Sparse Gaussian Process Temporal Difference Learning for Marine Robot Navigation

John Martin, Jinkun Wang, Brendan Englot
Department of Mechanical Engineering
Stevens Institute of Technology
{jmarti3, jwang92, benglot}@stevens.edu

Abstract: We present a method for Temporal Difference (TD) learning that addresses several challenges faced by robots learning to navigate in a marine environment. For improved data efficiency, our method reduces TD updates to Gaussian Process regression. To make predictions amenable to online settings, we introduce a sparse approximation with improved quality over current rejection-based methods. We derive the predictive value function posterior and use the moments to obtain a new algorithm for model-free policy evaluation, SPGP-SARSA. With simple changes, we show SPGP-SARSA can be reduced to a model-based equivalent, SPGP-TD. We perform comprehensive simulation studies and also conduct physical learning trials with an underwater robot. Our results show SPGP-SARSA can outperform the state-of-the-art sparse method, replicate the prediction quality of its exact counterpart, and be applied to solve underwater navigation tasks.

Keywords: Reinforcement Learning, Sparse Gaussian Process Regression

1 Introduction

Model-free Reinforcement Learning (RL) demands that robots produce lots of data to evaluate and improve their decision making policies. For marine robots, this can be challenging, since learning must be performed online, and their acoustics-based sensors produce data in low volumes. Here, we will present an algorithm that supports the learning of navigation policies with very little data.

Our algorithm belongs to the class of value estimation methods. Such methods are rooted in Bellman's equation, which describes the value of taking action $\mathbf{a}$ in state $\mathbf{s}$ as a sum of the expected reward and the forecasted value over a random transition $(\mathbf{s}, \mathbf{a}) \rightarrow (\mathbf{S}, \mathbf{A})$:

$$Q(\mathbf{s}, \mathbf{a}) = \mathbf{E}[R(\mathbf{s}, \mathbf{a})] + \gamma \mathbf{E}[Q(\mathbf{S}, \mathbf{A})].$$

The contractive nature of Bellman's equation motivates many standard algorithms, which apply the recursion to obtain better estimates of the true value [1, 2, 3]. Our method chooses a less conventional approach, but one which is more data-efficient, reducing Bellman updates to Gaussian Process (GP) regression [4].

GP regression is a flexible Bayesian method for learning unknown functions from data [5]. It uses a kernel-based covariance structure to promote a high degree of data association, well suited for learning when data is scarce. The main drawback is the prediction complexity, which scales prohibitively as $\mathcal{O}(N^3)$, where N is the number of data points [5]. To address this issue, we appeal to a sparse approximation.

Sparse approximations have been proposed to reduce the complexity of exact predictions [6, 4, 7, 8]. Many methods use an information criterion as the basis for rejecting redundant data. For GP-RL, Engel *et al.* employ the conditional covariance as a measure of relative error [4, 7], and they discard points that contribute little error to predictions. By limiting the active set to $M \ll N$ points, their predictions never cost more than NM^2 operations.

Besides discarding potentially useful information, the most prevalent issue with rejection sparsification is how it interferes with model selection. It can cause sharp changes to appear in the marginal likelihood and complicate evidence maximization with common optimizers, such as L-BFGS [9].

2nd Conference on Robot Learning (CoRL 2018), Zürich, Switzerland.

In what follows, we will describe a new method for GP value estimation that induces sparsity without discarding data. We approximate the exact TD regression framework of Engel *et al.* [4] with a smaller active set containing M adjustable support points. With this change, our method achieves the same prediction complexity as the state-of-the-art approximation, $\mathcal{O}(NM^2)$, while incurring less approximation error. Our method is based on the Sparse Pseudo-input Gaussian Process (SPGP) [10]. Therefore, it inherits many of SPGP's favorable characteristics, all of which we show support the unique needs of robots learning to navigate in a marine environment.

2 TD Value Estimation as GP Regression

TD algorithms recover the latent value function with data gathered in the standard RL fashion: at each step, the robot selects an action $\mathbf{a} \in \mathcal{A}$ based on its current state $\mathbf{s} \in \mathcal{S}$, after which it transitions to the next state $\mathbf{s}'$ and collects a reward $R \sim p_r(\cdot|\mathbf{s}, \mathbf{a})$. The repeated interaction is described as a Markov Decision Process, $(\mathcal{S}, \mathcal{A}, p_r, p_s, \gamma)$, associated with the transition distribution $\mathbf{s}' \sim p_s(\cdot|\mathbf{s}, \mathbf{a})$, stationary policy $\mathbf{a} \sim \pi(\cdot|\mathbf{s})$, and discount factor $\gamma \in [0, 1]$. As the name suggests, TD algorithms update a running estimate of the value function to minimize its difference from the Bellman estimate: $r + \gamma Q(\mathbf{s}', \mathbf{a}') - Q(\mathbf{s}, \mathbf{a})$; r being the observed reward. Once the estimate converges, a robot can navigate by selecting actions from the greedy policy π, such that $\mathbf{E}_\pi[\mathbf{A}|\mathbf{s}] = \arg\max_{\mathbf{a}\in\mathcal{A}} Q(\mathbf{s}, \mathbf{a})$.

