# Value Function Approximation through Sparse Bayesian Modeling

Nikolaos Tziortziotis and Konstantinos Blekas

Department of Computer Science, University of Ioannina,
P.O. Box 1186, 45110 Ioannina, Greece
{ntziorzi,kblekas}@cs.uoi.gr

**Abstract.** In this study we present a sparse Bayesian framework for value function approximation. The proposed method is based on the on-line construction of a dictionary of states which are collected during the exploration of the environment by the agent. A linear regression model is established for the observed partial discounted return of such dictionary states, where we employ the Relevance Vector Machine (RVM) and exploit its enhanced modeling capability due to the embedded sparsity properties. In order to speed-up the optimization procedure and allow dealing with large-scale problems, an incremental strategy is adopted. A number of experiments have been conducted on both simulated and real environments, where we took promising results in comparison with another Bayesian approach that uses Gaussian processes.

**Keywords:** Value function approximation, Sparse Bayesian modeling, Relevance Vector Machine, Incremental learning.

## 1 Introduction

Reinforcement learning (RL) [13] aims at controlling an autonomous agent in an environment which is usually unknown. The agent is only aware of a reward signal that is applied to it when acting with the environment. In this manner, the actions are evaluated and the learning process is designed on choosing the action with the optimum expected return. The goal of RL is to discover an optimal policy, where in most cases this is equivalent to estimating the value function of states. A plethora of methods has been proposed in the last decades using a variety of value-function estimation techniques [5]. Algorithms such as the Q-learning [18] and Sarsa [9,12] try to estimate the long-term expected value of each possible action given a particular state by choosing actions with the maximum value. However, these methods have some drawbacks that prevent them from using in large or continuous state spaces of real-world applications. Value function approximation approaches offer a nice solution to this problem. Least-squares temporal-difference (LSTD) learning [2] is a widely used algorithm for value function learning of a fixed policy. Also, the least-squares policy-iteration (LSPI) method [6] extends the LSTD by using it in the policy evaluation step of policy estimation.

Recently, kernelized reinforcement learning methods have been paid a lot of attention by employing all the benefits of kernel techniques [14]. In this manner, standard RL methods have been extended by mapping to kernel spaces, see for example [20,19]. One particularly elegant Bayesian RL formulation is the Gaussian Process Temporal Difference (GPTD) [3], that constitutes an efficient adaptation of the Gaussian processes to the problem of online value-function estimation. The GPTD employs a probabilistic generative model for the state value function, and the solution to the inference problem is given by the posterior distribution conditioned on the observed sequence of rewards. An on-line kernel sparsification algorithm has also been proposed in [3], by incrementally constructing an appropriate dictionary of representative states. Finally, the Kalman Temporal Differences (KTD) framework has been introduced only recently [4], where the value function approximation is stated as a filtering problem and nonstationarities are allowed through the specification of some evolution model for parameters.

In this study an alternative Bayesian scheme for value function approximation is presented. The key aspects of our method are the creation of a state dictionary and the partial discounted return which corresponds to the accumulated reward between two states that get placed in the dictionary. The advantages of this approach are threefold. First, it achieves a reduced computational complexity, since our analysis deals only with the states which are stored in the dictionary. At a second level, it manages to avoid making approximations when dealing with large-scale problems, as in the case of GPTD method for calculating the kernel covariance matrix. Finally, it offers enhanced modeling capabilities due to the embedded sparsity model properties. More specifically, the proposed method addresses the problem of value function approximation by appropriately creating a linear regression model. Training this model is achieved through a sparse Bayesian methodology [15,11] that offers many advantages in regression. Enforcing sparsity is a fundamental machine learning regularization principle that causes to obtain more flexible inference methods. In sparse Bayesian regression we employ models having initially many degrees of freedom, where we apply a heavy tail prior over coefficients. After training, only few coefficients will be maintained, since they will be automatically considered as significant. This is equivalent to retaining only a part of the dictionary which will be responsible for estimating the value function and designing the optimum policy. Furthermore, we have used a computationally efficient incremental strategy that presented in [16], in order to accelerate the optimization procedure. The proposed method was tested on a suite of benchmarks including known simulated environments, as well as real environments using a PeopleBot mobile robot. Comparison has been made using the sparse on-line version of the GPTD algorithm.

In section 2, we briefly describe the Markov Decision Processes (MDPs) and the GPTD method as a Bayesian framework for value function approximation. The proposed sparse regression model is then presented in section 3, along with an incremental learning procedure. To assess the performance of our methodology we present in section 4 numerical experiments with artificial and real test environments. Finally, in section 5 we give conclusions and suggestions for future research.

