ICML 2005

Reinforcement learning with Gaussian processes

Yaakov Engel YAKI@CS.UALBERTA.CA
Dept. of Computing Science, University of Alberta, Edmonton, Canada

Shie Mannor SHIE@ECE.MCGILL.CA
Dept. of Electrical and Computer Engineering, McGill University, Montreal, Canada

Ron Meir RMEIR@EE.TECHNION.AC.IL
Dept. of Electrical Engineering, Technion Institute of Technology, Haifa 32000, Israel

Abstract

Gaussian Process Temporal Difference (GPTD) learning offers a Bayesian solution to the policy evaluation problem of reinforcement learning. In this paper we extend the GPTD framework by addressing two pressing issues, which were not adequately treated in the original GPTD paper (Engel et al., 2003). The first is the issue of stochasticity in the state transitions, and the second is concerned with action selection and policy improvement. We present a new generative model for the value function, deduced from its relation with the discounted return. We derive a corresponding on-line algorithm for learning the posterior moments of the value Gaussian process. We also present a SARSA based extension of GPTD, termed GPSARSA, that allows the selection of actions and the gradual improvement of policies without requiring a world-model.

1. Introduction

In Engel et al. (2003) the use of Gaussian Processes (GPs) for solving the Reinforcement Learning (RL) problem of value estimation was introduced. Since GPs belong to the family of kernel machines, they bring into RL the high, and quickly growing representational flexibility of kernel based representations, allowing them to deal with almost any conceivable object of interest, from text documents and DNA sequence data to probability distributions, trees and graphs, to mention just a few (see Schölkopf & Smola, 2002, and references therein). Moreover, the use of Bayesian reasoning with GPs allows one to obtain not only value estimates, but also estimates of the *uncertainty* in the value, and this in large and even infinite MDPs.

However, both the probabilistic generative model and the corresponding Gaussian Process Temporal Differences (GPTD) algorithm proposed in Engel et al. (2003) had two major shortcomings. First, the original model is strictly correct only if the state transitions of the underlying Markov Decision Process (MDP) are deterministic[1], and if the rewards are corrupted by white Gaussian noise. While the second assumption is relatively innocuous, the first is a serious handicap to the applicability of the GPTD model to general MDPs. Secondly, in RL what we are really after is an optimal, or at least a good suboptimal action selection *policy*. Many algorithms for solving this problem are based on the Policy Iteration method, in which the value function must be estimated for a sequence of fixed policies, making value estimation, or policy evaluation, a crucial algorithmic component. Since the GPTD algorithm only addresses the value estimation problem, we need to modify it somehow, if we wish to solve the complete RL problem.

One possible heuristic modification, demonstrated in Engel et al. (2003), is the use of Optimistic Policy Iteration (OPI) (Bertsekas & Tsitsiklis, 1996). In OPI the learning agent utilizes a model of its environment and its current value estimate to guess the expected payoff for each of the actions available to it at each time step. It then greedily (or ε-greedily) chooses the highest ranking action. Clearly, OPI may be used only when a good model of the MDP is available to the agent. However, assuming that such a model is available as prior knowledge is a rather strong assumption inapplicable in many domains, while estimating such a model on-the-fly, especially when the state-transitions

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

[1] Or if the discount factor is zero, in which case the model degenerates to simple GP regression.

are stochastic, may be prohibitively expensive. In either case, computing the expectations involved in ranking the actions may itself be prohibitively costly. Another possible modification, one that does not require a model, is to estimate *state-action values*, or Q-values, using an algorithm such as Sutton's SARSA (Sutton & Barto, 1998).

The first contribution of this paper is a modification of the original GPTD model that allows it to learn value and value-uncertainty estimates in general MDPs, allowing for stochasticity in both transitions and rewards. Drawing inspiration from Sutton's SARSA algorithm, our second contribution is GPSARSA, an extension of the GPTD algorithm for learning a Gaussian distribution over state-action values, thus allowing us to perform model-free policy improvement.

2. Modeling the Value Via the Discounted Return

Let us introduce some definitions to be used in the sequel. A Markov Decision Process (MDP) is a tuple $(\mathcal{X},\mathcal{U},R,p)$ where $\mathcal{X}$ and $\mathcal{U}$ are the state and action spaces, respectively; $R:\mathcal{X}\to\mathbb{R}$ is the immediate reward, which may be random, in which case $q(\cdot|\mathbf{x})$ denotes the distribution of rewards at the state $\mathbf{x}$; and $p:\mathcal{X}\times\mathcal{U}\times\mathcal{X}\to[0,1]$ is the transition distribution, which we assume is stationary. A *stationary policy* $\mu:\mathcal{X}\times\mathcal{U}\to[0,1]$ is a mapping from states to action selection probabilities. Given a fixed policy μ, the transition probabilities of the MDP are given by the *policy-dependent state transition probability distribution* $p^\mu(\mathbf{x}'|\mathbf{x})=\int_{\mathcal{U}}d\mathbf{u}p(\mathbf{x}'|\mathbf{u},\mathbf{x})\mu(\mathbf{u}|\mathbf{x})$. The *discounted return* $D(\mathbf{x})$ for a state $\mathbf{x}$ is a random process defined by

$$D(\mathbf{x})=\sum_{i=0}^{\infty}\gamma^i R(\mathbf{x}_i)|\mathbf{x}_0=\mathbf{x},\ \text{with}\ \mathbf{x}_{i+1}\sim p^\mu(\cdot|\mathbf{x}_i). \tag{2.1}$$

Here, $\gamma\in[0,1]$ is a discount factor that determines the exponential devaluation rate of delayed rewards[2]. Note that the randomness in $D(\mathbf{x}_0)$ for any given state $\mathbf{x}_0$ is due both to the stochasticity of the sequence of states that follow $\mathbf{x}_0$, and to the randomness in the rewards $R(\mathbf{x}_0),R(\mathbf{x}_1),R(\mathbf{x}_2)\ldots$. We refer to this as the *intrinsic* randomness of the MDP. Using the stationarity of the MDP we may write

$$D(\mathbf{x})=R(\mathbf{x})+\gamma D(\mathbf{x}'),\ \text{with}\ \mathbf{x}'\sim p^\mu(\cdot|\mathbf{x}). \tag{2.2}$$

The equality here marks an equality in the distributions of the two sides of the equation. Let us define the expectation operator $\mathbf{E}_\mu$ as the expectation over all possible trajectories and all possible rewards collected in them. This allows us to define the *value function* $V(\mathbf{x})$ as the result of applying this expectation operator to the discounted return $D(\mathbf{x})$. Thus, applying $\mathbf{E}_\mu$ to both sides of Eq. (2.2), and using the conditional expectation formula (Scharf, 1991), we get

$$V(\mathbf{x})=\bar{r}(\mathbf{x})+\gamma\mathbf{E}_{\mathbf{x}'|\mathbf{x}}V(\mathbf{x}')\quad\forall\mathbf{x}\in\mathcal{X}, \tag{2.3}$$

which is recognizable as the fixed-policy version of the Bellman equation (Bertsekas & Tsitsiklis, 1996).

