Extending Model-based Policy Gradients for Robots in Heteroscedastic Environments

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

Abstract: In this paper, we consider the problem of learning robot control policies in heteroscedastic environments, whose noise properties vary throughout a robot's state and action space. We consider reinforcement learning algorithms that evaluate policies using learned models of the environment, and we extend this class of algorithms to capture heteroscedastic effects with two enchained Gaussian processes. We explore the capabilities and limitations of this approach, and demonstrate that it reduces model bias across a variety of simulated robotic systems.

Keywords: Reinforcement Learning, Gaussian Processes, Heteroscedasticity

1 Introduction

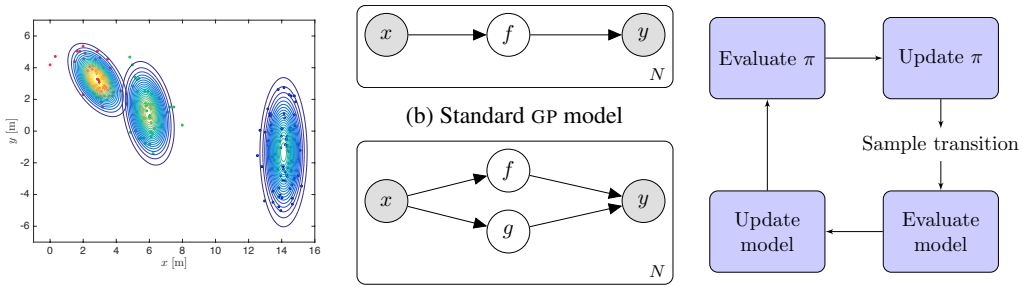
Model-based algorithms for Reinforcement Learning (RL) provide a data-efficient approach for learning robot control policies [1, 2, 3]. With these algorithms, robots use a model of the environment to obtain a simulation-based measure of a policy's utility [4]. When the model is easily computable and sufficiently consistent with the environment's true dynamics, simulations can accelerate learning by reducing the number of costly physical learning experiences [5].

Model-based algorithms have proven efficient in environments that exhibit homogeneous noise properties. However, there has been no work that extends their application to heteroscedastic domains; those for which noise properties vary throughout the state and action spaces. In our experience, localizing marine robots with a fixed acoustic beacon, heteroscedastic effects impart considerable uncertainty into a robot's transition dynamics (Fig. 1a). Failing to capture these effects in the RL process will result in inaccurate policy evaluation and poor task execution. To address this issue, we propose a method to extend the class of model-based RL algorithms with a richer transition model.

Several related methods employ Gaussian process (GP) regression to model transition dynamics [1, 3, 6]. GP regression offers a flexible and analytically-tractable framework for learning unknown functions [7]. Under standard assumptions, a robot perceives noisy outputs y from a latent function f expressing the robot's true dynamics. The observation process (Fig. 1b) is described by the model $y = f(\mathbf{x}) + \varepsilon$, where $f(\mathbf{x}) \sim \mathcal{N}(0, k_f)$ with covariance function $k_f(\mathbf{x}, \mathbf{x}')$. Furthermore, output dimensions are treated independently with i.i.d noise of constant variance $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. Since we are concerned with fielding robots in heteroscedastic environments, our approach considers a generative model with input-dependent noise, $y = f(\mathbf{x}) + \varepsilon(\mathbf{x})$, where $\varepsilon(\mathbf{x}) \sim \mathcal{N}(0, g(\mathbf{x}))$. Here, there are two latent functions, f and g, to describe the transition dynamics and the variance changes (Fig. 1c). By placing a GP prior over the logarithm of possible variance functions, $g(\mathbf{x}) = \exp(h(\mathbf{x}))$, where $h(\mathbf{x}) \sim \mathcal{N}(0, k_h)$, we obtain the hetroscedastic Gaussian process described originally by Goldberg *et. al* [8]. This model can be contextualized more generally as a Chained Gaussian Process using an identity and a log link function with two latent processes [9].

In this paper, we integrate the heteroscedastic GP model into a framework for indirect policy evaluation, whereby samples of the value function are obtained with rollouts from the model (Fig. 1d). We apply our framework to three systems; a point robot, an underactuated unmanned surface vessel (USV), and a six degree-of-freedom underwater vehicle. Our simulations demonstrate a significant reduction in model bias when using our framework over the standard model. Furthermore, our results indicate that model learning can be done efficiently, using reasonable amounts of data for short-range operations.

1st Conference on Robot Learning (CoRL 2017), Mountain View, United States.



(a) Heteroscedastic position data (b) Standard GP model (c) Heteroscedastic GP model (d) Model-based RL framework

Figure 1: **Motivating data and our models:** The robot localization data in Fig. 1a was obtained from an ultra-short baseline (USBL) acoustic positioning sensor located at the origin of the plot. It exhibits significant heteroscedastic effects as it tracks a marine robot that is placed at three different locations in a shallow-water environment. In order to model the changing variance in environments like these, we apply an extended model of the standard Gaussian process (GP) (Fig. 1b) that uses an additional latent function g for the noise process (Fig 1c). Both are integrated into a model-based framework for Reinforcement Learning (RL) (Fig. 1d), and their performance is compared in Sec. 3.