The Gaussian Process Temporal Difference (GPTD) framework improves upon the data efficiency of frequentist TD estimation by departing from the contractive nature of Bellman's equation, in favor of a convergence driven by non-parametric Bayesian regression. The data model is based on the random return $Z(\mathbf{x}) = \sum_{t=0}^{\infty} \gamma^t R(\mathbf{x}_t)$, expressed as a sum of its mean, $Q(\mathbf{x})$, and zero-mean residual, $\Delta Q(\mathbf{x}) = Z(\mathbf{x}) - Q(\mathbf{x})$. Model inputs are state-action vectors $\mathbf{x} \in \mathcal{X} = \mathcal{S} \times \mathcal{A}$, and value differences are used to describe the observation process:

$$R(\mathbf{x}) = Q(\mathbf{x}) - \gamma Q(\mathbf{x}') + [\Delta Q(\mathbf{x}) - \gamma \Delta Q(\mathbf{x}')] = Q(\mathbf{x}) - \gamma Q(\mathbf{x}') + \varepsilon(\mathbf{x}, \mathbf{x}'). \tag{1}$$

Our work makes the simplifying assumption that noise levels, $\varepsilon(\mathbf{x}, \mathbf{x}')$, are i.i.d random variables with constant parameters, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Under this assumption, transitions exhibit no serial correlation. Although it is straightforward to model serially-correlated noise [7], doing so would preclude application of our desired approximation. We elaborate on this point in Section 3.

Given a time-indexed sequence of transitions $(\mathbf{x}_t, R(\mathbf{x}_t), \mathbf{x}_{t+1})_{t=0}^{N-1}$, we stack the model variables into vectors, resulting in the complete data model: $\mathbf{r} = \mathbf{H}\mathbf{q}(\mathbf{x}) + \boldsymbol{\varepsilon}$, where $\mathbf{q} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{qq})$, and

$$\begin{pmatrix} R(\mathbf{x}_0) \\ R(\mathbf{x}_1) \\ \vdots \\ R(\mathbf{x}_{N-1}) \end{pmatrix} = \begin{pmatrix} 1 & -\gamma & 0 & \cdots & 0 \\ 0 & 1 & -\gamma & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \cdots & 1 & -\gamma \end{pmatrix} \begin{pmatrix} Q(\mathbf{x}_0) \\ Q(\mathbf{x}_1) \\ \vdots \\ Q(\mathbf{x}_N) \end{pmatrix} + \begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{pmatrix}. \tag{2}$$

Notice the commonality Equation 2 has with a standard GP likelihood model, $\mathbf{y} = \mathbf{f}(\mathbf{x}) + \boldsymbol{\varepsilon}$. Both models assume the outputs, $\mathbf{r} \sim \mathbf{y}$, are noisy observations of a latent function, $\mathbf{q} \sim \mathbf{f}$. What distingushes TD estimation is the presence of value correlations, imposed from Bellman's equation and encoded as temporal difference coefficients in $\mathbf{H}$. Used for exact GP regression, Equation 2 leads to the GP-SARSA algorithm: a non-parametric Bayesian method for recovering latent values [4].

As a Bayesian method, GP-SARSA computes a predictive posterior over the latent values by conditioning on observed rewards. The corresponding mean and variance are used for policy evaluation:

$$v(\mathbf{x}_*) = \mathbf{k}_{r*}^\top (\mathbf{K}_{rr} + \sigma^2 \mathbf{I})^{-1} \mathbf{r}, \qquad s(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_{r*}^\top (\mathbf{K}_{rr} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_{r*}. \tag{3}$$

Here, $\mathbf{K}_{qq}$ is the covariance matrix with elements $[\mathbf{K}_{qq}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{K}_{rr} = \mathbf{H}\mathbf{K}_{qq}\mathbf{H}^\top$, and $\mathbf{k}_{r*} = \mathbf{H}\mathbf{k}_*$, where $[\mathbf{k}_*]_i = k(\mathbf{x}_i, \mathbf{x}_*)$. Subscripts denote dimensionality, e.g. $\mathbf{K}_{qq} \in \mathbb{R}^{|\mathbf{q}|\times|\mathbf{q}|}$.

3 Sparse Pseudo-input Gaussian Process Temporal Difference Learning

The GP-SARSA method requires an expensive $N \times N$ matrix inversion, costing $\mathcal{O}(N^3)$. Since robots must estimate values online to improve their navigation policies, we appeal to an approximate method which induces sparsity in the standard data model (Equation 2). We expand the probability space with $M \ll N$ additional pseudo values, $\mathbf{u}$. Here, pseudo values serve a parametric role in a non-parametric

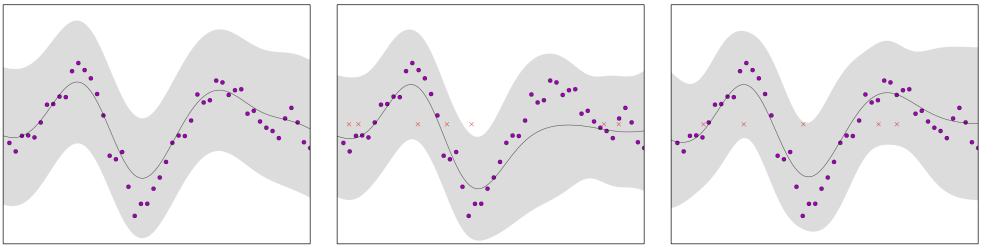


Figure 1: **Visualizing the approximate posterior:** We plot the predictive distributions of GP-SARSA (left) and our approximate method, SPGP-SARSA. We used $M = 7$ randomly-initialized pseudo inputs (red crosses) and $N = 50$ training samples (magenta) taken from the prior. The predictive mean is black, and two standard deviations of uncertainty are shown in gray. We plot SPGP-SARSA before (center) and after pseudo input optimization (right). After training, the sparse method is nearly identical to the exact method.

setting; they are free variables that provide probability mass at support locations $\mathbf{z} \in \mathbf{Z} \subset \mathcal{X}$. The extra latent variables obey the same data model as $\mathbf{q}$, but are predetermined, and thus, exhibit no noise. By conditioning $\mathbf{q}$ upon $\mathbf{u}$ and $\mathbf{Z}$, we can collapse the predictive probability space such that all dense matrix inversions are of rank M. Finally, we optimize $\mathbf{Z}$ to maximize the likelihood and produce a high-quality fit. This strategy is called Sparse Pseudo-input GP regression [10] (Figure 1).