2.1. The Value Model

The recursive definition of the discounted return (2.2) is the basis for our statistical generative model connecting values and rewards. Let us decompose the discounted return D into its mean V and a random, zero-mean residual ΔV,

$$D(\mathbf{x})=V(\mathbf{x})+\Delta V(\mathbf{x}), \tag{2.4}$$

where $V(\mathbf{x})=\mathbf{E}_\mu D(\mathbf{x})$. In the classic frequentist approach $V(\cdot)$ is no longer random, since it is the true value function induced by the policy μ. Adopting the Bayesian methodology, we may still view the value $V(\cdot)$ as a random entity by assigning it additional randomness that is due to our subjective uncertainty regarding the MDP's model (p,q). We do not know what the true functions p and q are, which means that we are also uncertain about the true value function. We choose to model this additional *extrinsic* uncertainty by defining $V(\mathbf{x})$ as a random process indexed by the state variable $\mathbf{x}$. This decomposition is useful, since it separates the two sources of uncertainty inherent in the discounted return process D: For a known MDP model, V becomes deterministic and the randomness in D is fully attributed to the intrinsic randomness in the state-reward trajectory, modeled by ΔV. On the other hand, in a MDP in which both transitions and rewards are deterministic but otherwise unknown, ΔV becomes deterministic (i.e., identically zero), and the randomness in D is due solely to the extrinsic uncertainty, modeled by V. For a more thorough discussion of intrinsic and extrinsic uncertainties see Mannor et al. (2004).

Substituting Eq. (2.4) into Eq. (2.2) and rearranging we get

$$R(\mathbf{x})=V(\mathbf{x})-\gamma V(\mathbf{x}')+N(\mathbf{x},\mathbf{x}'),\ \mathbf{x}'\sim p^\mu(\cdot|\mathbf{x}), \tag{2.5}$$

where $N(\mathbf{x},\mathbf{x}')\stackrel{\text{def}}{=}\Delta V(\mathbf{x})-\gamma\Delta V(\mathbf{x}')$. Suppose we are provided with a trajectory $\mathbf{x}_0,\mathbf{x}_1,\ldots,\mathbf{x}_t$, sampled from the MDP under a policy μ, i.e., from

[2] When $\gamma=1$ the policy must be proper (i.e., guaranteed to terminate), see Bertsekas and Tsitsiklis (1996).

$p_0(\mathbf{x}_0)\Pi_{i=1}^{t} p^{\mu}(\mathbf{x}_i|\mathbf{x}_{i-1})$, where p_0 is an arbitrary probability distribution for the first state. Let us write our model (2.5) with respect to these samples:

$$R(\mathbf{x}_i) = V(\mathbf{x}_i) - \gamma V(\mathbf{x}_{i+1}) + N(\mathbf{x}_i, \mathbf{x}_{i+1}),\ i = 0, \ldots, t-1. \tag{2.6}$$

Defining the finite-dimensional processes R_t, V_t, N_t and the $t \times (t+1)$ matrix $\mathbf{H}_t$

$$\begin{aligned} R_t &= (R(\mathbf{x}_0), \ldots, R(\mathbf{x}_t))^{\top}, \\ V_t &= (V(\mathbf{x}_0), \ldots, V(\mathbf{x}_t))^{\top}, \\ N_t &= (N(\mathbf{x}_0, \mathbf{x}_1), \ldots, N(\mathbf{x}_{t-1}, \mathbf{x}_t))^{\top}, \\ \mathbf{H}_t &= \begin{bmatrix} 1 & -\gamma & 0 & \ldots & 0 \\ 0 & 1 & -\gamma & \ldots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \ldots & 1 & -\gamma \end{bmatrix}, \end{aligned} \tag{2.7}$$

we may write the equation set (2.6) more concisely as

$$R_{t-1} = \mathbf{H}_t V_t + N_t. \tag{2.8}$$

In order to specify a complete probabilistic generative model connecting values and rewards, we need to define a prior distribution for the value process V and the distribution of the "noise" process N. As in the original GPTD model, we impose a Gaussian prior over value functions, i.e., $V \sim \mathcal{N}(0, k(\cdot,\cdot))$, meaning that V is a Gaussian Process (GP) for which, a priori, $\mathbf{E}(V(\mathbf{x})) = 0$ and $\mathbf{E}(V(\mathbf{x})V(\mathbf{x}')) = k(\mathbf{x}, \mathbf{x}')$ for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, where k is a positive-definite kernel function. Therefore, $V_t \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_t)$, where $\mathbf{0}$ is a vector of zeros and $[\mathbf{K}_t]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. Our choice of kernel function k should reflect our prior beliefs concerning the correlations between the values of states in the domain at hand.

To maintain the analytical tractability of the posterior value distribution, we model the residuals $\Delta V_t = (\Delta V(\mathbf{x}_0), \ldots, \Delta V(\mathbf{x}_t))^{\top}$ as a Gaussian process[3]. This means that the distribution of the vector ΔV_t is completely specified by its mean and covariance. Another assumption we make is that each of the residuals $\Delta V(\mathbf{x}_i)$ is generated independently of all the others. This means that, for any $i \neq j$, the random variables $\Delta V(\mathbf{x}_i)$ and $\Delta V(\mathbf{x}_j)$ correspond to two distinct experiments, in which two random trajectories starting from the states $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively, are generated independently of each other. We will discuss the implications of this assumption in Section 2.2. We are now ready to proceed with the derivation of the distribution of the noise process N_t.

