Transfer of Value Functions via Variational Methods

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider the problem of transferring value functions in reinforcement learning. We propose an approach that uses the given source tasks to learn a prior distribution over optimal value functions and provide an efficient variational approximation of the corresponding posterior in a new target task. We show our approach to be general, in the sense that it can be combined with complex parametric function approximators and distribution models, while providing two practical algorithms based on Gaussians and Gaussian mixtures. We theoretically analyze them by deriving a finite-sample analysis and provide a comprehensive empirical evaluation in four different domains.

1 Introduction

Recent advancements have allowed reinforcement learning (RL) [35] to achieve impressive results in a wide variety of complex tasks, ranging from Atari [26] through the game of Go [34] to the control of sophisticated robotics systems [16, 23, 22]. The main limitation is that these RL algorithms still require an enormous amount of experience samples before successfully learning such complicated tasks. One of the most promising solutions to alleviate this problem is transfer learning, which focuses on reusing past knowledge available to the agent in order to reduce the sample-complexity for learning new tasks. In the typical settings of transfer in RL [37], the agent is assumed to have already solved a set of *source tasks* generated from some unknown distribution. Then, given a *target task* (which is drawn from the same distribution, or a slightly different one), the agent can rely on knowledge from the source tasks to speed up the learning process. This reuse of knowledge constitutes a significant advantage over plain RL, where the agent learns each new task from scratch independently of any previous learning experience. Several algorithms have been proposed in the literature to transfer different elements involved in the learning process: experience samples [21, 36], policies/options [11, 18], rewards [17], features [5], parameters [10, 15], and so on. We refer the reader to [37, 19] for a thorough survey on transfer in RL.

Under the assumption that tasks follow a specific distribution, an intuitive choice for designing a transfer algorithm is to attempt at characterizing the uncertainty over the target task. Then, an ideal algorithm would leverage prior knowledge from the source tasks to interact with the target task to reduce the uncertainty as quickly as possible. This simple intuition makes Bayesian methods appealing approaches for transfer in RL, and many previous works have been proposed in this direction. In [39], the authors assume tasks share similarities in their dynamics and rewards and propose a hierarchical Bayesian model for the distribution of these two elements. Similarly, in [20], the authors assume that tasks are similar in their value functions and design a different hierarchical Bayesian model for transferring such information. More recently, [10], and its extension [15], consider tasks whose dynamics are governed by some hidden parameters, and propose efficient Bayesian models for quickly learning such parameters in new tasks. However, most of these algorithms require specific, and sometimes restrictive, assumptions (e.g., on the distributions involved or the function approximators adopted), which might limit their practical applicability. The importance of having transfer algorithms that alleviate the need for strong assumptions and that easily adapt to different contexts motivates us to take a more general approach.

Similarly to [20], we assume tasks to share similarities in their value functions and use the given source tasks to learn a distribution over such functions. Then, we use this distribution as a prior for learning the target task and we propose a variational approximation of the corresponding posterior that is computationally efficient. Leveraging on recent ideas from randomized value functions [27, 3], we design a Thompson Sampling-based algorithm which efficiently explores the target task by repeatedly sampling from the posterior and acting greedily w.r.t. (with respect to) the sampled value function. We show that our approach is very general, in the sense that it can work with any parametric function approximator and with any prior/posterior distribution models (in this paper we focus on the Gaussian and Gaussian mixture models). In addition to the algorithmic contribution, we also give a theoretical contribution by providing a finite-sample analysis of our approach and an experimental contribution showing its empirical performance on four domains with increasing level of difficulty.

