

---

# Bayesian Multi-Task Reinforcement Learning

---

Alessandro Lazaric

Mohammad Ghavamzadeh

INRIA Lille - Nord Europe, Team SequeL, FRANCE

ALESSANDRO.LAZARIC@INRIA.FR

MOHAMMAD.GHAVAMZADEH@INRIA.FR

## Abstract

We consider the problem of multi-task reinforcement learning where the learner is provided with a set of tasks, for which only a small number of samples can be generated for any given policy. As the number of samples may not be enough to learn an accurate evaluation of the policy, it would be necessary to identify classes of tasks with similar structure and to learn them jointly. We consider the case where the tasks share structure in their value functions, and model this by assuming that the value functions are all sampled from a common prior. We adopt the Gaussian process temporal-difference value function model and use a hierarchical Bayesian approach to model the distribution over the value functions. We study two cases, where all the value functions belong to the same class and where they belong to an undefined number of classes. For each case, we present a hierarchical Bayesian model, and derive inference algorithms for (i) joint learning of the value functions, and (ii) efficient transfer of the information gained in (i) to assist learning the value function of a newly observed task.

## 1. Introduction

Multi-task learning (MTL) is an important learning paradigm and has recently been an area of active research in machine learning (e.g., Caruana 1997; Baxter 2000; Yu et al. 2005; Xue et al. 2007; Bonilla et al. 2008). A common setup is that there are multiple related tasks for which we are interested in improving the performance over individual learning by sharing information across the tasks. This transfer of information is particularly important when we are provided with only a limited number of data to learn each task. Exploiting data from related problems provides more

training samples for the learner and can improve the performance of the resulting solution.

Most reinforcement learning (RL) algorithms (Sutton & Barto, 1998) often need a large number of samples to solve a problem and cannot directly take advantage of the information coming from other similar tasks. Nonetheless, recent work has shown that transfer and multi-task learning techniques can be employed in RL to reduce the number of samples needed to achieve nearly-optimal solutions. All approaches to multi-task RL (MTRL) assume that the tasks share similarity in some components of the problem such as dynamics, reward structure, or value function. While some methods explicitly assume that the shared components are drawn from a common generative model (Wilson et al., 2007; Mehta et al., 2008), this assumption is more implicit in others (Taylor et al., 2007; Lazaric et al., 2008). In Mehta et al. (2008), tasks share the same dynamics and reward features, and only differ in the weights of the reward function. The proposed method initializes the value function for a new task using the previously learned value functions as a prior. In Wilson et al. (2007), the distribution over the dynamics and the reward functions of the tasks is drawn from a hierarchical Bayesian model (HBM). Due to some similarity to our work, we discuss this method in more details in Section 5. Lazaric et al. (2008) implicitly assume that the tasks are drawn from a common distribution. They propose a method to selectively transfer samples from source tasks to a target task based on the likelihood of the target samples being generated by the models built for the source tasks. Finally, in Taylor et al. (2007), learning the value function of the target task is expedited using the solution learned in a source task with related, but different, state and action spaces.

In this paper, we study the MTRL scenario in which the learner is provided with a number of tasks with common state and action spaces. For any given policy, only a small number of samples can be generated in each task, which may not be enough to accurately

---

Appearing in *Proceedings of the 27<sup>th</sup> International Conference on Machine Learning*, Haifa, Israel, 2010. Copyright 2010 by the author(s)/owner(s).

evaluate the policy. In such a MTRL problem, it is necessary to identify classes of tasks with similar structure and to learn them jointly. In our work, we consider a particular class of MTRL problems in which the tasks share structure in their value functions. To allow the value functions to share a common structure, one way would be to assume that they are all sampled from a common prior. We adopt the Gaussian process temporal-difference (GPTD) value function model (Engel et al., 2005) for each task, model the distribution over the value functions using a HBM, and develop solutions to (i) joint learning of the value functions, and (ii) efficient transfer of the information acquired in (i) to facilitate learning the value function of a newly observed task. We refer to the above problems as *symmetric* and *asymmetric* multi-task learning, respectively. In Section 3, we present a HBM for the case in which all the value functions belong to the same class, and derive an EM algorithm to find MAP estimates of the value functions and the model’s hyper-parameters. However, as pointed out in Caruana (1997) and Baxter (2000), if the functions do not belong to the same class, simply learning them together can be detrimental (*negative* transfer). It is therefore important to have models that will generally benefit from related tasks and will not hurt performance when the tasks are unrelated. This is particularly important in RL as changing the policy at each step of the policy iteration algorithm can change the way tasks are clustered together. This means that even if we start with value functions belonging to the same class, after one iteration the new value functions may be clustered into several classes. In Section 4, we introduce a Dirichlet process (DP) based HBM for the case that the value functions belong to an undefined number of classes, and derive inference algorithms for both the symmetric and asymmetric scenario. In Section 5, we discuss the similarities and differences with closely related work. In Section 6, we report and analyze experimental results.

## 2. Preliminaries

The agent-environment interaction in RL is conventionally modelled as a Markov Decision Process (MDP). A MDP is a tuple  $\mathcal{M} = \langle \mathcal{X}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$  where  $\mathcal{X}$  and  $\mathcal{A}$  are the state and action spaces, respectively;  $\mathcal{R}$  is the probability distribution over rewards  $R$ ; and  $\mathcal{P}$  is the transition probability distribution. A *stationary* policy  $\pi : \mathcal{X} \times \mathcal{A} \rightarrow [0, 1]$  is a mapping from states to action selection probabilities. The MDP controlled by a policy  $\pi$  induces a Markov chain with transition probability distribution  $P^\pi(x'|x) = \int_{\mathcal{A}} P(x'|x, a)\pi(a|x)da$ . Given a policy  $\pi$ , the (possibly discounted,  $\gamma \in [0, 1]$ ) return for a state  $x$ ,  $D^\pi(x)$ , is a random process de-

finied by  $D^\pi(x) = \sum_{t=0}^{\infty} \gamma^t R(x_t) | x_0 = x$ , with  $x_{t+1} \sim P^\pi(\cdot|x_t)$ . The *value function*  $V^\pi(x)$  is the expected value of  $D^\pi(x)$  where the expectation is over all possible trajectories and rewards collected along them.