By definition (Eq. 2.4), $\mathbf{E}_{\mu}[\Delta V(\mathbf{x})] = 0$ for all $\mathbf{x}$, so we have $\mathbf{E}_{\mu}[N(\mathbf{x}_i, \mathbf{x}_{i+1})] = 0$. Turning to the covariance, we have $\mathbf{E}_{\mu}[N(\mathbf{x}_i, \mathbf{x}_{i+1})N(\mathbf{x}_j, \mathbf{x}_{j+1})] = \mathbf{E}_{\mu}[(\Delta V(\mathbf{x}_i) - \gamma \Delta V(\mathbf{x}_{i+1}))(\Delta V(\mathbf{x}_j) - \gamma \Delta V(\mathbf{x}_{j+1}))]$. According to our assumption regarding the independence of the residuals, for $i \neq j$, $\mathbf{E}_{\mu}\big[\Delta V(\mathbf{x}_i)\Delta V(\mathbf{x}_j)\big] = 0$. In contrast, $\mathbf{E}_{\mu}\big[\Delta V(\mathbf{x}_i)^2\big] = \mathbf{Var}_{\mu}\big[D(\mathbf{x}_i)\big]$ is generally larger than zero, unless both transitions and rewards are deterministic. Denoting $\sigma_i^2 = \mathbf{Var}_{\mu}\big[D(\mathbf{x}_i)\big]$, these observations may be summarized into the distribution of ΔV_t: $\Delta V_t \sim \mathcal{N}\big(\mathbf{0}, \mathrm{diag}(\boldsymbol{\sigma}_t)\big)$ where $\boldsymbol{\sigma}_t = [\sigma_0^2, \sigma_1^2, \ldots, \sigma_t^2]^{\top}$, and $\mathrm{diag}(\cdot)$ denotes a diagonal matrix whose diagonal entries are the components of the argument vector. To simplify the subsequent discussion let us assume[4] that, for all $i \in \{1, \ldots, t\}$, $\sigma_i = \sigma$, and therefore $\mathrm{diag}(\boldsymbol{\sigma}_t) = \sigma^2 \mathbf{I}$. Since $N_t = \mathbf{H}_t \Delta V_t$, we have $N_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_t)$ with,

$$\begin{aligned} \boldsymbol{\Sigma}_t &= \sigma^2 \mathbf{H}_t \mathbf{H}_t^{\top} \\ &= \sigma^2 \begin{bmatrix} 1+\gamma^2 & -\gamma & 0 & \ldots & 0 \\ -\gamma & 1+\gamma^2 & -\gamma & \ldots & 0 \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & \ldots & -\gamma & 1+\gamma^2 \end{bmatrix}. \end{aligned}$$

Let us briefly compare our new model with the original, deterministic GPTD model. Both models result in the same general form of Eq. (2.8). However, in the original GPTD model the Gaussian noise term N_t had a diagonal covariance matrix (white noise), while in the new model N_t is colored with a tridiagonal covariance matrix. Note also, that as the discount factor γ is reduced to zero, the two models tend to coincide. This is reasonable, since, the more strongly the future is discounted, the less it should matter whether the transitions are deterministic or stochastic. In Section 5 we perform an empirical comparison between the two corresponding algorithms.

Since both the value prior and the noise are Gaussian, by the Gauss-Markov theorem (Scharf, 1991), so is the posterior distribution of the value conditioned on an observed sequence of rewards $\mathbf{r}_{t-1} = (r_0, \ldots, r_{t-1})^{\top}$.

[3] This may not be a correct assumption in general; however, in the absence of any prior information concerning the distribution of the residuals, it is the *simplest* assumption we can make, since the Gaussian distribution possesses the highest entropy among all distributions with the same covariance. It is also possible to relax the Gaussianity requirement on both the prior and the noise. The resulting estimator may then be shown to provide the *linear minimum mean-squared error* estimate for the value.

[4] In practice it is also more likely that we will have some idea concerning the average variance of the discounted return, rather than a detailed knowledge of the variance at each individual state. Notwithstanding, the subsequent analysis is easily extended to include state dependent noise, see Engel (2005).

The posterior mean and variance of the value at some point $\mathbf{x}$ are given, respectively, by

$$\begin{aligned}\hat{v}_t(\mathbf{x}) &= \mathbf{k}_t(\mathbf{x})^\top \boldsymbol{\alpha}_t, \\ p_t(\mathbf{x}) &= k(\mathbf{x},\mathbf{x}) - \mathbf{k}_t(\mathbf{x})^\top \mathbf{C}_t \mathbf{k}_t(\mathbf{x}), \qquad (2.9)\\ \text{where } \mathbf{k}_t(\mathbf{x}) &= \left(k(\mathbf{x}_0,\mathbf{x}),\ldots,k(\mathbf{x}_t,\mathbf{x})\right)^\top, \\ \boldsymbol{\alpha}_t &= \mathbf{H}_t^\top \left(\mathbf{H}_t\mathbf{K}_t\mathbf{H}_t^\top + \boldsymbol{\Sigma}_t\right)^{-1} \mathbf{r}_{t-1}, \\ \mathbf{C}_t &= \mathbf{H}_t^\top \left(\mathbf{H}_t\mathbf{K}_t\mathbf{H}_t^\top + \boldsymbol{\Sigma}_t\right)^{-1} \mathbf{H}_t. \qquad (2.10)\end{aligned}$$

2.2. Relation to Monte-Carlo Simulation

Consider an episode in which a terminal state is reached at time step $t+1$. In this case, the last equation in our generative model should read $R(\mathbf{x}_t) = V(\mathbf{x}_t) + N(\mathbf{x}_t)$, since $V(\mathbf{x}_{t+1}) = 0$. Our complete set of equations is now

$$R_t = \mathbf{H}_{t+1} V_t + N_t, \qquad (2.11)$$

with $\mathbf{H}_{t+1}$ a square $(t+1)\times(t+1)$ matrix, given by $\mathbf{H}_{t+1}$ as defined in (2.7), with its last column removed. Note that $\mathbf{H}_{t+1}$ is also invertible, since its determinant equals 1.