## 2 Markov Decision Processes and GPTD

In the most standard formulation of the problem, the environment where an agent acts, is modeled as a Markov Decision Process (MDP) [13]. A MDP is denoted as a tuple $\{\mathcal{S}, \mathcal{A}, R, P, \gamma\}$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively; $R$ is a reward function that specifies the immediate reward for each transition; $P$ is the state transition distribution; and $\gamma \in [0, 1]$ is a discount factor that determines the importance of current and future rewards. A stationary policy $\pi$ defines a probability distribution over the action space condition on the states and can be seen as a mapping $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. The discounted return $D(s)$ for a state $s$ under a policy $\pi$, having a policy dependent state transition probability distribution $p^{\pi}(\cdot|s_t)$, is given by

$$D(s) = \sum_{t=0}^{\infty} \gamma^t R(s_t)|s_0 = s \, . \tag{1}$$

This can be written more concisely as

$$D(s) = R(s) + \gamma D(s') \text{ , where } s' \sim p^{\pi}(\cdot|s). \tag{2}$$

The objective of RL problems is to estimate an optimal policy $\pi^*$ which maximize the expected discounted return, $V^{\pi}(s) = E_{\pi}[D(s)]$. This can be translated into a value function approximation problem, according to the following recursive formulation:

$$V^{\pi}(s) = E_{\pi}\left[R(s_t) + \gamma V^{\pi}(s_{t+1})|s_t = s\right]. \tag{3}$$

Equation 3 is the Bellman equation for $V^{\pi}$ which expresses a relationship between the values of current and next state. Alternatively, the state-action value function usually used to facilitate policy improvement. This is the expected discounted return starting from state $s$, taking the action $a$ and then following the policy $\pi$:

$$Q^{\pi}(s, a) = E_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|s_0 = s, a_0 = a\right]. \tag{4}$$

Having found the optimal action-state value function $Q^*$, the optimal policy is given by $\pi^*(s) = \arg\max_a Q^*(s, a)$.

Gaussian Processes [8] have recently been used as a Bayesian framework for modeling RL tasks. Gaussian Process Temporal Difference (GPTD) [3] is based on describing the value function as a Gaussian process. In particular, a decomposition of the discounted return $D(s)$ is first considered into its mean value and a zero mean residual:

$$D(s) = V(s) + (D(s) - V(s)) = V(s) + \mathit{\Delta} V(s) \, . \tag{5}$$

By combining Eqs. 5, 2 we obtain the following rule:

$$R(s) = V(s) - \gamma V(s') + N(s, s') \, , \tag{6}$$

where $N(s, s') = \mathit{\Delta} V(s) - \gamma \mathit{\Delta} V(s')$ is the difference between residuals. Given a sample trajectory of states $\{s_1, \ldots, s_t\}$, the model results in a set of $t - 1$ linear equations

$$R_t = H_t V_t + N_t, \tag{7}$$

where $R_t, V_t, N_t$ are vectors of rewards, value functions and residuals, respectively. Additionally, the $H_t$ is a matrix of size $(t-1) \times t$ and is given by

$$H_t = \begin{bmatrix} 1 & -\gamma & 0 & \cdots & 0 \\ 0 & 1 & -\gamma & \cdots & 0 \\ \vdots & & & \vdots & \\ 0 & 0 & \cdots & 1 & -\gamma \end{bmatrix}. \tag{8}$$

By considering the above equation as a Gaussian Process, a zero-mean Gaussian prior is assumed over the value functions $V_t$, i.e., $V_t \sim \mathcal{N}(\mathbf{0}, K_t)$, where $K_t$ is a kernel covariance matrix over states. Also, the residuals $N_t$ is assumed to be zero-mean Gaussian, $N_t \sim \mathcal{N}(\mathbf{0}, \Sigma_t)$, where the covariance matrix is calculated as $\Sigma_t = \sigma_t^2 H_t H_t^\top$. In this way, at each time a state $s$ is visited, the value function of the state is given by the posterior distribution, which is also Gaussian, $(V(s)|R_t) \sim \mathcal{N}(\hat{V}(s), p_t(s))$, where

$$\hat{V}(s) = k_t(s)^\top \alpha_t, \qquad \alpha_t = H_t^\top (H_t K_t H_t^\top + \Sigma_t)^{-1} R_t,$$
$$\text{and} \quad p_t(s) = k(s,s) - k_t(s)^\top C_t k_t(s), \qquad C_t = H_t^\top (H_t K_t H_t^\top + \Sigma_t)^{-1} H_t.$$

