Off-Policy Reinforcement Learning with Gaussian Processes

Girish Chowdhary
Oklahoma State University
218 Engineering South, Stillwater, OK

Miao Liu
Duke University
104B Hudson Hall, Durham, NC

Robert Grande
MIT
77 Massachusetts Ave, Cambridge, MA

Thomas Walsh
MIT
77 Massachusetts Ave, Cambridge, MA

Jonathan How
MIT
77 Massachusetts Ave, Cambridge, MA

Lawrence Carin
Duke University
128 Hudson Hall, Durham, NC

Abstract

An off-policy Bayesian nonparameteric approximate reinforcement learning framework, termed as GPQ, that employs a Gaussian Processes (GP) model of the value (Q) function is presented in both the batch and online settings. Sufficient conditions on GP hyperparameter selection are established to guarantee convergence of off-policy GPQ in the batch setting, and theoretical and practical extensions are provided for the online case. Empirical results demonstrate GPQ has competitive learning speeds in addition to its convergence guarantees and its ability to automatically choose its own bases locations.

1 Introduction

Reinforcement learning (RL) [39] in continuous or large state spaces often relies on function approximation to maintain a compact model of the value (Q) function [12, 14, 29]. The performance of approximate RL algorithms employing fixed feature function approximators, such as Radial Bases Function networks with a priori distributed centers, often depends on the choice of those features, which can be difficult to allocate. Gaussian Processes (GPs) [34] are Bayesian Nonparametric (BNP) models that are capable of automatically adjusting features based on the observed data. GPs have been successfully employed in high-dimensional approximate RL domains, such as a simulated octopus arm [13], but several aspects of their use, particularly convergence guarantees and off-policy learning have not been fully addressed.

More specifically, unlike RL algorithms employing a tabular representation, and even some function approximation techniques [14, 30], no convergence results for RL algorithms with GPs exist. Also, existing RL methods with GPs have either required burdensome computation in the form of a planner [33, 8, 18] or require that the policy being learned is the same as the one being executed (on-policy learning) [13]. The latter approach is less general than off-policy RL, which enables learning the optimal value function using samples collected with a safe or exploratory policy.

In this paper, we present a method for approximate RL using GPs that has provable convergence guarantees in the off-policy setting. More specifically, a model-free off-policy approximate reinforcement learning technique, termed as GPQ, that uses a GP model to approximate the value function is presented. GPQ does not require a planner, and because it is off-policy it can be used in both online or batch settings even when the data is obtained using a safe or exploratory policy. In addition, we present an extension of GPQ that uses a heuristic exploration strategy based on the GP's inherent measures on predictive confidence. In addition to presenting the GPQ framework, sufficient conditions for convergence of GPQ to the best achievable optimal Q-function given the data (Q^*) are presented in the batch and online setting, and it is shown that these properties hold even as new features are added or less-important features removed to maintain computational feasibility. Our work also contributes to off-policy approximate RL in general, because unlike other recent papers on off-policy RL with fixed-parameter linear function approximation [40, 29, 21, 27], our approach allows the basis functions to be automatically identified from data. Furthermore, our condition for convergence reveals why in the batch case GPQ or kernel base Fitted Q-Iteration [14] could lead to divergence in the worst case, and how the divergence can be prevented by tuning a regularization-like parameter of the GP. A practical online implementation of this framework that makes use of a recent budgeted online sparse GP inference algorithm [7] is empirically demonstrated, and it is theoretically shown how the sparsification affects GPQ's performance in the batch case. Our theoretical and empirical results show off-policy RL using a GP provides provable convergence guarantees and competitive learning speeds.

2 Background

The problem of sequential decision making under uncertainty can be formulated as a *Markov Decision Process* (MDP) [39]: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, with state space $\mathcal{S}$, action set $\mathcal{A}$, transition function $\mathcal{P} = p(s_{t+1}|s_t, a_t)$, reward function $\mathcal{R}(s,a) \mapsto \Re$, and discount factor $\gamma \in [0,1)$. A deterministic *policy* $\pi : \mathcal{S} \to Pr[\mathcal{A}]$ maps states to actions. Together with the initial state s_0, a policy forms a *trajectory* $\zeta = \{[s_0, a_0, r_0], [s_1, a_1, r_1], \cdots\}$ where $a_t = \pi(s_t)$, and both r_t and s_t are sampled from the reward and transition functions.

The state-action *value function* or *Q-function* of each state-action pair under policy π is defined as $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1}(s_{t+1}, a_{t+1}) | s_0 = s, a_0 = a\right]$ which is the expected sum of discounted rewards obtained starting from state s, taking action a and following π thereafter. The optimal Q-function Q^* satisfies $Q^* = \max_\pi Q^\pi(s,a)$ and is captured by the Bellman Equation: $Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r(s_t, a_t) + \max_{a'} \gamma Q^*(s_{t+1}, a')\right]$.

Throughout this paper, bounds are derived on $||\hat{Q} - Q^*||$ where $\hat{Q}$ is an approximation of Q^*. Deriving such bounds provides a bound on the value of the optimal policy derived from $\hat{Q}$ when it is executed in the real world, specifically an error factor of $\frac{2\gamma||\hat{Q}-Q^*||}{1-\gamma}$ is introduced [36].

2.1 Reinforcement Learning

Reinforcement learning is concerned with finding the optimal policy $\pi^*(s) = \operatorname{argmax}_a Q^*(s,a)$ when $\mathcal{P}$ and $\mathcal{R}$ are unknown. In *online* RL, an agent chooses actions to sample trajectories from the environment. In *batch* RL, a collection of trajectories is provided to the learning agent. In either case, *model-free* value-based reinforcement learning methods update an estimate of $Q(s,a)$ directly based on the samples. When an RL method learns the value function of the same policy with which samples were collected it is classified as *on-policy*; when the policy employed to obtain samples is different, it is termed *off-policy*. For instance, the Q-learning algorithm [44] performs an off-policy update of the form $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \Delta_t$, with a time dependent learning rate α_t and Temporal Difference (TD) error $\Delta_t = r(s_t, a_t) + \gamma \max_{a'} \big(Q_t(s_{t+1}, a')\big) - Q_t(s_t, a_t)$, where a' is the action that maximizes $\big(Q_t(s_{t+1}, a')\big)$. The on-policy counterpart to Q-learning is SARSA [39], which uses a different TD error based on the current policy: $\Delta_t = r(s_t, a_t) + \gamma\big(Q_t(s_{t+1}, a_{t+1})\big) - Q_t(s_t, a_t)$. Off-policy algorithms can be used in situations where batch data or "safe" exploration policies are used.

For continuous RL domains, linear value function approximation is often used to model the Q-function as a weighted combination of fixed bases $\phi(s,a)$, that is $Q(s_t, a_t) = \phi^T(s_t, a_t)\theta$. Then, the off-policy gradient descent operation from Q-learning can be used to update θ using the following TD error with $\phi_t = \phi(s_t, a_t)$: $\Delta_t = (r_t + \gamma \max_{a'} \phi^T_{t+1_{\max \theta_t}} \theta_t) - \phi_t^T \theta_t$. However, function approximation can cause divergence for off-policy RL methods, including Q-learning and linear-least squares approaches [1, 41]. It is possible to guarantee convergence of some off-policy algorithms under certain constraints. For instance, Melo et al. [30] provide sufficient conditions for approximate Q-learning's convergence: $\mathbb{E}_{\pi}\left[\phi_t \phi_t^T\right] \geq \gamma^2 \mathbb{E}\left[\phi_{t_{\max \theta_t}} \phi_{t_{\max \theta_t}}^T\right]$. This generally requires the initial policy to be very close to π^* and is heavily tied to the choice of bases. The potential for divergence in off-policy settings can limit the applicability of RL algorithms to safety critical domains.

2.2 Related Work

A series of papers [12, 13] showed GPs could be used to capture the value function using on-policy updates, specifically SARSA and Approximate Policy Iteration, which may not be able to learn the value of a new policy from batch data. Others have used GPs in model-based RL to learn the MDP parameters ($\mathcal{P}$ and $\mathcal{R}$) [33, 8], including a variant of the $R_{\max}$ algorithm that explored through an "optimism in the face of uncertainty" heuristic [18]. However, model-based approaches require a planner to determine the best action to take after each update, which could be computationally costly.