Our model's validity may be substantiated by performing a *whitening* transformation on Eq. (2.11). Since the noise covariance matrix $\boldsymbol{\Sigma}_t$ is positive definite, there exists a square matrix $\mathbf{Z}_t$ satisfying $\mathbf{Z}_t^\top \mathbf{Z}_t = \boldsymbol{\Sigma}_t^{-1}$. Multiplying Eq. (2.11) by $\mathbf{Z}_t$ we then get $\mathbf{Z}_t R_t = \mathbf{Z}_t\mathbf{H}_{t+1}V_t + \mathbf{Z}_t N_t$. The transformed noise term $\mathbf{Z}_t N_t$ has a covariance matrix given by $\mathbf{Z}_t\boldsymbol{\Sigma}_t\mathbf{Z}_t^\top = \mathbf{Z}_t(\mathbf{Z}_t^\top\mathbf{Z}_t)^{-1}\mathbf{Z}_t^\top = \mathbf{I}$. Thus the transformation $\mathbf{Z}_t$ whitens the noise. In our case, a whitening matrix is given by

$$\mathbf{Z}_t = \mathbf{H}_{t+1}^{-1} = \begin{bmatrix} 1 & \gamma & \gamma^2 & \ldots & \gamma^t \\ 0 & 1 & \gamma & \ldots & \gamma^{t-1} \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix}.$$

The transformed model is $\mathbf{Z}_t R_t = V_t + N'_t$ with white Gaussian noise $N'_t = \mathbf{Z}_t N_t \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Let us look at the i'th equation (i.e., row) of this transformed model:

$$R(\mathbf{x}_i) + \gamma R(\mathbf{x}_{i+1}) + \ldots + \gamma^{t-i} R(\mathbf{x}_t) = V(\mathbf{x}_i) + N'_i,$$

with $N'_i \sim \mathcal{N}(0,\sigma^2)$. This is exactly the generative model we would have used had we wanted to learn the value function by performing GP regression using Monte-Carlo samples of the discounted-return as our targets. The major benefit in using the GPTD formulation is that it allows us to perform exact updates of the parameters of the posterior value mean and covariance *on-line*.

3. An On-Line Algorithm

Computing the parameters $\boldsymbol{\alpha}_t$ and $\mathbf{C}_t$ of the posterior moments (2.10) is computationally expensive for large samples, due to the need to store and invert a matrix of size $t\times t$. Even when this has been performed, computing the posterior moments for every new query point requires that we multiply two $t\times 1$ vectors for the mean, and compute a $t\times t$ quadratic form for the variance. These computational requirements are prohibitive if we are to compute value estimates on-line, as is usually required of RL algorithms. Engel et al. (2003) used an on-line kernel sparsification algorithm that is based on a view of the kernel as an inner-product in some high dimensional feature space to which raw state vectors are mapped[5]. This sparsification method incrementally constructs a dictionary $\mathcal{D} = \left\{\tilde{\mathbf{x}}_1,\ldots,\tilde{\mathbf{x}}_{|\mathcal{D}|}\right\}$ of representative states. Upon observing $\mathbf{x}_t$, the distance between the feature-space image of $\mathbf{x}_t$ and the span of the images of current dictionary members is computed. If the squared distance exceeds some positive threshold ν, $\mathbf{x}_t$ is added to the dictionary, otherwise, it is left out. Determining this squared distance, δ_t, involves solving a simple least-squares problem, whose solution is a $|\mathcal{D}|\times 1$ vector $\boldsymbol{a}_t$ of optimal approximation coefficients, satisfying

$$\boldsymbol{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1}\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t), \quad \delta_t = k(\mathbf{x}_t,\mathbf{x}_t) - \boldsymbol{a}_t^\top \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t), \qquad (3.12)$$

where $\tilde{\mathbf{k}}_t(\mathbf{x}) = \left(k(\tilde{\mathbf{x}}_1,\mathbf{x}),\ldots,k(\tilde{\mathbf{x}}_{|\mathcal{D}_t|},\mathbf{x})\right)^\top$ is a $|\mathcal{D}_t|\times 1$ vector, and $\tilde{\mathbf{K}}_t = \left[\tilde{\mathbf{k}}_t(\tilde{\mathbf{x}}_1),\ldots,\tilde{\mathbf{k}}_t(\tilde{\mathbf{x}}_{|\mathcal{D}_t|})\right]$ a square $|\mathcal{D}_t|\times|\mathcal{D}_t|$, symmetric, positive-definite matrix.

By construction, the dictionary has the property that the feature-space images of all states encountered during learning may be approximated to within a squared error ν by the images of the dictionary members. The threshold ν may be tuned to control the sparsity of the solution. Sparsification allows kernel expansions, such as those appearing in Eq. 2.10, to be approximated by kernel expansions involving only dictionary members, by using

$$\mathbf{k}_t(\mathbf{x}) \approx \mathbf{A}_t\tilde{\mathbf{k}}_t(\mathbf{x}), \quad \mathbf{K}_t \approx \mathbf{A}_t\tilde{\mathbf{K}}_t\mathbf{A}_t^\top. \qquad (3.13)$$

The $t\times|\mathcal{D}_t|$ matrix $\mathbf{A}_t$ contains in its rows the approximation coefficients computed by the sparsification algorithm, i.e., $\mathbf{A}_t = [\boldsymbol{a}_1,\ldots,\boldsymbol{a}_t]^\top$, with padding zeros placed where necessary, see Engel et al. (2003).

The end result of the sparsification procedure is that the posterior value mean $\hat{v}_t$ and variance p_t may be compactly approximated as follows (compare to

[5] For completeness, we repeat here some of the details concerning this sparsification method.

Online Sparsification Algorithm

Kernels

$k: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, $\phi: \mathbb{R}^d \to \mathcal{F}$

think SVM for 

- defines a high dim feature space, $\mathcal{F}$, implicitly
  - calculates the similarity in a transformed space
  - without tsf., would just be an inner prod

→ both have similar conditions

$k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{F}}$ ← is defined

Reproducing Kernel Hilbert Space, $\mathcal{H}$

- the space of functions, f, that can be expressed as a finite L.C. of the same $k(\cdot)$
  - different functions define different RKHSs (including params, e.g. length scale)

→ $f(x) = \sum_{i=1}^{N} a_i k(x_i, x)$

s.t. $x_i \in \mathbb{R}^d$ are datapoints,
$a_i \in \mathbb{R}$ are coeffs.