2 Preliminaries

We consider a distribution $\mathcal{D}$ over tasks, where each task $\mathcal{M}_\tau$ is modeled as a discounted Markov Decision Process (MDP). We define an MDP as a tuple $\mathcal{M}_\tau = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}_\tau, \mathcal{R}_\tau, p_0, \gamma \rangle$, where $\mathcal{S}$ is the state-space, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}_\tau(\cdot|s,a)$ is the distribution of the next state s' given that action a is taken in state s, $\mathcal{R}_\tau : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, p_0 is the initial-state distribution, and $\gamma \in [0,1)$ is the discount factor. We assume the reward function to be uniformly bounded by a constant $R_{max} > 0$. A deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ is a mapping from states to actions. At the beginning of each episode of interaction, the initial state s_0 is drawn from p_0. Then, the agent takes the action $a_0 = \pi(s_0)$, receives a reward $\mathcal{R}_\tau(s_0, a_0)$, transitions to the next state $s_1 \sim \mathcal{P}_\tau(\cdot|s_0, a_0)$, and the process is repeated. The goal is to find the policy maximizing the long-term return over a possibly infinite horizon: $\max_\pi J(\pi) \triangleq \mathbb{E}_{\mathcal{M}_\tau,\pi}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_\tau(s_t, a_t)]$. To this end, we define the optimal value function of task $\mathcal{M}_\tau$, $Q^*_\tau(s,a)$, as the expected return obtained by taking action a in state s and following an optimal policy thereafter. Then, an optimal policy π^*_τ is a policy that is greedy with respect to the optimal value function, i.e., $\pi^*_\tau(s) = \operatorname{argmax}_a Q^*_\tau(s,a)$ for all states s. It can be shown (e.g., [28]) that Q^*_τ is the unique fixed-point of the optimal Bellman operator T_τ defined by $T_\tau Q(s,a) = \mathcal{R}_\tau(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{P}_\tau}[\max_{a'} Q(s',a')]$ for any value function Q. From now on, we adopt the term Q-function to denote any plausible value function, i.e., any function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ uniformly bounded by $\frac{R_{max}}{1-\gamma}$. In the following, to avoid cluttering the notation, we will drop the subscript τ whenever there is no ambiguity.

We consider a parametric family of Q-functions, $\mathcal{Q} = \left\{ Q_{\boldsymbol{w}} : \mathcal{S} \times \mathcal{A} \to \mathbb{R} \mid \boldsymbol{w} \in \mathbb{R}^d \right\}$, and we assume each function in $\mathcal{Q}$ to be uniformly bounded by $\frac{R_{max}}{1-\gamma}$. When learning the optimal value function, a quantity of interest is how close a given function $Q_{\boldsymbol{w}}$ is to the fixed-point of the Bellman operator. A possible measure is its Bellman error (or Bellman residual), defined by $B_{\boldsymbol{w}} \triangleq TQ_{\boldsymbol{w}} - Q_{\boldsymbol{w}}$. Notice that $Q_{\boldsymbol{w}}$ is optimal if and only if $B_{\boldsymbol{w}}(s,a) = 0$ for all s, a. If we assume the existence of a distribution ν over $\mathcal{S} \times \mathcal{A}$, a sound objective is to directly minimize the squared Bellman error of $Q_{\boldsymbol{w}}$ under ν, denoted by $\|B_{\boldsymbol{w}}\|_\nu^2$. Unfortunately, it is well-known that an unbiased estimator of this quantity requires two independent samples of the next state s' for each s, a (e.g., [25]). In practice, the Bellman error is typically replaced by the TD error $b(\boldsymbol{w})$, which approximates the former using a single transition sample $\langle s, a, s', r \rangle$, $b(\boldsymbol{w}) = r + \gamma \max_{a'} Q_{\boldsymbol{w}}(s', a') - Q_{\boldsymbol{w}}(s,a)$. Finally, given a dataset $D = \langle s_i, a_i, r_i, s'_i \rangle_{i=1}^N$ of N samples, the squared TD error is computed as $\|B_{\boldsymbol{w}}\|_D^2 = \frac{1}{N} \sum_{i=1}^N (r_i + \gamma \max_{a'} Q_{\boldsymbol{w}}(s'_i, a') - Q_{\boldsymbol{w}}(s_i, a_i))^2 = \frac{1}{N} \sum_{i=1}^N b_i(\boldsymbol{w})^2$. Whenever the distinction is clear from the context, with a slight abuse of terminology, we refer to the squared Bellman error and squared TD error as Bellman error and TD error, respectively.