A key problem in RL is to learn the value function of a given policy, which is called *policy evaluation* (Sutton & Barto, 1998). Loosely speaking, in policy evaluation the goal is to find a “close enough” approximation  $V$  of the value function  $V^\pi$ . Unlike in *supervised learning*, the target function  $V^\pi$  is not known in advance and its values have to be inferred from the observed rewards. Therefore, it is required to define a stochastic generative model connecting the underlying hidden value function with the observed rewards. In this paper, we adopt the GPTD value function model proposed in Engel et al. (2005), in which the discounted return  $D$  is decomposed into its mean  $V$  and a random zero-mean residual  $\Delta V$ ,  $D(x) = V(x) + \Delta V(x)$ . Combining this decomposition with the Bellman equation, we get

$$R(x) = V(x) - \gamma V(x') + \epsilon(x, x'), \quad x' \sim P^\pi(\cdot|x), \quad (1)$$

where  $\epsilon(x, x') \stackrel{\text{def}}{=} \Delta V(x) - \gamma \Delta V(x')$ . Suppose we are provided with a set of samples  $\mathcal{D} = \{(x_n, x'_n, r_n)\}_{n=1}^N$ , where  $r_n$  and  $x'_n$  are the reward and the next state observed by following policy  $\pi$  in state  $x_n$ , respectively. By writing the model of Eq. (1) w.r.t. these samples, we obtain  $R = \mathbf{H}V + \mathcal{E}$ , where  $\mathbf{H} \in \mathbb{R}^{N \times 2N}$  and

$$R^\top = (r_n)_{n=1}^N; \quad \mathbf{H} = \begin{bmatrix} 1 & -\gamma & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & -\gamma & \dots & 0 & 0 \\ \vdots & & & & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & -\gamma \end{bmatrix};$$

$$\mathcal{E}^\top = (\epsilon(x_n, x'_n))_{n=1}^N; \quad V^\top = (V(x_n), V(x'_n))_{n=1}^N.$$

Note that if the samples are generated from a single trajectory then  $x'_n = x_{n+1}$  and  $\mathbf{H}$  is of the form defined by Eq. (2.7) in Engel et al. (2005). In order to specify a complete probabilistic generative model connecting values and rewards, we need to define a prior distribution for the value function  $V$  and the distribution of the noise  $\epsilon$ . Similar to Engel et al. (2005), we model the value function as a Gaussian process (GP), and the noise vector as  $\mathcal{E} \sim \mathcal{N}(\mathbf{0}, \mathbf{S})$ , where  $\mathbf{S}$  is the noise covariance matrix modelling the correlation of the noise between different states. In the following we write  $\mathbf{S} = \sigma^2 \mathbf{P}$ , where  $\sigma^2$  and  $\mathbf{P}$  are the variance and the correlation matrix of the noise, respectively. For a more extended discussion about different models of noise we refer readers to Section 8.4 in Lazaric & Ghavamzadeh (2010). The value function  $V$  may be represented either in parametric or non-parametric form. In this paper we use the parametric representation to make

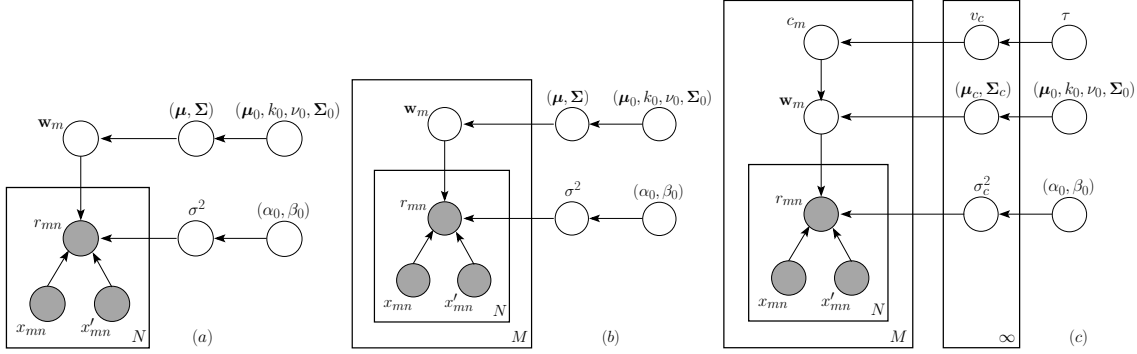


Figure 1. Graphical representations for (a) the single-task model — an extension of GPTD by defining a hyper-prior over the parameters, (b) the single-class multi-task model of Section 3, and (c) the multi-class multi-task model of Section 4.

the formulation easier to follow, but all the results can be extended to the non-parametric case following similar steps as in Section 5.2 of Yu et al. (2005). In the parametric form, the value function is represented by a finite set of  $d$  features  $\phi(\cdot) = (\phi_1(\cdot), \dots, \phi_d(\cdot))^\top$  and a weight vector  $\mathbf{w} = (w_1, \dots, w_d)^\top$  as  $V(\cdot) = \phi(\cdot)^\top \mathbf{w}$ . The randomness in  $V$  is now due to  $\mathbf{w}$  being a random vector with Gaussian prior  $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ . The model equation now becomes  $R = \mathbf{H}\boldsymbol{\Phi}^\top \mathbf{w} + \mathcal{E}$ , where  $\boldsymbol{\Phi} = [\phi(x_1), \phi(x'_1), \dots, \phi(x_N), \phi(x'_N)]$  is a  $d \times 2N$  matrix. Fig. 1(a) shows the graphical representation of this model used for single-task learning (STL) in the experiments of this paper. It is an extension of the original GPTD model by defining Normal-inverse-Wishart and inverse-Gamma hyper-priors parametrized by  $\psi_0 = (\boldsymbol{\mu}_0, k_0, \nu_0, \boldsymbol{\Sigma}_0, \alpha_0, \beta_0)$  over the model parameters  $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \sigma^2)$ . This allows us to optimize the model parameters given the data.