A limitation to the application of the GPTD is the computational complexity that increases linearly with time $t$. To solve this problem, an on-line kernel sparsification algorithm has been proposed in [3] which is based on the construction of a dictionary of representative states, $\mathcal{D}_{t-1} = \{\tilde{s}_1, \dots, \tilde{s}_{d_{t-1}}\}$. An approximate linear dependence (ALD) analysis is performed in order to examine whether or not a visited state $s_t$ must be entered into the dictionary. This is achieved according to a least squares problem, where we test if the image of the candidate state, $\phi(s_t)$, can be adequately approximated by the elements of the current dictionary [3], i.e.

$$\delta_t = \min_{\boldsymbol{a}} \left\| \sum_j a_j \phi(\tilde{s}_j) - \phi(s_t) \right\|^2 \leq \nu, \tag{9}$$

where $\nu$ is a positive threshold that controls the level of sparsity. The sparse on-line version of GPTD makes further approximations for calculating the kernel matrix $K_t$, where it uses only the dictionary members for this purpose; for more details see [3].

## 3 The Proposed Method

An advanced methodology to the task of control learning is presented in this section that employs the Relevance Vector Machine (RVM) [15] generative model for value function approximation. We start our analysis by taking into account the stationarity property of the MDP, which allows us to rewrite the discounted return of Eq. 1 at time step $t$ as

$$D(s_t) = \mathcal{R}(s_t) + \gamma^{k_t} D(s_{t+k_t}), \tag{10}$$

where $\mathcal{R}(s_t) = \sum_{j=0}^{k_t-1} \gamma^j R(s_{t+j})$ is the *partial discounted return* of a state-reward subsequence. The term $k_t$ denotes the time difference between two states that have been observed. According to the Eq. 5, the discounted return can be decomposed into its mean and a zero-mean residual. Considering this assumption and substituting Eq. 5 into Eq. 10 leads us to the following rule:

$$\mathcal{R}(s_t) = V(s_t) - \gamma^{k_t} V(s_{t+k_t}) + N(s_t, s_{t+k_t}), \tag{11}$$

where $N(s_t, s_{t+k_t}) = \Delta V(s_t) - \gamma^{k_t} \Delta V(s_{t+k_t})$ is the residuals difference.

Thus, assuming a dictionary of $n$ states $\mathcal{D}_t = \{\tilde{s}_1, \dots, \tilde{s}_n\}$, we obtain a set of $n-1$ equations:

$$\mathcal{R}(\tilde{s}_i) = V(\tilde{s}_i) - \gamma^{k_i} V(\tilde{s}_{i+1}) + N(\tilde{s}_i, \tilde{s}_{i+1}), \quad \text{for } i = 1, \dots, n-1\,, \tag{12}$$

which can be written more concisely as

$$\mathcal{R}_n = H_n V_n + N_n, \tag{13}$$

where $\mathcal{R}_n = (\mathcal{R}(\tilde{s}_1), \dots, \mathcal{R}(\tilde{s}_{n-1}))^T$, $V_n = (V(\tilde{s}_1), \dots, V(\tilde{s}_n))^T$ and $N_n = (N(\tilde{s}_1, \tilde{s}_2), \dots, N(\tilde{s}_{n-1}, \tilde{s}_n))^T$. The matrix $H_n$ is of size $(n-1) \times n$ and has the following form

$$H_n = \begin{bmatrix} 1 & -\gamma^{k_1} & 0 & \cdots & 0 \\ 0 & 1 & -\gamma^{k_2} & \cdots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \cdots & 1 & -\gamma^{k_{n-1}} \end{bmatrix}. \tag{14}$$

Moreover, we assume that the (hidden) vector of the value functions is described with the functional form of a linear model

$$V_n = \Phi_n \mathbf{w}_n\,, \tag{15}$$

where $\mathbf{w}_n$ is the vector of the $n$ unknown model regression coefficients. $\Phi_n = [\phi_1^T \dots \phi_n^T]$ is a kernel 'design' matrix that contains $n$ basis functions $\phi_i$, where the values of their components have been calculated using a kernel function $\phi_i(\tilde{s}_j) \equiv k(\tilde{s}_i, \tilde{s}_j)$. It must be noted that during our experimental study we have considered Gaussian type of kernels governed by a scalar parameter (kernel width). Thus, Eq. 13 can be written as

$$\mathcal{R}_n = H_n \Phi_n \mathbf{w}_n + N_n, \tag{16}$$

that can be further simplified as:

$$\mathbf{y}_n = \Phi_n \mathbf{w}_n + \mathbf{e}_n\,. \tag{17}$$

The above equation describes a linear regression model that fits the modified observations, $\mathbf{y}_n = (H_n^\top H_n)^{-1} H_n^\top \mathcal{R}_n$. The term $\mathbf{e}_n$ plays the role of the stochastic model noise and is assumed to be a zero-mean Gaussian with precision $\beta_n$, i.e. $\mathbf{e}_n \sim \mathcal{N}(0, \beta_n^{-1} I)$. Under this prism, the conditional probability density of the sequence $\mathbf{y}_n$ is also Gaussian, i.e.