Although SPGP regression is well-known, it has never been applied to TD estimation, where latent variables exhibit serial correlation. Therefore, current results from the sparse GP literature do not apply. In what follows we apply the SPGP principles to derive a new method for TD value estimation, which we call SPGP-SARSA. The method is data-efficient and fast enough for online prediction.

3.1 Latent Value Likelihood Model

Given an input location, $\mathbf{x}$, the likelihood of the latent value, $Q(\mathbf{x})$, is the conditional probability

$$p(Q|\mathbf{x}, \mathbf{Z}, \mathbf{u}) = \mathcal{N}(Q|\mathbf{k}_u^\top \mathbf{K}_{uu}^{-1}\mathbf{u}, k(\mathbf{x},\mathbf{x}) - \mathbf{k}_u^\top \mathbf{K}_{uu}^{-1}\mathbf{k}_u), \tag{4}$$

where $[\mathbf{K}_{uu}]_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$, $[\mathbf{k}_u]_i = k(\mathbf{z}_i, \mathbf{x})$. The complete data likelihood is obtained by stacking the N independent single transition likelihoods

$$p(\mathbf{q}|\mathbf{X}, \mathbf{Z}, \mathbf{u}) = \prod_{t=1}^{N} p(Q_t|\mathbf{x}_t, \mathbf{Z}, \mathbf{u}) = \mathcal{N}(\mathbf{g}, \widetilde{\mathbf{K}}); \tag{5}$$

where we defined $\mathbf{g} = \mathbf{K}_{qu}\mathbf{K}_{uu}^{-1}\mathbf{u}$, $\widetilde{\mathbf{K}} = \mathrm{diag}(\mathbf{K}_{qq} - \mathbf{K}_{qu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uq})$, with $[\mathbf{K}_{qu}]_{ij} = k(\mathbf{x}_i, \mathbf{z}_j)$.

3.2 Conditioned Data Likelihood Model

Here we derive the likelihood distribution of the observed rewards conditioned on the pseudo values $p(\mathbf{r}|\mathbf{X}, \mathbf{Z}, \mathbf{u})$. Consider the transformed joint distribution over values, rewards, and pseudo values

$$\begin{pmatrix}\mathbf{q}\\ \mathbf{r}\\ \mathbf{u}\end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{pmatrix}, \begin{pmatrix}\mathbf{K}_{qq} & \mathbf{K}_{qq}\mathbf{H}^\top & \mathbf{K}_{qu}\\ \mathbf{H}\mathbf{K}_{qq} & \mathbf{H}\mathbf{K}_{qq}\mathbf{H}^\top + \sigma^2\mathbf{I} & \mathbf{H}\mathbf{K}_{uq}\\ \mathbf{K}_{uq} & \mathbf{K}_{uq}\mathbf{H}^\top & \mathbf{K}_{uu}\end{pmatrix}\right).$$

The likelihood is obtained by conditioning $\mathbf{r}$ on $\mathbf{u}$ and invoking transition independence:

$$p(\mathbf{r}|\mathbf{X}, \mathbf{Z}, \mathbf{u}) = \mathcal{N}(\mathbf{K}_{ru}\mathbf{K}_{uu}^{-1}\mathbf{u}, \mathbf{Q} + \sigma^2\mathbf{I}), \qquad \mathbf{Q} = \mathrm{diag}(\mathbf{K}_{rr} - \mathbf{K}_{ru}\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}). \tag{6}$$

3.3 Posterior of Pseudo Values

To obtain the posterior $p(\mathbf{u}|\mathbf{r}, \mathbf{X}, \mathbf{Z})$, we use Bayes' rule. Given the marginal $p(\mathbf{u}|\mathbf{Z}) = \mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K}_{uu})$ and the conditional for $\mathbf{r}$ given $\mathbf{u}$ (Equation 6), the posterior for $\mathbf{u}$ given $\mathbf{r}$ is

$$p(\mathbf{u}|\mathbf{r}, \mathbf{X}, \mathbf{Z}) = \mathcal{N}(\mathbf{u}|\mathbf{L}\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}(\mathbf{Q} + \sigma^2\mathbf{I})^{-1}\mathbf{r}, \mathbf{L}); \tag{7}$$
$$\mathbf{L} = \mathbf{K}_{uu}(\mathbf{K}_{uu} + \mathbf{K}_{ur}(\mathbf{Q} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{ru})^{-1}\mathbf{K}_{uu}.$$

3.4 Latent Value Predictive Posterior

The predictive posterior is obtained by marginalizing the pseudo values:

$$p(Q_*|\mathbf{x}_*, \mathbf{r}, \mathbf{X}, \mathbf{Z}) = \int p(Q_*|\mathbf{x}_*, \mathbf{Z}, \mathbf{u}) p(\mathbf{u}|\mathbf{r}, \mathbf{X}, \mathbf{Z}) d\mathbf{u}.$$