In the MTRL setting of this paper, the learner is provided with  $M$  tasks or MDPs with common state and action spaces  $\mathcal{M}_m = \langle \mathcal{X}, \mathcal{A}, \mathcal{R}_m, \mathcal{P}_m \rangle$ ,  $m = 1, \dots, M$ . Given a fixed policy,  $N$  samples are generated in each task, i.e.,  $\mathcal{D}_m = \{(x_{mn}, x'_{mn}, r_{mn})\}_{n=1}^N$ , which may not be enough to have an accurate evaluation of the policy. We consider the case in which the tasks share structure in their value functions. In the parametric value function model discussed above, this can be interpreted as the value functions share the same feature space and their weight vectors are sampled independently from a common prior, i.e.,  $V_m(\cdot) = \phi(\cdot)^\top \mathbf{w}_m$ ;  $\mathbf{w}_m \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ . In the next two sections, we study two different scenarios: 1) when all the tasks belong to the same class, i.e., they share the same prior, and 2) when they can be clustered into an undefined number of classes.

### 3. Single-class Multi-task Learning

In this section, we consider the case where all the tasks belong to the same class, i.e., they share the

same distribution over their value functions  $\mathbf{w}_m \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ ,  $m = 1, \dots, M$ ; and the same observation noise  $\sigma^2$ . The goal in the symmetric form of this problem is to estimate  $\{\mathbf{w}_m\}_{m=1}^M$  from the data  $\{\mathcal{D}_m\}_{m=1}^M$ , whereas in the asymmetric case we are interested in estimating the parameters  $\theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma}, \sigma^2)$  from the data in order to use them as a prior for a newly observed task (e.g., task  $\mathcal{M}_{M+1}$ ). We use a parametric HBM for this problem. HBMs allow us to model both the individuality of the tasks and the correlation between them. In HBMs, individual models with task specific parameters are usually located at the bottom, and at the layer above, tasks are connected together via a common prior placed over those parameters. Learning the common prior is a part of the training process in which data from all the tasks contribute to learning, thus making it possible to share information between the tasks usually via sufficient statistics. Then given the learned prior, individual models are learned independently. As a result, learning at each task is affected by both its own data and by data from the other tasks related through the common prior.

#### 3.1. The Model

We assume a normal-inverse-Wishart and an inverse-Gamma hyper-priors for  $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$  and  $\sigma^2$ , respectively.

$$\begin{aligned} p(\theta|\psi_0) &= p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \times p(\sigma^2) \\ &= \mathcal{N}(\boldsymbol{\mu}; \boldsymbol{\mu}_0, \boldsymbol{\Sigma}/k_0) \mathcal{IW}(\boldsymbol{\Sigma}; \nu_0, \boldsymbol{\Sigma}_0) \times \mathcal{IG}(\sigma^2; \alpha_0, \beta_0). \end{aligned} \quad (2)$$

These distributions are the conjugate priors for multivariate Gaussian distributions  $p(\mathbf{w}_m|\boldsymbol{\mu}, \boldsymbol{\Sigma})$  and  $p(R_m|\mathbf{w}_m, \sigma^2) = \mathcal{N}(\mathbf{H}\boldsymbol{\Phi}_m^\top \mathbf{w}_m, \sigma^2 \mathbf{P})$ , respectively. This leads to the following generative model for the data,  $\{\mathcal{D}_m\}$ . Fig. 1(b) shows the graphical representation of this model. The details of the model can be found in Lazaric & Ghavamzadeh (2010).

**Single-Class Model:** Given the hyper-parameters  $\psi_0 = (\boldsymbol{\mu}_0, k_0, \nu_0, \boldsymbol{\Sigma}_0, \alpha_0, \beta_0)$ ,

1. The parameters  $\theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma}, \sigma^2)$  are sampled once from the hyper-prior as in Eq. (2),
2. For each task  $\mathcal{M}_m$  (value function  $V_m$ ), the weight vector is sampled as  $\mathbf{w}_m \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ ,
3. Given  $\{(x_{mn}, x'_{mn})\}_{n=1}^N$ ,  $R_m = \mathbf{H}\boldsymbol{\Phi}_m^\top \mathbf{w}_m + \mathcal{E}$ , where  $\mathcal{E} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{P})$ ,  $m = 1, \dots, M$ .

### 3.2. Inference

This model can be learned by optimizing the penalized likelihood  $p(\{R_m\}|\{(x_{mn}, x'_{mn})\}, \theta)p(\theta)$  w.r.t. the parameters  $\theta = (\boldsymbol{\mu}, \boldsymbol{\Sigma}, \sigma^2)$  using an EM algorithm. In the rest of the paper, we refer to this algorithm as SCMTL, for single-class multi-task learning.

**E-step:** Since the posterior distribution of the latent variables  $p(\{\mathbf{w}_m\}|\{\mathcal{D}_m\}, \theta)$  is a product of  $M$  Gaussian posterior distributions  $p(\mathbf{w}_m|\mathcal{D}_m, \theta) = \mathcal{N}(\boldsymbol{\mu}'_{0m}, \boldsymbol{\Sigma}'_{0m})$ , for each task  $m$ , we compute the mean and covariance as

$$\begin{aligned} \boldsymbol{\mu}'_{0m} &= \boldsymbol{\Sigma}'_{0m} \left[ \frac{1}{\sigma^2} \boldsymbol{\Phi}_m \mathbf{H}^\top \mathbf{P}^{-1} R_m + \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} \right], \\ \boldsymbol{\Sigma}'_{0m} &= \left[ \frac{1}{\sigma^2} \boldsymbol{\Phi}_m \mathbf{H}^\top \mathbf{P}^{-1} \mathbf{H} \boldsymbol{\Phi}_m^\top + \boldsymbol{\Sigma}^{-1} \right]^{-1}. \end{aligned}$$