Several recent approaches ensure convergence in the online case by adding some form of regularization to TD algorithms for policy evaluation, as done in TDC [40]; GQ [29], LARS-TD [21], and RO-TD [27]. These algorithms use different versions of the SARSA-style TD error, but the regularization ensures their convergence even when the samples are obtained off-policy. In contrast to these algorithms, the GPQ algorithm presented here is an off-policy learning algorithm that attempts to directly minimize the Q-learning TD error. This TD error, which is directly derived from the Bellman optimality equations, is better suited for off-policy learning because it is the best approximation to the optimal Q-function given the samples that have been seen. Furthermore, many existing approaches rely on a fixed set of bases stipulated at initialization. In particular, [29] assumed the existence of a fixed set of features and Liu et al. [27] assume a fixed set of features from which features of less value are removed. In contrast, GPQ leverages a BNP approach to automatically choose features based on the data. Other algorithms, such as Bellman Error Basis Functions [32] dynamically construct features for linear function approximators, but these methods are primarily used in on-policy evaluation and, depending on their potential space of features, may not have the representational power of GPs. In the batch case, several algorithms belonging to the Fitted Q-Iteration (FQI) family of algorithms [14] can guarantee off-policy convergence with specific function approximators, including averagers [31] and certain forms of regularized least squares [15]. In contrast, our approach leverages GPs, which are powerful BNP function approximators. We show that Gaussian Processes can, with a prescribed setting of one regularization parameter, guarantee convergence in the batch case (which can be related to an FQI-like algorithm) and then describe how to extend this batch algorithm to the online case. In the online case, Geist and Pietquin [16] use a measurement model similar to GPQ and FQI, and handle the nonlinearity of that equation by using Extended/Unscented Kalman Filter like approaches, which can potentially diverge.

In Section 4.2 we introduce an exploration method based on "optimism in the face of uncertainty" using GP confidence intervals. Others have used confidence intervals on the MDP parameters themselves in a similar way in the discrete-state case [38]. Previous work has done similar exploration using GPs in supervised learning [22] and the bandit setting with continuous actions [9], but the latter is only for single-state RL whereas we explore in a full MDP. Other RL exploration heuristics that utilize GP variance include a strategy based on information theory [6], which more explicitly targets uncertain areas, but was designed for on-policy RL and requires a model to calculate long-term information gains.

Although rooted in the Bayesian framework, GPs have a deep connection with Reproducing Kernel Hilbert Space techniques [35], therefore, this work also augments the growing body of literature in Kernel based approximate MDPs and RL [24, 2, 5, 45].

2.3 Gaussian Processes

Gaussian Processes (GPs) [34] are BNP function approximation models: they do not specify a model structure a priori and explicitly model noise and uncertainty. A GP is defined as a collection of random variables, any finite subset of which has a joint Gaussian distribution with mean (prediction) function $m(z)$ and covariance kernel, such as a Radial Basis Function (RBF), $k(z', z)$, for input points z and z'. In the case of modeling the Q-function, the input domain is the space of all state action pairs and the model captures a distribution over possible Q-functions.

Let $Z = [z_1, \dots, z_\tau]$ be a set of state action pairs observed at discrete sample times, where each state action pair is concatenated as z for ease of exposition. In our analysis, we assume a finite set of actions, but all analysis extends to the continuous action space as well. Let $\vec{y} = [y_1, \dots, y_\tau]^T$ denote the vector of observation values at the respective state action pairs. Given some set of data points $\vec{y}$ at corresponding locations in the input domain Z, we would like to predict the expected value of the Q-function $y_{\tau+1}$ at some possibly new location $z_{\tau+1}$. Define $K(Z, Z)$ as the kernel matrix with entries $K_{l,m} = k(z_l, z_m)$. $\mathbf{k}(Z, z_{\tau+1}) \in \mathbb{R}^\tau$ denotes the kernel vector corresponding to the $\tau + 1^{th}$ measurement, and ω_n^2 represents the variance of the uncertainty in our measurement. Using Bayes law and properties of Gaussian distributions, the conditional probability can be calculated as a normal variable [34] with mean $m(z_{\tau+1}) = \alpha^T \mathbf{k}(Z, z_{\tau+1})$, where $\alpha = [K(Z, Z) + \omega_n^2 I]^{-1}\vec{y}$ are the kernel weights, and covariance

$$\Sigma(z_{\tau+1}) = k(z_{\tau+1}, z_{\tau+1}) + \omega_n^2 - \mathbf{k}^T(Z, z_{\tau+1})[K(Z, Z) + \omega_n^2 I]^{-1}\mathbf{k}(Z, z_{\tau+1}). \tag{1}$$

Performing batch prediction using GPs requires an expensive inversion of a matrix that scales as $O(\tau^3)$ with the size of the data. Many sparsification schemes have been proposed to reduce this computational complexity to $O(\tau m^2)$ [7, 26], where m is a number of reduced parameters. In this paper, we use the sparsification method of [7], which allows for sequential updates. This sparsification algorithm works by building a dictionary of basis vector points that adequately describe the input domain without including a basis point at the location of every observed data point. A full description of this algorithm is available in Appendix B. In order to determine when a new point should be added to the dictionary, a linear independence test is performed:

$$\beta_{\tau+1} = k(z_{\tau+1}, z_{\tau+1}) - \mathbf{k}(Z_d, z_{\tau+1})^T K(Z_d, Z_d)^{-1}\mathbf{k}(Z_d, z_{\tau+1}). \tag{2}$$

When $\beta_{\tau+1}$ is larger than a specified threshold β_{tol}, then a new data point is added to the dictionary. Otherwise, the weights α_τ are updated, but the dimensionality of α_τ remains the same.

3 Batch Off-Policy RL with a GP

As a slight abuse of notation, we let Q^* represent the best possible representation of the true Q-function given the data available. In GPQ, we model Q^* as a GP with mean function $m^*(s, a)$ and positive semi-definite covariance kernel $k([s, a], [s', a'])$. To do so, we place a zero mean Gaussian prior on Q, so $Q(s, a) \sim \mathcal{N}(0, k(\cdot, \cdot))$. Our goal is to perform posterior inference using available information so that the current estimate of the mean $\hat{m}$ approaches the mean of Q^*. Let the current estimate of the mean of the Q-function be $\hat{Q}(s, a) = \hat{m}(s, a)$. Since samples of Q^* are not available, posterior inference needs to be performed using the best estimate of Q^* at the current time as:

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \max_{a'}(\hat{Q}(s_{t+1}, a')). \tag{3}$$

Everytime we update the model of $\hat{Q}(s_t, a_t)$ with a new observation, the accuracy of the observation is dependent on the accuracy of the current model. Typically, the parameter ω_n^2 is viewed as a uncorrelated, Gaussian measurement noise in GP literature. Here, we offer an alternative interpretation of ω_n^2 as a regularization term which accounts for the fact that current measurements are not necessarily drawn from the true model and therefore prevents our model from converging too quickly to an incorrect estimate of Q^*. As we will show later, ω_n^2 plays a pivotal role in preventing divergence.

We now consider using the update rule in (3) with a GP model in the batch setting.

3.1 Batch GP-Fitted Q-Iteration

Using GPs and the update rule in Equation (3) in the batch setting gives us Algorithm 1, which we call GP-FQI because it is a member of the Fitted Q-Iteration family of algorithms. At each iteration, the values of the stored points are updated based on the stored rewards and transitions as well as the previous iteration's approximation of the Q-values, which is the form of Fitted Q-iteration.

