,

$$\kappa((\mathbf{s},\mathbf{a}),(\mathbf{s}',\mathbf{a}')) = \exp\{-\frac{1}{2}((\mathbf{s},\mathbf{a})-(\mathbf{s}',\mathbf{a}'))\Sigma((\mathbf{s},\mathbf{a})-(\mathbf{s}',\mathbf{a}')^T\} \tag{10}$$

motivated by the fact that a continuous function over a compact set may be approximated uniformly by a function in a RKHS equipped with a universal kernel [31].

To apply the Representer Theorem, we require the cost to be coercive in Q [30], which may be satisfied through use of a Hilbert-norm regularizer, so we define the regularized cost functional $J(Q) = L(Q) + (\lambda/2)\|Q\|^2_{\mathcal{H}}$ and solve the regularized problem (7), i.e.

$$Q^* = \underset{Q\in\mathcal{H}}{\operatorname{argmin}} J(Q) = \underset{Q\in\mathcal{H}}{\operatorname{argmin}} L(Q) + \frac{\lambda}{2}\|Q\|^2_{\mathcal{H}}. \tag{11}$$

Thus, finding a locally optimal action-value function in an MDP amounts to solving the RKHS-valued compositional stochastic program with a non-convex objective defined by the Bellman optimality equation (4). This action-value function can then be used to obtain the optimal policy (2). In the following section, we turn to iterative stochastic methods to solve (11). We point out that this is a step back from the original intent of solving (7) to then find optimal policies π^* using (2). This is the case because the assumption we have made about Q^* being representable in the RKHS $\mathcal{H}$ need not be true. More importantly, the functional $J(Q)$ is not convex in Q and there is no guarantee that a local minimum of $J(Q)$ will be the optimal policy Q^*. This is a significant difference relative to policy evaluation problems [22].

III. Stochastic Quasi-Gradient Method

To solve 11, we propose applying a functional variant of stochastic quasi-gradient (SQG) descent to the loss function $J(Q)$ [cf. (11)]. The reasoning for this approach rather than a stochastic gradient method is the nested expectations cause the functional stochastic gradient to be still dependent on a second expectation which is not computable, and SQG circumvents this issue. Then, we apply the Representer Theorem (9) ("kernel trick") to obtain a tractable parameterization of this optimization sequence, which unfortunately has per-iteration complexity. We then mitigate this untenable complexity growth while preserving optimality using greedy compressive methods, inspired by [25], [22].

To find a stationary point of (11) we use quasi-gradients $\nabla_Q J(Q)$ of the functional $J(Q)$ relative to the function Q in an iterative process. To do so, introduce an iteration index t and let Q_t be the estimate of the stationary point at iteration t. Further consider a random

state action pair $(\mathbf{s}_t, \mathbf{a}_t)$ independently and randomly chosen from the distribution $\mathbb{P}(d\mathbf{s}, d\mathbf{a})$. Action $\mathbf{a}_t$ is executed from state $\mathbf{s}_t$ resulting in the system moving to state $\mathbf{s}'_t$. This outcome is recorded along with reward $r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}'_t)$ and the action a'_t that maximizes the action-value function Q_t when the system is in state $\mathbf{s}'_t$,

$$\mathbf{a}'_t := \underset{a'}{\operatorname{argmax}}\, Q_t(\mathbf{s}'_t, \mathbf{a}'). \tag{12}$$

The state (S) $\mathbf{s}_t$, action (A) $\mathbf{a}_t$, reward (R) $r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}'_t)$, state (S) $\mathbf{s}'_t$, action (A) $\mathbf{a}'_t$ are collectively referred to as the SARSA tuple at time t.

Further consider the expressions for $J(Q)$ in (11) and $L(Q)$ in (6) and exchange order of the expectation and differentiation operators to write the gradient of $J(Q)$ as

$$\nabla_Q J(Q_t) = \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t}\Big[f(Q_t; \mathbf{s}_t, \mathbf{a}_t) \times \nabla_Q f(Q_t; \mathbf{s}_t, \mathbf{a}_t) \Big] + \lambda Q_t. \tag{13}$$

Observe that to obtain samples of $\nabla_Q J(Q, \mathbf{s}, \mathbf{a}, \mathbf{s}')$ we require two different queries to a simulation oracle: one to approximate the inner expectation over the Markov transition dynamics defined by $\mathbf{s}'$, and one for *each initial pair* $\mathbf{s}, \mathbf{a}$ which defines the outer expectation. This complication, called the "double sampling problem," was first identified in [21], [32], [13], has been ameliorated through use of two time-scale stochastic approximation, which may be viewed as a stochastic variant of quasi-gradient methods [28].