2 Model-based RL with Heteroscedastic Gaussian Processes

Here we describe the fundamentals of RL using heteroscedastic GPs for indirect policy evaluation. We present the transition and noise processes, and we show how to perform approximate inference to evaluate policies. Our procedure applies to a single output dimension, though when multiple outputs are considered, the analysis can be replicated for each additional dimension.

2.1 The Transition Process

As a robot operates, it collects sequences of transition samples $(\mathbf{x}_i, y_i)$, for $i = 1, \cdots, N$. We stack these into a design matrix $\mathbf{X}$ of state-action vectors $\mathbf{x} \in \mathbb{R}^n$ and a target vector $\mathbf{y}$ of state variables y. Together, the pair forms the training set of the f-process, $\mathcal{D}_f = \{\mathbf{X}, \mathbf{y}\}$. Observations $\mathbf{y}$ are assumed to be draws from a GP prior, where $\mathbf{f}(\mathbf{x})$ is a vector-valued transition function drawn from a GP.

Given the corresponding latent variances, $\mathbf{g} = (k_g(\mathbf{x}_1), \cdots, k_g(\mathbf{x}_N))^\top$, Goldberg *et. al* provides the conditional distribution $p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}, k_{g*}, \mathbf{g}) = \mathcal{N}(f_*|\mu, v)$, which we apply to predict transitions:

$$\mu(\mathbf{x}_*) = \mathbf{k}_{f*}^\top (\mathbf{K}_f + \mathbf{K}_g)^{-1}\mathbf{y}, \qquad v(\mathbf{x}_*) = k_f(\mathbf{x}_*, \mathbf{x}_*) + k_g(\mathbf{x}_*) - \mathbf{k}_{f*}^\top (\mathbf{K}_f + \mathbf{K}_g)^{-1}\mathbf{k}_{f*}. \quad (1)$$

Here, we have denoted $\mathbf{k}_{f*} = (k_f(\mathbf{x}_*, \mathbf{x}_1), \cdots, k_f(\mathbf{x}_*, \mathbf{x}_N))^\top$ for some input $\mathbf{x}_*$, and $\mathbf{K}_f$ to be the Gram matrix with elements $[\mathbf{K}_f]_{ij} = k_f(\mathbf{x}_i, \mathbf{x}_j)$. $\mathbf{K}_g$ is a diagonal matrix whose elements are the exponentiated mean function of the h-process, $k_g(\mathbf{x}) = \exp(\mu_h(\mathbf{x}))$.

Due to the presence of the exponential on $\mathbf{h}(\mathbf{x})$, the sum of $\mathbf{f}$ and $\mathbf{g}$ is no longer a GP. As a result, it will be impossible to marginalize the latent variables $\mathbf{g}_* = (k_g(\mathbf{x}_{*,1}), \cdots, k_g(\mathbf{x}_{*,N}))^\top$ and $\mathbf{g}$ from

$$p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}, \mathbf{g}_*, \mathbf{g}) p(\mathbf{g}_*, \mathbf{g}|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) d\mathbf{g}_* d\mathbf{g}. \quad (2)$$

Therefore, exact inference with the heteroscedastic GP is intractable. Several approximations to Equation 2 have been proposed. These include MCMC [8], variational inference [9, 10], Laplace approximation [11], expectation propagation [12], and maximum likelihood [13].

In robotics, the uncertainty of a given sensor usually constitutes a small fraction of the measurment's overall magnitude. For these applications, the probability mass of $p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y})$ will be concentrated highly around the mean. Therefore, a maximum likelihood assumption will provide a fair approximation of the true posterior.

2.2 The Noise Process

We proceed under the maximum likelihood assumptions of Kersting *et. al* [13]. The approximate posterior is given by

$$p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) \approx p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}, \mathbf{g}_*^*, \mathbf{g}^*), \qquad \{\mathbf{g}_*^*, \mathbf{g}^*\} = \arg\max_{\mathbf{g}_*, \mathbf{g}} p(\mathbf{g}_*, \mathbf{g}|\mathbf{x}_*, \mathbf{X}, \mathbf{y}), \tag{3}$$

where $\mathbf{g}_*$ and $\mathbf{g}$ are replaced by the maximizing variables of $p(\mathbf{g}_*, \mathbf{g}|\mathbf{x}_*, \mathbf{X}, \mathbf{y})$. To compute k_g, we condition $\mathbf{h} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_h)$ on the observations $\mathcal{D}_h = \{\mathbf{X}, \mathbf{z}\}$ and apply the exponential transformation to the standard GP posterior without noise $g \sim \exp(\mathcal{N}(h|\mu_h, v_h))$. The posterior functions of h are

$$\mu_h(\mathbf{x}) = \mathbf{k}_{h*}^\top \mathbf{K}_h^{-1} \mathbf{z}, \qquad v_h(\mathbf{x}) = k_h(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_{h*}^\top \mathbf{K}_h^{-1} \mathbf{k}_{h*}. \tag{4}$$

In practice, adding noise to the covariance matrix $\mathbf{K}_h$ increases numerical stability during inversion.