**M-step:** We optimize  $\theta$  to maximize the penalized expected log-likelihood of complete data  $\log p(\{\mathcal{D}_m\}, \{\mathbf{w}_m\}|\theta)$  over the posterior distribution estimated in the E-step and obtain the new parameters

$$\begin{aligned} \boldsymbol{\mu}_{\text{new}} &= \frac{1}{M + k_0} \left( k_0 \boldsymbol{\mu}_0 + \sum_{m=1}^M \boldsymbol{\mu}'_{0m} \right), \\ \boldsymbol{\Sigma}_{\text{new}} &= \frac{1}{M + \nu_0 + d + 2} \left\{ k_0 (\boldsymbol{\mu} - \boldsymbol{\mu}_0) (\boldsymbol{\mu} - \boldsymbol{\mu}_0)^\top + \boldsymbol{\Sigma}_0 \right. \\ &\quad \left. + \sum_{m=1}^M [(\boldsymbol{\mu}'_{0m} - \boldsymbol{\mu}) (\boldsymbol{\mu}'_{0m} - \boldsymbol{\mu})^\top + \boldsymbol{\Sigma}'_{0m}] \right\}, \\ \sigma_{\text{new}}^2 &= \frac{1}{MN + 2(1 + \alpha_0)} \left\{ 2\beta_0 + \sum_{m=1}^M [\text{tr}(\mathbf{P}^{-1} \mathbf{H} \boldsymbol{\Phi}_m^\top \boldsymbol{\Sigma}'_{0m} \boldsymbol{\Phi}_m \mathbf{H}^\top) \right. \\ &\quad \left. + (R_m - \mathbf{H} \boldsymbol{\Phi}_m^\top \boldsymbol{\mu}'_{0m}) \mathbf{P}^{-1} (R_m - \mathbf{H} \boldsymbol{\Phi}_m^\top \boldsymbol{\mu}'_{0m})^\top \right\}. \end{aligned}$$

## 4. Multi-class Multi-task Learning

In this section, we consider the case where the tasks belong to an undefined number of classes. Tasks in the same class  $\{\mathcal{M}_m | c_m = c\}$  share the same distribution over their value functions  $\mathbf{w}_m \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ , and the same observation noise  $\sigma_c^2$ . We use a nonparametric HBM for this problem. In the HBM proposed in this paper, the common prior is drawn from a Dirichlet process (DP). DP is powerful enough to model the parameters of individual classes, to fit them well without any assumption about the functional form of the prior, and to automatically learn the number of underlying classes.

### 4.1. The Model

We place a  $\text{DP}(\tau, G_0)$  prior over the class assignment and the class parameters. The concentration parameter  $\tau$  and the base distribution  $G_0$  can be considered as

priors over the number of classes and the class parameters  $\theta_c = (\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c, \sigma_c^2)$ , respectively.  $G_0$  is the product of a  $d$ -dimensional normal-inverse-Wishart and a 1-dimensional inverse-Gamma distributions, with parameters  $\psi_0 = (\boldsymbol{\mu}_0, k_0, \nu_0, \boldsymbol{\Sigma}_0, \alpha_0, \beta_0)$ , (see Eq. 2). We employ the stick-breaking representation of the DP prior (Ishwaran & James, 2001), and define a task-to-class assignment variable  $(c_{m1}, \dots, c_{m\infty})$  for each task  $m$ , whose elements are all zero except that the  $c$ th element is equal to one if task  $m$  belongs to class  $c$ . Given the above, the data  $\{\mathcal{D}_m\}$  can be seen as drawn from the following generative model, whose graphical representation is shown in Fig. 1(c).

**Multi-Class Model:** Hyper-parameters  $(\tau, \psi_0)$ ,

1. Stick-breaking view: Draw  $v_c$  from the Beta distribution  $\mathcal{B}e(1, \tau)$ ,  $\pi_c = v_c \prod_{i=1}^{c-1} (1 - v_i)$ , and independently draw  $\theta_c \sim G_0$ ,  $c = 1, \dots, \infty$ ,
2. Task-to-class assignment: Draw the indicator  $(c_{m1}, \dots, c_{m\infty})$  from a multinomial distribution  $\mathcal{M}_\infty(1; \pi_1, \dots, \pi_\infty)$ ,  $m = 1, \dots, M$ ,
3. The weight vector is sampled as  $\mathbf{w}_m \sim \mathcal{N}(\boldsymbol{\mu}_{c_m}, \boldsymbol{\Sigma}_{c_m})$ ,  $m = 1, \dots, M$ ,
4. Given  $\{(x_{mn}, x'_{mn})\}_{n=1}^N$ ,  $R_m = \mathbf{H}\boldsymbol{\Phi}_m^\top \mathbf{w}_m + \mathcal{E}$ , where  $\mathcal{E} \sim \mathcal{N}(\mathbf{0}, \sigma_{c_m}^2 \mathbf{P})$ ,  $m = 1, \dots, M$ .

### 4.2. Inference

We are interested in the posterior distribution of the latent variables  $\mathcal{Z} = \{\{\mathbf{w}_m\}, \{c_m\}, \{\theta_c\}\}$  given the observed data and the hyper-parameters  $\tau$  and  $\psi_0$ , i.e.,  $p(\mathcal{Z}|\{\mathcal{D}_m\}, \tau, \psi_0) \propto p(\{\mathcal{D}_m\}|\mathcal{Z}, \tau, \psi_0)p(\mathcal{Z}|\tau, \psi_0)$ . In the following we outline the main steps of the algorithm used to solve this inference problem, which we refer to as MCMTL, for multi-class multi-task learning (see Fig. 2). MCMTL combines the SCMTL algorithm of Sec. 3.2 for class parameters estimation, with a Gibbs sampling algorithm for learning the class assignments (Neal, 2000). The main advantage of such combination is that at each iteration, given the current estimate of the weights, we take advantage of the conjugate priors to derive an efficient Gibbs sampling procedure.