Following this line of reasoning, we build up the total expectation of one of the terms in (13) while doing stochastic descent with respect to the other. In principle, it is possible to build up the expectation of either term in (13), but the mean of the difference of kernel evaluations is of infinite complexity. On the other hand, the *temporal action difference*, defined as the difference between the action-value function evaluated at state-action pair $(\mathbf{s}, \mathbf{a})$ and the action-value function evaluated at next state and the instantaneous maximizing action $(\mathbf{s}', \mathbf{a}')$, i.e.,

$$\delta := r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma Q(\mathbf{s}', \mathbf{a}') - Q(\mathbf{s}, \mathbf{a}) \tag{14}$$

is a *scalar*, and thus so is its total expected value. Therefore, for obvious complexity motivations, we build up the total expectation of (14). To do so, we propose recursively averaging realizations of (14) through the following auxiliary sequence z_t, initialized as null $z_0 = 0$:

$$\begin{aligned} \delta_t &:= r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}'_t) + \gamma Q(\mathbf{s}'_t, \mathbf{a}'_t) - Q(\mathbf{s}_t, \mathbf{a}_t)\,, \\ z_{t+1} &= (1 - \beta_t) z_t + \beta_t \delta_t \end{aligned} \tag{15}$$

where $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}'_t)$ is an independent realization of the random triple $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ and $\beta_t \in (0,1)$ is a learning rate.

To define the stochastic descent step, we replace the first term inside the outer expectation in (13) with its instantaneous approximation $[\gamma \kappa((\mathbf{s}', \mathbf{a}'), \cdot) - \kappa((\mathbf{s}, \mathbf{a}), \cdot)]$ evaluated at a sample triple $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}'_t)$, which yields the stochastic quasi-gradient step:

$$Q_{t+1} = (1 - \alpha_t \lambda) Q_t(\cdot) - \alpha_t (\gamma \kappa(\mathbf{s}'_t, \mathbf{a}'_t, \cdot) - \kappa(\mathbf{s}_t, \mathbf{a}_t, \cdot)) z_{t+1}) \tag{16}$$

where the coefficient $(1 - \alpha_t \lambda)$ comes from the regularizer and α_t is a positive scalar learning rate. Moreover, $\mathbf{a}'_t = \operatorname{argmax}_{\mathbf{b}} Q_t(\mathbf{s}', \mathbf{b})$ is the instantaneous Q-function maximizing action. Now, using similar logic to [33], we may extract a computationally tractable parameterization of the infinite dimensional function sequence (16), exploiting properties of the RKHS (8).

Kernel Parametrization Suppose $Q_0 = 0 \in \mathcal{H}$. Then the update in (16) at time t, inductively making use of the Representer Theorem, implies the function Q_t is a kernel expansion of past state-action tuples $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}'_t)$

$$Q_t(s, a) = \sum_{n=1}^{2(t-1)} w_n \kappa(\mathbf{v}_n, (\mathbf{s}, \mathbf{a})) = \mathbf{w}_t^T \kappa_{\mathbf{X}_t}((\mathbf{s}, \mathbf{a})) \tag{17}$$

The kernel expansion in (17), together with the functional update (16), yields the fact that functional SQG in $\mathcal{H}$ amounts to updating the kernel dictionary $\mathbf{X}_t \in \mathbb{R}^{p \times 2(t-1)}$ and coefficient vector $\mathbf{w}_t \in \mathbb{R}^{2(t-1)}$ as

$$\begin{aligned} \mathbf{X}_{t+1} &= [\mathbf{X}_t, (\mathbf{s}_t, \mathbf{a}_t), (\mathbf{s}'_t, \mathbf{a}'_t)]\,, \\ \mathbf{w}_{t+1} &= [(1 - \alpha_t \lambda)\mathbf{w}_t, \alpha_t z_{t+1}, -\alpha_t \gamma z_{t+1}] \end{aligned} \tag{18}$$

In (18), the coefficient vector $\mathbf{w}_t \in \mathbb{R}^{2(t-1)}$ and dictionary $\mathbf{X}_t \in \mathbb{R}^{p \times 2(t-1)}$ are defined as

$$\begin{aligned} \mathbf{w}_t &= [w_1, \ldots, w_{2(t-1)}]\,, \\ \mathbf{X}_t &= [(\mathbf{s}_1, \mathbf{a}_1), (\mathbf{s}'_1, \mathbf{a}'_1), \ldots, (\mathbf{s}_{t-1}, \mathbf{a}_{t-1}), (\mathbf{s}'_{t-1}, \mathbf{a}'_{t-1})], \end{aligned} \tag{19}$$

and in (17), we introduce the notation $\mathbf{v}_n = (\mathbf{s}_n, \mathbf{a}_n)$ for n even and $\mathbf{v}_n = (\mathbf{s}'_n, \mathbf{a}'_n)$ for n odd. Moreover, in (17), we make use of a concept called the empirical kernel map associated with dictionary $\mathbf{X}_t$, defined as

$$\begin{aligned} \kappa_{\mathbf{X}_t}(\cdot) = [&(\kappa((\mathbf{s}_1, \mathbf{a}_1), \cdot), \kappa((\mathbf{s}'_1, \mathbf{a}'_1), \cdot), \ldots, \\ &\ldots, \kappa((\mathbf{s}_{t-1}, \mathbf{a}_{t-1}), \cdot), \kappa((\mathbf{s}'_{t-1}, \mathbf{a}'_{t-1}), \cdot)]^T\,. \end{aligned} \tag{20}$$