In Equation 4, the log variances $\mathbf{z}$ function as observed variables. However, in reality, the variance is a latent variable; it must be estimated from data when it cannot be marginalized. Kersting *et. al* proposes a simple sampling-based method for doing this using the empirical variance [13]. Draw M samples of the approximate posterior $\mathcal{N}(\mathbf{f}|\mu, v)$, assuming that each draw is an independent observation of the noise-free f-process. Using the noisy targets $\mathbf{y}$, the i-th variance observation is estimated with $z_i \approx \log\left(\frac{1}{M}\sum_{j=1}^{M} 0.5 \cdot (y_i - f_j)^2\right)$. This implicitly assumes that f_j and y_i are observed with equal likelihood for $i = 1, \cdots, N$. For large enough M, this estimate will minimize the average distance between the predictive distribution and the observations $\mathbf{y}$. We chose $M = 100$.

2.3 Model-based RL

Our objective for RL is to compute a deterministic policy, π, that maximizes the expected discounted return $\eta = \mathbf{E}_{\rho^\pi}\left[\sum_{t=0}^{T} \gamma^t R(\mathbf{x}_t) \mid \mathbf{x}_0 = \mathbf{x}\right]$. The distribution $\rho^\pi(\mathbf{x}_{0:T}) = p_0(\mathbf{x}_0)\prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1})$ depends on the latent transition dynamics, which our model will aid in evaluating. T is some finite or infinite horizon time span, and the constant $\gamma \in [0, 1]$ is used, optionally, to decay future rewards. To ensure consistency with our noise model, transitions are assumed to occur independently. In this paper, we treat decision-making policies $\pi_{\boldsymbol{\theta}}$ as deterministic mappings from states to actions, parameterized by the vector $\boldsymbol{\theta} \in \mathbb{R}^d$. The reward function $R(\mathbf{x})$ describes the robot's task and provides a real-valued measure of a policy's utility over a single transition.

Computing expectations under ρ^π can be analytically intractable. In continuous domains, there are an overwhelming number of terms to evaluate. By using a model-based framework (Fig. 1d), the expectation can be approximated by sampling ρ^π with model rollouts. Starting from some initial state and action $\mathbf{x}$, the model predicts the future transitions induced by using the policy π. Summing the associated rewards provides the agent with a sample of η.

The approach to RL we have just described shares many features with Bayesian optimization (BO) [14, 15]. Central to both learning methods is an unknown objective function, $\eta(\mathbf{x})$, that is expensive to evaluate and must be maximized. In BO, the acquisition function selects the evaluation points $\mathbf{x}$, whereas the policy serves this purpose in RL by inducing transitions for new evaluation data. Our approach uses a model to incorporate prior knowledge about the stochastic process of objective function evaluations. In contrast, BO places a prior directly over the objective function. The direct approach is most similar to value methods for Bayesian RL [16, 17, 18]. Even though the two methods have similar features, their overall purposes are different. RL seeks an optimal decision policy for its respective task, while BO seeks only to find the maximizing argument $\mathbf{x}^* = \arg\max_{\mathbf{x}} \eta(\mathbf{x})$.

An alternative method for approximating the full expectation η is presented by Deisenroth and Rasmussen [1]. Here, the authors obtain a closed-form result by modeling the randomness of transition perturbations, $\mathbf{x}' - \mathbf{x}$. Transitions are treated as independent and Gaussian, and moment-matching is applied to obtain expressions for η at every time step. Deisenroth and Rasmussen achieve unprecedented efficiency, but at the cost of considerable expressiveness in the reward function and policy. Both these functions must be polynomial or exponential to guarantee analytical tractability. While such requirements are not restrictive for a large variety of applications, having the freedom to select flexible policy classes is partially why RL is such a powerful framework for decision making.

2.4 Gradient-based Policy Improvement

We maximize the objective η by adjusting policy parameters $\boldsymbol{\theta}$ with gradient ascent. Direct methods for gradient-based policy improvement are referred to as Policy Gradient algorithms [19, 20, 21, 22]. For each update, the parameters $\boldsymbol{\theta}$ are adjusted in steps of $\beta \in \mathbb{R}$ toward local regions of increasing return. This is highly applicable to robots with sensitive dynamics, because changes in the policy can be configured to evolve smoothly:

$$\boldsymbol{\theta}_{j+1} \leftarrow \boldsymbol{\theta}_j + \beta_j \nabla \eta_j. \tag{5}$$

Provided that rewards are bounded and the learning rate β_j is chosen from a Robbins-Monroe sequence, policy gradient methods are guaranteed to converge to a local optimum [23]. By computing these numerically, we can support a large class of policies, even those with discontinuous parameterizations.

For d policy parameters, the partial derivative $\frac{\partial \eta}{\partial \theta_j}$ can be estimated by perturbing the j-th parameter θ_j by an amount $\Delta\theta_j$ and computing the resulting change in return $\Delta\eta_j = \eta(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}_j) - \eta(\boldsymbol{\theta} - \Delta\boldsymbol{\theta}_j)$. Sample returns come from model rollouts, where $\Delta\boldsymbol{\theta}_j$ is an elementary vector scaled by $\Delta\theta_j$ in the j-th dimension. Gradients are estimated with $\nabla\eta(\boldsymbol{\theta}) \approx \Delta\boldsymbol{\eta}/2\Delta\boldsymbol{\theta}$.