Method ← initialization

- w.r.t. posteriors $(f = \{\mu(x), \sigma(x)\}$ in an online setting
- maintain sparsity by keeping a subset, dictionary, $\mathcal{D}_t$, of representative data, $\{\tilde{x}_i\}$

at t:

$$\hat{y}_t = f_{t-1}(x_t) = \sum_{i=1}^{|\mathcal{D}_{t-1}|} a_i^{(t-1)} \cdot k(\tilde{x}_i, x_t)$$

$$e_t = y_t - \hat{y}_t$$

$a^{(t)} = \tilde{K}_{t-1}^{-1} \tilde{k}_{t-1}(x_t)$ ← look at paper for expansion

$\delta_t = k(x_t, x_t) - a^{(t-1)\top} \tilde{k}(x_t)$ ← existing contribution

Isq. distance → norm$_{\mathcal{F}}$

if $\delta_t > \nu$ ← threshold param.

include $x_t \in \mathcal{D}_t$

Eq. 2.9, 2.10)

$$\hat{v}_t(\mathbf{x}) = \tilde{\mathbf{k}}_t(\mathbf{x})^\top \tilde{\boldsymbol{\alpha}}_t,$$
$$p_t(\mathbf{x}) = k(\mathbf{x},\mathbf{x}) - \tilde{\mathbf{k}}_t(\mathbf{x})^\top \tilde{\mathbf{C}}_t \tilde{\mathbf{k}}_t(\mathbf{x}), \qquad (3.14)$$

where $\tilde{\boldsymbol{\alpha}}_t = \tilde{\mathbf{H}}_t^\top \left(\tilde{\mathbf{H}}_t \tilde{\mathbf{K}}_t \tilde{\mathbf{H}}_t^\top + \boldsymbol{\Sigma}_t\right)^{-1} \mathbf{r}_{t-1}$

$$\tilde{\mathbf{C}}_t = \tilde{\mathbf{H}}_t^\top \left(\tilde{\mathbf{H}}_t \tilde{\mathbf{K}}_t \tilde{\mathbf{H}}_t^\top + \boldsymbol{\Sigma}_t\right)^{-1} \tilde{\mathbf{H}}_t, \qquad (3.15)$$

and $\tilde{\mathbf{H}}_t = \mathbf{H}_t \mathbf{A}_t$.

The parameters that the GPTD algorithm is required to store and update in order to evaluate the posterior mean and variance are now $\tilde{\boldsymbol{\alpha}}_t$ and $\tilde{\mathbf{C}}_t$, whose dimensions are $|\mathcal{D}_t| \times 1$ and $|\mathcal{D}_t| \times |\mathcal{D}_t|$, respectively. In many cases this results in significant computational savings, both in terms of memory and time, when compared with the exact non-sparse solution.

The derivation of the recursive update formulas for the mean and covariance parameters $\tilde{\boldsymbol{\alpha}}_t$ and $\tilde{\mathbf{C}}_t$, for a new sample $\mathbf{x}_t$, is rather long and tedious due to the added complication arising from the non-diagonality of the noise covariance matrix $\boldsymbol{\Sigma}_t$. We therefore refer the interested reader to (Engel, 2005, Appendix A.2.3) for the complete derivation (with state dependent noise). In Table 1 we present the resulting algorithm in pseudocode.

Some insight may be gained by noticing that the term $r_{t-1} - \Delta\tilde{\mathbf{k}}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1}$ appearing in the update for d_t is a temporal difference term. From Eq. (3.14) and the definition of $\Delta\tilde{\mathbf{k}}_t$ (see Table 1) we have $r_{t-1} - \Delta\tilde{\mathbf{k}}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1} = r_{t-1} + \gamma \hat{v}_{t-1}(\mathbf{x}_t) - \hat{v}_{t-1}(\mathbf{x}_{t-1})$. Consequently, d_t may be viewed as a linear filter driven by the temporal differences. The update of $\tilde{\boldsymbol{\alpha}}_t$ is simply the output of this filter, multiplied by the gain vector $\tilde{\mathbf{c}}_t / s_t$. The resemblance to the Kalman Filter updates is evident. It should be noted that it is indeed fortunate that the noise covariance matrix vanishes except for its three central diagonals. This relative simplicity of the noise model is the reason we were able to derive simple and efficient recursive updates, such as the ones described above.

Table 1. The On-Line Monte-Carlo GPTD Algorithm

Parameters: ν , σ
Initialize $\mathcal{D}_0 = \{\mathbf{x}_0\}$, $\tilde{\mathbf{K}}_0^{-1} = 1/k(\mathbf{x}_0,\mathbf{x}_0)$, $\boldsymbol{a}_0 = (1)$,
$\tilde{\boldsymbol{\alpha}}_0 = 0$, $\tilde{\mathbf{C}}_0 = 0$, $\tilde{\mathbf{c}}_0 = 0$, $d_0 = 0$, $1/s_0 = 0$
for $t = 1, 2, \ldots$
  observe $\mathbf{x}_{t-1}$, r_{t-1}, $\mathbf{x}_t$
  $\boldsymbol{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$
  $\delta_t = k(\mathbf{x}_t,\mathbf{x}_t) - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \boldsymbol{a}_t$
  $\Delta\tilde{\mathbf{k}}_t = \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_{t-1}) - \gamma \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$
  if $\delta_t > \nu$
    $\tilde{\mathbf{K}}_t^{-1} = \frac{1}{\delta_t} \begin{bmatrix} \delta_t \tilde{\mathbf{K}}_{t-1}^{-1} + \boldsymbol{a}_t \boldsymbol{a}_t^\top & -\boldsymbol{a}_t \\ -\boldsymbol{a}_t^\top & 1 \end{bmatrix}$
    $\boldsymbol{a}_t = (0, \ldots, 1)^\top$
    $\tilde{\mathbf{h}}_t = (\boldsymbol{a}_{t-1}, -\gamma)^\top$
    $\Delta k_{tt} = \boldsymbol{a}_{t-1}^\top \left(\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_{t-1}) - 2\gamma \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)\right) + \gamma^2 k(\mathbf{x}_t,\mathbf{x}_t)$
    $\tilde{\mathbf{c}}_t = \frac{\gamma\sigma^2}{s_{t-1}} \begin{pmatrix} \tilde{\mathbf{c}}_{t-1} \\ 0 \end{pmatrix} + \tilde{\mathbf{h}}_t - \begin{pmatrix} \tilde{\mathbf{C}}_{t-1} \Delta\tilde{\mathbf{k}}_t \\ 0 \end{pmatrix}$
    $d_t = \frac{\gamma\sigma^2}{s_{t-1}} d_{t-1} + r_{t-1} - \Delta\tilde{\mathbf{k}}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1}$
    $s_t = (1+\gamma^2)\sigma^2 + \Delta k_{tt} - \Delta\tilde{\mathbf{k}}_t^\top \tilde{\mathbf{C}}_{t-1} \Delta\tilde{\mathbf{k}}_t + \frac{2\gamma\sigma^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1}^\top \Delta\tilde{\mathbf{k}}_t - \frac{\gamma^2\sigma^4}{s_{t-1}}$
    $\tilde{\boldsymbol{\alpha}}_{t-1} = \begin{pmatrix} \tilde{\boldsymbol{\alpha}}_{t-1} \\ 0 \end{pmatrix}$
    $\tilde{\mathbf{C}}_{t-1} = \begin{bmatrix} \tilde{\mathbf{C}}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix}$
  else
    $\tilde{\mathbf{h}}_t = \boldsymbol{a}_{t-1} - \gamma \boldsymbol{a}_t$
    $\Delta k_{tt} = \tilde{\mathbf{h}}_t^\top \Delta\tilde{\mathbf{k}}_t$
    $\tilde{\mathbf{c}}_t = \frac{\gamma\sigma^2}{s_{t-1}} \tilde{\mathbf{c}}_t + \tilde{\mathbf{h}}_t - \tilde{\mathbf{C}}_{t-1} \Delta\tilde{\mathbf{k}}_t$
    $s_t = (1+\gamma^2)\sigma^2 + \Delta\tilde{\mathbf{k}}_t^\top \left(\tilde{\mathbf{c}}_t + \frac{\gamma\sigma^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1}\right) - \frac{\gamma^2\sigma^4}{s_{t-1}}$
  end if
    $\tilde{\boldsymbol{\alpha}}_t = \tilde{\boldsymbol{\alpha}}_{t-1} + \frac{\tilde{\mathbf{c}}_t}{s_t} d_t$
    $\tilde{\mathbf{C}}_t = \tilde{\mathbf{C}}_{t-1} + \frac{1}{s_t} \tilde{\mathbf{c}}_t \tilde{\mathbf{c}}_t^\top$
end for
return $\mathcal{D}_t$, $\tilde{\boldsymbol{\alpha}}_t$, $\tilde{\mathbf{C}}_t$