More formally, given an arbitrary initial class assignment  $\{c_m\}$ , a distinct EM algorithm is run on each class  $c = 1, \dots, C$  (with  $C$  the current estimate of the number of classes) and returns  $M$  distributions  $\mathcal{N}(\boldsymbol{\mu}'_{0m}, \boldsymbol{\Sigma}'_{0m})$ . Given the weights estimated at the previous step,  $\hat{\mathbf{w}}_m = \boldsymbol{\mu}'_{0m}$ , the Gibbs sampling solves the DP inference by drawing samples from the posterior distribution  $p(\{c_m\}|\{R_m\}, \{\hat{\mathbf{w}}_m\}, \tau, \psi_0)$ . In particular, the state of the Markov chain simulated in the Gibbs sampling is the class assignment  $\{c_m\}$ , i.e., the vector of the classes each task belongs to. At each iteration, each component  $c_m$  of the state is updated by sampling from the following distribution

```

MCMTL( $\{R_m\}, \tau, \psi_0$ )
Initialize  $\{c_m\}$ 
repeat
  for  $c = 1, \dots, C$  do
    Initialize  $\theta_c$ 
    repeat
      for  $m : c_m = c$  do
         $p(\mathbf{w}_m | R_m, \theta_c) = \mathcal{N}(\boldsymbol{\mu}'_{0m}, \boldsymbol{\Sigma}'_{0m})$  (E-step)
      end for
      Optimize  $\theta_c$  (M-step)
    until convergence
  end for
  Set  $\widehat{\mathbf{w}}_m = \boldsymbol{\mu}'_{0m}$ ,  $m = 1, \dots, M$ 
   $p(\{c_m\} | \{\widehat{\mathbf{w}}_m\}, \{R_m\}, \tau, \psi_0)$ 
until convergence
return  $\{\widehat{\mathbf{w}}_m\}$  and  $\{c_m\}$ 
    
```

Figure 2. The inference algorithm for the multi-class multi-task learning (MCMTL) scenario.

$$\begin{aligned}
 \text{If } c = c_{m'}, m' \neq m: & p(c_m = c | \{c_{m'}\}, R_m, \widehat{\mathbf{w}}_m, \tau, \psi_0) \\
 &= b \frac{M_{-m,c}}{M-1+\tau} \int p(R_m, \widehat{\mathbf{w}}_m | \theta_c) p(\theta_c | \{c_{m'}\}, \psi_0) d\theta_c, \\
 \text{else: } & p(c_m \neq c_{m'}, m' \neq m | \{c_{m'}\}, R_m, \widehat{\mathbf{w}}_m, \tau, \psi_0) \\
 &= b \frac{\tau}{M-1+\tau} \int p(R_m, \widehat{\mathbf{w}}_m | \theta) p(\theta | \psi_0) d\theta, \quad (3)
 \end{aligned}$$

where  $M_{-m,c}$  is the number of tasks in class  $c$  except task  $m$ , and  $b$  is a normalizing constant. While the first term in Eq. (3) is the probability of task  $m$  to belong to an existing class  $c$ , the second term returns the probability of assigning task  $m$  to a new class. Thanks to the conjugate base distribution  $G_0$ , the integrals in Eq. (3) can be solved analytically. In fact

$$\begin{aligned}
 p(R_m, \widehat{\mathbf{w}}_m | \theta) p(\theta | \psi_0) &= p(R_m | \widehat{\mathbf{w}}_m, \sigma^2) p(\widehat{\mathbf{w}}_m | \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\
 &\quad \times p(\boldsymbol{\mu}, \boldsymbol{\Sigma} | \boldsymbol{\mu}_0, k_0, \nu_0, \boldsymbol{\Sigma}_0) p(\sigma^2 | \alpha_0, \beta_0) \\
 &\propto \mathcal{N}(\boldsymbol{\mu}; \boldsymbol{\mu}'_0, \boldsymbol{\Sigma} / k'_0) \mathcal{IW}(\boldsymbol{\Sigma}; \nu'_0, \boldsymbol{\Sigma}'_0) \times \mathcal{IG}(\sigma^2; \alpha'_0, \beta'_0), \quad (4)
 \end{aligned}$$

where  $\psi'_0 = (\boldsymbol{\mu}'_0, k'_0, \nu'_0, \boldsymbol{\Sigma}'_0, \alpha'_0, \beta'_0)$  are the posterior parameters of  $G_0$  given the weight  $\widehat{\mathbf{w}}_m$  and the rewards  $R_m$  (see Lazaric & Ghavamzadeh 2010 for their definition). Using the posterior hyper-parameters, the second integral in Eq. (3) can be written as

$$\begin{aligned}
 & \int p(R_m, \widehat{\mathbf{w}}_m | \theta) p(\theta | \psi_0) d\theta \\
 &= \left( \frac{k_0}{\pi k'_0} \right)^{\frac{d}{2}} \frac{|\boldsymbol{\Sigma}_0|^{\nu_0/2}}{|\boldsymbol{\Sigma}'_0|^{\nu'_0/2}} \frac{\Gamma\left(\frac{\nu'_0}{2}\right)}{\Gamma\left(\frac{\nu_0-d}{2}\right)} \times (2\pi|\mathbf{P}|)^{-\frac{N}{2}} \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)} \frac{\Gamma(\alpha'_0)}{\beta_0^{\alpha'_0}}. \quad (5)
 \end{aligned}$$

In the first integral of Eq. (3), the density function  $p(\theta_c | \{c_{m'}\}, \psi_0)$  is the posterior probability over the class parameters  $\theta_c$  given the data from all the tasks belonging to  $c_m$  according to the current class assignment  $\{c_{m'}\}$ . Similar to Eq. (4), we compute the posterior hyper-parameters  $\psi_{0c}$  of the

normal-inverse-Wishart and inverse-Gamma distributions given  $\{\widehat{\mathbf{w}}_{m'}\}$  and  $\{R_{m'}\}$ , with  $m' \neq m$  and  $c_{m'} = c_m$ . Finally, the integral can be analytically calculated as in Eq. (5), where the hyper-parameters  $\psi_0$  and the posterior hyper-parameters  $\psi'_0$  are replaced by  $\psi_{0c}$  and  $\psi'_{0c}$ , respectively.

### 4.3. Symmetric vs. Asymmetric Learning

The MCMTL algorithm returns both the distribution over the weights for each task and the learned hierarchical model (task-class assignments). While the former can be used to evaluate the learning performance in the symmetric case, the latter provides a prior for learning a new task in the asymmetric scenario.