While convergence guarantees exist for some function approximators used in FQI, such as Regression trees, others, such as Neural Nets, are known to be potentially divergent. Below, we prove that GP-FQI can diverge if the regularization parameter is not properly set. However, we also prove that for any set of hyperparameters and desired density of data, a proper regularization constant can be

Algorithm 1 Batch GPQ (GP-FQI)

```
1: Input: Experience tuples $\langle s, a, r, s' \rangle_{1...N}$
2: Output: A GP representing $\hat{Q}^*$
3: $\hat{Q} \leftarrow$ Initialized GP.
4: repeat
5:     $\hat{Q}' \leftarrow$ Initialized GP.
6:     for each experience tuple $\langle s, a, r, s' \rangle_i$ do
7:         $y_i = r_i + \gamma \max_b \hat{Q}(s', b)$
8:     Train $\hat{Q}$ on all $(\langle s_i, a_i \rangle, y \rangle_i$
9:     $\hat{Q} = \hat{Q}'$
10: until The convergence condition is satisfied
```

Algorithm 2 Online GPQ

```
1: for for each time step $\tau$ do
2:     Choose $a_\tau$ from $s_\tau$, using $\epsilon$-greedy exploration
3:     Take action $a_\tau$, observe $r_\tau$, $s_{\tau+1}$
4:     Let $z_\tau = \langle s, a \rangle$ and $y_\tau = r + \gamma \max_b \hat{Q}(s', b)$
5:     if $\beta_{\tau+1} > \beta_{tol}$ then
6:         Add $z_\tau$ to the $\mathcal{BV}$ set.
7:     Compute $\mathbf{k}_{z_{\tau+1}}$ and $\alpha_{\tau+1}$ according to [7]
8:     if $|\mathcal{BV}| >$ Budget then
9:         Delete $z_i \in \mathcal{BV}$ with lowest score according to [7]
10:    update $\hat{Q}(z_{\tau+1}) = \sum_{i=1}^{\infty} \alpha_i k(z_i, \cdot)$
```

determined to ensure convergence. We begin with a counter-example showing divergence in the batch setting if ω_n^2 is insufficient, but show convergence when ω_n^2 is large enough.

Consider a system with three nodes located along the real line at locations -1, 0,and 1. At each time step, the agent can move deterministically to any node or remain at its current node. The reward associated with all actions is zero. All algorithms are initialized with $\hat{Q}(z) = 1 \forall z$, $\gamma = 0.9999$, and we use a RBF kernel with bandwidth $\sigma = 1$ in all cases. We consider two settings of the regularization parameter, $\omega_n^2 = 0.1$ and $\omega_n^2 = 1$. Figure 1 shows that when ω_n^2 is set too low, the Bellman operation can produce divergence in the batch setting. If the regularization is set to the higher value, GP-FQI converges. In the following sections, we show that determining the sufficient regularization parameter ω_n^2 depends only on the density of the data and the hyperparameters, not the initialization value of $\hat{Q}$ or γ.

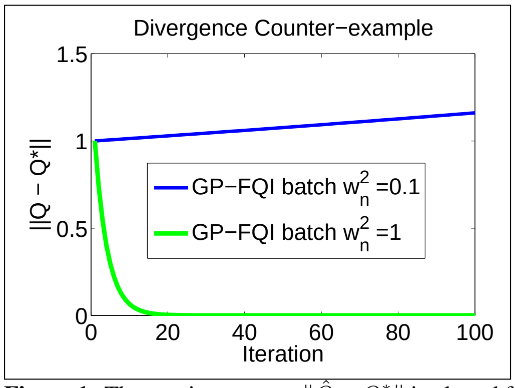

Figure 1: The maximum error $\|\hat{Q} - Q^*\|$ is plotted for GP-FQI with insufficient regularization $\omega_n^2 = 0.1$ and sufficient regularization $\omega_n^2 = 1$.

Let T denote the approximate Bellman operator that updates the mean of the current estimate of the Q-function using the measurement model of (3), that is $\hat{m}_{k+1} = T\hat{m}_k$, we argue that T is a contraction, so a fixed point exists. For a GP model, we define the approximate Bellman operator in the batch case as training a new GP with the observations $y_i = r(s_i, a_i) + \gamma \max_b \hat{Q}(s_i', b)$ at the input locations $z_i = (s_i, a_i)$.

In the following theorem, we show that in the case of finite data, a finite regularization term always exists which guarantees convergence. In the next theorem, we show that for a GP with infinite data which only adds data points which exceed the linear independence test β_{tol}, that a finite regularization term also exists. In theorem 3, we show that a finite bound exists, but do not compute the bound exactly as this depends on the topology of the space and kernel function.

Theorem 1. *Given a GP with data Z of finite size N, and Mercer kernel that is bounded above by $k_{\max}$, there exists a finite regularization parameter ω_n^2 such that the T is a contraction in the batch setting. In particular, $\omega_n^2 = 2(\|K(Z,Z)\|_\infty - k_{\max}) \leq 2N$*

Proof. Let $\|(\cdot)\|$ denote the infinity norm. Consider the approximate Bellman operator T

$$\|TQ(Z) - TP(Z)\| = \|K(Z,Z)(K(Z,Z) + \omega_n^2 I)^{-1}(\gamma Q(Z') - \gamma P(Z'))\| \quad (4)$$

$$\leq \gamma \|K(Z,Z)\| \|(K(Z,Z) + \omega_n^2 I)^{-1}\| \|(Q(Z) - P(Z))\| \quad (5)$$

If $\|K(Z,Z)\|\|(K(Z,Z)+\omega_n^2 I)^{-1}\| \leq 1$, then T is a contraction. By the structure of the kernel, we know $\|K(Z,Z)\| = \max_j \sum_i k(z_j, z_i)$ For finite data, the sum is a finite sum of finite numbers, so $\|K(Z,Z)\|$ is finite. For the second term, we assume that ω_n^2 is large and use [43] to bound the infinity norm of an inverse matrix that is strictly diagonally dominant. In order to be considered strictly diagonally dominant, a matrix A must have the property $a_{ii} > \sum_{j\neq i} a_{ij}, \forall i$. In this case,

$$\|(K(Z,Z)+\omega_n^2 I)^{-1}\| \leq \frac{1}{k_{\max} + \omega_n^2 - (\|K(Z,Z)\| - k_{\max})} \tag{6}$$

Therefore, setting $\omega_n > 2\|K(Z,Z)\| - 2k_{\max}$, we have

$$\|TQ(Z) - TP(Z)\| \quad \leq \quad \gamma\|Q(Z) - P(Z)\| \tag{7}$$

□

Theorem 2. *Given a GP with infinite data generated using a sparse approximation with acceptance tolerance β_{tol}, and given a Mercer kernel function that decays exponentially, there exists a finite regularization parameter ω_n^2 such that T is a contraction in the batch setting.*

Proof. To prove a contraction, we prove that $\|K(Z,Z)\|$ is finite so by Theorem 1 there exists a finite ω_n^2 such that (4) is a contraction. The norm of $\|K(Z,Z)\|$ is given by $\|K(Z,Z)\| = \max_j \sum_i k(z_j, z_i)$. We argue that the sum is convergent for an infinite number of data points selected using the linear independence test in (12). As the volume of the input domain increases to infinity, the number of data points added by a sparse selection increases polynomially, due to Ehrhart's Theorem, while the value they add to the sum decays exponentially fast. This means that the sum is convergent as the number of BV goes to infinity. The result follows from Theorem 1. For details, the reader is referred to Appendix C.1 □

Theorem 2 provides a powerful insight into the convergence properties of GPs. As the density of basis vectors increases or as the bandwidth of the kernel function grows, corresponding to decreasing β_{tol}, the basis vector weights α_i becomes increasingly correlated. As the weights become correlated, changing the weight at one basis vector also changes the weights of nearby basis vectors. It is this sharing of weights that can result in divergence, as seen in [1]. Theorem 2 shows that for a given β_{tol} and kernel function, there exists a finite regularization parameter ω_n^2 that will prevent divergence, however. This regularization technique can also be applied to provide convergence guarantees for FQI using a linear function approximator. In related work, [28] provides convergence guarantees for linear FQI using a similar regularization technique solved using an optimization-based framework. In practice, ω_n^2 does not have to be set very large to prevent divergence. Both Theorem 2 and [28] consider worst case analysis, which generally is not encountered in practice. For most applications, reasonable values of $\omega_n^2 \in [0.01, 1]$ will prevent divergence. In the next theorem, we bound the approximation error from using a sparse representation of a GP versus a full GP.