4. Policy Improvement with GPSARSA

As mentioned above, SARSA is a fairly straightforward extension of the TD algorithm (Sutton & Barto, 1998), in which state-action values are estimated, thus allowing policy improvement steps to be performed without requiring any additional knowledge on the MDP model. The idea is to use the stationary policy μ being followed in order to define a new, augmented process, the state space of which is $\mathcal{X}' = \mathcal{X} \times \mathcal{U}$, (i.e., the original state space augmented by the action space), maintaining the same reward model. This augmented process is Markovian with transition probabilities $p'(\mathbf{x}',\mathbf{u}'|\mathbf{x},\mathbf{u}) = p^\mu(\mathbf{x}'|\mathbf{x})\mu(\mathbf{u}'|\mathbf{x}')$. SARSA is simply the TD algorithm applied to this new process. The same reasoning may be applied to derive a GPSARSA algorithm from the GPTD algorithm. All we need is to define a covariance kernel function over state-action pairs, i.e., $k : (\mathcal{X} \times \mathcal{U}) \times (\mathcal{X} \times \mathcal{U}) \to \mathbb{R}$. Since states and actions are different entities it makes sense to decompose k into a state-kernel k_x and an action-kernel k_u: $k(\mathbf{x},\mathbf{u},\mathbf{x}',\mathbf{u}') = k_x(\mathbf{x},\mathbf{x}')k_u(\mathbf{u},\mathbf{u}')$. If both k_x and k_u are kernels we know that k is also a legitimate kernel (Schölkopf & Smola, 2002), and just as the state-kernel codes our prior beliefs concerning correlations between the values of different states, so should the action-kernel code our prior beliefs on value correlations between different actions.

All that remains now is to run GPTD on the augmented state-reward sequence, using the new state-

action kernel function. Action selection may be performed by ε-greedily choosing the highest ranking action, and slowly decreasing ε toward zero. However, we may run into difficulties trying to find the highest ranking action from a large or even infinite number of possible actions. This may be solved by sampling the value estimates for a few randomly chosen actions and maximize only among these, or using a fast iterative maximization method, such as the quasi-Newton method or conjugate gradients. Ideally, we should design the action kernel in such a way as to provide a closed-form expression for the greedy action.

5. Experiments

As an example let us consider a two-dimensional continuous world residing in the unit square $\mathcal{X} = [0, 1]^2$. A RL agent is required to solve a maze navigation problem in this world and find the shortest path from any point in $\mathcal{X}$ to a goal region, while avoiding obstacles. From any state, the agent may take a 0.1-long step in *any direction*. For each time step until it reaches a goal state the agent is penalized with a negative reward of -1; if it hits an obstacle it is returned to its original position. Let us represent an action as the unit vector pointing in the direction of the corresponding move, thus making $\mathcal{U}$ the unit circle. We leave the space kernel k_x unspecified and focus on the action kernel. Let us define k_u as follows, $k_u(\mathbf{u}, \mathbf{u}') = 1 + \frac{(1-b)}{2}(\mathbf{u}^\top \mathbf{u}' - 1)$, where b is a constant in $[0, 1]$. Since $\mathbf{u}^\top \mathbf{u}'$ is the cosine of the angle between $\mathbf{u}$ and $\mathbf{u}'$, $k_u(\mathbf{u}, \mathbf{u}')$ attains its maximal value of 1 when the two actions are the same, and its minimal value of b when the actions are 180 degrees apart. Setting b to a positive value is reasonable, since even opposite actions from the same state are expected, a priori, to have positively correlated values. However, the most valuable feature of this kernel is its linearity, which makes it possible to maximize the value estimate over the actions analytically.

Assume that the agent runs GPSARSA, so that it maintains a dictionary of state-action pairs $D_t = \{(\tilde{\mathbf{x}}_i, \tilde{\mathbf{u}}_i)\}_{i=1}^m$. The agent's value estimate for its current state $\mathbf{x}$ and an arbitrary action $\mathbf{u}$ is $\hat{v}(\mathbf{x}, \mathbf{u}) = \sum_{i=1}^m \tilde{\boldsymbol{\alpha}}_i k_x(\tilde{\mathbf{x}}_i, \mathbf{x}) k_u(\tilde{\mathbf{u}}_i, \mathbf{u}) = \sum_{i=1}^m \tilde{\boldsymbol{\alpha}}_i k_x(\tilde{\mathbf{x}}_i, \mathbf{x}) \left(1 + \frac{(1-b)}{2}(\tilde{\mathbf{u}}_i^\top \mathbf{u} - 1)\right)$. Maximizing this expression w.r.t. $\mathbf{u}$ amounts to maximizing $\sum_{i=1}^m \beta_i(\mathbf{x}) \tilde{\mathbf{u}}_i^\top \mathbf{u}$ subject to the constraint $\|\mathbf{u}\| = 1$, where $\beta_i(\mathbf{x}) \stackrel{\text{def}}{=} \tilde{\boldsymbol{\alpha}}_i k_x(\tilde{\mathbf{x}}_i, \mathbf{x})$. Solving this problem using a single Lagrange multiplier results in the greedy action $\mathbf{u}^* = \frac{1}{\lambda} \sum_{i=1}^m \beta_i(\mathbf{x}) \tilde{\mathbf{u}}_i$ where λ is a normalizing constant. It is also possible to maximize the variance estimate. This may be used to select non-greedy exploratory moves, by choosing the action the value of which the agent is least certain about. Performing this maximization amounts to solving a 2×2 Eigenvalue problem, which, for lack of space, we defer to a longer version of this paper.