**Symmetric Learning.** According to the generative model in Section 4.1, the task weights are distributed according to the normal distribution  $\mathcal{N}(\boldsymbol{\mu}'_{0m}, \boldsymbol{\Sigma}'_{0m})$ , where  $\boldsymbol{\mu}'_{0m}$  and  $\boldsymbol{\Sigma}'_{0m}$  are the posterior mean and covariance of the weight vector  $\mathbf{w}_m$  returned by the MCMTL algorithm. Since  $V_m(x) = \phi(x)^\top \mathbf{w}_m$ , the value of  $V_m$  at a test state  $x_*$  is distributed as

$$p(V_m(x_*) | x_*, \boldsymbol{\mu}'_{0m}, \boldsymbol{\Sigma}'_{0m}) = \mathcal{N}(\phi(x_*)^\top \boldsymbol{\mu}'_{0m}, \phi(x_*)^\top \boldsymbol{\Sigma}'_{0m}^{-1} \phi(x_*)).$$

If MCMTL successfully clusters the task, we expect the value function prediction to be more accurate than learning each task independently.

**Asymmetric Learning.** In the asymmetric setting the class of the new task is not known in advance. The inference problem is formalized as  $p(\mathbf{w}_{M+1} | R_{M+1}, \psi_0, \{c_m\}_{m=1}^M)$ , where  $\mathbf{w}_{M+1}$  and  $R_{M+1}$  are the weight vector and rewards of the new task  $\mathcal{M}_{M+1}$ , respectively. Similar to Section 4.2, this inference problem cannot be solved in closed form, thus, we must apply the MCMTL algorithm to the new task. The main difference with the symmetric learning is that the class assignments  $\{c_m\}$  and weights  $\{\widehat{\mathbf{w}}_m\}$  for all the previous tasks are kept fixed, and are used as a prior over the new task learned by the MCMTL algorithm. As a result, the Gibbs sampling reduces to a one-step sampling process assigning the new task either to one of the existing classes or to a new class. If  $\mathcal{M}_{M+1}$  belongs to a new class, then the inference problem becomes  $p(\mathbf{w}_{M+1} | R_{M+1}, \psi_0)$ , that is exactly the same as in STL. On the other hand, if  $\mathcal{M}_{M+1}$  belongs to class  $c$ , the rewards and weights  $\{R_{m'}\}, \{\mathbf{w}_{m'}\}$  of the tasks in class  $c$  can be used to compute the posterior hyper-parameters  $\psi'_{0c}$  as in Eq. (4), and to solve the inference problem  $p(\mathbf{w}_{M+1} | R_{M+1}, \psi'_{0c})$ .

## 5. Related Work

In RL, the approach of this paper is mainly related to Wilson et al. (2007). Although we both use a DP-based HBM to model the distribution over the common structure of the tasks, in Wilson et al. (2007) the

tasks share structure in their dynamics and reward function, while we consider the case that the similarity is in the value function. There are scenarios in which significantly different MDPs and policies may lead to very similar value functions. In such scenarios, the method proposed in this paper would still be able to leverage on the commonality of the value functions, thus performing better than single-task learning. Moreover in [Wilson et al. \(2007\)](#), the setting is incremental, i.e., the tasks are observed as a sequence, and there is no restriction on the number of samples generated by each task. The focus is not on joint learning with finite number of samples, it is on using the information gained from the previous tasks to facilitate learning in a new one. This setting is similar to the asymmetric learning considered in our work.

In supervised learning, our work is related to [Yu et al. \(2005\)](#) and [Xue et al. \(2007\)](#). In [Yu et al. \(2005\)](#), the authors present a single-class HBM for learning multiple related functions using GPs. Our single-class model of Section 3 is an adaptation of this work for RL using GPTD. Besides, our multi-class model of Section 4 extends this method to the case with an undefined number of classes. In [Xue et al. \(2007\)](#), a DP-based HBM is used to learn the extent of similarity between classification problems. The problem considered in our paper is regression, the multi-class model of Section 4 is more complex than the one used in [Xue et al. \(2007\)](#), and the inference algorithms of Section 4 are based on Gibbs sampling, where a variational method is used for inference in [Xue et al. \(2007\)](#).

## 6. Experiments

In this section, we report empirical results applying the Bayesian multi-task learning (BMTL) algorithms presented in this paper to a regression problem and a benchmark RL problem, inverted pendulum. We compare the performance of single-task learning (STL) with single-class multi-task learning (SCMTL), i.e., all tasks are assumed to belong to the same class, and multi-class multi-task learning (MCMTL), i.e., tasks belong to a number of classes not known in advance. By STL, we refer to running the EM algorithm of Section 3.2 for each task separately. The reason to use the regression problem in our experiments is that it allows us to evaluate our BMTL algorithms when the tasks are generated exactly according to the generative models of Sections 3 and 4.

### 6.1. A Regression Problem

In this problem, tasks are functions in the linear space spanned by a feature space  $\phi(x) = (1, x, x^2, x^3, x^4, x^5)^\top$  on the domain  $\mathcal{X} = [-1, 1]$ . The weights for the tasks are drawn from four different classes, i.e., four 6-dim

multivariate Gaussian distributions, with the parameters shown in Fig. 3(a). The noise covariance matrix  $\mathbf{S} = \text{diag}(\sigma^2)$  for all the algorithms. We evaluate the performance of each BMTL algorithm by computing its relative mean squared error (MSE) improvement over STL:  $(MSE_{\text{STL}} - MSE_{\text{BMTL}}) / MSE_{\text{STL}}$ . The MSEs are computed over  $N' = 1000$  test samples. All the reported results are averaged over 200 independent runs.

In the first experiment, we draw all the tasks from class  $c_2$ . Fig. 3(c) shows the performance of SCMTL for different number of tasks ( $M$ ) and samples per task ( $N$ ). SCMTL achieves an improvement over STL that varies from  $29.86\% \pm 0.9$  for  $N = 100$  and 20 tasks to  $67.64\% \pm 0.8$  for 100 tasks with only 20 samples each. The results indicate that SCMTL successfully takes advantage of the samples coming from all the tasks to build a more accurate prior than the one obtained by considering each task separately as in STL. However, the advantage of SCMTL over STL declines as  $N$  is increased. In fact, as STL converges, SCMTL cannot make further improvement. We repeated the same experiment for the other classes. The minimum and maximum performance of SCMTL for all the classes (all obtained for  $N = 100, M = 20$  and  $N = 20, M = 100$ , respectively) are summarized in Fig. 3(b).