$$p(\mathbf{y}_n | \mathbf{w}_n, \beta_n) = \mathcal{N}(\mathbf{y}_n | \Phi_n \mathbf{w}_n, \beta_n^{-1} I). \tag{18}$$

An important issue is how to define the optimal order of the above regression model. Sparse Bayesian methodology offers an advanced solution to this problem by penalizing large order models. This is the idea behind the Relevance Vector Machines (RVM) [15]. More specifically, a heavy-tailed prior distribution, $p(\mathbf{w}_n)$, is imposed over the regression coefficients $\mathbf{w}_n$ to zero out most of the weights $w_{ni}$ after training. This is achieved in an hierarchical way: First, a zero-mean Gaussian distribution is considered

$$p(\mathbf{w}_n|\boldsymbol{\alpha}_n) = \mathcal{N}(\mathbf{w}_n|0, A_n^{-1}) = \prod_{i=1}^{n} \mathcal{N}(w_{ni}|0, \alpha_{ni}^{-1}), \tag{19}$$

where $A_n$ is a diagonal matrix containing the $n$ elements of the precision vector $\boldsymbol{\alpha}_n = (\alpha_{n1}, \ldots, \alpha_{nn})^\top$. At a second level, a Gamma hyperprior is imposed over each hyperparameter, $\alpha_{ni}$,

$$p(\boldsymbol{\alpha}_n) = \prod_{i=1}^{n} Gamma(\alpha_{ni}|a, b). \tag{20}$$

It must be noted that both Gamma parameters $a, b$, are *a priori* set to zero in order to make these priors uninformative.

This two-stage hierarchical prior is actually a Student's-t distribution that provides sparseness to the model [15], since it enforces most of the parameters $\alpha_{ni}$ to become large and as a result the corresponding weights $w_{ni}$ are set to zero. In this way, the complexity of the regression models is controlled automatically, while at the same time over-fitting is avoided. Furthermore, we can obtain the marginal likelihood distribution of sequence $\mathbf{y}_n$ by integrating out the weights $\mathbf{w}_n$. This gives a zero mean Gaussian:

$$p(\mathbf{y}_n|\boldsymbol{\alpha}_n, \beta_n) = \int p(\mathbf{y}_n|\mathbf{w}_n, \beta_n) p(\mathbf{w}_n|\boldsymbol{\alpha}_n) dw = \mathcal{N}(0, C_n), \tag{21}$$

where the covariance matrix has the form, $C_n = \Phi_n A_n^{-1} \Phi_n^\top + \beta_n^{-1} I$.

From the Bayes rule, the posterior distribution of the weights can be also obtained as [15]:

$$p(\mathbf{w}_n|\mathbf{y}_n, \boldsymbol{\alpha}_n, \beta_n) = \mathcal{N}(\mathbf{w}_n|\boldsymbol{\mu}_n, \Sigma_n), \tag{22}$$

where

$$\boldsymbol{\mu}_n = \beta_n \Sigma_n \Phi_n^\top \mathbf{y}_n, \quad \Sigma_n = (\beta_n \Phi_n^\top \Phi_n + A_n)^{-1}. \tag{23}$$

The maximization of the log-likelihood function of Eq. 21, leads to the following update rules for the model parameters [15]:

$$\alpha_{ni} = \frac{\gamma_i}{\mu_{ni}^2}, \tag{24}$$

$$\beta_n^{-1} = \frac{\|\mathbf{y}_n - \Phi_n \boldsymbol{\mu}_n\|^2}{n - \sum_{i=1}^{n} \gamma_i}, \tag{25}$$

where $\gamma_i = 1 - \alpha_{ni}[\Sigma_n]_{ii}$ and $[\Sigma_n]_{ii}$ is the i-th diagonal element of the matrix $\Sigma_n$. Thus, Equations 23, 24 and 25 are applied iteratively until convergence. The mean values of weights, $\boldsymbol{\mu}_n$, are finally used for the value function approximation of a state $s$, i.e. $\tilde{V}(s) = \boldsymbol{\phi}(s)^T \boldsymbol{\mu}_n$, where $\boldsymbol{\phi}(s) = (k(s, \tilde{s}_1), \ldots, k(s, \tilde{s}_n))^T$.