The top-level policy search algorithm is described in Algorithm 2. For each update cycle, N training trajectories are drawn from the true environment and added to the dataset $\mathcal{D}_f$. **Het_Update** is called to refine the current model predictions. The model is subsequently used to estimate the policy gradient and update the parameters $\boldsymbol{\theta}$. Policy updates are made with Algorithm 3. Given the current parameter sets $\boldsymbol{\theta}$, $\boldsymbol{\theta}_f$, and $\boldsymbol{\theta}_h$, along with the model data $\mathcal{D}_f$, $\mathcal{D}_h$, the time horizon T, and discount factor γ, we estimate the policy gradient numerically.

Better estimates of the policy gradient lead to improved policy updates. These are obtained as the transition predictions become more consistent with those in the true environment. As this process is repeated, the policy converges onto a parameterization that locally maximizes the expected return η.

Algorithm 1 Heteroscedastic Model Update

1: **Het_Update**$(\boldsymbol{\theta}_f, \boldsymbol{\theta}_h, \mathcal{D}_f, M)$
2: Given initial parameters $\boldsymbol{\theta}_f$, $\boldsymbol{\theta}_h$, dataset $\mathcal{D}_f$, and the variance sample size M, compute
3: $\mathcal{D}_h \leftarrow \emptyset$
4: **for all** $(\mathbf{x}_i, y_i) \in \mathcal{D}_f$ **do**
5: Compute the f-process posterior $\mathcal{N}(\mu(\mathbf{x}_i), v(\mathbf{x}_i))$
6: Draw M samples $\{f_1, \cdots, f_M\}$ from $\mathcal{N}(\mu(\mathbf{x}_i), v(\mathbf{x}_i))$
7: $z_i \leftarrow \log\left(\frac{1}{M}\sum_{j=1}^{M} 0.5 \cdot (y_i - f_j)^2\right)$
8: $\mathcal{D}_h \leftarrow \mathcal{D}_h \cup (\mathbf{x}_i, z_i)$
9: Adjust $\boldsymbol{\theta}_f$ to maximize Equation (6)
10: Repeat from Line 3 until $\boldsymbol{\theta}_f$ converges
11: **return** $\boldsymbol{\theta}_f, \mathcal{D}_h$

2.5 Approximate Evidence Maximization for the Heteroscedastic Gaussian Process

When the robot obtains transition data from the environment, we update the model to reflect the new information and promote monotonic policy improvement [4, 24]. This amounts to adjusting the covariance function parameters, $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_h$, to maximize the evidence of observations $p(\mathbf{y}|\mathbf{X})$. In our experiments, we consider the exponentiated quadratic covariance function

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left[-0.5(\mathbf{x} - \mathbf{x}')^\top \mathbf{L}(\mathbf{x} - \mathbf{x}')\right].$$

For the f-process, $\mathbf{L}$ is a diagonal matrix with elements $\ell_{f,1}^{-2}, \cdots, \ell_{f,n}^{-2}$. Each $\ell_{f,i}$ represents a characteristic length scale of the i-th input dimension, and σ_f^2 modulates the variance of the output. The complete parameter set is $\boldsymbol{\theta}_f = \{\ell_{f,1}, \cdots, \ell_{f,n}, \sigma_f\}$. We also selected the exponentiated quadratic for the h-process; though, $\boldsymbol{\theta}_h \neq \boldsymbol{\theta}_f$. Generally speaking, the choice of covariance function is problem specific and should reflect any prior knowledge concerning the correlation of outputs; see [7] for a broader treatment of this subject.

Algorithm 2 Heteroscedastic Policy Gradient

HPG$(\boldsymbol{\beta}, U, N, M, \boldsymbol{\theta}_f, \boldsymbol{\theta}_h)$
Given a learning rate schedule $\boldsymbol{\beta}$, the number of policy updates U, trajectory sample size N, variance sample size M, and initial parameter sets $\boldsymbol{\theta}_f$, $\boldsymbol{\theta}_h$, compute
for $i = 0, \cdots, U-1$ **do**
  $\mathcal{D}_f \leftarrow \emptyset$
  for $j = 1, \cdots, N$ **do**
    Sample trajectory τ_j using π_θ
    $\mathcal{D}_f \leftarrow \mathcal{D}_f \cup \{\tau_j\}$
  $\boldsymbol{\theta}_f, \mathcal{D}_h \leftarrow$**Het_Update**$(\boldsymbol{\theta}_f, \boldsymbol{\theta}_h, \mathcal{D}_f, M)$
  $\nabla\eta_i \leftarrow$ **PGEv**$(\boldsymbol{\theta}_i, \boldsymbol{\theta}_f, \boldsymbol{\theta}_h, \mathcal{D}_f, \mathcal{D}_h, T, \gamma)$
  $\boldsymbol{\theta}_{i+1} \leftarrow \boldsymbol{\theta}_i + \beta_i \nabla\eta_i$
return $\boldsymbol{\theta}, \boldsymbol{\theta}_f, \boldsymbol{\theta}_h$