In the second experiment, we draw the tasks randomly from the four classes. We first apply SCMTL to this problem. Fig. 4(a) shows the SCMTL’s performance. As it can be seen, the results are worse than those in the first experiment (Fig. 3(c)), varying from  $30.15\% \pm 4.8$  to  $54.05\% \pm 1.2$ . By clustering all the tasks together, SCMTL takes advantage of all the available samples, thus, performs better than STL. However, when the tasks are drawn from significantly different distributions, it learns a very general prior which does not allow a significant improvement over STL. We then apply MCMTL to this problem. MCMTL’s performance (Fig. 4(b)) varies from  $45.64\% \pm 5.6$  to  $77.65\% \pm 0.8$  and is significantly better than SCMTL’s (Fig. 4(a)). In order to evaluate how well MCMTL classifies the tasks, we also compare its performance to a version of MCMTL in which each task is assigned to the right class in advance. The difference between the two algorithms is statistically significant only for  $N = 20$  (with the maximum of 5.08% for  $M = 20$ ), where the noise on the samples makes it more difficult to discriminate between the distributions generating the tasks, and thus, to classify them correctly.

Finally, we compare SCMTL and MCMTL in the asymmetric setting. At the end of each run, we draw 100 new test tasks at random from the same four classes used to generate the training tasks. We run the

	$\mu_c$	$\sigma_c^2$
$c_1$	(50 10 5 0 0 0)	1.0
$c_2$	(0 0 0 0.5 0.1 0.05)	3.0
$c_3$	(0 -20 0 -0.5 0 -0.05)	5.0
$c_4$	(-50 0 -5 0 -0.1 0)	7.5

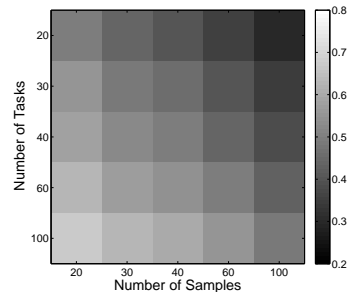
  

	$\Sigma_c$
$c_1$	diag(20.0 10.0 5.0 0.0 0.0 0.0)
$c_2$	diag(0.0 0.0 0.0 0.5 0.1 0.05)
$c_3$	diag(0.0 5.0 0.0 0.5 0.0 0.01)
$c_4$	diag(20.0 0.0 5.0 0.0 0.1 0.0)

(a)

	Min	Max
$c_1$	37.35% $\pm$ 1.2	64.78% $\pm$ 0.9
$c_2$	29.86% $\pm$ 0.9	67.64% $\pm$ 0.8
$c_3$	66.69% $\pm$ 0.8	91.81% $\pm$ 0.6
$c_4$	56.47% $\pm$ 1.2	75.78% $\pm$ 0.3

(b)



(c)

Figure 3. (a) class parameters, (b) minimum and maximum improvement of SCMTL over STL in each class, (c) relative MSE improvement of SCMTL over STL when all the tasks are drawn from class  $c_2$ .

asymmetric algorithm described in Section 4.3 on each of the test tasks separately. Fig. 4(c) shows the performance of SCMTL and MCMTL for different number of training tasks and  $N$  fixed to 20. The results indicate that MCMTL performs relatively better than SCMTL as the number of training tasks increases.

## 6.2. Inverted Pendulum

The experiments of Section 6.1 indicate that when the tasks are generated exactly according to the generative models of Sections 3 and 4, the BMTL methods can significantly improve the performance of a regression problem w.r.t. STL. As discussed in Section 2, the policy evaluation step of policy iteration can be casted as a regression problem, thus, similar improvement can be expected. In this section, we compare our BMTL algorithms with STL in the problem of learning a control policy for balancing an inverted pendulum. Dynamics, reward function, and basis functions are the same as in Lagoudakis & Parr (2003). Each task is generated by drawing the parameters of the dynamics (pole mass, pole length, cart mass, and noise on the actions) from Gaussian distributions with means and variances summarized in Fig. 5(a). The distribution over the two classes is uniform. It is worth noting that, unlike the regression experiments, here we have no guarantee that the weights of the value functions will follow the generative models assumed by the BMTL methods. We use policy iteration with 10 iterations and the noise correlation matrix  $\mathbf{P}^{-1} = \Phi_m^\top (\Phi_m \Phi_m^\top)^{-1} \Phi_m$  for all the algorithms (see Lazaric & Ghavamzadeh 2010 for details). In STL, each policy evaluation step is solved using the EM algorithm of Section 3.2 for each task separately, where in BMTL, it is solved by running SCMTL or MCMTL over all the tasks. All the results are averaged over 150 independent runs.

Fig. 5(b) shows the performance of the policy learned by STL, SCMTL, and MCMTL for  $M = 10$  tasks and different (up to 500) number of samples per task. Note that STL converges at about 1200 samples per task

with an average performance of  $2473 \pm 61.9$  balanced steps. As it can be seen, both BMTL methods outperform STL, and MCMTL achieves a better performance than SCMTL as the number of samples is increased. Since SCMTL forces all the tasks to belong to a common distribution, it learns a very general prior, and thus, it cannot approximate the value functions as accurate as MCMTL, which is able to correctly discriminate between class  $c_1$  and  $c_2$ . In order to show how the performance changes with different number of tasks, we compute the area ratio (Taylor et al., 2007) on the first 500 samples as  $\rho_{\text{BMTL}} = \frac{A_{\text{BMTL}} - A_{\text{STL}}}{A_{\text{STL}}}$ , where  $A_{\text{BMTL}}$  ( $A_{\text{STL}}$ ) is the area under the learning curve of STL (BMTL) from 100 to 500 samples. Fig. 5(c) shows that MCMTL has significantly better area ratio than SCMTL for all values of  $M$  except very small ones.