Observe that (18) causes $\mathbf{X}_{t+1}$ to have two more columns than its predecessor $\mathbf{X}_t$. We define the *model order* as the number of data points (columns) M_t in the dictionary at time t, which for functional stochastic quasi-gradient descent is $M_t = 2(t-1)$. Asymptotically, then, the complexity of storing $Q_t(\cdot)$ is infinite, and even for moderately large training sets is untenable. Next, we address this intractable complexity blowup, inspired by [25], [22], using greedy compression methods [24].

Sparse Stochastic Subspace Projections Since the update step (16) has complexity at least $\mathcal{O}(t)$ due to the parametrization induced by the RKHS, it is impractical in settings with streaming data or arbitrarily large training sets. We address this issue by replacing the stochastic quasi-descent step (16) with an orthogonally projected

variant, where the projection is onto a low-dimensional functional subspace of the RKHS $\mathcal{H}_{\mathbf{D}_{t+1}} \subset \mathcal{H}$

$$Q_{t+1} = \mathcal{P}_{\mathcal{H}_{\mathbf{D}_{t+1}}}[(1-\alpha_t\lambda)Q_t(\cdot) - \alpha_t(\gamma\kappa(\mathbf{s}_t',\mathbf{a}_t',\cdot)-\kappa(\mathbf{s}_t,\mathbf{a}_t,\cdot))z_{t+1})] \quad (21)$$

where $\mathcal{H}_{\mathbf{D}_{t+1}} = \text{span}\{((\mathbf{s}_n,\mathbf{a}_n),\cdot)\}_{n=1}^{M_t}$ for some collection of sample instances $\{(\mathbf{s}_n,\mathbf{a}_n)\} \subset \{(\mathbf{s}_t,\mathbf{a}_t)\}_{u\leq t}$. We define $\kappa_{\mathbf{D}}(\cdot) = \{\kappa((\mathbf{s}_1,\mathbf{a}_1),\cdot)\dots\kappa((\mathbf{s}_M,\mathbf{a}_M),\cdot)\}$ and $\kappa_{\mathbf{D},\mathbf{D}}$ as the resulting kernel matrix from this dictionary. We seek function parsimony by selecting dictionaries $\mathbf{D}$ such that $M_t << \mathcal{O}(t)$. Suppose that Q_t is parameterized by model points $\mathbf{D}_t$ and weights $\mathbf{w}_t$. Then, we denote $\tilde{Q}_{t+1}(\cdot) = (1-\alpha_t\lambda)Q_t(\cdot) - \alpha_t(\gamma\kappa(\mathbf{s}_t',\mathbf{a}_t',\cdot) - \kappa(\mathbf{s}_t,\mathbf{a}_t,\cdot))z_{t+1})$ as the SQG step without projection. This may be represented by dictionary and weight vector [cf. (18)]:

$$\tilde{\mathbf{D}}_{t+1} = [\mathbf{D}_t, (\mathbf{s}_t,\mathbf{a}_t), (\mathbf{s}_t',\mathbf{a}_t')] ,$$
$$\tilde{\mathbf{w}}_{t+1} = [(1-\alpha_t\lambda)\mathbf{w}_t, \alpha_t z_{t+1}, -\alpha_t\gamma z_{t+1}] , \quad (22)$$

where z_{t+1} in (22) is computed by (15) using Q_t obtained from (21):

$$\delta_t := r(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_t') + \gamma Q_t(\mathbf{s}_t',\mathbf{a}_t') - Q_t(\mathbf{s}_t,\mathbf{a}_t) ,$$
$$z_{t+1} = (1-\beta_t)z_t + \beta_t\delta_t . \quad (23)$$