## 3.1 Incremental Optimization

The application of RVM in large scaling problem is problematic, since it requires the computation of matrix $\Sigma_n$ in Eq. 23. In our case this is happening when the size of the dictionary ($n$) becomes large. To deal with this problem we can follow an incremental learning algorithm that has been proposed in [16]. The method initially assumes that all states in the dictionary (all basis functions) have been pruned due to the sparsity constraint. This is equivalent to assuming that $\alpha_{ni} = \infty, \forall i = \{1, \dots, n\}$. Then, at each iteration a basis function is examined whether to be either added to the model, or removed from the model, or re-estimated. When adding a basis function, the value of the hyperparameter $\alpha_{ni}$ is estimated according to the maximum likelihood criterion.

In particular, it is easily to show that the term of the marginal likelihood of Eq. 21 which referred to the single parameter $\alpha_{ni}$ is [16]

$$\ell(\alpha_{ni}) = \frac{1}{2}\left(\log \alpha_{ni} - \log(\alpha_{ni} + s_{ni}) + \frac{q_{ni}^2}{\alpha_{ni} + s_{ni}}\right), \tag{26}$$

where

$$s_{ni} = \frac{\alpha_{ni} S_{ni}}{\alpha_{ni} - S_{ni}}, \qquad q_{ni} = \frac{\alpha_{ni} Q_{ni}}{\alpha_{ni} - S_{ni}}, \tag{27}$$

and $S_{ni} = \boldsymbol{\phi}_i^\top C_n^{-1} \boldsymbol{\phi}_i$, $Q_{ni} = \boldsymbol{\phi}_i^\top C_n^{-1} \mathbf{y}_n$, $\boldsymbol{\phi}_i = (\phi_i(\tilde{s}_1), \dots, \phi_i(\tilde{s}_n))^\top$. Note that in this manner the matrix inversion is avoided by using the Woodbury identity [16]. It has been shown in [16] that the log-likelihood has a single maximum at:

$$\alpha_{ni} = \frac{s_{ni}^2}{q_{ni}^2 - s_{ni}}, \quad \text{if } q_{ni}^2 > s_{ni}, \tag{28}$$

$$\alpha_{ni} = \infty, \quad \text{if } q_{ni}^2 \leq s_{ni}. \tag{29}$$

Thus, a basis function $\boldsymbol{\phi}_i$ is added (and so the corresponding state of the dictionary becomes active) in the case of $q_{ni}^2 > s_{ni}$. In the opposite case, this basis function is removed.

## 3.2 Working in Episodic Tasks and Unknown Environments

So far we have focused on solving continuing tasks, where the agent is placed initially to a random state and then is let to wander-off indefinitely. Since most of the RL tasks are episodic, a modification to our approach is needed to meet these requirements. During episodic tasks the agent will reach a terminal state within a finite number of steps. After that, a new episode (epoch) begins by placing the agent to a random initial position. An absorbing state may be thought as a state that only zero-rewards are received and that each action plays no role. In this case, the partial discounted return $\mathcal{R}(\tilde{s}_n)$ of the last inserted state ($\tilde{s}_n$) in an episode, is actually the discounted return $D(\tilde{s}_n)$, itself. Also, the discount factor $\gamma$ for the subsequent dictionary state ($\tilde{s}_{n+1}$) will be set to zero. Thus, the matrix $H_{n+1}$ will take the following form

$$H_{n+1} = \begin{bmatrix} 1 & -\gamma^{k_1} & 0 & \cdots & 0 & 0 \\ 0 & 1 & -\gamma^{k_2} & \cdots & 0 & 0 \\ \vdots & \vdots & & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -\gamma^{k_{n-1}} & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix}. \tag{30}$$

This new form of the matrix $H$ is the only modification to our approach in order to deal with episodic tasks.

Finally, in our study we have considered transitions between state-action pairs instead of single states, since the model environment is completely unknown. This is achieved by determining the kernel function as a product of state kernel $k_s$ and action kernel $k_a$, i.e. $k(s,a,s',a') = k_s(s,s')k_a(a,a')$ (legitimate kernel [10,1]). Therefore, we have considered the approximation of the optimal state-value function $Q$.