3 Variational Transfer Learning

In this section, we describe our variational approach to transfer in RL. In Section 3.1, we start by introducing our algorithm from a high-level perspective, in such a way that any choice of prior and posterior distributions is possible. Then, in Sections 3.2 and 3.3, we propose practical implementations based on Gaussians and mixtures of Gaussians, respectively. We conclude with some considerations on how to optimize the proposed objective in Section 3.4.

Algorithm 1 Variational Transfer

Require: Target task $\mathcal{M}_\tau$, source Q-function weights $\mathcal{W}_s$, batch size M

1: Estimate prior $p(\boldsymbol{w})$ from $\mathcal{W}_s$
2: Initialize variational parameters: $\boldsymbol{\xi} \leftarrow \operatorname{argmin}_{\boldsymbol{\xi}} KL(q_{\boldsymbol{\xi}} || p)$
3: Initialize replay buffer: $D = \emptyset$
4: **repeat**
5:     Sample initial state: $s_0 \sim p_0$
6:     **while** s_h is not terminal **do**
7:         Sample weights: $\boldsymbol{w} \sim q_{\boldsymbol{\xi}}(\boldsymbol{w})$
8:         Take action $a_h = \operatorname{argmax}_a Q_{\boldsymbol{w}}(s_h, a)$
9:         Observe transition $s_{h+1} \sim \mathcal{P}_\tau(\cdot|s_h, a_h)$ and collect reward $r_{h+1} = \mathcal{R}_\tau(s_h, a_h)$
10:         Add sample to the replay buffer: $D \leftarrow D \cup \langle s_h, a_h, r_{h+1}, s_{h+1}\rangle$
11:         Sample mini-batch $D' = \langle s_i, a_i, r_i, s_i' \rangle_{i=1}^M$ from D
12:         Estimate the gradient $\nabla_{\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{\xi})$ using D'
13:         Update $\boldsymbol{\xi}$ in the direction of $-\nabla_{\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{\xi})$ using any stochastic optimizer (e.g., ADAM)
14:     **end while**
15: **until** forever

3.1 Algorithm

Let us observe that the distribution $\mathcal{D}$ over tasks induces a distribution over optimal Q-functions. Furthermore, for any MDP, learning its optimal Q-function is sufficient for solving the problem. Thus, we can safely replace the distribution over tasks with the distribution over their optimal value functions. In our parametric settings, we reduce the latter to a distribution $p(\boldsymbol{w})$ over weights. Assume, for the moment, that we know the distribution $p(\boldsymbol{w})$ and consider a dataset $D = \langle s_i, a_i, r_i, s_i' \rangle_{i=1}^N$ of samples from some task $\mathcal{M}_\tau \sim \mathcal{D}$ that we want to solve. Then, we can compute the posterior distribution over weights given such dataset by applying Bayes theorem as $p(\boldsymbol{w}|D) \propto p(D|\boldsymbol{w})p(\boldsymbol{w})$. Unfortunately, this cannot be directly used in practice since we do not have a model of the likelihood $p(D|\boldsymbol{w})$. In such case, it is very common to make strong assumptions on the MDPs or the Q-functions to get tractable posteriors. However, in our transfer settings, all distributions involved depend on the family of tasks under consideration and making such assumptions is likely to limit the applicability to specific problems. Thus, we take a different approach to derive a more general, but still well-grounded, solution. Notice that our final goal is to move the total probability mass over the weights minimizing some empirical loss measure, which in our case is the TD error $\|B_{\boldsymbol{w}}\|_D^2$. Then, given a prior $p(\boldsymbol{w})$, we know from PAC-Bayesian theory that the optimal Gibbs posterior q minimizing an oracle upper bound on the expected loss takes the form (e.g., [8]):

$$q(\boldsymbol{w}) = \frac{e^{-\Lambda \|B_{\boldsymbol{w}}\|_D^2} p(\boldsymbol{w})}{\int e^{-\Lambda \|B_{\boldsymbol{w}'}\|_D^2} p(d\boldsymbol{w}')}, \tag{1}$$