Let $\mathbf{M} = \mathbf{K}_{uu} + \mathbf{K}_{ur}(\mathbf{Q} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{ru}$. Our new predictive is Gaussian with mean and variance

$$\tilde{v}(\mathbf{x}_*) = \mathbf{k}_{u*}^\top \underbrace{\mathbf{M}^{-1}\mathbf{K}_{ur}(\mathbf{Q} + \sigma^2\mathbf{I})^{-1}\mathbf{r}}_{\boldsymbol{\alpha}_\pi}, \qquad \tilde{s}(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_{u*}^\top \underbrace{(\mathbf{K}_{uu}^{-1} - \mathbf{M}^{-1})}_{\boldsymbol{\Lambda}_\pi} \mathbf{k}_{u*}. \tag{8}$$

Equation 8 represents our main contribution. The parameters $\boldsymbol{\alpha}_\pi$ and $\boldsymbol{\Lambda}_\pi$ are independent of the input, and their expressions depend on two inverses. The first is the dense M-rank matrix, $\mathbf{K}_{uu}$. The second is the N-rank diagonal matrix $(\mathbf{Q} + \sigma^2\mathbf{I})$. When $M \ll N$, both matrices are easier to invert than a dense N-rank matrix. Thus, Equation 8 provides motivation for estimating and predicting latent values efficiently.

3.5 Assumptions and Related Work

There are several key assumptions underpinning our results. To guarantee the likelihood can be factored, we need to assume that transitions are uncorrelated. Had we modeled serial correlation in the noise process, ε would be distributed with a tridiagonal covariance [7], which cannot be factored directly. It is possible to obtain a factorable model by applying a whitening transform. However, this changes the observation process to Monte Carlo samples, which are known to be noisy [11]. Under our simpler set of assumptions, we obtain an efficiently-computable, sparse representation of the value posterior that is amenable to smooth evidence maximization. Section 4.1 details how to select the hyperparameters and pseudo inputs with gradient-based optimization.

The most relevant methods to our work apply GP regression to estimate the latent value function in a TD setting [4, 7, 12, 13]. This class of algorithms is distinct from those which apply GP regression in the absence of sequential correlation, with Monte Carlo sample returns [14], and methods whose convergence behavior is driven primarily by the Bellman contraction [15, 16]. As a GP-TD algorithm, the policy update process (Algorithm 2) depends only on GP regression. This convergence is known to be data efficient and asymptotically unbiased [5]. We do not prove convergence for GP-TD algorithms here; however, we mention the convergence behavior to underscore our method's relevance to robots learning with limited volumes of data. We also note these methods have been scaled to high-dimensional systems with complex, continously-varying dynamics [13].

When it comes to other approximate GP-TD methods, the state-of-the-art uses a low-rank approximation to the full covariance matrix [7]: $\mathbf{K}_{qq} \approx \mathbf{A}\tilde{\mathbf{K}}^{-1}\mathbf{A}^\top$, where $\mathbf{A}$ is a projection matrix. Before adding new data to the active set, LOWRANK-SARSA checks if new data points increase the conditional covariance by a desired error threshold, ν. In Section 5.1, we compare the approximation quality of LOWRANK-SARSA and SPGP-SARSA.

4 New Algorithms for Robot Navigation

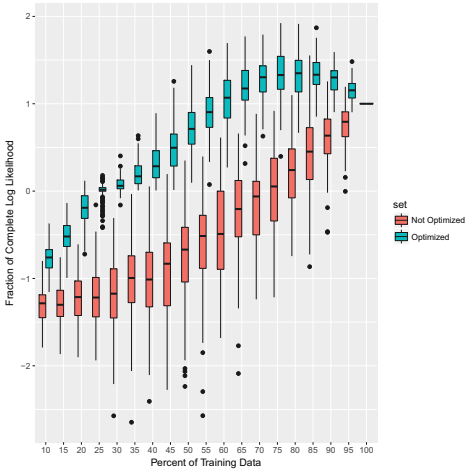
Algorithm 1 SPGP-SARSA

input: $\mathbf{x}_0$, π, $\boldsymbol{\theta}$, $\mathbf{Z}$
for $t = 1, \cdots, N$ **do**
  Observe $\mathbf{x}_{t-1}, r_{t-1}, \mathbf{x}_t$
  $\boldsymbol{\alpha}_\pi \leftarrow \mathbf{M}^{-1}\mathbf{K}_{ur}(\mathbf{Q} + \sigma^2\mathbf{I})^{-1}\mathbf{r}$
  $\boldsymbol{\Lambda}_\pi \leftarrow \mathbf{K}_{uu}^{-1} - \mathbf{M}^{-1}$
  Maximize Eq. 9 for optimal $\boldsymbol{\theta}$, $\mathbf{Z}$ (Optional)
output: $\boldsymbol{\alpha}_\pi, \boldsymbol{\Lambda}_\pi$

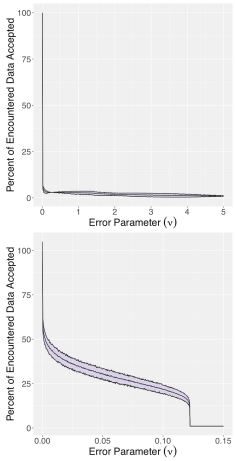
Algorithm 2 POLICY-ITERATION

input: $\mathbf{x}_0$, $\boldsymbol{\theta}$, $\mathbf{Z}$
$\pi \leftarrow \pi_{\text{init}}$
repeat:
  $\boldsymbol{\alpha}_\pi, \boldsymbol{\Lambda}_\pi \leftarrow$SPGP-SARSA$(\mathbf{x}_0, \pi, \boldsymbol{\theta}, \mathbf{Z})$
  $\pi \leftarrow$GREEDY-UPDATE$(\boldsymbol{\alpha}_\pi, \boldsymbol{\Lambda}_\pi)$
until: π converges
output: π