Theorem 3. *If the sparse GP algorithm is used, the error $\|\mathbb{E}[\hat{Q} - Q^*]\|$ is uniformly, ultimately bounded for the approximate Bellman operator.*

Proof. Let $\hat{Q}(s_t, a_t) = \hat{Q}_t$ for notational convenience. $\mathbb{E}[Q^*] = m^*(z) = \sum_{i\in\mathcal{BV}} \mathbf{k}^T(Z, z_i)\alpha_i^* + \sum_{i\notin\mathcal{BV}} \mathbf{k}^T(Z, z_i)\alpha_i^*$. Let $V(Q_t) = \|\mathbb{E}[\hat{Q}_t - Q^*_{\mathcal{BV}}]\|_\infty$ be a positive definite Lyapunov candidate. Let T denote the approximate Bellman operator applied to a sparse GP.

$$\begin{aligned} \|TE[\hat{Q}_t] - TE[Q^*]\| \quad &\leq \quad \|T\|\|\gamma \max_{a'} \sum_{i\in\mathcal{BV}} \mathbf{k}(Z,z)^T(\alpha_i^t - \alpha_i^*)\| + \|\sum_{i\notin\mathcal{BV}} \mathbf{k}(Z,z_i)\alpha_i^*\| &(8)\\ &\leq \quad \gamma\|T\|\|\mathbb{E}[\hat{Q}_t - Q_t^*]\| + |q_t|_\infty \beta_{tol}. &(9)\end{aligned}$$

where $q_t = (m(z_t) - \hat{m}(z_t))/(\Sigma(z_t))$ is the approximation error given in [7]. Let $c_1 = \gamma\|T\|$ noting that $c_1 < 1$. Let $c_2 = |q_t|\beta_{tol}$, which is finite. Hence $V(Q_{t+1}) - V(Q_t) \leq (c_1 - 1)V(Q_t) + c_2$, indicating that whenever $\|\mathbb{E}[\hat{Q}_t - Q^*_{BV}]\| \geq \frac{c_2}{1-c_1}$, $V(Q_{t+1}) - V(Q_t) \leq 0$. Therefore, the set $\left\{\hat{Q}_t : \|\mathbb{E}[\hat{Q}_t - Q^*_{BV}]\| \geq \frac{c_2}{1-c_1}\right\}$ is positively invariant, indicating that Q_t approaches and stays bounded within a compact neighborhood of Q^*. Furthermore, as $\beta_{tol} \to 0$, $Q_t \to Q^*$ uniformly.
□

In the proof Theorem 3, it should be noted that q_{t_∞} is always upper bounded. In particular, [20] shows that the upper bound on q_{t_∞} depends on the sparsification tolerance ϵ_{tol} in the online GP algorithm in [7].

4 Online Learning with GPQ

In the online case, an agent must collect data on its own, and the set of samples will increase in size. First consider a naïve implementation of GP-FQI in the online setting in which at every step, we run the GP-FQI algorithm to convergence. This leads to the following changes to GP-FQI:

- At each timestep t, the greedy action $a^* = \mathrm{argmax}_a \hat{Q}(s_t, a)$ is selected with probability $1 - \epsilon$, and with probability ϵ a random action is chosen.
- After each $\langle s, a, r, s' \rangle$ experience, GP_a is updated by adding the point $s, a, r + \gamma \max_b \hat{Q}(s', b)$. Determine ω_n^2 using Theorem 1 and perform GP-FQI as in Algorithm 1 .

In the batch sequential version of GP-FQI, ω_n^2 can be determined at each step by directly computing $\|K(Z,Z)\|$. Alternatively, one can use the linear independence test and only accept data points with $\beta_{\tau+1} \geq \beta_{tol}$. Given this knowledge, $\|K(Z,Z)\|$ can be computed a priori.

Corollary 1. *If Theorem 1 and Theorem 2 hold, then GP-FQI converges in the batch sequential algorithm provided the data points are sufficiently dense in the domain:* $\forall z \in S \times A : \exists c \in Z$ *s.t.* $k(z,z) - k(z,c)^2/k(c,c) \leq \beta_{tol}$.

At every iteration, GP-FQI will converge to a fixed estimate of Q^* at the current data locations. Provided that the policy used for obtaining samples ensures ergodicity of the induced Markov chain, the algorithm will converge to an estimate of Q^* everywhere. If we use a non-sparse GP with infinite data, we add the condition that there is zero probability of sampling a finite set of data points infinitely often in order to prevent $\|K(Z,Z)\|$ from growing to infinity. Additionally, if a sparse GP is used, then we know from Theorem 3 that our error is ultimately bounded.

4.1 Online GPQ

In theory, GP-FQI provides a method with provable convergence, however the computational requirements can be intense. In our empirical results, we employ several approximations to GP-FQI to reduce the computational burden. We call this modified algorithm *Online GPQ* and display it in Algorithm 2. At each step of Online GPQ, we take an action according to some policy π that ensures ergodicity of the induced Markov chain, and observe the value $y_\tau = r + \gamma \max_b \hat{Q}_\tau(s', b)$ at location z_τ. The sparse online GP algorithm of [7] is used to determine whether or not to add a new basis vector to the active bases set $\mathcal{BV}$ and then update the kernel weights. We provide a set of sufficient conditions for convergence in the online case.

Theorem 4. *For an ergodic sample obtaining policy π, and for each active basis set, a sufficient condition for convergence of $\hat{m}(z_t) \to m^*(z_t)$ as $t \to \infty$ online GPQ is* $\mathbb{E}_\pi\left[C_t \mathbf{k}_t \mathbf{k}_t^T + K_t^{-1} \mathbf{k}_t \mathbf{k}_t^T\right] \geq \gamma \mathbb{E}_\pi\left[C_t \mathbf{k}_t \mathbf{k}_t^\alpha + K_t^{-1} \mathbf{k}_t \mathbf{k}_t^\alpha\right]$, *where* $\mathbf{k}_t^\alpha \alpha_t = \max_{a'}(\mathbf{k}^T(x_{t+1}, a'))\alpha_t$.

Here, C_t is a negative definite and K_t^{-1} is a positive definite matrix related to the posterior and the prior covariance explained in [7]. These sufficient conditions for convergence are less restrictive than [30] for Q-learning. The proof style follows closely to that of [30]. For details, the reader is referred to Appendix C.2.

It should be noted that while the convergence results presented here are significant because no such results have been available before, these results only guarantee the asymptotic convergence of the Q function to the approximate Bellman operator's fixed point, within the projection of the selected bases. Which means the algorithm will eventually converge, yet no guarantees on the rate of convergence are provided here. Such guarantees are expected to depend on the choice of the exploration scheme, the algorithm's eventual selection of bases, and the rate at which the predictive variance decreases.

4.2 Optimistic Exploration for GPQ

The Online GPQ algorithm above used an ϵ-greedy exploration strategy, which may not collect samples in an efficient manner. We now consider a more targeted exploration heuristic facilitated by the GP representation. Others have considered similar heuristics based on information theory [6]. Here we use a simpler strategy based on the "optimism in the face of uncertainty" principle, which has been a cornerstone of efficient exploration algorithms (e.g. [38]).

In the discrete-state case, optimistic value functions can be maintained by initializing Q-values to $R_{\max}/(1-\gamma)$ and performing updates that maintain optimism until the values are nearly accurate. Pairing this over-estimate with greedy action selection causes the agent to explore areas of the state space that may yield higher long-term values, but not at the expense of "known" areas which have higher values grounded in real data. We propose using the upper confidence tails from the GP as an optimistic value function. Specifically, for any point $\langle s, a\rangle$, the GP will report an upper confidence tail of $m(s) + 2\mathbf{\Sigma}(s_{\tau+1})$ where m and $\mathbf{\Sigma}$ are defined in Section 2.3. We modify GPQ to use these optimistic estimates in two ways: change value used in GP update to $\hat{Q}(s_i, a_i) = r(s_i) + \gamma \max_a[\hat{Q}(s_{i+1}, a) + 2\mathbf{\Sigma}(s, a)]$, always take actions that are greedy with respect to the upper confidence tail.

The first change uses the upper tail of the next state's Q-value in the Bellman update to maintain optimism of the value function and is reminiscent of the backups performed in Model-Based Interval Estimation [38]. The second change makes the algorithm select greedy actions with respect to an optimistic Q-function.

A degree of caution needs to be used when employing this optimistic strategy because of potentially slow convergence rates for GPQ when large amounts of data has been collected. Once a large amount of data has been collected at a certain point, the confidence interval at that location can converge before GPQ has centered the mean around the true Q-value. These temporarily incorrect Q-values will still be optimistic and encourage exploration, but it may take a large amount of time for these points to converge to their true values, meaning this technique is prone to *over*-exploration. However, in many of our empirical tests, this variance-based exploration significantly outperformed ϵ-greedy exploration.