Our experimental test-bed is the continuous state-action maze described above. In order to introduce stochasticity into the transitions, beyond the randomness inherent in the ε-greedy policy, we corrupt the moves chosen by the agent with a zero-mean uniformly distributed angular noise in the range of ± 30 degrees.

A GPSARSA-learning agent was put through 200 episodes, each of which consisted of placing it in a random position in the maze, and letting it roam the maze until it reaches a goal position (success) or until 100 time-steps elapse, whichever happens first. At each episode, ε was set to $10/(10 + T)$, T being the number of successful episodes completed up to that point. The σ parameter of the intrinsic variance was fixed at 1 for all states. The state kernel was Gaussian $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') = c \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2/(2\sigma_k^2)\right)$, with $\sigma_k = 0.2$ and a $c = 10$ (c is the prior variance of V, since $k(\mathbf{x}, \mathbf{x}) = c$). The action kernel was the linear kernel described above. The parameter ν controlling the sparsity of the solution was $\nu = 0.1$, resulting in a dictionary that saturates at 150-160 state-action pairs. The results of such a run on a maze similar to the one used in Engel et al. (2003) are shown in Fig. 1. The next experiment shows how the original GPTD algorithm of Engel et al. (2003) fails when the state dynamics become non-deterministic, as opposed to the MC-GPTD algorithm. To demonstrate this we use a simple 10 state random-walk MDP. The 10 states are arranged linearly from state 1 on the left to state 10 on the right. The left-hand wall is a retaining barrier, meaning that if a left step is made from state 1, the state transitioned to is again state 1. State 10 is a zero reward absorbing state. The stochasticity in the transitions is introduced by the policy, which is defined by the single parameter $\Pr(right)$ – the probability of making a step to the right ($\Pr(left) = 1 - \Pr(right)$). Each algorithm was run for 400 episodes, where each episode begins at state 1 and ends when the agent reaches state 10. This was repeated 10 times for each $\Pr(right) \in \{0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1\}$. The mean rewards equal 1 for states 1-9 and 0 for state 10. The observed rewards for states 1-9 were obtained by corrupting the mean rewards with a 0.1 standard deviation IID Gaussian noise. The kernel used was Gaussian with a standard deviation of 1 (for the purpose of computing the kernel, the states may be considered to be embedded in the real interval $[1, 10]$, with the Euclidean distance as the metric).

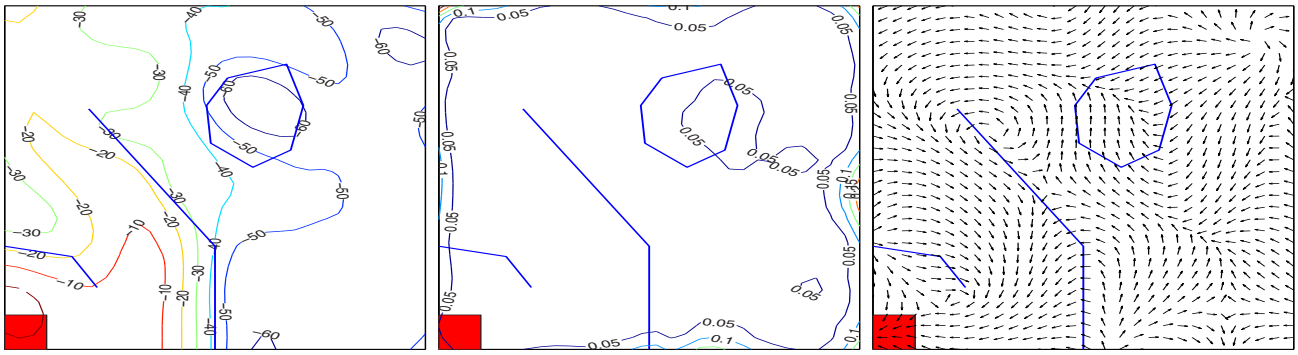

Figure 1. The posterior value mean (left), posterior value variance (center), and the corresponding greedy policy (right), for the maze shown here, after 200 learning episodes. The goal is at the bottom left.

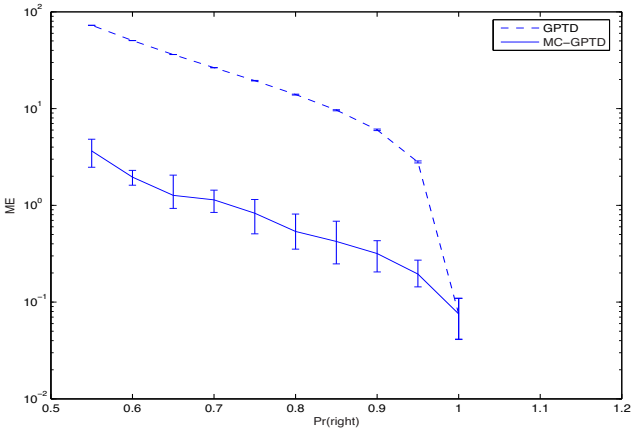

Figure 2. MC-GPTD compared with the original GPTD on the random-walk MDP. The mean error after 400 episodes for each algorithm is plotted against Pr(*right*).

We then computed the root-mean-squared error (RMSE) between the resulting value estimates and the true value function (which is easily computed). In all of the experiments (except when $\Pr(right) = 1$, as explained below), the error of the original GPTD converged well before the 100 episode mark. The results of this experiment are shown in Fig. 2. Both algorithms converge to the same low error for $\Pr(right) = 1$ (i.e. a deterministic policy), with almost identical learning curves (not shown). However, as $\Pr(right)$ is lowered, making the transitions stochastic, the original GPTD algorithm converges to inconsistent value estimates, whereas MC-GPTD produces consistent results. This bias was already observed and reported in Engel et al. (2003) as the "dampening" of the value estimates.