Navigation tasks are specified through the reward function. As a robot transitions through its operating space, it should assign the highest value to states and actions that bring it closer to the goal, and the



(a) SPGP-SARSA likelihood disparity



(b) LOWRANK-SARSA rejection statistics

Figure 2: **Visualizing the effect of parameter changes on approximation quality:** Figure 2a shows how the inclusion of more pseudo-inputs improves approximation quality, and how optimizating their locations is additionally benificial. We fixed a synthetic dataset of 100 samples from the prior, $\mathcal{N}(\mathbf{0}, \mathbf{K}_{qq})$. As a reflection of quality, we uniformly sampled subsets of the data, computed the log likelihood with SPGP-SARSA, both before and after optimization, and normalized it by the log likelihood of the full set. Boxplots were computed for 100 random subsets. Figure 2b shows how the low-rank method cannot always induce sparsity. We vary the error threshold, ν, and plot the percentage of samples LOWRANK-SARSA retained from 100 random trajectories. Trajectories from the Mountain Car system (top), and prior, $\mathcal{N}(\mathbf{0}, \mathbf{K}_{qq})$ (bottom) are shown.

lowest values around obstacles and other forbidden regions. We provide examples of such functions in Section 5.

Given a suitable reward function, Algorithm 1 implements SPGP-SARSA regression, where the posterior value function parameters $\boldsymbol{\alpha}_\pi, \boldsymbol{\Lambda}_\pi$ are learned with sequentially-observed data. The policy is updated using standard policy iteration, described in Algorithm 2.

4.1 Finding the Hyper-parameters and Pseudo-inputs

We use the marginal likelihood to fit the hyper-parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}, \sigma^2\}$ and pseudo inputs $\mathbf{Z}$ to the observed data, $\mathbf{r}, \mathbf{X}$. Unlike rejection-based sparsification, our method has the benefit of being continuous in nature. This allows for the variables to be tuned precisely to achieve a high-quality fit. The marginal likelihood is a Gaussian, given by

$$p(\mathbf{r}|\mathbf{X}, \mathbf{Z}) = \int p(\mathbf{r}|\mathbf{X}, \mathbf{Z}, \mathbf{u}) p(\mathbf{u}|\mathbf{Z}) d\mathbf{u} = \mathcal{N}(\mathbf{r}|\mathbf{0}, \mathbf{Q} + \sigma^2 \mathbf{I} + \mathbf{K}_{ru}\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}). \tag{9}$$

Instead of optimizing Equation 9 directly, we maximize its logarithm, $\mathcal{L} = \log p(\mathbf{r}|\mathbf{X}, \mathbf{Z})$. Given the gradient of $\mathcal{L}$, we can use an iterative method, such as L-BFGS, to find the optimal parameters. Full details of the gradient computation are provided in the supplement.

4.2 Training Considerations

The frequency with which model parameters are optimized can greatly influence the runtime of Algorithm 1. For D-dimensional inputs, SPGP-SARSA must fit $DM + |\boldsymbol{\Theta}|$ variables; whereas, GP-SARSA must fit only $|\boldsymbol{\Theta}|$. Although it is not strictly necessary to refit model parameters at each time step, the frequency which updates are needed will depend on how well $\mathbf{X}$ and $\mathbf{Z}$ reflect the support of the operating space. As new regions are explored, the model will need to be refit.

Even with a strategy to fit model parameters, we must still choose the number of pseudo inputs, M. In Figure 2, we examine the tradeoff between efficiency and accuracy in relation to M. As M increases, $\mathbf{Z}$ begins to coincide with $\mathbf{X}$, and prediction efficiency reduces, but the approximation improves to match the exact predictive posterior.

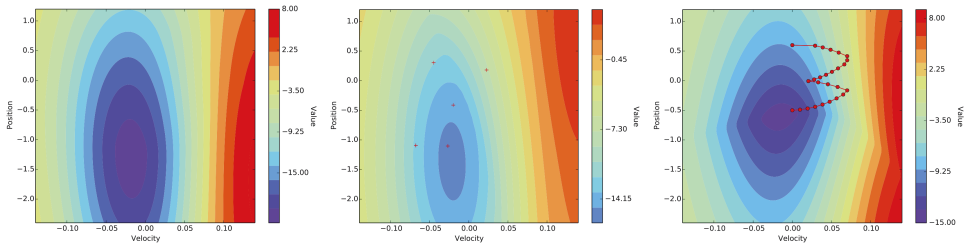
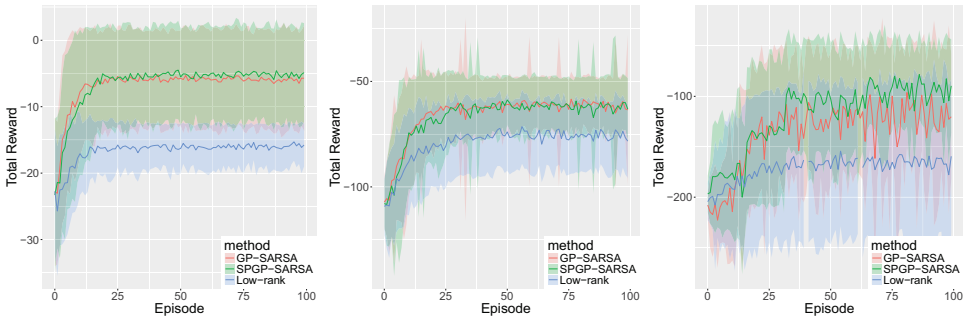


Figure 3: **SPGP-SARSA replicates value functions with a small active set:** Using the same covariance parameters, we plot value predictions from GP-SARSA (left) and SPGP-SARSA before (center) and after (right) pseudo-input optimization. With only a 5×5 matrix inversion, SPGP-SARSA nearly replicates the true value landscape, while GP-SARSA used a 302×302 inversion. The pseudo inputs (red crosses) moved beyond plot boundaries after optimization. A sample trajectory is shown at right.