5 Results and Conclusions

Our experiments cover three domains, a discrete 5×5 Gridworld, a continuous state Inverted Pendulum (see [25]), and continuous state Puddle-World [4]. Specific details of the domains are listed in Appendix A. These domains pose increasingly more difficult challenges to GPQ when choosing basis points. The algorithms compared in each domain include the two variants of Online GPQ (ϵ-greedy and optimistic), both using a budgeting scheme [7]. We also implemented a tabular Q-learner (QL-Tab) using discretization, Q-learning with fixed-basis linear function approximation (QL-FB), and the GQ algorithm [29]. We chose these algorithms because they are each off-policy and model-free online approaches for dealing with continuous state spaces. We report results for the best case parameter settings (see Appendix A) of 1) the learning rate (QL-tab, FA-FB, GQ), 2) the exploration rate (QL-tab, FA-FB, GQ, GPQ-ϵ-greedy), 3) bandwidth of kernel (FA-FB, GQ, GPQ), 4) the position of kernels (GQ, FA-FB, GPQ) and 5) the number of kernels (quantization level for QL-tab). After cross-validation, the policy learned from all these methods for the three domains are evaluated based on discounted cumulative reward averaged over 20 independent runs and are shown in Figure 2.

The Gridworld consisted of 25 cells, noisy transitions, and a goal reward of 1 (step reward of 0). While all of the algorithms find the optimal policy, the GPQ based methods converge much faster by quickly identifying important areas for basis points. We also see that optimistic exploration using the GP's variance is advantageous, as the algorithm very quickly uncovers the optimal policy.

The Inverted Pendulum is a continuous 2-dimensional environment with the reward defined as the difference between the absolute value of the angle for two consecutive states. Again, GPQ quickly finds adequate bases and converges to a near optimal policy while GQ requires more samples. Q-learning with fixed bases and a tabular representation achieve adequate policies as well but require thousands of more samples. Optimistic exploration is not as helpful in this domain since the pen-

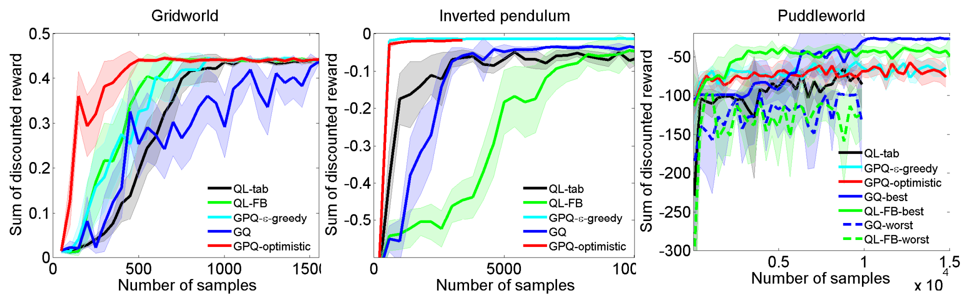

Figure 2: Average sum of discounted rewards for the experimental domains. The GQ and the QL variants are given more information because their bases are specified a priori, yet GPQ is able to reach comparable performance (often faster) while choosing its own basis functions.

dulum usually starts near the goal so targeted exploration is not required to find the goal region. Additional graphs in the appendix show that optimistic exploration is beneficial for certain parameter settings. The graphs in the appendix also demonstrate that GPQ methods are more resilient against small quantizations (budgets) because they are able to select their own bases, while GPQ and QL-FB are far more sensitive to the number and placement of bases.

Finally, we performed experiments in the Puddle-World domain, a continuous 2-dimensional environment with Gaussian transition noise and a high-cost puddle between the agent and the goal. Because of the large range of values around the puddle, basis placement is more challenging here than in the other domains and GPQ sometimes converges to a cautious (puddle adverse) policy. Multiple lines are shown for QL-FB and GQ, depicting their best and worst case in terms of parameter settings, as they were extremely sensitive to these settings in this domain. While the best case versions of GQ and QL-FB reached better policies than GPQ, in the worst case, their Q-values appear to *diverge*. While this is not immediately evident from the discounted-reward graph, an additional graph in the appendix shows the average steps to the goal, which more clearly illustrates this divergence. While GQ has convergence guarantees when data comes from a fixed policy, those conditions are violated here, hence the potential for divergence. In summary, while very careful selection of parameters for QL-FB and GQ leads to slightly better performance, GPQ performs almost as well as their best case with less information (since it does not need the bases a priori) and far outperforms their worst-case results.

6 Conclusions

In this paper, we presented a nonparametric Bayesian framework (GPQ) that uses GPs to approximate the value function in off-policy RL. We presented algorithms using this framework in the batch and online case and provided sufficient conditions for their convergence. We also described exploration scheme for the online case that made use of the GP's prediction confidence. Our empirical results show that GPQ's representational power allows it to perform as well or better than other off-policy RL algorithms. Our results also reveal that a regularization-like term can help decouple parameters such that divergence is avoided in an application of off-policy approximate reinforcement learning employing Gaussian kernels. In future work, we plan to analyze the theoretical properties of our optimistic exploration scheme and deploy the algorithm on practical robotics platforms.

Appendices

A Details of Empirical Results

Our experiments cover three domains, a discrete 5×5 Gridworld, a continuous state Inverted Pendulum (similar to [25]), and continuous state Puddle-World [4]. The Gridworld consisted of 25 cells

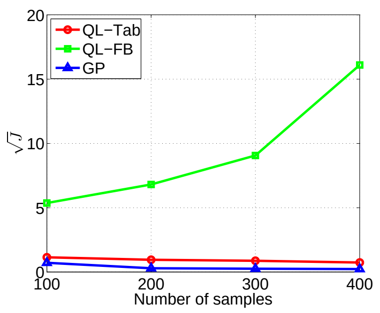

Figure 3: The performance of GPQ with ϵ-greedy exploration on Baird's counterexample (the "Star" problem) which can cause divergence with fixed-basis linear function approximation.

with the agent always starting in the lower-left corner, a 0 step cost, and a goal state in the upper-right corner with reward 1 and $\gamma = 0.9$. Transitions were noisy with a $.1$ probability of the agent staying in its current location. We also performed experiments on the Inverted Pendulum, a continuous 2-dimensional environment with three actions: applying forces of -50, 0, or 50 Newtons. The goal is to balance the pendulum upright within a threshold of $(-\pi/2, \pi/2)$ and the reward is defined as the difference between the absolute value of the angle for two consecutive states.

The Puddle-World domain is a continuous 2-dimensional environment with an agent moving in the four compass directions and Gaussian noise added to its movements. The initial state is at the bottom left of the domain and the goal region is near the top-right corner. The goal region and puddle placement follows the description of [4]. The "puddle" (which is really two overlapping puddles) between the agent and the goal causes negative reward proportional to the agent's distance to the center of each puddle. Steps outside of the puddle cause a reward of -1 except at the goal where the reward is 0.

The results are influenced by several parameters, including 1) the learning rate (QL-tab, FA-FB, GQ), 2) the exploration rate (QL-tab, FA-FB, GQ, GPQ-ϵ-greedy), 3) bandwidth of kernel (FA-FB, GQ, GPQ), 4) the position of kernels (GQ, FA-FB, GPQ) and 5) the number of kernels (quantization level for QL-tab). The learning rate is set as $0.5/t^{\alpha}$ with $\alpha \in \{0.1, 0.3, 0.5\}$, the exploration rate is set according to $1/t^{\beta}$ with $\beta \in \{0.1, 0.3, 0.5\}$. For Gridworld the kernel budget is 25; for Inverted Pendulum, the budget is chosen from $\{36, 100\}$ and for Puddle World we use budgets from $\{100, 400\}$. The quantization level for tabular Q-Learning is set the same as the budget for those function approximation methods. For each of the algorithms, the best combination of parameter settings was used for the figures in the main paper.

The following experiments from the Inverted Pendulum domain show the robustness of GPQ against changes in the quantization (budgeting) level and also the sensitivity of the fixed basis methods to the number of basis points. Figure 4 demonstrates that GPQ methods are more resilient against small budgets because they are able to select their own bases, while GPQ and QL-FB are far more volatile. These graphs also show that optimistic exploration is beneficial for certain parameter settings.

The graphs evaluating performance in Puddle World in Figure 5 include an additional graph that shows the average steps to the goal, which more clearly illustrates the divergence of GQ and QL-FB. The divergence in these worst cases is caused by having many bases (> 100) with overlapping bandwidths ($> .05$). In contrast, a smaller number of bases with the same small bandwidth actually produces the best performances for these algorithms, because weights are updated almost independently. Figure 6 elaborates on this sensitivity by showing the performance of all the algorithm under different budget (quantization) constraints with all other parameters fixed. We see GQ and QL-FB are very sensitive to the budget, and QL-tab is sensitive to the quantization level, while GPQ is relatively stable. In summary, while very careful selection of parameters for QL-FB and GQ leads to slightly better performance, GPQ performs almost as well as their best case with less information (since it does not need the bases a priori) and far outperforms their worst-case results.