Observe that $\tilde{\mathbf{D}}_{t+1}$ has $\tilde{M}_{t+1} = M_t + 2$ columns which is the length of $\tilde{\mathbf{w}}_{t+1}$. We proceed to describe the construction of the subspaces $\mathcal{H}_{\mathbf{D}_{t+1}}$ onto which the SQG iterates are projected in (21). Specifically, we select the kernel dictionary $\mathbf{D}_{t+1}$ via greedy compression. We form $\mathbf{D}_{t+1}$ by selecting a subset of M_{t+1} columns from $\tilde{\mathbf{D}}_{t+1}$ that best approximates $\tilde{Q}_{t+1}$ in terms of Hilbert norm error. To accomplish this, we use kernel orthogonal matching pursuit [25], [22] with error tolerance ε_t to find a compressed dictionary $\mathbf{D}_{t+1}$ from $\tilde{\mathbf{D}}_{t+1}$, the one that adds the latest samples. For a fixed dictionary $\mathbf{D}_{t+1}$, the update for the kernel weights is a least-squares problem on the coefficient vector:

$$\mathbf{w}_{t+1} = \kappa_{\mathbf{D}_{t+1}\mathbf{D}_{t+1}}^{-1}\kappa_{\mathbf{D}_{t+1}\tilde{\mathbf{D}}_{t+1}}\tilde{\mathbf{w}}_{t+1} \quad (24)$$

We must also tune ε_t to ensure both stochastic descent and finite model order.

We summarize the proposed method, KQ-Learning, in Algorithm 1, the execution of the stochastic projection of the functional SQG iterates onto subspaces $\mathcal{H}_{\mathbf{D}_{t+1}}$. We begin with the initial function null $Q_0 = 0$, with an empty dictionary and coefficients (Step 1). At each step, given an i.i.d. sample $(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_t')$ and step-size α_t, β_t (Steps 2-5), we compute the unconstrained functional SQG iterate $\tilde{Q}_{t+1}(\cdot) = (1-\alpha_t\lambda)Q_t(\cdot) - \alpha_t(\gamma\kappa(\mathbf{s}_t',\mathbf{a}_t',\cdot) - \kappa(\mathbf{s}_t,\mathbf{a}_t,\cdot))z_{t+1})$ parametrized by $\tilde{\mathbf{D}}_{t+1}$ and $\tilde{\mathbf{w}}_{t+1}$ (Steps 6-7), which are fed into KOMP (Algorithm 2) [25] with budget ε_t, (Step 8).

In order to implement Algorithm 1, we require the evaluation of the instantaneous maximizing action $\mathbf{a}_t = \text{argmax}_{\mathbf{a}} Q_t((\mathbf{x},\mathbf{a})$. We use simulated annealing to approximate the maximum for a mixture of Gaussians (Radial Basis Functions, RBF). In Section IV, we assume that the argmax is evaluated exactly.

Algorithm 1 KQ-Learning

Input: $\{\alpha_t,\beta_t,\varepsilon_t\}_{t=0,1,2\dots}$

1: $Q_0(\cdot) = 0, D_0 = [], w_0 = [], z_0 = 0$
2: **for** $t = 0,1,2,\dots$ **do**
3: Obtain trajectory $(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_t')$ via exploratory policy
4: Compute max action: $\mathbf{a}' = \pi_t(\mathbf{s}') = \text{argmax}_{\mathbf{a}'} Q_t(\mathbf{s}_t',\mathbf{a}_t')$
5: Update temporal action diff. δ_t and aux. seq. z_{t+1}

$$\delta_t = r(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_t') + \gamma Q_t(\mathbf{s}_t',\mathbf{a}_t') - Q_t(\mathbf{s}_t,\mathbf{a}_t)$$
$$z_{t+1} = (1-\beta_t)z_t + \beta_t\delta_t .$$

6: Compute functional stochastic quasi-gradient step

$$\tilde{Q}_{t+1} = (1-\alpha_t\lambda)Q_t(\cdot) - \alpha_t(\gamma\kappa(\mathbf{s}_t',\mathbf{a}_t',\cdot) - \kappa(\mathbf{s}_t,\mathbf{a}_t,\cdot))z_{t+1}).$$

7: Update dictionary $\tilde{D}_{t+1} = [D_t,(\mathbf{s},\mathbf{a}),(\mathbf{s}',\mathbf{a}')]$, weights $\tilde{w}_{t+1} = [(1-\alpha_t\lambda)w_t, \alpha_t z_{t+1}, -\alpha_t\gamma z_{t+1}]$.
8: Greedily compress function with KOMP $(Q_{t+1},D_{t+1},w_{t+1}) = \textbf{KOMP}(\tilde{Q}_{t+1},\tilde{D}_{t+1},\tilde{w}_{t+1})$
9: **end for**
10: **return** Q

IV. Convergence Analysis

In this section, we shift focus to the task of establishing that the sequence of action-value function estimates generated by Algorithm 1 actually yield a locally optimal solution to the Bellman optimality equation, which, given intrinsic the non-convexity of the problem setting, is the best one may hope for in general through use of numerical stochastic optimization methods. Our analysis extends the ideas of coupled supermartingales in reproducing kernel Hilbert spaces [22], which have been used to establish convergent policy evaluation approaches in infinite MDPs (a convex problem), to non-convex settings, and further generalizes the non-convex vector-valued setting of [28].