(a) Mountain Car (b) USV (c) UUV

Figure 4: **Learning performance during policy improvement:** We plot average total reward over 100 episodes, along with one standard deviation. In each case, SPGP-SARSA performs as well as GP-SARSA; LOWRANK-SARSA is the least optimal, and all policies exhibit considerable variance.

Another consideration, when training GP models, is preventing overfitting. Overfitting occurs when the predictive variance collapses around the training data. It can be prevented by adding a regularization term to the log of Equation 9, penalizing the magnitude of covariance parameters and pseudo inputs.

4.3 SPGP TD Learning

To reduce SPGP-SARSA to its model-based equivalent, we let $\mathcal{X} = \mathcal{S}$ and swap the state-action transition process for the associated state transition process (Table 1); the analysis from Section 2 and 3 follows directly. The new input variable, $\mathbf{x} = \mathbf{s}$, simply reduces the latent function space to one over $\dim(\mathbf{s})$ variables. Equation 8 describes the state value posterior moments, analogous to frequentist TD [11] and standard GP-TD [4, 12]. We call the resulting algorithm SPGP-TD.

Target value $\tilde{v}(\mathbf{x})$	Input space $\mathcal{X}$	Transition Dist. $p_\pi(\mathbf{x}_t \vert \mathbf{x}_{t-1})$
$V(\mathbf{s})$	$\mathcal{S}$	$\int_{\mathcal{A}} p(\mathbf{s}_t \vert \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) \pi(\mathbf{a}_{t-1} \vert \mathbf{s}_{t-1}) d\mathbf{a}$
$Q(\mathbf{s}, \mathbf{a})$	$\mathcal{S} \times \mathcal{A}$	$p(\mathbf{s}_t \vert \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) \pi(\mathbf{a}_t \vert \mathbf{s}_t)$

Table 1: **Mapping from SARSA to TD:** Our method can estimate values under two processes.

5 Experimental Results

We presented SPGP-SARSA as a method for value estimation and explained how it applies to learning navigation policies. Now, we empirically verify several prior assertions: SPGP-SARSA is data efficient; it provides a more flexible and accurate approximation to GP-SARSA than LOWRANK-SARSA; it is suitable for online applications to marine robots. Evidence to support these claims is provided with several targeted simulation studies and a physical experiment using a BlueROV underwater robot.

All experiments use the same covariance function: $k(\mathbf{x}, \mathbf{x}') = \sigma_f \exp\left[-0.5(\Delta\mathbf{x}^\top \boldsymbol{\ell} \Delta\mathbf{x})\right]$, where $\Delta\mathbf{x} = (\mathbf{x} - \mathbf{x}')$, and $\boldsymbol{\ell}$ is a diagonal matrix of length scales.

5.1 Comparing Approximation Quality

To facilitate comparison between the approximation quality of SPGP-SARSA and LOWRANK-SARSA, we analyzed a measure of evidence maximization: the ratio of sparse-to-complete log-likelihood, $\mathcal{L}_{\text{SPARSE}}/\mathcal{L}_{\text{GPTD}}$. We found the low-rank approximation causes sharp changes in the likelihood, and its magnitude often varied around 10^3. Although this precluded visual comparisons, we are still able to show SPGP-SARSA provides a tight approximation. The ratio varies smoothly in relation to different levels of sparsity and improves further as pseudo inputs are optimized (Figure 2a).

In a second study, we examined the range of LOWRANK-SARSA's adjustability. In principle, the error threshold ν can be tuned to any positive number. However, results show the available range can be limited and result in extreme amounts of data retention or rejection (Figure 2b).

As marine robots require learning algorithms that are simultaneously data-efficient and online-viable, it can be problematic if one quality is missing. Retaining an excessive amount of data reduces the computational benefit of the low-rank approximation. Conversely, rejecting too much data is counterproductive when very little arrives. SPGP-SARSA offers a good middle ground, because it retains all observations while still achieving fast predictions at any level of sparsity.

5.2 Simulated Navigation Tasks

For this experiment, we solve the complete RL problem. Employing Algorithm 2, we compare performance of each value estimator, {GP, SPGP, LOWRANK}-SARSA, as they inform policy updates on simulated navigation tasks. The purpose is to understand each algorithm's learning performance.

From the data (Figure 4), it is clear that all algorithms converge quickly: in around fifty episodes. These results are consistent with prior work by Engel *et al.* [13], where GP-SARSA was applied to control a high dimensional octopus arm having 88 continuous state variables and 6 action variables. Performance differences between the two approximate methods are due to their approximation quality (Figure 2). As expected, SPGP-SARSA is able to learn on par with the exact method, because it can replicate the predictive posterior better than LOWRANK-SARSA.

In each experiment, robots learn over 100 episodes and select actions with unique ϵ-greedy policies. The number of pseudo inputs, M, was selected for a fair comparison. Specifically, after finding a ν that induced approximately 50% sparsity, we choose M so both methods converged with active sets of the same size. All pseudo inputs were initialized randomly.