## 7. Conclusions

We presented hierarchical Bayesian models (HBMs) and inference algorithms for multi-task reinforcement learning (RL) where the tasks share structure in their value functions. To the best of our knowledge, this is the first work that models value function similarity using HBMs. In particular, we considered two cases, where all the value functions belong to the same class, and where they belong to an undefined number of classes. In these cases, we modelled the distribution over the value functions using a parametric HBM and a Dirichlet process (DP) based non-parametric HBM, respectively. For each case, we derived inference algorithms for learning the value functions jointly and to transfer the knowledge acquired in the joint learning to improve the performance of learning the value function of a new task. We first applied our proposed Bayesian multi-task learning (BMTL) algorithms to a regression problem, in which the tasks are drawn from the generative models used by the BMTL methods. The results indicate that BMTL algorithms achieve significant improvement over single-task learning (STL) in both symmetric and asymmetric settings. We then ap-

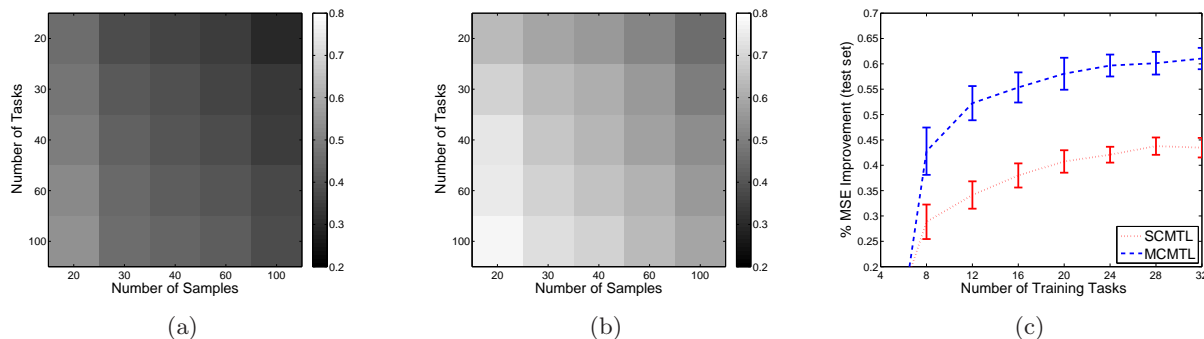


Figure 4. Results for the case that the tasks are drawn randomly from the four classes: (a) relative MSE improvement of SCMTL over STL, (b) relative MSE improvement of MCMTL over STL, (c) asymmetric performance of MCMTL and SCMTL for  $N = 20$ .

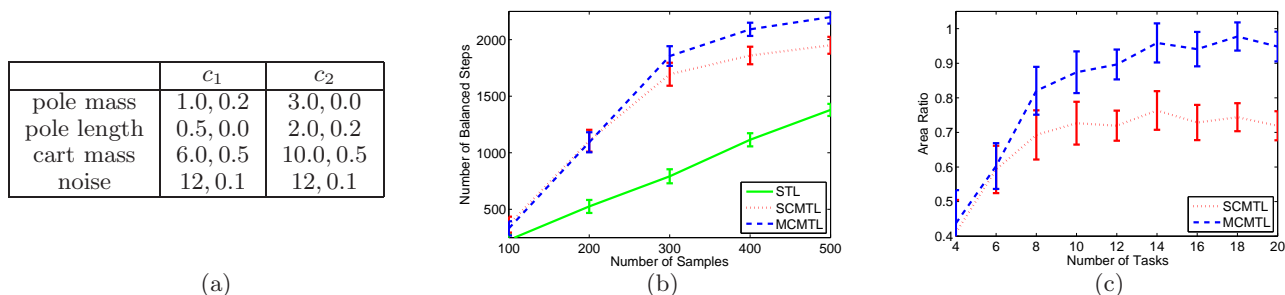


Figure 5. Results for the inverted pendulum problem: (a) distributions of the parameters of the dynamics, (b) comparing the performance of STL, SCMTL, and MCMTL in terms of the number of balanced steps for  $M = 10$ , (c) comparing the performance of SCMTL and MCMTL in terms of the area ratio on the first 500 samples.

plied our BMTL algorithms to a benchmark RL problem, inverted pendulum. Although the tasks are no longer generated according to the models used by the BMTL algorithms, they still outperform STL.

**Acknowledgments** Experiments presented in this paper were carried out using the Grid5000 experimental testbed (<https://www.grid5000.fr>). This work was supported by French National Research Agency (ANR) (project EXPLO-RA  $n^\circ$  ANR-08-COSI-004).

## References

- Baxter, J. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.
- Bonilla, E., Chai, K., and Williams, C. Multi-task Gaussian process prediction. In *Proceedings of NIPS 20*, pp. 153–160, 2008.
- Caruana, R. Multitask learning. *Machine Learning*, 28(1): 41–75, 1997.
- Engel, Y., Mannor, S., and Meir, R. Reinforcement learning with Gaussian processes. In *Proceedings of ICML 22*, pp. 201–208, 2005.
- Ishwaran, H. and James, L. Gibbs sampling methods for stick-breaking priors. *J. Amer. Statistical Assoc.*, 96: 161–173, 2001.
- Lagoudakis, M. and Parr, R. Least-squares policy iteration. *JMLR*, 4:1107–1149, 2003.
- Lazaric, A. and Ghavamzadeh, M. Bayesian multi-task reinforcement learning. Technical Report inria-00475214, INRIA, 2010.
- Lazaric, A., Restelli, M., and Bonarini, A. Transfer of samples in batch reinforcement learning. In *Proceedings of ICML 25*, pp. 544–551, 2008.
- Mehta, N., Natarajan, S., Tadepalli, P., and Fern, A. Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 73(3):289–312, 2008.
- Neal, R. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.
- Sutton, R. and Barto, A. *An Introduction to Reinforcement Learning*. MIT Press, 1998.
- Taylor, M., Stone, P., and Liu, Y. Transfer learning via inter-task mappings for temporal difference learning. *JMLR*, 8:2125–2167, 2007.
- Wilson, A., Fern, A., Ray, S., and Tadepalli, P. Multi-task reinforcement learning: A hierarchical Bayesian approach. In *Proceedings of ICML 24*, pp. 1015–1022, 2007.
- Xue, Y., Liao, X., Carin, L., and Krishnapuram, B. Multi-task learning for classification with dirichlet process priors. *JMLR*, 8:35–63, 2007.
- Yu, K., Tresp, V., and Schwaighofer, A. Learning Gaussian processes from multiple tasks. In *Proceedings of ICML 22*, pp. 1012–1019, 2005.