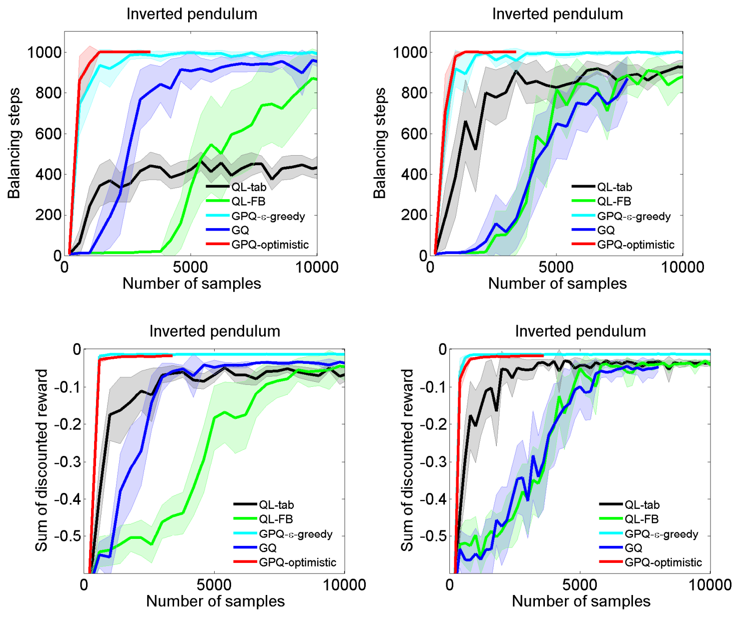

Figure 4: The performance for Inverted Pendulum under different budget (quantization levels), 36 (left) and 100 (right). GPQ is not sensitive to the quantization level because it selects its own bases, while the quantization level impacts the other algorithms significantly. For example, with higher a quantization level GQ converges slowly, and with a lower quantization level QL-tab performances poorly.

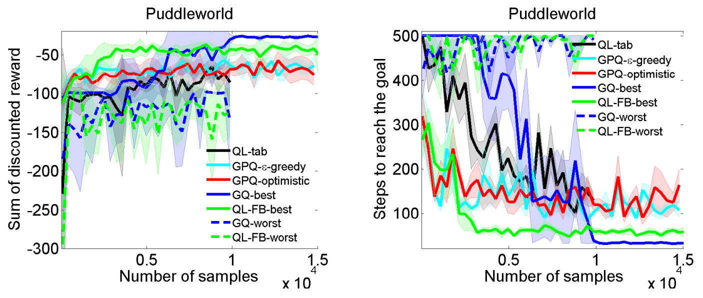

Figure 5: The performance of the algorithms in Puddleworld. The bandwidth is set to be 0.1 for the basis function in all methods. The number of basis (budget) is set to be 400. FA-FB and GQ diverge when the bases are placed uniformly over the state space.

B Sequential GP Updates

Performing batch prediction using GPs requires the inversion of a matrix that scales with the size of the data. Inverting this matrix at every iteration is computationally taxing, and many sparsification schemes have been proposed to reduce this burden [7, 37, 26]. In this paper, we use the sparsification method used in [7], which allows for sequential updates. The sparsification algorithm in [7] works building a dictionary of basis points that adequately describe the input domain without including a basis point at the location of every observed data points.

Given a dictionary of bases, Z_d, the prediction and covariance equations are computed as,

$$m(z_{\tau+1}) = \alpha_t^T k(Z_d, z_{\tau+1}) \tag{10}$$

$$\mathbf{\Sigma}(z_{\tau+1}) = k(z_{\tau+1}, z_{\tau+1}) + k^T(Z_d, z_{\tau+1}) C_t k(Z_d, z_{\tau+1}) \tag{11}$$

where $C = -(K + \omega^2 I)^{-1}$, i.e. C is the negative of the inverse of the regularized covariance matrix which is computed recursively. A natural and simple way to determine whether to add a new point

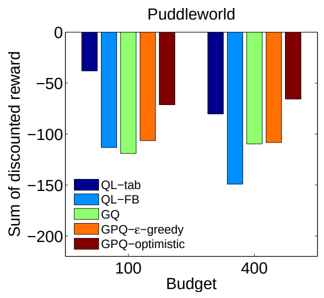

Figure 6: The performance for Puddleworld under different budget (quantization levels).

to the subspace is to check how well it is approximated by the elements in Z. This is known as the kernel linear independence test [7], and has deep connections to reproducing kernel Hilbert spaces (RKHS) [34]. The linear independence test measures the length of the basis vector $\phi(z_{\tau+1})$ that is perpendicular to the linear subspace spanned by the current bases. For GPs, the linear independence test is computed as

$$\beta_{\tau+1} = k(z_{\tau+1}, z_{\tau+1}) - k(Z_d, z_{\tau+1})^T K(Z_d, Z_d)^{-1} k(Z_d, z_{\tau+1}). \tag{12}$$

When $\beta_{\tau+1}$ is larger than a specified threshold ϵ_{tol}, then a new data point should be added to the dictionary. Otherwise, the associated weights α_τ are updated, but the dimensionality of α_τ remains the same. When incorporating a new data point into the GP model, the inverse kernel matrix and weights can be recomputed with a rank-1 update. To compute the updates in an online fashion, first define the scalar quantities

$$q^{(\tau+1)} = \frac{y - \alpha_\tau^T k_{x_\tau}}{\omega_n^2 + k^T(Z_d, z_{\tau+1}) C_t k(Z_d, z_{\tau+1}) + k(z_\tau, z_\tau)}, \tag{13}$$

$$r^{(\tau+1)} = -\frac{1}{\omega_n^2 + k^T(Z_d, z_{\tau+1}) C_t k(Z_d, z_{\tau+1}) + k(z_\tau, z_\tau)}, \tag{14}$$

Let $e_{\tau+1}$ be the $(\tau+1)$ coordinate vector, and let $T_{\tau+1}(\cdot)$ and $U_{\tau+1}(\cdot)$ denote operators that extend a τ-dimensional vector and matrix to a $(\tau+1)$ vector and $(\tau+1) \times (\tau+1)$ matrix by appending zeros to them, respectively. The GP parameters can be solved recursively by using the equations

$$\begin{aligned} \alpha_{\tau+1} &= T_{\tau+1}(\alpha_\tau) + q^{(\tau+1)} s_{\tau+1}, \\ C_{\tau+1} &= U_{\tau+1}(C_\tau) + r^{(\tau+1)} s_{\tau+1} s_{\tau+1}^T, \\ s_{\tau+1} &= T_{\tau+1}(C_\tau k_{x_{\tau+1}}) + e_{\tau+1}. \end{aligned} \tag{15}$$

The inverse of the matrix $K(Z,Z)$, denoted by P, needed to solve for $\gamma_{\tau+1}$ is updated online through the equation

$$P_{\tau+1} = U_{\tau+1}(P_\tau) + \gamma_{\tau+1}^{-1} \left(T_{\tau+1}(\hat{e}_{\tau+1}) - e_{\tau+1}\right) \left(T_{\tau+1}(\hat{e}_{\tau+1}) - e_{\tau+1}\right)^T \tag{16}$$

where $\hat{e}_{\tau+1} := P_\tau k(z_{\tau+1}, Z_d)^T$. In order to maintain a dictionary of fixed size, many point deletion criterion can be used [7]. Alternatively, [11] shows that if data is drawn from a Banach space and if $\epsilon_{tol} > 0$, then the number of points added to the dictionary will always be finite. If points are not deleted from the dictionary, the approximation error can be bounded analytically. In this paper, we consider a sparse representation of GPs without deletion of dictionary elements in order to facilitate analysis. Empirically, we have found that maintaining a fixed budget size does not significantly decrease performance.