Algorithm 3 Policy Gradient Evaluation

PGEv$(\boldsymbol{\theta}_0, \boldsymbol{\theta}_f, \boldsymbol{\theta}_h, \mathcal{D}_f, \mathcal{D}_h, T, \gamma)$
Given the parameters $\boldsymbol{\theta}_0$, $\boldsymbol{\theta}_f$, $\boldsymbol{\theta}_h$, the current model data $\mathcal{D}_f$, $\mathcal{D}_h$, the time horizon T, and the discount factor γ, compute
for $i = 1, \cdots, d$ **do**
  $\eta(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}_i) = \sum_{t=0}^{T} \gamma^t r$ from rollout
  $\eta(\boldsymbol{\theta} - \Delta\boldsymbol{\theta}_i) = \sum_{t=0}^{T} \gamma^t r$ from rollout
  $\Delta\eta_i \leftarrow \eta(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}_i) - \eta(\boldsymbol{\theta} - \Delta\boldsymbol{\theta}_i)$
  $\nabla\eta_i \leftarrow \frac{\Delta\eta_i}{2\Delta\theta_i}$
return $\nabla\eta$

We use an approximate form of evidence maximization starting from the marginal likelihood

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{X}, \mathbf{g}) p(\mathbf{g}|\mathbf{X}) d\mathbf{g}.$$

Since we are unable to marginalize $\mathbf{g}$, we invoke the maximum likelihood approximation for the conditional $p(\mathbf{y}|\mathbf{X}) \approx p(\mathbf{y}|\mathbf{X}, \mathbf{g}^*)$, where $\mathbf{g}^* = \arg\max_{\mathbf{g}} p(\mathbf{g}|\mathbf{X})$. The new likelihood, $p(\mathbf{y}|\mathbf{X}, \mathbf{g}^*)$, provides a lower bound on the exact marginal since our optimization procedure (Alg. 1) cannot guarantee that $\mathbf{g}^*$ is a global maximizer. We obtain our optimization objective by replacing $\mathbf{K}_g$ with the maximizing variant $\mathbf{K}_g^*$ and taking the logarithm of $p(\mathbf{y}|\mathbf{X}, \mathbf{g}^*)$:

$$\log p(\mathbf{y}|\mathbf{X}, \mathbf{g}^*) = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{K}_f + \mathbf{K}_g^*| - \frac{1}{2}\mathbf{y}^\top(\mathbf{K}_f + \mathbf{K}_g^*)^{-1}\mathbf{y}. \tag{6}$$

Our complete optimization procedure is outlined in Algorithm 1. It follows the Expectation-Maximization algorithm described by Kersting *et. al* [13]. The E step estimates latent variances $\mathbf{z}$, and the M step maximizes the hyperparameters of the f-process. We used an inner conjugate gradient optimization to fit the data. As with the standard EM algorithm, this procedure is not guaranteed to converge. However, we found that it performs reliably in practice.

Evaluating the likelihood costs $\mathcal{O}(N^3)$ operations, the same as a standard GP. Although, the precise runtime of Algorithm 1 depends on the convergence tolerance and inner optimization of Equation 6.

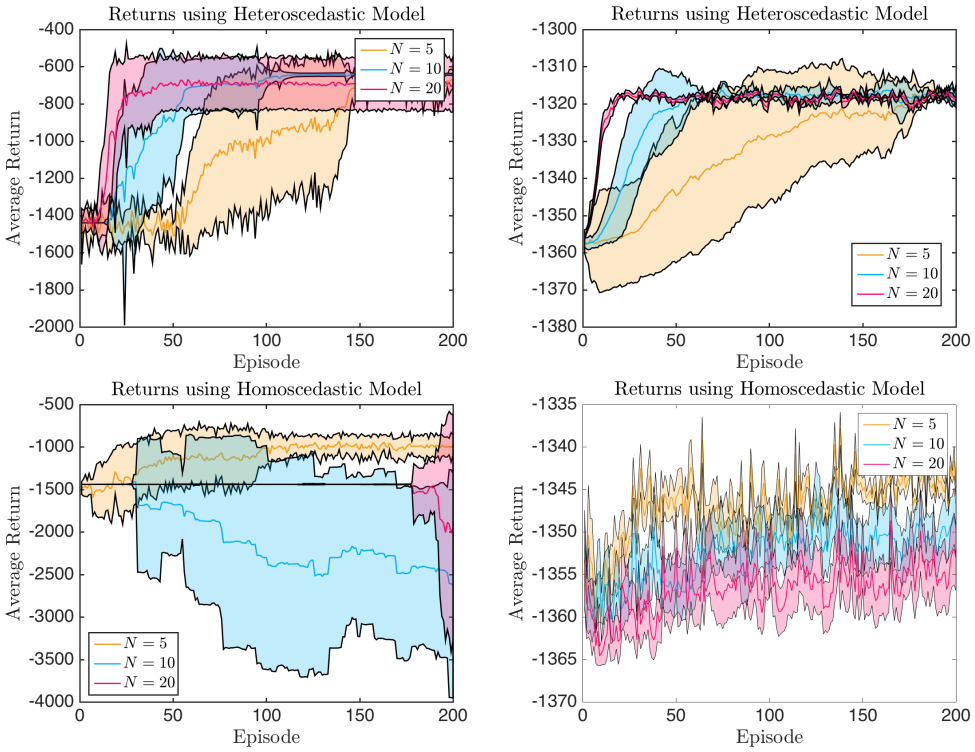
3 Simulation Results

In this section, we analyze the result of using the maximum-likelihood heteroscedastic GP for learning robot control policies. We perform simulated learning trials with three marine-inspired systems whose observations are corrupted with spatially-varying noise. We consider a two degree-of-freedom point robot, a three degree-of-freedom underactuated USV, and a six degree-of-freedom underwater vehicle. Comparisons are made to a similar model-based algorithm that uses a standard GP to predict transitions, and a performance comparison is intended to expose any aggregate effects of model bias.