for some parameter $\Lambda > 0$. Since Λ is typically chosen to increase with the number of samples N, in the remaining, we set it to $\lambda^{-1}N$, for some constant $\lambda > 0$. Notice that, whenever the term $e^{-\Lambda \|B_{\boldsymbol{w}}\|_D^2}$ can be interpreted as the actual likelihood of D, q becomes a classic Bayesian posterior. Although we now have an appealing distribution, the integral at the denominator of (1) is intractable to compute even for simple Q-function models. Thus, we propose a variational approximation $q_{\boldsymbol{\xi}}$ by considering a simpler family of distributions parameterized by $\boldsymbol{\xi} \in \Xi$. Then, our problem reduces to finding the variational parameters $\boldsymbol{\xi}$ such that $q_{\boldsymbol{\xi}}$ minimizes the Kullback-Leibler (KL) divergence w.r.t. the Gibbs posterior q. From the theory of variational inference (e.g., [6]), this can be shown to be equivalent to minimizing the well-known (negative) *evidence lower bound* (ELBO):

$$\min_{\boldsymbol{\xi} \in \Xi} \mathcal{L}(\boldsymbol{\xi}) = \mathbb{E}_{\boldsymbol{w} \sim q_{\boldsymbol{\xi}}} \left[\|B_{\boldsymbol{w}}\|_D^2 \right] + \frac{\lambda}{N} KL \left(q_{\boldsymbol{\xi}}(\boldsymbol{w}) \mid\mid p(\boldsymbol{w}) \right). \tag{2}$$

The approximate posterior balances between placing probability mass over those weights $\boldsymbol{w}$ that have low expected TD error (first term), and staying close to the prior distribution (second term). Assuming that we can compute the gradients of (2) w.r.t. the variational parameters $\boldsymbol{\xi}$, our objective can be optimized using any stochastic optimization algorithm, as shown in the next subsections.

We now highlight our general transfer procedure in Algorithm 1, while deferring a description of specific choices for the involved distributions to the next two subsections. Given a set of weights $\mathcal{W}_s$ from the source tasks' optimal Q-functions, we start by estimating the prior distribution (line 1), and we initialize the variational parameters by minimizing the KL divergence w.r.t. such distribution (line 2).[1] Then, at each time step of interaction, we re-sample the weights from the current approximate posterior and act greedily w.r.t. the corresponding Q-function (lines 7,8). After collecting the new experience (lines 9-10), we draw a mini-batch of samples from the replay buffer (line 11), use this to estimate the objective function gradient (line 12), and update the variational parameters (line 13).

The key property of our approach is the weight resampling at line 7, which resembles the well-known Thompson sampling approach adopted in multi-armed bandits [7] and closely relates to the recent value function randomization [27, 3]. At each time we guess what is the task we are trying to solve based on our current belief and we act as if such guess were true. This mechanism allows an efficient adaptive exploration of the target task. Intuitively, during the first steps of interaction, the agent is very uncertain about the current task, and such uncertainty induces stochasticity in the chosen actions, allowing a rather informed exploration to take place. Consider, for instance, that actions that are bad on average for all tasks are improbable to be sampled, while this cannot happen in uninformed exploration strategies, like ϵ-greedy, before learning takes place. As the learning process goes on, the algorithm will quickly figure out which task is solving, thus moving all the probability mass over the weights minimizing the TD error. From that point, sampling from the posterior is approximately equivalent to deterministically taking such weights, and no more exploration will be performed.

Finally, notice the generality of the proposed approach: as far as the objective $\mathcal{L}$ is differentiable in the variational parameters $\boldsymbol{\xi}$, and its gradients can be efficiently computed, any approximator for the Q-function and any prior/posterior distributions can be adopted. For the latter, we describe two practical choices in the next two sections.

3.2 Gaussian Variational Transfer

We now restrict to a specific choice of the prior and posterior families that makes our algorithm very efficient and easy to implement. We assume that optimal Q-functions (or better, their weights) follow a multivariate Gaussian distribution. That is, we model the prior as $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ and we learn its parameters from the set of source weights using maximum likelihood estimation (with small regularization to make sure the covariance is positive definite). Then, our variational family is the set of all well-defined Gaussian distributions, i.e., the variational parameters are $\Xi = \left\{(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \mid \boldsymbol{\mu} \in \mathbb{R}^d, \boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}, \boldsymbol{\Sigma} \succ 0\right\}$. To prevent the covariance from becoming not positive definite, we consider its Cholesky decomposition $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$ and learn the lower-triangular Cholesky factor $\boldsymbol{L}$ instead. In this case, deriving the gradient of the objective is very simple. Both the KL between two multivariate Gaussians and its gradients have a simple closed-form expression. The expected log-likelihood, on the other hand, can be easily differentiated by adopting the reparameterization trick (e.g., [13, 29]). We report these results in Appendix B.2.