C Convergence Proofs

C.1 Batch GP-FQI

Theorem 2. *Given a GP with infinite data generated using a sparse approximation with acceptance tolerance β_{tol}, and given a Mercer kernel function that decays exponentially, there exists a finite regularization parameter ω_n^2 such that the Bellman operator T is a contraction in the batch setting.*

Proof. We show that for a mercer kernel that decays exponentially, there always exists a finite ω_n^2 such that the Bellman operator results in a contraction in the supremum norm. For ease of exposition, all norms are assumed to be the infinity norm unless otherwise stated. Consider the Bellman operator

$$\begin{aligned} \|TQ(Z) - TP(Z)\| &= \|K(Z,Z)(K(Z,Z)+\omega_n^2 I)^{-1}(\gamma\vec{Q}(Z') - \gamma\vec{P}(Z'))\| & (17)\\ &\le \gamma\|K(Z,Z)\|\|(K(Z,Z)+\omega_n^2 I)^{-1}\|\|(Q(Z)-P(Z))\| & (18)\end{aligned}$$

To bound $\|K(Z,Z)\|\|(K(Z,Z)+\omega_n^2 I)^{-1}\| \le 1$, first consider the norm of $\|K(Z,Z)\|$. By the structure of the kernel, we know,

$$\|K(Z,Z)\| = \max_j \sum_i k(z_j, z_i) \qquad (19)$$

We now argue that the norm of $K(Z,Z)$ is bounded for an infinite number of basis vectors selected using the linear independence test in (19). The number of points that can be included in the basis vector set given the sparse selection procedure grows only polynomially as the volume of the input space increases. However, the kernel function decays exponentially fast as a function of distance. Therefore, as the volume of the input domain increases, the number of added kernels increases polynomially while the value they add to the sum (19) decays exponentially fast. This means that the sum is convergent as the number of BV goes to infinity. We formalize this notion.

The following argument assumes a squared exponential kernel function $k(z_i, z_j) = \exp(-\frac{1}{2}(z_i - z_j)^T M(z_i - z_j))$, $M > 0$ however, the results generalize to a variety of kernels that decay exponentially fast. Given some sparsity parameter β_{tol}, there exists an associated weighted distance δ_{tol} such that all points $\forall z \in \mathcal{BV}$, $(z_i - z_j)^T M(z_i - z_j) \ge \delta_{tol}^2, \forall i \ne j$. In particular, this bound can be computed as $\delta_{tol}^2 = -\ln(1-\beta_{tol})$.

To compute δ_{tol}^2, consider that the linear independence test $\beta_{z'} = k(z', z') - k(Z, z')^T K(Z,Z)^{-1} k(Z, z')$ is also the conditional variance of the point z' given the exact values of the GP mean at locations Z (the conditional variance with zero measurement noise) . Conditional variance is a monotonically decreasing function as a function of number of basis vector points [23]. That is, given more observations, the conditional variance will never increase. Therefore,

$$\begin{aligned} \beta_{z'} &\le k(z', z') - k(z_i, z')^T k(z_i, z_i)^{-1} k(z_i, z'), \quad \forall i & (20)\\ &\le 1 - \exp(-\frac{1}{2}(z_i - z')^T M(z_i - z'))^2 & (21)\end{aligned}$$

Therefore, if $\beta_{z'} \ge \beta_{tol}$, this implies $1 - \exp(-\frac{1}{2}(z_i - z')^T M(z_i - z'))^2 \ge \beta_{tol}$. In order to bound a minimum weighted distance $(z_i - z_j)^T M(z_i - z_j)$ between any two points such that both are could be included in a sparse GP,

$$\begin{aligned} \beta_{tol} &= 1 - \exp(-\frac{1}{2}(z_i - z')^T M(z_i - z'))^2 & \\ ((z_i - z')^T M(z_i - z'))^2 &= -\ln(1-\beta_{tol}) & (22)\\ \delta_{tol}^2 &= -\ln(1-\beta_{tol}) & (23)\end{aligned}$$

This gives us a bound on how close any two points can be in a sparse GP. We now assume the worst case analysis that all points are as compact as possible while satisfying 22. Given a volume in the input domain, distributing a maximal number of basis vectors is equivalent to performing a patterning operation with a lattice.

From Ehrhart's Theory[10] , we know that the number of lattice points in a polytope scales polynomially with the axis scaling factor, t, with maximum degree equal to the dimension of the space. In order to place an upper bound on (19) to show convergence, we begin by considering the case of concentric hyperspheres of radii $n\delta_{tol}$, $n \in \mathcal{Z}$ around a basis vector location. Each hypersphere contains a number of lattice points that grows polynomially as some function $\mathcal{L}(n\delta_{tol})$. To conservatively estimate $\sum_i k(z_j, z_i)$, one can add the total number of lattice points contained in the interior of the hyperspere $n+1$,$\mathcal{L}((n+1)\delta_{tol})$, at the distance $n\delta_{tol}$. This estimate is conservative in two ways. First, all of the lattice points inside hypersphere n are counted multiple times in the summation. Secondly, all of the lattice points at a distance between $[n\delta_{tol}, (n+1)\delta_{tol}]$ are given weight at distance $(n\delta_{tol})$. Summing over all hyperspheres, we have for any z_j,

$$\sum_i k(z_j, z_i) \leq \sum_{n=1}^{\infty} f((n+1)\delta_{tol}, \text{dim})\text{exp}(-(n\delta_{tol})^2) \tag{24}$$

where $f((n+1)\delta_{tol}, \text{dim})\text{exp}(-(n\delta_{tol})^2)$ is a polynomial function with highest degree dim. Substituting,

$$\|K(Z, Z)\| \leq \sum_{n=1}^{\infty} f((n+1)\delta_{tol}, \text{dim})\text{exp}(-(n\delta_{tol})^2) \tag{25}$$

One can easily verify that the right side of 25 converges by the integral test. Since $\|K(Z, Z)\|$ is finite, we know that there exists a finite ω_n^2 that results in contraction for the Bellman operator. □

C.2 Online GPQ

In the following we present a sufficient condition for convergence of online GPQ (see Section 4.1). The main idea is to show that the mean of the GP estimate of the Q-function, $\hat{m}$, converges to the mean m^*. Note that $m^*(z) = \mathbf{k}^T(Z, z)\alpha^*$, where α^* is the unique minimizer of the regularized least squares cost function $\sum_t \|Q^*(z_t) - \mathbf{k}^T(Z, z_t)\alpha^*\|^2$. For ease of exposition, we define the following notation: $\mathbf{k}(Z_t, z_t) = \mathbf{k}_t$, $K_t = K(Z_t, Z_t)$, and $\mathbf{k}_t^\alpha \alpha_t = \max_{a'}(\mathbf{k}^T(x_{t+1}, a'))\alpha_t$. Similar to proofs of other online RL algorithms, including TD learning ([42] and Q-learning [30], an ODE approach is used to establish stability [3]. The update in (15) is rewritten here for the case when the active basis set is fixed, that is, when $\beta_t \leq \beta_{tol}$

$$\begin{aligned} \alpha_{\tau+1} &= \alpha_\tau + \zeta_\tau q^{(\tau+1)}(s_{\tau+1}), \\ C_{\tau+1} &= (C_\tau) + \zeta_\tau r^{(\tau+1)} s_{\tau+1} s_{\tau+1}^T, \\ s_{\tau+1} &= (C_\tau k_{x_{\tau+1}}). \end{aligned} \tag{26}$$

where $\hat{s}_{t+1} = C_t \mathbf{k}_{t+1} + K_t^{-1}\mathbf{k}_{t+1}$ and ζ_τ is a decreasing sequence with $\sum_\tau \zeta_\tau = \infty$.