Before proceeding with technical assumptions, we introduce a few definitions: the functional stochastic quasi-gradient of the regularized objective is defined as

$$\hat{\nabla}_Q J(Q_t,z_{t+1};\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_t') = (\gamma\kappa((\mathbf{s}_t',\mathbf{a}_t'),\cdot) - \kappa((\mathbf{s}_t,\mathbf{a}_t),\cdot))z_{t+1} + \lambda Q_t , \quad (25)$$

and its sparse-subspace projected variant as

$$\tilde{\nabla}_Q J(Q_t,z_{t+1};\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_t') = (Q_t - \mathcal{P}_{\mathcal{H}_{D_{t+1}}}[Q_t - \alpha_t\hat{\nabla}_Q J(Q_t,z_{t+1};\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_t')])/\alpha_t . \quad (26)$$

Note that the update may be rewritten as a stochastic projected quasi-gradient step rather than a stochastic quasi-gradient step followed by a set projection, i.e.,

$$Q_{t+1} = Q_t - \alpha_t \tilde{\nabla}_Q J(Q_t, z_{t+1}; \mathbf{s}_t, \mathbf{a}_t, \mathbf{s}'_t) \,. \quad (27)$$

With these definitions, we may state our main assumptions required to establish convergence of Algorithm 1.

Assumption 1 *The state space $\mathcal{S} \subset \mathbb{R}^p$ and action space $\mathcal{A} \subset \mathbb{R}^q$ are compact, and the reproducing kernel map may be bounded as*

$$\sup_{\mathbf{s}\in\mathcal{S},\mathbf{a}\in\mathcal{A}} \sqrt{\kappa((\mathbf{s},\mathbf{a}),(\mathbf{s},\mathbf{a}))} = K < \infty \quad (28)$$

Assumption 2 *The temporal action difference δ and auxiliary sequence z satisfy the zero-mean, finite conditional variance, and Lipschitz continuity conditions:*

$$\mathbb{E}[\delta \,|\, \mathbf{s},\mathbf{a}] = \bar{\delta} \,,\ \mathbb{E}[(\delta - \bar{\delta})^2] \le \sigma_\delta^2 \,,\ \mathbb{E}[z^2 \,|\, \mathbf{s},\mathbf{a}] \le G_\delta^2 \quad (29)$$

where σ_δ and G_δ are positive scalars, and $\bar{\delta} = \mathbb{E}\{\delta \mid \mathbf{s},\mathbf{a}\}$ is defined as the expected value of the temporal action difference conditioned on state $\mathbf{s}$ and action $\mathbf{a}$.

Assumption 3 *The functional gradient of the temporal action difference is an unbiased estimate for $\nabla_Q J(Q)$ and the difference of the reproducing kernels expression has finite conditional variance:*

$$\mathbb{E}[(\gamma\kappa((\mathbf{s}'_t,\mathbf{a}'_t),\cdot) - \kappa((\mathbf{s}_t,\mathbf{a}_t),\cdot))\delta] = \nabla_Q J(Q) \quad (30)$$

$$\mathbb{E}\{\|\gamma\kappa((\mathbf{s}'_t,\mathbf{a}'_t),\cdot) - \kappa((\mathbf{s}_t,\mathbf{a}_t),\cdot)\|^2_{\mathcal{H}} | \mathcal{F}_t\} \le G_Q^2 \quad (31)$$

Moreover, the projected stochastic quasi-gradient of the objective has finite second conditional moment as

$$\mathbb{E}\{\|\tilde{\nabla}_Q J(Q_t, z_{t+1}; \mathbf{s}_t, \mathbf{a}_t, \mathbf{s}'_t)\|^2_{\mathcal{H}} | \mathcal{F}_t\} \le \sigma_Q^2 \quad (32)$$

and the temporal action difference is Lipschitz continuous with respect to the action-value function Q. Moreover, for any two distinct δ and $\bar{\delta}$, we have

$$\|\delta - \bar{\delta}\| \le L_Q \|Q - \tilde{Q}\|_{\mathcal{H}} \quad (33)$$

with Q, $\tilde{Q} \in \mathcal{H}$ distinct Q-functions; $L_Q > 0$ is a scalar.

Assumption 1 regarding the compactness of the state and action spaces of the MDP holds for most application settings and limits the radius of the set from which the MDP trajectory is sampled. The mean and variance properties of the temporal difference stated in Assumption 2 are necessary to bound the error in the descent direction associated with the stochastic sub-sampling and are required to establish convergence of stochastic methods. Assumption 3 is similar to Assumption 2, but instead of establishing bounds on the stochastic approximation error of the temporal difference, limits stochastic error variance in the RKHS. Moreover, (33) is justified since the maximum of a continuous function is Lipschitz in the infinity norm, which can be related to the Hilbert norm through a constant factor. These are natural extensions of the conditions needed for vector-valued stochastic compositional gradient methods.

