A Bayesian Reinforcement Learning framework using Relevant Vector Machines

Nikolaos Tziortziotis and Konstantinos Blekas
Department of Computer Science, University of Ioannina,
45110 Ioannina, Greece
Tel. +30 26510 08816, Email: {ntziorzi,kblekas}@cs.uoi.gr

Abstract

In this work we present an advanced Bayesian formulation to the task of control learning that employs the Relevance Vector Machines (RVM) generative model for describing value functions. The key aspect of the proposed method is the design of the discount return as a generalized linear model that constitutes a well-known probabilistic approach. This allows to augment the model with advantageous sparse priors provided by the RVM's regression framework. We have also taken into account the significant issue of selecting the proper parameters of the kernel design matrix. We have demonstrated that our method produces improved performance in both simulated and real test environments.
Additional information on this work can be found at: http://www.cs.uoi.gr/∼ntziortzi/AAAI2011/

Introduction

Reinforcement Learning (RL) (Sutton and Barto 1998) refers to learning to control a system. It is a paradigm of Machine Learning (ML) in which rewards and punishments guide the learning process. In reinforcement learning an agent interacts with its environment, receiving observations and selecting actions to maximize a scalar reward signal that is provided by the environment. This interaction is usually modeled by Markov Decision Processes (MDPs). A standard MDP consists of $(\mathcal{X}, \mathcal{U}, R, p)$, where $\mathcal{X}$ and $\mathcal{U}$ are the state and action spaces, respectively. Further, $R : \mathcal{X} \to \Re$ is the reward function, while $p : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0, 1]$ is the state transition distribution. A policy $\mu : \mathcal{X} \times \mathcal{U} \to [0, 1]$ is a mapping from states to action selection probabilities. The *discounted return* for $D(\mathbf{x})$ for a state $\mathbf{x}$ under policy μ is a random process defined as

$$D(\mathbf{x}) = \sum_{i=0}^{\infty} \gamma^i R(\mathbf{x}_i) | \mathbf{x}_0 = \mathbf{x}, \tag{1}$$

where $\mathbf{x}_{i+1} \sim p^\mu(\cdot|\mathbf{x}_i)$ is the policy-dependent state transition probability distribution and $\gamma \in [0, 1]$ is the discount factor that determines the present value of future rewards. The above Equation can be written more concisely as

$$D(\mathbf{x}) = R(\mathbf{x}) + \gamma D(\mathbf{x}') \quad \text{where } \mathbf{x}' \sim p^\mu(\cdot|\mathbf{x}). \tag{2}$$

Copyright © 2012, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The RL goal is to estimate a *value function* for each state which is the expected discounted return under a policy μ with respect to the randomness in the trajectories and in the rewards collected therein, so $V(\mathbf{x}) = \mathbf{E}_\mu D(\mathbf{x})$. In this way, agent will select actions that drive to states with the best expected reward. During learning the agent must balance between selecting actions to achieve high reward and selecting actions to gain information about the environment. This is called exploration vs. exploitation trade off.

Gaussian Processes (Rasmussen and Williams 2006) have been recently used to optimal control problems providing an elegant Bayesian modeling RL framework. The Gaussian Process Temporal Difference (GPTD) (Engel, Mannor, and Meir 2005) assumes that value functions are modeled with Gaussian processes. In particular, a decomposition of the discounted return $D(x)$ is first considered into its mean value and a zero-mean residual ΔV, i.e.

$$D(\mathbf{x}) = V(\mathbf{x}) + \Delta V(\mathbf{x}). \tag{3}$$

Thus by combing Eqs. (3), (2) we obtain the following rule

$$R(\mathbf{x}) = V(\mathbf{x}) - \gamma V(\mathbf{x}') + N(\mathbf{x}, \mathbf{x}'), \tag{4}$$

where $\mathbf{x}' \sim p^\mu(\cdot|\mathbf{x})$. Given a sample trajectory $\mathbf{x_0}, \mathbf{x_1}, \ldots, \mathbf{x_t}$ the model results in the following set of t linear equations $R_t = H_t V_t + N_t$, where $R_t = (R(\mathbf{x}_0), \ldots, R(\mathbf{x}_t))$, $V_t = (V(\mathbf{x}_0), \ldots, V(\mathbf{x}_t))$, $N_t = (N(\mathbf{x}_0, \mathbf{x}_1), \ldots, N(\mathbf{x}_{t+1}, \mathbf{x}_t))$, and

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

According to the Gaussian processes' strategy, the value function is assumed to be zero-mean Gaussian, i.e. $V_t \sim \mathcal{N}(0, K_t)$, with a covariance matrix K_t and $N_t \sim \mathcal{N}(0, \Sigma_t)$ with $\Sigma_t = \sigma_t^2 H_t H_t^T$.

The application of the Gaussian Process to this regression problem provides with update rules for the value function according to the computation of the marginal posterior distribution of the value. However, the computational cost of the above method is huge since the size of matrices grow linearly with the number of states visited. For this reason, a sparse online algorithm has been proposed in (Engel, Mannor, and Meir 2005) by constructing a *dictionary* of input

states. Every time a state $\mathbf{x}_t$ is visited, a procedure is performed in order to decide whether or not to be entered into dictionary. This is made by examining if the feature space image of tested state $\phi(\mathbf{x}_t)$ can be approximated by the the existing dictionary with a maximum allowed square error. The sparse version of GPTD makes further some approximations for calculating efficiently the kernel matrix K_t with a reduced computational cost (for details see (Engel, Mannor, and Meir 2005)).