First, we considered a canonical RL navigation problem, the Mountain Car [11]. With limited power, the robot must learn to exploit its dynamics to reach the crest of a hill. The state is given by $\mathbf{s} = (s, \dot{s})$, where $s \in [-1.2, 0.6]$, and $\dot{s} \in [-0.07, 0.07]$. Rewards are $R = \varepsilon - s$, with $\varepsilon \sim \mathcal{N}(0, 0.001)$, until the goal is reached, where $R = 1$. Episodes start at $\mathbf{s}_0 = (-0.5, 0.0)$, and the goal is $s_{\text{goal}} = 0.6$. We let $M = 5$, $\nu = 0.1$, and learning evolve over 50 transitions. One action, $a \in [-1, 1]$ controls the robot's motion (Figure 3) .

Our second system is a planar Unmanned Surface Vehicle (USV), which has been considered in prior learning experiments [17, 18]. The robot must navigate within 10m of $\mathbf{x}_{\text{goal}} = (50\text{m}, 50\text{m})$ using 100 transitions. States $\mathbf{s} = (x, y, \theta, \dot{\theta})$ contain position, x, y, heading. θ, and heading rate $\dot{\theta}$. The speed is held constant at $V = 3$ m/s, and the angular rate $\omega \in [-15°/\text{s}, 15°/\text{s}]$ controls the robot through

$$x_{t+1} = x_t + \Delta V \cos\theta_t, \qquad y_{t+1} = y_t + \Delta V \sin\theta_t,$$
$$\theta_{t+1} = \theta_t + \Delta\dot{\theta}_t, \qquad \dot{\theta}_t = \dot{\theta}_t + \frac{\Delta}{T}(\omega_t - \dot{\theta}_t).$$

We use time steps of $\Delta = 1.0$s, and a $T = 3$ step time delay for the command ω to be realized. The delay models resistance of surface currents and actuator limitations. Rewards are assigned with $R = R_{\text{min}} - (R_{\text{goal}} - R_{\text{min}}) \exp(-d/\delta) + \varepsilon$, where $R_{\text{min}} = -1.0$, $R_{\text{goal}} = 10.0$, ε as before, $d = ||\mathbf{x}_{\text{goal}} - \mathbf{x}||$, and $\delta = 10$. We chose a linear policy, $\omega = K_\omega e_\theta$, where $e_\theta = \arctan[(50 - y)/(50 - x)] - \theta$. K_ω was updated with a line search, to maximize the first moment of the value posterior. We selected $\nu = 0.1$ and $M = 50$.

For the third system we consider a common Unmanned Underwater Vehicle (UUV) design, with differential control. The robot commands forward acceleration, v, and turn rate, ω, through port, a_{port}, and starboard, a_{star}, actions. The dynamics are an extension of the USV with the additional

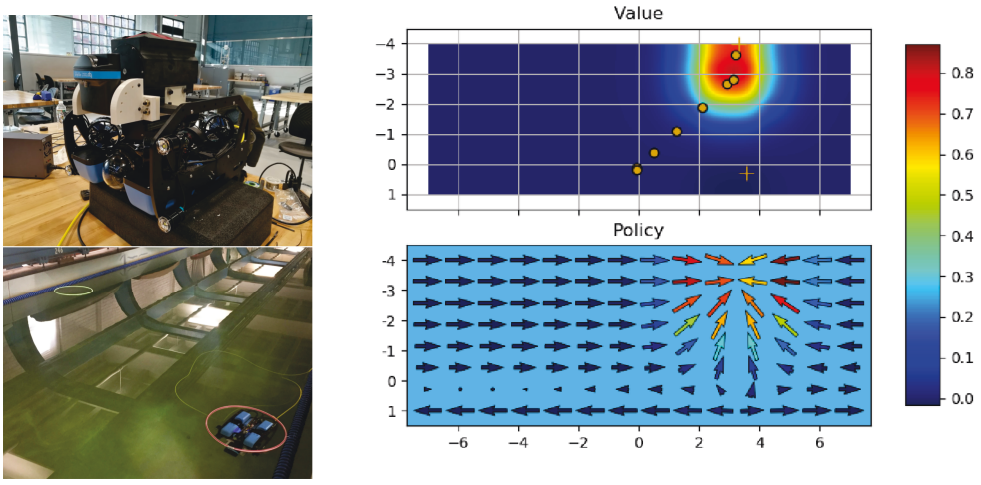


Figure 5: **Underwater robot navigation:** The robot (top-left) learns to navigate between two rings (bottom-left). With only a demonstration of eight transitions, SPGP-SARSA recreates a near-optimal value (top-right) and policy (bottom-right). Robot trajectory and the pseudo inputs are plotted in gold.

dimension, $V_{t+1} = V_t + \Delta v_t$. The policy uses Fourier basis functions, with e_θ defined as before, and $e_r = ||\mathbf{x}_{\text{goal}} - \mathbf{x}||$:

$$v = K_r e_r \cos(e_\theta), \qquad \omega = K_r \cos(e_\theta) \sin(e_\theta) + K_\theta e_\theta.$$

These map to actions through $v_t = K_v(a_{\text{port},t} + a_{\text{star},t})$ and $\omega_t = K_w(a_{\text{port},t} - a_{\text{star},t})$, where we let $K_v = K_w = 1$. Policy updates select parameters K_r and K_θ to maximize the value posterior mean - found through a 100×100 grid search. Learning occurs over 100 episodes of 200 transitions, with $\nu = 5$ and $M = 50$.