The compactness of $\mathcal{S}$ and $\mathcal{A}$ (Assumption 1) implies that $\mathcal{H}$ is a compact function space, which together with the closedness of Hilbert subspaces $\mathcal{H}_{D_t}$, mean Q_t is contained within compact sets for all t due to the use of set projections in (21), meaning

$$\|Q_t\|_{\mathcal{H}} \le D \text{ for all } t \,, \quad (34)$$

where $D > 0$ is some positive constant.

Theorem 1 *Consider the sequence z_t and $\{Q_t\}$ as stated in Algorithm 1. Assume the regularizer is positive $\lambda > 0$, Assumptions 1-3 hold, and the step-size conditions hold, with $C > 0$ a positive constant:*

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \sum_{t=1}^{\infty} \beta_t = \infty, \sum_{t=1}^{\infty} \alpha_t^2 + \beta_t^2 + \frac{\alpha_t^2}{\beta_t} < \infty, \varepsilon_t = C\alpha_t^2 \quad (35)$$

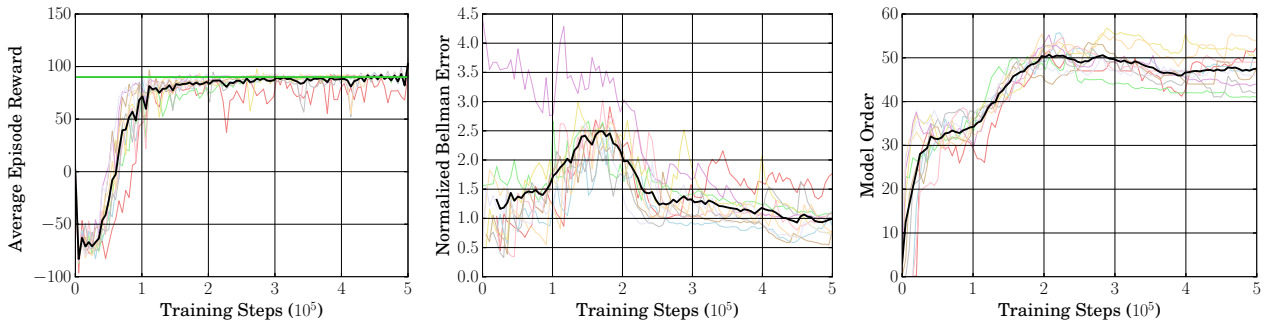
Then $\|\nabla_Q J(Q)\|_{\mathcal{H}}$ converges to null with probability 1, and hence Q_t attains a stationary point of (11). In particular, the limit of Q_t achieves the regularized Bellman fixed point restricted to the RKHS.

Theorem 1 establishes that Algorithm 1 converges almost surely to a stationary solution of the problem (11) defined by the Bellman optimality equation in a continuous MDP. See [1] for the proof. This is one of the first Lyapunov stability results for Q-learning in continuous state-action spaces with nonlinear function parameterizations, which are intrinsically necessary when the Q-function does not admit a lookup table (matrix) representation, and should form the foundation for value-function based reinforcement learning in continuous spaces. A key feature of this result is that the complexity of the function parameterization will not grow untenably large due to the use of our KOMP-based compression method which ties the sparsification bias ε_t to the algorithm step-size α_t. In the following section, we investigate the empirical validity of the proposed approach on a classic autonomous control task called the Continuous Mountain Car.

V. Experiments

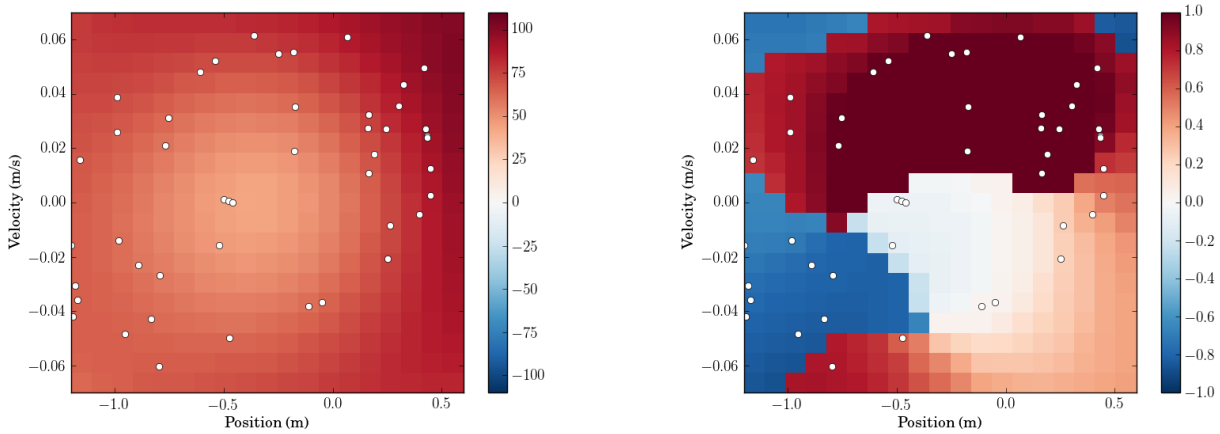
We benchmark KQ-Learning (Algorithm 1) on a classic control problem, the Continuous Mountain Car [30], which is featured in OpenAI Gym [35]. In this problem, the state space is $p = 2$ dimensional, consisting of position and velocity, bounded within $[-1.2, 0.6]$ and $[-0.07, 0.07]$, respectively. The action space is $q = 1$ dimensional: force on the car, within the interval $[-1, 1]$. The reward function is 100 when the car reaches the goal at position 0.6, and $-0.1a^2$ for any action a.