3.1 General Configuration

The simulated environment implements each robot's true dynamics according to the generative model $y = f(\mathbf{x}) + \varepsilon(\mathbf{x})$. Here, f is unique to each system, but ε is common; we use $\varepsilon(\mathbf{x}) = \alpha_a \exp(\alpha_x \mathbf{x}^\top \mathbf{x}) \cdot \delta$, where $\delta \sim \mathcal{N}(0, 1)$. This is intended to model the planar positioning variance of a fixed USBL acoustic beacon (Fig. 1a) by adjusting the noise of planar position variables in proportion to their distance from the origin. We chose $\alpha_a = 1$, $\alpha_x = 10^{-4}$.

Rewards are assigned with respect to a linear quadratic function $R(\mathbf{x}) = \bar{\mathbf{x}}^\top \mathbf{Q} \bar{\mathbf{x}}$, where we define $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}_{\text{goal}}$, and $\mathbf{Q} = -0.1\mathbf{I}$. Learning trials occur in episodes with $\gamma = 1$ and T specified uniquely for each system. Sample trajectories were generated uniformly from the state space using the current policy.



(a) Point Robot

(b) Unmanned Surface Vessel

Figure 2: **Naive models fail in heteroscedastic environments:** We plot the evolution of cumulative reward for a point robot (Fig. 2a) and a USV (Fig. 2b). The experiments were repeated with $N = 5, 10, 20$ sample trajectories and averaged over 10 independent trials. The standard Gaussian process model (bottom row) suffers greatly from bias and produces suboptimal behavior.

The top-level **Het_Update** loop is terminated when subsequent parameter updates differ less than 10^{-4}. For m output variables, there are m different parameter vectors $\boldsymbol{\theta}_{f,1}, \cdots, \boldsymbol{\theta}_{f,m}$. The termination criterion is satisfied when $\max_{\Delta\theta}\{\Delta\theta_1, \cdots, \Delta\theta_m\}$ falls below our threshold, or 1000 iterations are executed. Here, $\Delta\theta$ is the magnitude of the difference vectors, $\Delta\theta(t) = ||\boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1)||_2$.

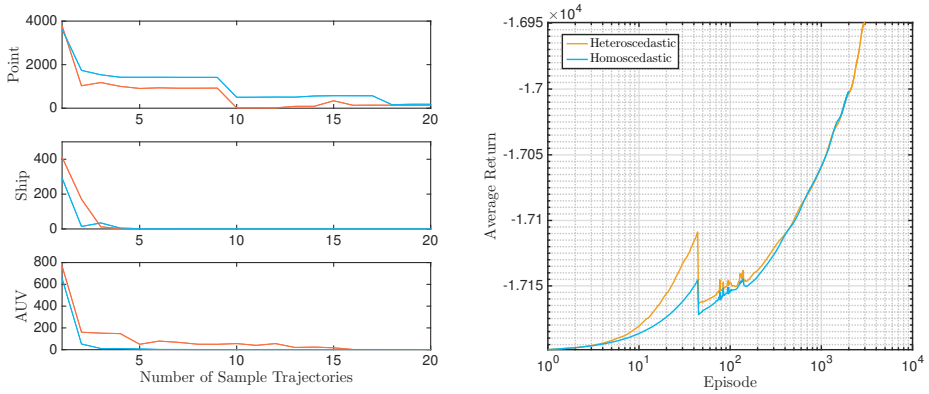
3.2 Heteroscedastic GP Training Test

In addition to policy learning, we also perform isolated model tests to characterize the convergence of Algorithm 1. We hold the time horizon fixed at $T = 20$ steps and vary the number of sample trajectories $N = 1, \cdots, 20$ used to determine the observation batch size. We compared predictions used by the model to those generated by the true environment on a test trajectory produced from a random policy initialization.

The results are shown in Figure 3a. Most of the time, the target tolerance, 10^{-4}, was reached in less than 10 iterations. Intuitively, more training data results in more accurate predictions. Training data comes from the true environment, so it always informs a more accurate model. In testing, we observed predictions converge at two different rates; those corrupted with heteroscedastic noise took longest to converge (Fig. 3a). The noiseless coordinates converged rapidly, using only one or two trajectories. The corrupted variables managed to achieve reasonable accuracy with less than 20 sample trajectories. Interestingly, the point robot took longest to converge, even though it has the simplest dynamics. This results from using an unconstrained action space; the robot is able to explore deeper into state space where prior observations do not correlate strongly.