The proposed method

The contribution of the proposed method is twofold: At first it establishes a better value function modeling with the use of a sparse Bayesian framework. At a second level it manages to overcome some limitations of the GPTD and its sparse version, arisen from the enormous computational complexity that lead to approximated solutions. Our scheme deals with the construction of a proper state dictionary and the cumulative discount rewards between successive states entered in it. Using the stationarity of the MDP we may rewrite the discount return (Eq. 2) for a visited state $\mathbf{x}_t$ as

$$D(\mathbf{x}_t) = SR(\mathbf{x}_t) + \gamma^{k_t} D(\mathbf{x}_{t+k_t}) \,, \tag{5}$$

where $SR(\mathbf{x}_t) = \sum_{j=0}^{k_t-1} \gamma^j R(\mathbf{x}_{t+j})$ is the partial discount return of the state-reward sequence $\mathbf{x}_t, R(\mathbf{x}_t), \ldots, \mathbf{x}_{t+k_t-1}, R(\mathbf{x}_{t+k_t-1})$, and k_t is the time difference between successive samples entering in the dictionary. Using the above formulation and considering the decomposition of GPTD (Eq. 3) we can obtain the next rule:

$$SR(\mathbf{x}_t) = V(\mathbf{x}_t) - \gamma^{k_t} V(\mathbf{x}_{t+k_t}) + N(\mathbf{x}_t, \mathbf{x}_{t+k_t}) \,, \tag{6}$$

where $N(\mathbf{x}_t, \mathbf{x}_{t+k_t}) = \Delta V(\mathbf{x}_t) - \gamma^{k_t} \Delta V(\mathbf{x}_{t+k_t})$. Therefore, for each sample $\tilde{\mathbf{x}}_n$ inserted into the dictionary, we can obtain a set of n model equations for states $\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n\}$:

$$SR_n = H_n V_n + N_n \,, \tag{7}$$

where $SR_n = (SR(\tilde{\mathbf{x}}_1), \ldots, SR(\tilde{\mathbf{x}}_n))$, while matrix H takes the following form

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

In our approach, we further assume that the sequence of value function $V_n = (V(\tilde{\mathbf{x}}_1), \ldots, V(\tilde{\mathbf{x}}_n))$ can be described as a linear model, i.e.

$$V_n = \Phi_n w_n \,, \tag{8}$$

where Φ_n is a kernel design matrix of size $n \times n$, while w_n is the vector of the (unknown) model regression coefficients. Thus, Equation 7 can be written as

$$y_n = \Phi_n w_n + e_n \,, \tag{9}$$

where $y_n = (H_n^T H_n)^{-1} H_n^T SR_n$. In the above formulation the term e_n plays the role of the model noise and is assumed to be zero mean Gaussian with a spherical covariance, i.e. $e_n \sim \mathcal{N}(0, \beta_n^{-1} I)$ (β_n^{-1} is the precision parameter).

The policy estimation problem is turned into a regression coefficients estimation problem. In this direction, sparse Bayesian methodology offers an advanced solution by introducing sparse priors that has been successfully employed in the Relevance Vector Machine (RVM) model (Tipping 2001). In particular, a heavy tailed prior distribution is imposed on the regression parameters that causes to zero out most of them and maintain only a few which are considered significant after training. RVM has the advantages to increase the flexibility of the inference process, to control automatically the model complexity and to avoid overfitting. Sparse prior is defined in a hierarchical way by considering first a zero-mean Gaussian distribution over w_n, where their precision are considered as hyperparameters following a Gamma hyperprior. This two-stage hierarchical prior is actually a Student's-t distribution that is sparse. Training of the RVM under the maximum-likelihood framework, leads to closed form update equations that are applied iteratively until convergence (Tipping 2001).

Special care has been also given to the construction of the design matrix. During our study we have adopted the Gaussian kernel function $K(x_i, x_j) = \exp(|x_i - x_j|^2/(2\lambda))$, which is governed by the scalar parameter λ. As experiments have shown, this parameter plays an important role that may affect significantly the performance. We have introduced here a multi-kernel scheme by assuming a mixture of M kernel matrices $\Phi_n = \sum_{m=1}^{M} u_m \Phi_{nm}$. Each matrix has its own kernel parameter value λ_m and a weight u_m, where $\sum_{m=1}^{M} u_m = 1$. During the RVM training, there mixing weights $\{u_m\}$ are estimated by considering a convex quadratic programming problem with constraints (equalities and inequalities). Experiments have shown that the proposed scheme improves significantly the performance and makes the method independent on the scalar parameter choice.

We have experimentally studied the proposed method to known simulated environments, such as the Cart-pole and the Mountain Car, as well as to several simulated and real environments of a Pioneer/PeopleBot mobile robot which is built on the robust P3-DX base. Comparison has been made using the sparse version of the GPTD and the typical SARSA(λ) temporal difference algorithm (Sutton and Barto 1998). Our approach manages to yield significantly better asymptotic performance in much faster rate and to improve learning procedure.

References

Engel, Y.; Mannor, S.; and Meir, R. 2005. Bayesian reinforcement learning with gaussian process temporal difference methods. In *International Conference on Machine Learning*, 201–208.

Rasmussen, C., and Williams, C. 2006. *Gaussian Processes for Machine Learning*. The MIT Press.

Sutton, R., and Barto, A. 1998. *Introduction to reinforcement learning*. MIT Press Cambridge, USA.

Tipping, M. E. 2001. Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research* 1:211–244.