## 4 Experimental Results

The performance of our model (called as RVMTD) has been studied to several simulated and real environments. In all cases, two evaluation criteria have been used: the mean number of steps, as well as the mean return with respect to episodes. Comparison has been made with the on-line GPTD algorithm [3]. It must be noted that both methods use the same criterion for adding a new state to the dictionary (Eq. 9) with the same threshold parameter $\nu$. Also, the proper value for the scalar parameter of the Gaussian kernel in each problem was found experimentally. However, in some cases (mostly on simulated environments) the sensitivity of the performance of both approaches to this parameter was significant. Finally, in all cases, the decay parameter $\gamma$ was set to $0.99$.

### 4.1 Experiments on Simulated Environments

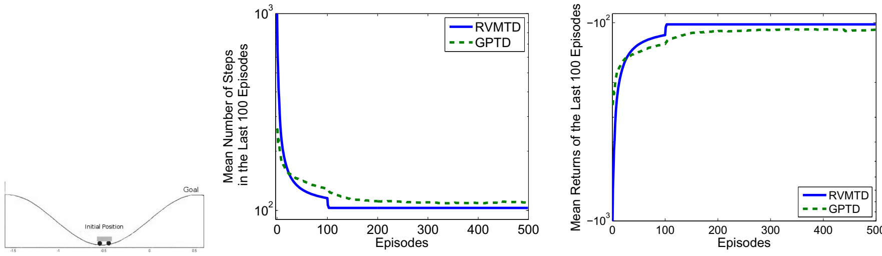
The first series of experiments was made using two well-known benchmarks [1]. The first one is the mountain car [7], where the objective of this task is to drive an underpowered car up a steep mountain road from a valley to tophill, as illustrated in the Fig. 1(a). Due to the force of gravity, the car cannot accelerate up to the tophill and thus it must go to the opposite slope to acquire enough momentum, so as to reach the goal on the right slope. The environmental states consist of two continuous variables: the position ($p_t \in [-1.5, +0.5]$) and the current velocity ($v_t \in [-0.07, 0.07]$) of the car. Initially, the car is standing motionless ($v_0 = 0$) at the position $p_0 = -0.5$. At each time step, it receives a negative reward $r = -1$. Three are the possible actions: +1 (full throttle forward), -1 (full throttle reverse) and 0 (zero throttle). An episode is terminated either when the car reaches the goal at the right tophill, or the total number of steps exceeds a maximum allowed value (1000). The state kernel $k_s$ was set as $k_s = k(s,s') = \exp\left(-\sum_{i=1}^{2}(s_i - s'_i)^2/(2\sigma_i^2)\right)$, where $\sigma_1^2 = 5\times10^{-2}$ and $\sigma_2^2 = 5\times10^{-4}$. On the other hand we have used a simple action kernel of type: $1$ when the actions are the same, $0.5$ when differs by one and $0$ otherwise. Finally, the parameter $\nu$ that

[1] Both simulators have been downloaded from http://www.dacya.ucm.es/jam/download.htm

specifies the sparsity of our model was set to $\nu = 0.001$, resulting in a dictionary that contains about 150 states.

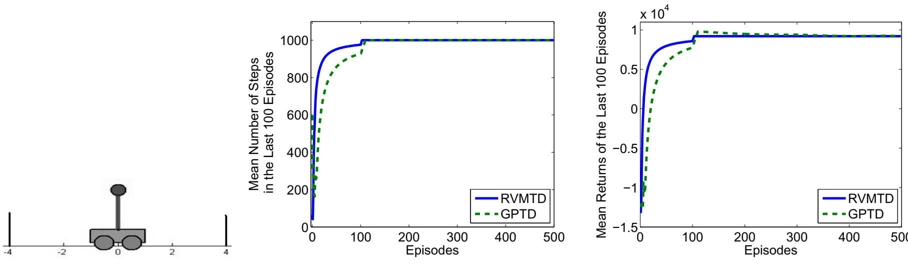
Another test environment is the famous cart pole shown in Fig. 2(a), where the objective is to keep the pole balanced and the cart within its limits by applying a fixed magnitude force either to the left or to the right. The states consists of four continuous variables: the horizontal position $(x)$ and the velocity $(\dot{x})$ of the cart, and the angle $(\theta)$ and the angular velocity $(\dot{\theta})$ of the pole. There are 21 possible discrete actions from -10 to 10, while the reward received by the environment takes the form $r = 10 - 10|10\theta|^2 - 5|x| - 10\dot{\theta}$. The cart is initially positioned in the middle of the track having zero velocity, and the pole is parallel to the vertical line having velocity $\dot{\theta} = 0.01$. An episode terminates when either the cart moves off the track, or the pole falls, or the pole is successfully balanced for 1000 time steps. Similarly, we have considered a Gaussian state kernel with different scalar parameter $(\sigma_i^2)$ per variable: $\sigma_1^2 = 2$, $\sigma_2^2 = 0.5$, $\sigma_3^2 = 0.008$ and $\sigma_4^2 = 0.1$. The action kernel was also Gaussian with variance, $\sigma^2 = 1$. Finally, the parameter $\nu$ was set to 0.1, resulting in a dictionary of size 100.