3.3 Point Robot Control

Here the latent transition dynamics are linear. The robot starts from the position $(40\text{m}, 40\text{m})$, and the goal is to maximize the expected reward over $T = 20$ time steps of $\Delta = 0.5\text{s}$ by navigating



(a) Model Training (b) Underwater Robot

Figure 3: **Model training and Underwater Robot results:** In Figure 3a, we plot the total mean-squared error between a random test trajectory and one generated with the same random actions using the heteroscedastic GP model. Horizontal position error is plotted in red and vertical position error in blue. The training set size was varied with $N = 1, \cdots, 20$ trajectories of length $T = 20$. Results are averaged over 10 independent trials. Figure 3b plots the cumulative reward as the underwater robot learns to navigate with a horizon of $T = 100$ steps. Propagating the model over a long horizon allows bias to enter the learning process. Initially, our method achieves good performance, but it eventually behaves the same as the standard model when more error is introduced from bias.

toward the position $(100\text{m}, 100\text{m})$. States are four-dimensional vectors $\mathbf{s} = (x, \dot{x}, y, \dot{y})^\top$ containing positions x, y and velocities $\dot{x}, \dot{y}$. Actuation is applied with two actions $\mathbf{a} = (u_x, u_y)^\top$, which are selected according to a linear policy $\mathbf{a} = \mathbf{K}\mathbf{s}$. Here, $\mathbf{K}$ is a weight matrix. Results are shown in Figure 2 and discussed in Section 3.6. Fixed learning rates were used, and the true dynamics are

$$x_{t+1} = x_t + \Delta\dot{x}_t, \qquad y_{t+1} = y_t + \Delta\dot{y}_t,$$
$$\dot{x}_{t+1} = \dot{x}_t + u_x \qquad \dot{y}_{t+1} = \dot{y}_t + u_y.$$

3.4 Underactuated USV Steering

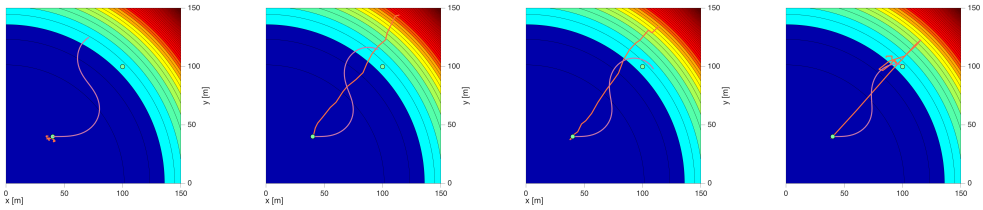
The maneuvering of a planar USV represents a more challenging control task, with two additional degrees of freedom. We assume the task is required in a GPS-denied environment, subject to the same noise properties as above. This problem was inspired by an experiment in [25], highlighting a case in which a robot's actuation does not provide full controllability. Similar to the point robot task, the USV must maximize the expected reward over $T = 40$ time steps while trying to navigate toward the position $(100\text{m}, 100\text{m})$. The USV moves at a constant speed of $V = 5$ m/s and asserts control by setting its angular rate $\omega \in [\pm 15°/\text{s}]$. States $\mathbf{s} = (x, y, \theta, \dot{\theta})^\top$ contain positions x, y, heading θ, and yaw rate $\dot{\theta}$. Time steps occur in multiples of $\Delta = 0.5$s. The dynamics are

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

The model includes a $\tau = 3$-step time delay for the command ω to be realized. This is intended to model delays from surface currents and actuator limitations. Learning rates remained fixed, and actions were chosen with a linear policy $\omega = \sum_{i=1}^{4} \mathbf{w}^\top \phi_i(\mathbf{s})$ using four basis tiles ϕ_i split around the goal position. Convergence results are shown in Figure 2 and discussed in Section 3.6.

3.5 Autonomous Underwater Vehicle Control

Here we evaluate the robustness of our method in the presence of considerable model propagation error. Propagation error occurs when making long-range predictions with test points having little to no correlation with the training data.



(a) 50 Episodes (b) 100 Episodes (c) 150 Episodes (d) 200 Episodes

Figure 4: **The evolution of policy improvement:** We show sample trajectories from the point robot (orange) and USV (purple) as their policy improves. Initially, the path is suboptimal. After convergence, each robot is able to navigate to the goal position, $(100\text{m}, 100\text{m})$. Learning occurs in the presence of exponentially worsening noise whose severity is indicated by the color gradient.

The system we consider is a twelve-dimensional, six-degree-of-freedom autonomous underwater vehicle (AUV). We fix the number of model updates to $N = 5$ and extend the time horizon to $T = 100$ steps of $\Delta = 0.2$s. The AUV state contains position, x, y, z, orientation $\phi, \theta\psi$, and body rates u, v, w, p, q, r. The control system allows for forward acceleration $u_{\text{f}} \in [\pm 0.5\ \text{m/s}^2]$, vertical acceleration $u_{\text{v}} \in [\pm 1\ \text{m/s}^2]$, and heading acceleration $u_{\text{d}} \in [\pm 0.5\ \text{rad/s}^2]$. The system is underactuated since it cannot control sway, bank or attitude instantaneously. The position rates are

$$\begin{aligned}
\dot{u}_{t+1} &= a_1 v_t r_t + a_2 u_t + a_3 u_t |u_t| + u_{\text{f}}, \\
\dot{v}_{t+1} &= b_1 u_t r_t + b_2 v_t + b_3 v_t |v_t|, \\
\dot{w}_{t+1} &= c_1 w_t + c_2 w_t |w_t| + u_{\text{v}}.
\end{aligned}$$

Yaw motion is described by $\dot{r}_{t+1} = d_1 r_t + d_2 r_t |r_t| + u_{\text{d}}$. Values for all the numerical constants can be found in the original paper [26]. Similar to before, the system starts from $(40\text{m}, 40\text{m}, 0\text{m})$. Now we task it to regulate toward $(100\text{m}, 100\text{m}, 100\text{m})$. Forward and vertical actions are computed with a linear policy $\mathbf{a}_{\text{lin}} = \mathbf{K}\mathbf{s}$, and heading actions are computed with $u_{\text{d}} = \sum_{i=1}^{4} \mathbf{w}_i^\top \phi_i(\mathbf{s})$, using the same features ϕ as the USV problem. Results are shown in Figure 3b and discussed in Section 3.6.