Theorem 4. *For an ergodic sample obtaining policy π, and for each active basis set $\mathcal{BV}$, a sufficient condition for convergence with probability 1 of $\hat{m}(z_t) \to m^*(z_t)$ as $t \to \infty$ when using the online sparse GP algorithm of* (26) *with the measurement model $\hat{Q}$ in (3) of the main paper is*

$$\mathbb{E}_\pi \left[C_t \mathbf{k}_t \mathbf{k}_t^T + K_t^{-1} \mathbf{k}_t \mathbf{k}_t^T \right] \geq \gamma \mathbb{E} \left[C_t \mathbf{k}_t \mathbf{k}_t^\alpha + K_t^{-1} \mathbf{k}_t \mathbf{k}_t^\alpha \right]. \tag{27}$$

Proof. Ensuring that assumptions required for Theorem 17 of [3] hold, the following ODE representation of the α update equation of 26 exists:

$$\dot{\alpha}(t) = \mathbb{E}_\pi \left[q_t S_t \right]. \tag{28}$$

In the following it is shown that $\alpha \to \alpha^*$ for each active basis set. Let $\tilde{\alpha} = \alpha - \alpha^*$ and consider the continuously differentiable function $V(\tilde{\alpha}) = \frac{1}{2}\tilde{\alpha_t}^T \tilde{\alpha}_t$. $V(\tilde{\alpha}_t) > 0$ for all $\tilde{\alpha}_t \neq 0$, therefore, V is a Lyapunov like candidate function [17, 19]. The first derivative of V along the trajectories of $\alpha(t)$ is

$$\dot{V}(\tilde{\alpha}) = \tilde{\alpha}_t^T \mathbb{E}_\pi \left[\frac{(C_t \mathbf{k}_t + K_t^{-1} \mathbf{k}_t)}{\sigma_x^2} (r_t + \gamma \mathbf{k}_t^\alpha \alpha_t - \mathbf{k}_t^T \alpha_t) \right], \tag{29}$$

where we have set $\sigma_x^2 = \omega_n^2 + \mathbf{k}^T(Z, z) C_t \mathbf{k}(Z, z') + k(z, z')$ for notational convenience. Note that $m^*(z) = \mathbf{k}^T(Z, z)\alpha^*$, and add and subtract $\mathbf{k}_t^T \alpha^*$ we have

$$\dot{V}(\tilde{\alpha}) = -\tilde{\alpha}_t^T \mathbb{E}_\pi \left[\frac{(C_t \mathbf{k}_t + K_t^{-1} \mathbf{k}_t)}{\sigma_x^2} \mathbf{k}_t^T \tilde{\alpha} + \tilde{\alpha}^T \mathbb{E}_\pi \left(\frac{(C_t \mathbf{k}_t + K_t^{-1} \mathbf{k}_t)}{\sigma_x^2} \right) (r_t + \gamma \mathbf{k}_t^\alpha \alpha_t - \mathbf{k}_t^T \alpha_t^*) \right]. \tag{30}$$

Adding and subtracting $\gamma \mathbf{k}_t^{al^*} \alpha^*$, noting that $\mathbf{k}_t^{al^*} \alpha^* \geq \mathbf{k}^\alpha \alpha^*$, and $m_t^* = r_t + \mathbf{k}^{\alpha^*} \alpha^*$ due to the Bellman equation, we have

$$\dot{V}(\tilde{\alpha}) = -\tilde{\alpha}_t^T \mathbb{E}_\pi \left[\frac{(C_t \mathbf{k}_t + K_t^{-1} \mathbf{k}_t)}{\sigma_x^2} \mathbf{k}_t^T - \gamma \frac{(C_t \mathbf{k}_t + K_t^{-1} \mathbf{k}_t)}{\sigma_x^2} \mathbf{k}_t^\alpha \right] \tilde{\alpha}. \tag{31}$$

Hence, if the condition in (27) is satisfied, $\dot{V}(\tilde{\alpha}) < 0$ and $\alpha \to \alpha^*$ w.p. 1 for each active basis set. □

The argument above can be extended to show that $m(z) \to m^*(z)$ even as new bases are added to the active basis set. To see this, let k be the time instant when a new basis is added and let α^*_{k+1} be the associated ideal weight vector. Note that the nature of the sparse GP algorithm ensures that $k+1 \geq k + \Delta T$ with $\Delta T > 0$. If ΔT is sufficiently large such that $\mathbb{E}_\pi(V(\alpha^*_{k+1}) - V(\alpha^*_k)) < 0$ then the convergence is uniform across all bases.

References

[1] L. C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *ICML*, pages 30–37, 1995.

[2] A. d. M. S. Barreto, D. Precup, and J. Pineau. Reinforcement learning using kernel-based stochastic factorization. In *NIPS*, pages 720–728, 2011.

[3] A. Benveniste, P. Priouret, and M. Métivier. *Adaptive algorithms and stochastic approximations*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[4] J. Boyan and A. Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Neural Information Processing Systems 7*, pages 369–376, 1995.

[5] X. Chen, Y. Gao, and R. Wang. Online selective kernel-based temporal difference learning. 2013.

[6] J. J. Chung, N. R. Lawrance, and S. Sukkarieh. Gaussian processes for informative exploration in reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013.

[7] L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[8] M. P. Deisenroth. *Efficient reinforcement learning using Gaussian processes*. PhD thesis, Karlsruhe Institute of Technology, 2010.

[9] T. Desautels, A. Krause, and J. W. Burdick. Parallelizing exploration-exploitation tradeoffs with gaussian process bandit optimization. In *ICML*, 2012.

[10] E. Ehrhart. Geometrie diophantienne-sur les polyedres rationnels homothetiques an dimensions. *Comptes rendus hebdomadaires des seances de l'academia des sciences*, 254(4):616, 1962.

[11] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *Signal Processing, IEEE Transactions on*, 52(8):2275–2285, 2004.

[12] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *International Conference on Machine Learning (ICML)*, 2005.

[13] Y. Engel, P. Szabo, and D. Volkinshtein. Learning to control an octopus arm with gaussian process temporal difference methods. In *Advances in Neural Information Processing Systems 18*, 2005.

[14] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, April 2005.

[15] A. M. Farahmand, M. Ghavamzadeh, C. Szepesvári, and S. Mannor. Regularized fitted q-iteration for planning in continuous-space markovian decision problems. In *Proceedings of the 2009 American Control Conference*, pages 725–730, 2009.

[16] M. Geist and O. Pietquin. Kalman temporal differences. *Journal of Artificial Intelligence Research (JAIR)*, 2010.

[17] W. M. Haddad and V. Chellaboina. *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach*. Princeton University Press, Princeton, 2008.

[18] T. Jung and P. Stone. Gaussian processes for sample efficient reinforcement learning with rmax-like exploration. In *European COnference on Machine Learning (ECML)*, 2012.

[19] H. K. Khalil. *Nonlinear Systems*. Macmillan, New York, 2002.

[20] H. Kingravi. *Reduced-Set Models for Improving the Training and Execution Speed of Kernel Methods*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 2013.

[21] J. Z. Kolter and A. Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[22] A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *Proceedings of the 24th international conference on Machine learning*, 2007.

[23] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *The Journal of Machine Learning Research*, 9:235–284, 2008.

[24] B. Kveton and G. Theocharous. Kernel-based reinforcement learning on representative states. In *AAAI*, 2012.

[25] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research (JMLR)*, 4:1107–1149, 2003.

[26] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse spectrum gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

[27] B. Liu, S. Mahadevan, and J. Liu. Regularized off-policy td-learning. In *Proceedings of Neural Information and Processing Systems*, Lake Tahoe, NV, Dec 2012.

[28] D. J. Lizotte. Convergent fitted value iteration with linear function approximation. In *Advances in Neural Information Processing Systems*, pages 2537–2545, 2011.

[29] H. R. Maei, C. Szepesvri, S. Bhatnagar, and R. S. Sutton. Toward off-policy learning control with function approximation. In *Proceedings of the International Conference on Machine Learning*, Haifa, Israel, 2010.

[30] F. S. Melo, S. P. Meyn, and M. I. Ribeiro. An analysis of reinforcement learning with function approximation. In *International Conference on Machine Learning (ICML)*, pages 664–671, 2008.

[31] D. Ormoneit and S. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2-3):161–178, 2002.

[32] R. Parr, C. Painter-Wakefield, L. Li, and M. L. Littman. Analyzing feature generation for value-function approximation. In *International Conference on Machine Learning (ICML)*, pages 737–744, 2007.

[33] C. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 751–759, 2004.

[34] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[35] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, December 2005.

[36] S. P. Singh and R. C. Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233, 1994.

[37] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1257–1264, 2006.

[38] A. L. Strehl and M. L. Littman. A theoretical analysis of model-based interval estimation. In *International Conference on Machine Learning (ICML)*, pages 856–863, 2005.

[39] R. Sutton and A. Barto. *Reinforcement Learning, an Introduction*. MIT Press, Cambridge, MA, 1998.

[40] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *ICML*, 2009.

[41] J. N. Tsitsiklis and B. V. Roy. An analysis of temporal difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, May 1997.

[42] J. N. Tsitsiklis and V. B. Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions On Automatic Control*, 42(5):674–690, 1997.

[43] J. M. Varah. A lower bound for the smallest singular value of a matrix. *Linear Algebra and Its Applications*, 11(1):3–5, 1975.

[44] C. J. Watkins. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[45] X. Xu, D. Hu, and X. Lu. Kernel-based least squares policy iteration for reinforcement learning. *Neural Networks, IEEE Transactions on*, 18(4):973–992, 2007.