6. Discussion

In Engel et al. (2003) GPTD was presented as an algorithm for learning a posterior distribution over value functions, for MDPs with stochastic rewards, but deterministic transitions. This was done by connecting the hidden values and the observed rewards by a generative model of the form of Eq. 2.8, in which the covariance of the noise term was diagonal: $\mathbf{\Sigma}_t = \sigma^2\mathbf{I}$. In the present paper we derived a second GPTD algorithm that overcomes the limitation of the first algorithm to deterministic transitions. We did this by invoking a useful decomposition of the discounted return random process into the value process, modeling our uncertainty concerning the MDP's model, and a zero-mean residual (2.4), modeling the MDP's intrinsic stochasticity; and by additionally assuming independence of the residuals. Surprisingly, all that this amounts to is the replacement of the diagonal noise covariance, employed in Engel et al. (2003), with a tridiagonal, correlated noise covariance: $\mathbf{\Sigma}_t = \sigma^2\mathbf{H}_t\mathbf{H}_t^\top$. This change induces a model that we have shown to be effectively equivalent to GP regression of Monte-Carlo samples of the discounted return. This should help uncover the implicit assumption underlying some of the most prevalent MC value estimation methods (e.g., TD(1) and LSTD(1)), namely, that the samples of the discounted return used are IID. Although in most realistic problems this assumption is clearly wrong, we nevertheless know that MC estimates, although not necessarily optimal, are asymptotically consistent.

We are therefore inclined to adopt a broader view of GPTD as a general GP-based framework for Bayesian modeling of value functions, encompassing all generative models of the form $R = \mathbf{H}_t V + N$, with $\mathbf{H}_t$ given by (2.7), a Gaussian prior placed on V, and an arbitrary zero-mean Gaussian noise process N. No doubt, most such models will be meaningless from a value estimation point of view, while others will not admit efficient recursive algorithms for computing the posterior value moments. However, if the noise covariance $\mathbf{\Sigma}_t$ is suitably chosen, and if it is additionally *simple*

in some way, we may be able to derive such a recursive algorithm to compute complete posterior value distributions, on-line. For instance, it turns out that by employing alternative forms of noise covariance, we are able to obtain GP-based variants of LSTD(λ) (Engel, 2005).

The second contribution of this paper is the extension of GPTD to the estimation of state-action values, or Q-values, leading to the GPSARSA algorithm. Learning Q-values makes the task of policy improvement in the absence of a transition model tenable, even when the action space is continuous, as demonstrated by the example in Section 5. The availability of confidence intervals for Q-values significantly expands the repertoire of possible exploration strategies. In finite MDPs, strategies employing such confidence intervals have been experimentally shown to perform more efficiently then conventional ε-greedy or Boltzmann sampling strategies, e.g., Kaelbling (1993); Dearden et al. (1998); Even-Dar et al. (2003). GPSARSA allows such methods to be applied to infinite MDPs, and it remains to be seen whether significant improvements can be so achieved for realistic problems with continuous space and action spaces.

In Rasmussen and Kuss (2004) an alternative approach to employing GPs in RL is proposed. The approach in that paper is fundamentally different from the generative approach of the GPTD framework. In Rasmussen and Kuss (2004) one GP is used to learn the MDP's transition model, while another is used to estimate the value. This leads to an inherently off-line algorithm, which is not capable of interacting with the controlled system directly and updating its estimates as additional data arrive. There are several other shortcomings that limit the usefulness of that framework. First, the state dynamics is assumed to be factored, in the sense that each state coordinate is assumed to evolve in time independently of all others. This is a rather strong assumption that is not likely to be satisfied in many real problems. Moreover, it is also assumed that the reward function is completely known in advance, and is of a very special form – either polynomial or Gaussian. Finally, the covariance kernels used are also restricted to be either polynomial or Gaussian or a mixture of the two, due to the need to integrate over products of GPs. This considerably diminishes the appeal of employing GPs, since one of the main reasons for using them, and kernel methods in general, is the richness of expression inherent in the ability to construct arbitrary kernels, reflecting domain and problem-specific knowledge, and defined over sets of diverse objects, such as text documents and DNA sequences (to name only two), and not just points in metric space.

Preliminary results on a high dimensional control task, in which GPTD is used in learning to control a simulated robotic "Octopus" arm, with 88 state variables, suggest that the kernel-based GPTD framework is not limited to low dimensional domains such as those experimented with here (Engel et al., 2005). Standing challenges for future work include balancing exploration and exploitation in RL using the value confidence intervals provided by GPTD methods; further exploring the space of GPTD models by considering additional noise covariance structures; application of the GPTD methodology to POMDPs; creating a GP-Actor-Critic architecture; GPQ-Learning for off-policy learning of the optimal policy; and analyzing the convergence properties of GPTD.

References

Bertsekas, D., & Tsitsiklis, J. (1996). *Neuro-dynamic programming.* Athena Scientific.

Dearden, R., Friedman, N., & Russell, S. (1998). Bayesian Q-learning. *Proc. of the Fifteenth National Conference on Artificial Intelligence.*

Engel, Y. (2005). *Algorithms and Representations for Reinforcement Learning.* Doctoral dissertation, The Hebrew University of Jerusalem. `www.cs.ualberta.ca/∼yaki`.

Engel, Y., Mannor, S., & Meir, R. (2003). Bayes meets Bellman: The Gaussian process approach to temporal difference learning. *Proc. of the 20th International Conference on Machine Learning.*

Engel, Y., Szabo, P., & Volkinstein, D. (2005). Learning to control an Octopus arm using Gaussian process temporal difference learning. `www.cs.ualberta.ca/∼yaki/reports/octopus.pdf`.

Even-Dar, E., Mannor, S., & Mansour, Y. (2003). Action elimination and stopping conditions for reinforcement learning. *Proc. of the 20th International Conference on Machine Learning.*

Kaelbling, L. P. (1993). *Learning in embedded systems.* MIT Press.

Mannor, S., Simester, D., Sun, P., & Tsitsiklis, J. (2004). Bias and variance in value function estimation. *Proc. of the 21st International Conference on Machine Learning.*

Rasmussen, C., & Kuss, M. (2004). Gaussian processes in reinforcement learning. *Advances in Neural Information Processing Systems 16.* Cambridge, MA: MIT Press.

Scharf, L. (1991). *Statistical signal processing.* Addison-Wesley.

Schölkopf, B., & Smola, A. (2002). *Learning with Kernels.* Cambridge, MA: MIT Press.

Sutton, R., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* MIT Press.