3.6 Results

Figure 2 shows the average return of our model-based RL framework (Alg. 2) and another that ignores heteroscedastic noise with a standard GP model. The evolution of policy improvement can be visualized in Figure 4. In each simulation, our method achieves convergence within 200 episodes. Although, convergence is not always possible using the naive method. For each case, we observe an association between the rate of convergence and the observation batch size, N. As N increases, so does the convergence rate. This result is expected, because convergence is determined by the algorithm's ability to evaluate its policy accurately with model predictions. Training with more sample trajectories reduces model bias and produces more accurate gradient estimates. Conversely, ignoring heteroscedastic effects compounds model bias, which can lead to divergence from poor policy evaluations.

Figure 3b shows the average return taken over 10 trials on a log-scale plot. Both methods perform equally poorly when making long-range predictions. These results make intuitive sense; in the extrapolation regime, correlations are low. Thus, model bias has a more significant effect.

4 Conclusion

We have presented a new method for model-based RL that uses a chained Gaussian process to model the transition dynamics of robots in the presence of input-dependent noise. Our approach leverages maximum likelihood assumptions in order to make inference tractable for predictions and training. Our tests show that model learning can be done with reasonable amounts of data for common marine robotic systems. Our comparative studies indicate the extent to which heteroscedastic effects contribute to model bias and impact the optimality of learned policies. We believe that GP-based methods for RL are poised for advancing robot autonomy. Subsequent efforts on this topic will be to evaluate the benefit of richer models and to perform physical learning trials on a mobile robot.

Acknowledgments

This research has been supported in part by the National Science Foundation, grant number IIS-1652064, and the Office of Naval Research, grant number N00014-10-1-0652. 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] M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[2] T. Hester, M. Quinlan, and P. Stone. Generalized model learning for reinforcement learning on a humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2369–2374, 2010.

[3] J. Ko, D. J. Klein, D. Fox, and D. Haehnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 742–747, 2007.

[4] T. Hester and P. Stone. *Learning and Using Models*. Springer, 2011.

[5] C. G. Atkeson and J. C. Santamaria. A comparison of direct and model-based reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 3557–3564, 1997.

[6] D. Nguyen-tuong, J. R. Peters, and M. Seeger. Local Gaussian process regression for real time online model learning. In *Advances in Neural Information Processing Systems 21*, pages 1193–1200, 2009.

[7] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[8] P. W. Goldberg, C. K. I. Williams, and C. M. Bishop. Regression with input-dependent noise: A Gaussian process treatment. In *Advances in Neural Information Processing Systems 10*, pages 493–499, 1998.

[9] A. D. Saul, J. Hensman, A. Vehtari, and N. D. Lawrence. Chained Gaussian processes. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 1431–1440, 2016.

[10] M. Titsias and M. Lázaro-Gredilla. Variational heteroscedastic Gaussian process regression. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[11] J. Vanhatalo, J. Riihimäki, J. Hartikainen, P. Jylänki, V. Tolvanen, and A. Vehtari. GPstuff: Bayesian modeling with Gaussian processes. *Journal of Machine Learning Research*, 14(Apr): 1175–1179, 2013.

[12] D. Hernández-lobato, V. Sharmanska, K. Kersting, C. H. Lampert, and N. Quadrianto. Mind the nuisance: Gaussian process classification using privileged noise. In *Advances in Neural Information Processing Systems 27*, pages 837–845, 2014.

[13] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic Gaussian process regression. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[14] J. M. Assael, Z. Wang, and N. de Freitas. Heteroscedastic treed Bayesian optimisation. *CoRR*, abs/1410.7172, 2014.

[15] E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010.

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

[17] C. E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *Advances in Neural Information Processing Systems 16*, pages 751–759, 2004.

[18] A. Wilson, A. Fern, and P. Tadepalli. Incorporating domain models into Bayesian optimization for RL. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 467–482, 2010.

[19] J. Peters and S. Schaal. Policy gradient methods for robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2219–2225, 2006.

[20] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21:682–697, 2008.

[21] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[22] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[23] R. Sutton, D. Mcallester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063, 2000.

[24] C. Szepesvári. *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2010.

[25] M. Ghavamzadeh, Y. Engel, and M. Valko. Bayesian policy gradient and actor-critic algorithms. *Journal of Machine Learning Research*, 17(66):1–53, 2016.

[26] O. Eidsvik. Identification of hydrodynamic parameters for remotely operated vehicles. Master's thesis, Norweigian University of Science and Technology, 2015.