We used Gaussian RBFs with a fixed non-isotropic kernel bandwidth $\sigma_1 = 0.8$, $\sigma_2 = 0.07$, $\sigma_3 = 1.0$ for all experiments. The relevant parameters are the step-sizes α and β, the regularizer λ, and the approximation



(a) Average Training Reward (b) Normalized Test Bellman Error (c) Model Order of Q

Fig. 1: Results of 10 experiments over 500,000 training steps were averaged (black curve) to demonstrate the learning progress for the effective, convergent, and parsimonious solution. Fig. 1a shows the average reward obtained by the ε-greedy policy during training. An average reward over 90 (green) indicates that we have solved Continuous Mountain Car, steering towards the goal location. Fig. 1b shows the Bellman error for testing samples (6) normalized by the Hilbert norm of Q, which converges to a small non-zero value. Fig. 1c shows the number of points parameterizing the kernel dictionary of Q during training, which remains under 55 on average. Overall, we solve Continuous Mountain Car with a complexity reduction by orders of magnitude relative to existing methods [14], [34].



(a) Value function $V(\mathbf{s})$ derived from limiting $Q(\mathbf{s},\mathbf{a})$ (b) Policy $\pi(\mathbf{s})$ derived from limiting $Q(\mathbf{s},\mathbf{a})$

Fig. 2: The learned Q-function is easily interpretable: we may visualize the value function, $V(s) = \max_a Q(s,a)$ (2a) and corresponding policy $\pi(s) = \text{argmax}_a Q(s,a)$ (2b). In Fig. 2a, the color indicates the value of the state, which is highest (dark red) near the goal 0.6. At this position, for any velocity, the agent receives an award of 100 and concludes the episode. In Fig. 2b, the color indicates the force on the car (action), for a given position and velocity (state). The learned policy takes advantage of the structure of the environment to accelerate the car without excess force inputs. The dictionary points are pictured in white and provide coverage of the state-action space.

error constant, C, where we fix the compression budget $\varepsilon = C\alpha^2$. These learning parameters were tuned through a grid search procedure, which yielded the following selections: $\gamma = 0.99$, $\sigma = [0.8, 0.07, 1.0]$, $\lambda = 10^{-6}$, $\varepsilon = 0.1$, T = 5×10^5, $\alpha = 0.5$, $\beta = 0.5$. As the agent traverses the environment, we select actions randomly, initially with probability 1, and then exponentially decay this likelihood to 0.1 after 10^5 exploratory training steps.

The results of this experiment are given in Figure 1: here we plot the normalized Bellman test error Fig. 1b, defined by the sample average approximation of (6) divided by the Hilbert norm of Q_t over a collection of generated test trajectories, as well as the average rewards during training (Fig. 1a), and the model order, i.e., the number of training examples in the kernel dictionary (Fig. 1c), all relative to the number of training samples processed. Observe that the Bellman test error converges and the interval average rewards approach 90, which is the benchmark used to designate a policy as "solving" Continuous Mountain Car. This is comparable to existing top entries on the OpenAI Leaderboard [35], such as Deep Deterministic Policy Gradient [34] . Moreover, we

obtain this result with a complexity reduction by orders of magnitude relative to existing methods for Q-function and policy representation.

Additionally, few heuristics are required to ensure KQ-Learning converges, which is in contrast to neural network approaches to Q-learning. One shortcoming of our implementation is its sample efficiency, which could improved through an experience replay buffer. Such methods re-reveal past trajectory data to the agent based on the magnitude of their temporal action difference, for example, and have been shown to accelerate learning. Alternative, variance reduction, acceleration, or Quasi-Newton methods would improve the learning rate.

An additional feature of our method is the interpretability of the resulting Q function, which we use to plot the value function (2a) and policy (2b). One key metric is the coverage of the kernel points in the state-action space. We can make conclusions about the generalizability of the policy by the density of the model points throughout the space. Also, we can interpret which past experiences make an impact on the current value and policy evaluation. This may have particular importance in mechanical or econometric applications, where the model points represent physical phenomena or specific events in financial markets.