5.3 Learning to Navigate with a BlueROV

Solving the complete RL problem with a BlueROV (Figure 5) presented a unique set of challenges. States $\mathbf{s} = (x, y, \theta)$ derived from a Doppler velocity log, with estimates that drifted on the order of 1m every 1-2 min; this ultimately bounded the number of transitions per episode. While initiating the learning process at depth, disturbances from the data tether introduced uncertainty in the initial position and heading. The robot's speed is also constrained to facilitate accurate localization. To move, the robot can yaw and translate back and forth for a variable length of time.

With numerous limitations imposed on the learning process, we evaluated what could be learned from a single demonstration of only eight transitions. In practice, learning from demonstrations can reduce trial and error [19, 20]. Results show, even with minimal information, that SPGP-SARSA was able to recover a near-optimal value function and policy (Figure 5). The experiment was repeated twenty times and the average policy update time took $0.013 \pm 7 \cdot 10^{-4}$s with a 2.8GHz i7 processor. We achieve only a modest improvement in prediction time, since $N = 8$, and we use $M = 2$ pseudo inputs. Despite this fact, our results confirm SPGP-SARSA can support efficient robot learning.

6 Conclusion

This paper presented an algorithm that supports learning navigation policies with very little data. We argued for the use of GP-TD algorithms to replace standard Bellman recursion, because non-parametric regression can be more data-efficient for learning value functions. We derived SPGP-SARSA as a sparse approximation to GP-SARSA and showed it is more flexible and its predictions are more accurate than the state-of-the-art low-rank method. SPGP-SARSA was applied to a physical marine robot and learned a near-optimal value function from a single demonstration. In closing, we believe our results highlight the efficiency of GP-TD algorithms and the utility of SPGP-SARSA as a marine robot learning method.

Acknowledgments

We thank the anonymous reviewers for their excellent feedback. This research has been supported in part by the National Science Foundation, grant number IIS-1652064. This work was also supported in part by the U.S. Department of Homeland Security under Cooperative Agreement No. 2014-ST-061-ML0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

[1] C. Szepesvári. *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2010. URL http://www.sztaki.hu/~szcsaba/papers/RLAlgsInMDPs-lecture.pdf.

[2] R. S. Sutton. Learning to predict by the methods of temporal differences. In *MACHINE LEARNING*, pages 9–44. Kluwer Academic Publishers, 1988.

[3] G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical report, Cambridge University, 1994.

[4] Y. Engel, S. Mannor, and R. Meir. Bayes meets bellman: The gaussian process approach to temporal difference learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003.

[5] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[6] L. Csato and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 2002.

[7] Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 2005.

[8] H. Jakab and L. Csato. Improving gaussian process value function approximation in policy gradient algorithms. In *Artificial Neural Networks and Machine Learning*, 2011.

[9] D. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[10] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, 2006.

[11] R. Sutton and A. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

[12] Y. Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, The Hebrew University, 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*. MIT Press, 2006.

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

[15] M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomput.*, 2009.

[16] C. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[17] M. Ghavamzadeh, Y. Engel, and M. Valko. Bayesian policy gradient and actor-critic algorithms. *J. Mach. Learn. Res.*, 2016.

[18] J. Martin and B. Englot. Extending model-based policy gradients for robots in heteroscedastic environments. In *1st Annual Conference on Robot Learning*, 2017.

[19] B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 2009.

[20] J. Kober, A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 2013.

7 Supplement

7.1 Model Parameter Optimization

Most optimization packages require an objective function and its gradient to optimize. In the full paper, we described the object function as the log likelihood of the value posterior. Below, we provide the full details of the corresponding gradient computation.

7.2 Objective Gradient

Let $\mathbf{K}_r = \mathbf{Q} + \sigma^2\mathbf{I} + \mathbf{K}_{ru}\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}$ and ξ_j be the j-th optimization variable. The gradient with respect to ξ_j is

$$\frac{\partial \mathcal{L}}{\partial \xi_j} = -\frac{1}{2}\text{tr}(\mathbf{K}_r^{-1}\mathbf{J}_r) + \frac{1}{2}\mathbf{r}^\top\mathbf{K}_r^{-1}\mathbf{J}_r\mathbf{K}_r^{-1}\mathbf{r}. \tag{10}$$

Here, $\mathbf{J}_r$ is the tangent matrix of $\mathbf{K}_r$ with respect to ξ_j. The full equations for computing $\mathbf{J}_r$ are described in the next section.

7.3 Likelihood Covariance Tangent Matrix

Denote ξ_j to be a generic optimization variable. Then the matrix $\mathbf{J}_r$ used in Equation 10 is given by

$$\begin{aligned}
\mathbf{J}_r &= \frac{\partial \mathbf{Q}}{\partial \xi_j} + \frac{\partial}{\partial \xi_j}\sigma^2\mathbf{I} + \mathbf{K}_{ru}\frac{\partial}{\partial \xi_j}(\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}) + \mathbf{J}_{ru}(\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}),\\
\frac{\partial \mathbf{Q}}{\partial \xi_j} &= \text{diag}\big(\mathbf{J}_{rr} - \mathbf{K}_{ru}\frac{\partial}{\partial \xi_j}(\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}) - \mathbf{J}_{ru}(\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur})\big),\\
\frac{\partial}{\partial \xi_j}(\mathbf{K}_{uu}^{-1}\mathbf{K}_{ur}) &= \mathbf{K}_{uu}^{-1}\mathbf{J}_{ur} + \frac{\partial \mathbf{K}_{uu}^{-1}}{\partial \xi_j}\mathbf{K}_{ur},\\
\frac{\partial \mathbf{K}_{uu}^{-1}}{\partial \xi_j} &= -\mathbf{K}_{uu}^{-1}\mathbf{J}_u\mathbf{K}_{uu}^{-1}.
\end{aligned}$$