The depicted results on these problems are illustrated in Figs. 1 and 2, respectively. As it is obvious our method achieves to find the same or improved policy in comparison



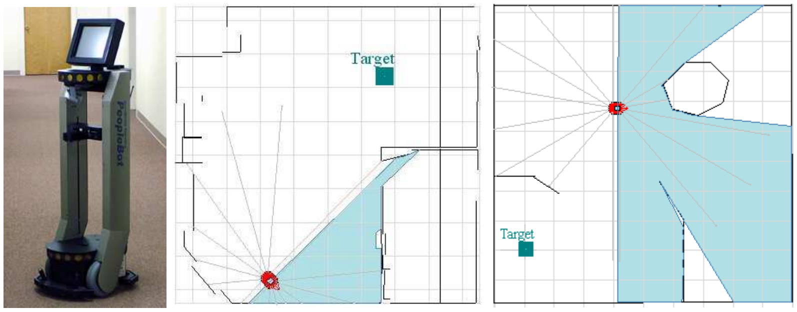
(a) mountain car (b) Mean number of steps (c) Mean return

**Fig. 1.** Experimental results with the mountain car simulator



(a) cart pole (b) Mean number of steps (c) Mean return

**Fig. 2.** Experimental results with the cart pole simulator



(a) PeopleBot (b) Stage world $S_1$ (c) Stage world $S_2$

**Fig. 3.** The mobile robot and the 2D-grid maps used in our experiments. These are two snapshots from the simulator with visualization of the robot's laser and sonar range scanner.

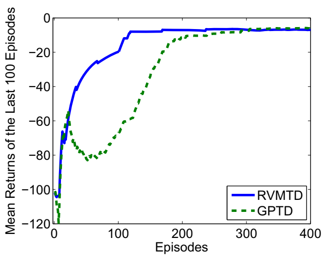
with the online GPTD. However, our approach has the tendency to convergence to the optimum solution much faster than GPTD. Especially in the case of the cart-pole, only a few episodes were capable of reaching the optimum policy with a smaller in size dictionary. It is interesting to note that, although both approaches converged to almost the same policy, the GPTD method requires a larger dictionary (almost double size), as well as higher execution time.

## 4.2 Experiments on a Mobile Robot

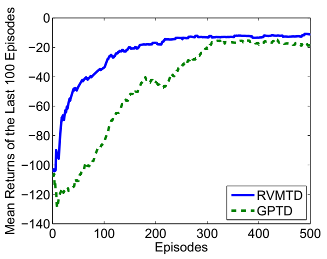
The performance of the proposed method have also been studied to a PeopleBot mobile robot, shown in Fig. 3, which is based on the robust P3-DX base. This is a wheeled mobile robot occupied with advanced tools for communication, through the ARIA (Advanced Robot Interface for Applications) library and various sensors, such as sonar, laser and a pan-tilt camera. In this work, only the sonar and laser sensors were used for obstacle avoidance. There is also available the MobileSim simulation environment built on the Stage platform which manages to simulate the real environment with satisfactory precision[2].

Two different grid maps (stage worlds) have been selected during our experiments, as shown in Fig.3. Note that the first one was obtained by mapping our laboratory using the PeopleBot robot and the MobileEyes software. In this study, the objective is to find a steady landmark (shown with a rectangular box in both maps of Fig.3) starting from any position in the world with the minimum number of steps. The robot receives a reward of -1 per time step, except when it finds an obstacle where the reward is -100. In our study we have discretized the action space into the 8 major compass winds, while the length of each step was $0.5m$. Also, the maximum allowed number of steps per episode was set to 100. Finally, we have used a Gaussian type of kernel for the environmental state with a scalar parameter value $\sigma^2 = 1$. The action kernel takes $4$ possible values: $1$ (when actions are the same), 0.6 (when differs $45°$), 0.3 (when differs $90°$), $0$ (otherwise). Note that, while we have used the same state kernel for both methods, in the case of the

[2] more details can be found at http://robots.mobilerobots.com/wiki/MobileSim

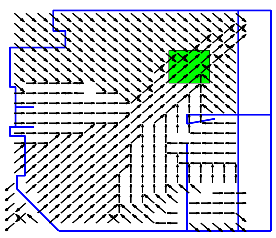


(a) Results on world $S_1$

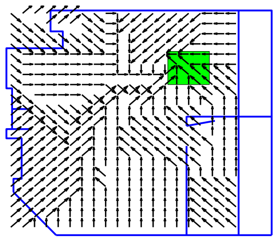


(b) Results on world $S_2$

**Fig. 4.** Plots of the mean return as estimated by both methods in two maps



RVMTD policy



GPTD policy

**Fig. 5.** Learned policies by both comparative methods in the case of test world $S_1$

online GPTD we have adopted the action kernel function described in [3], since it gave better performance.