REFERENCES

[1] A. Koppel, E. Tolstaya, E. Stump, and A. Ribeiro, "Nonparametric stochastic compositional gradient descent for q-learning in continuous markov decision problems," *IEEE Trans. Autom. Control (under preparation)*, 2017. [Online]. Available: http://koppel.bitballoon.com/assets/papers/2017_kqlearning_report.pdf

[2] R. Bellman, "The theory of dynamic programming," DTIC Document, Tech. Rep., 1954.

[3] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, p. 0278364913495721, 2013.

[4] D. P. Bertsekas and S. E. Shreve, *Stochastic optimal control: The discrete time case*. Academic Press, 1978, vol. 23.

[5] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 2018.

[6] K. Mitkovska-Trendova, R. Minovski, and D. Boshkovski, "Methodology for transition probabilities determination in a markov decision processes model for quality-accuracy management," *Journal of Engineering Management and Competitiveness (JEMC)*, vol. 4, no. 2, pp. 59–67, 2014.

[7] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[8] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[9] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, UK, May 1989.

[10] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

[11] J. N. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," *Machine Learning*, vol. 16, no. 3, pp. 185–202, 1994.

[12] R. S. Sutton, H. R. Maei, and C. Szepesvári, "A convergent $o(n)$ temporal-difference algorithm for off-policy learning with linear function approximation," in *Advances in neural information processing systems*, 2009, pp. 1609–1616.

[13] S. Bhatnagar, D. Precup, D. Silver, R. S. Sutton, H. R. Maei, and C. Szepesvári, "Convergent temporal-difference learning with arbitrary smooth function approximation," in *Advances in Neural Information Processing Systems*, 2009, pp. 1204–1212.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[15] G. Kimeldorf and G. Wahba, "Some results on tchebycheffian spline functions," *Journal of mathematical analysis and applications*, vol. 33, no. 1, pp. 82–95, 1971.

[16] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Int. Conf. on Computational Learning Theory*. Springer, 2001, pp. 416–426.

[17] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *In Proc. of the Twelfth Int. Conf. on Machine Learning*. Morgan Kaufmann, 1995, pp. 30–37.

[18] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Trans. Autom. Control*, vol. 42, no. 5, pp. 674–690, 1997.

[19] N. K. Jong and P. Stone, "Model-based function approximation in reinforcement learning," in *Proc. int. joint conf. on Autonomous agents and multiagent systems*. ACM, 2007, p. 95.

[20] A. Shapiro, D. Dentcheva *et al.*, *Lectures on stochastic programming: modeling and theory*. Siam, 2014, vol. 16.

[21] Y. Ermoliev, "Stochastic quasigradient methods and their application to system optimization," *Stochastics: An International Journal of Probability and Stochastic Processes*, vol. 9, no. 1-2, pp. 1–36, 1983.

[22] A. Koppel, G. Warnell, E. Stump, P. Stone, and A. Ribeiro, "Breaking bellman's curse of dimensionality: Efficient kernel gradient temporal difference," *arXiv preprint arXiv:1709.04221 (Submitted to AAAI 2017)*, 2017.

[23] D. Ormoneit and Ś. Sen, "Kernel-based reinforcement learning," *Machine learning*, vol. 49, no. 2-3, pp. 161–178, 2002.

[24] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. on signal processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[25] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, "Parsimonious Online Learning with Kernels via Sparse Projections in Function Space," *ArXiv e-prints*, Dec. 2016.

[26] E. J. Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, no. 9, pp. 589–592, 2008.

[27] Y. Engel, S. Mannor, and R. Meir, "Bayes meets bellman: The gaussian process approach to temporal difference learning," in *Proc. of the 20th Int. Conf. on Machine Learning*, 2003.

[28] M. Wang, E. X. Fang, and H. Liu, "Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions," *Mathematical Programming*, vol. 161, no. 1-2, pp. 419–449, 2017.

[29] A. W. Moore, "Efficient memory-based learning for robot control," University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-209, Nov. 1990.

[30] A. Argyriou, C. A. Micchelli, and M. Pontil, "When is there a representer theorem? vector versus matrix regularizers," *J. Mach. Learn. Res.*, vol. 10, no. Nov, pp. 2507–2529, 2009.

[31] C. A. Micchelli, Y. Xu, and H. Zhang, "Universal kernels," *J. Mach. Learn. Res.*, vol. 7, no. Dec, pp. 2651–2667, 2006.

[32] V. S. Borkar and S. P. Meyn, "The ode method for convergence of stochastic approximation and reinforcement learning," *SIAM J. Control Optim.*, vol. 38, no. 2, pp. 447–469, 2000.

[33] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," in *Advances in neural information processing systems*, 2002, pp. 785–792.

[34] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[35] Openai gym - continuous mountain car. [Online]. Available: https://gym.openai.com/envs/MountainCarContinuous-v0/