The experimental results of the two worlds are shown in Fig. 4 which gives the plots of the mean returns received by the agent in the last 100 episodes. Obviously, the proposed method manages to discover a more optimal policy in a higher rate in comparison with the GPTD. This is more apparent in Fig. 5, where we show the estimated trajectories following the learned policy of each method during studying the stage word $S_1$ (Fig. 3b). In the proposed method, the robot is close to the optimal path between any start point and destination (target), and as a result it reaches the destination reliably and faster. We took the same performance behavior with the other stage word $S_2$ of Fig. 3.

## 5 Conclusions

In this paper we have proposed an advanced methodology for model-free value function approximation using the RVM regression framework as the generative model. The key aspect of the proposed technique lies on the introduction of the partial discounted returns that are observed during the creation of a state dictionary. This sequential data of rewards is modeled using a sparse Bayesian framework as employed by the RVM method that incorporates powerful modeling properties. We have also applied an incremental learning strategy that accelerates the optimization procedure and makes the method to be practical for large scale problems. As experiments have shown, our method is able to achieve better performance and to learn significantly more optimal policies. A future research direction to our study is to further improve the regression method and the kernel design matrix specification, by incorporating a mechanism for adapting the scale parameter of the Gaussian kernel function [17]. Another interesting topic for future study is to work on different schemes for the on-line dictionary construction that allowing the dictionary to be dynamically adjusted during learning.

## References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
2. Bradtke, S.J., Barto, A.G.: Linear least-squares algorithms for temporal difference learning. Machine Learning 22, 33–57 (1996)
3. Engel, Y., Mannor, S., Meir, R.: Reinforcement learning with gaussian process. In: International Conference on Machine Learning, pp. 201–208 (2005)
4. Geist, M., Pietquin, O.: Kalman Temporal Differences. Journal of Artificial Intelligence Research 39, 483–532 (2010)
5. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of Artificial Inteligence Research 4, 237–285 (1996)
6. Lagoudakis, M.G., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research 4, 1107–1149 (2003)
7. Moore, A.: Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In: Machine Learning: Proceedings of the Eighth International Conference. Morgan Kaufmann (June 1991)
8. Rasmussen, C., Williams, C.: Gaussian Processes for Machine Learning. MIT Press (2006)
9. Rummery, G.A., Niranjan, M.: On-line q-learning using connectionist systems. Tech. rep., Cambridge University Engineering Department (1994)
10. Scholkopf, B., Smola, A.: Learning with Kernels. MIT Press (2002)
11. Seeger, M.: Bayesian Inference and Optimal Design for the Sparse Linear Model. Journal of Machine Learning Research 9, 759–813 (2008)
12. Singh, S., Sutton, R.S., Kaelbling, P.: Reinforcement learning with replacing eligibility traces. Machine Learning, 123–158 (1996)
13. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
14. Taylor, G., Parr, R.: Kernelized value function approximation for reinforcement learning. In: International Conference on Machine Learning, pp. 1017–1024 (2009)
15. Tipping, M.E.: Sparse bayesian learning and the relevance vector machine. Journal of Machine Learning Research 1, 211–244 (2001)
16. Tipping, M.E., Faul, A.C.: Fast marginal likelihood maximization for sparse bayesian models. In: Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics (2003)
17. Tzikas, D., Likas, A., Galatsanos, N.: Sparse Bayesian modeling with adaptive kernel learning. IEEE Trans. on Neural Networks 20(6), 926–937 (2009)
18. Watkins, C., Dayan, P.: Q-learning. Machine Learning 8(3), 279–292 (1992)
19. Xu, X., Hu, D., Lu, X.: Kernel-based least squares policy iteration for reinforcement learning. IEEE Transactions on Neural Networks 18(4), 973–992 (2007)
20. Xu, X., Xie, T., Hu, D., Lu, X.: Kernel least-squares temporal difference learning. International Journal of Information Technology 11(9), 54–63 (2005)