3.3 Mixture of Gaussian Variational Transfer

Although the Gaussian assumption of the previous section is very appealing as it allows for a simple and efficient way of computing the variational objective and its gradients, in practice it rarely allows us to describe the prior distribution accurately. In fact, even for families of tasks in which the reward and transition models are Gaussian, the Q-values might be far from being normally distributed. Depending on the family of tasks under consideration and, since we are learning a distribution over weights, on the chosen function approximator, the prior might have arbitrarily complex shapes. When the information loss due to the Gaussian approximation becomes too severe, the algorithm is likely to fail at capturing any similarities between the tasks. We now propose a variant to successfully solve this problem, while keeping the algorithm efficient and simple enough to be applied in practice.

Given the source tasks' weights $\mathcal{W}_s$, we model our estimated prior as a mixture with equally weighted isotropic Gaussians centered at each weight: $p(\boldsymbol{w}) = \frac{1}{|\mathcal{W}_s|}\sum_{\boldsymbol{w}_s \in \mathcal{W}_s} \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_s, \sigma_p^2 \boldsymbol{I})$. This model resembles a kernel density estimator [31] with bandwidth σ_p^2 and, due to its nonparametric nature, it allows capturing arbitrarily complex distributions. Consistently with the prior, we model our approximate posterior as a mixture of Gaussians. Using C components, our posterior is $q_{\boldsymbol{\xi}}(\boldsymbol{w}) = \frac{1}{C}\sum_{i=1}^{C} \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, with variational parameters $\boldsymbol{\xi} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_C, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_C)$. Once again,

[1] If the prior and approximate posterior were in the same family of distributions we could simply set $\boldsymbol{\xi}$ to the prior parameters. However, we are not making this assumption at this point.

we learn Cholesky factors instead of full covariances. Finally, since the KL divergence between two mixtures of Gaussians has no closed-form expression, we rely on an upper bound to such quantity, so that the negative ELBO still upper bounds the KL between the approximate and the exact posterior. Among the many upper bounds available, we adopt the one proposed in [12] (see Appendix B.3).

3.4 Minimizing the TD Error

From Sections 3.2 and 3.3, we know that differentiating the negative ELBO $\mathcal{L}$ w.r.t. $\boldsymbol{\xi}$ requires differentiating $\|B_{\boldsymbol{w}}\|_D^2$ w.r.t. $\boldsymbol{w}$. Unfortunately, the TD error is well-known to be non-differentiable due to the presence of the max operator. This issue is rarely a problem since typical value-based algorithms are semi-gradient methods, i.e., they do not differentiate the targets (see, e.g., Chapter 11 of [35]). However, our transfer settings are quite different from common RL. In fact, our algorithm is likely to start from Q-functions that are very close to an optimum and aims only to adapt the weights in some direction of lower error so as to quickly converge to the solution of the target task. Unfortunately, this property does not hold for most semi-gradient algorithms. Even worse, many online RL algorithms combined with complex function approximators (e.g., DQNs) are well-known to be unstable, especially when approaching an optimum, and require many tricks and tuning to work well [30, 38]. This property is clearly undesirable in our case, as we only aim at adapting already good solutions. Thus, we consider using a residual gradient algorithm [4]. To differentiate the targets, we replace the optimal Bellman operator with the mellow Bellman operator introduced in [2], which adopts a softened version of max called *mellowmax*:

$$\operatorname*{mm}_a Q_{\boldsymbol{w}}(s,a) = \frac{1}{\kappa}\log\frac{1}{|\mathcal{A}|}\sum_a e^{\kappa Q_{\boldsymbol{w}}(s,a)} \tag{3}$$

where κ is a hyperparameter and $|\mathcal{A}|$ is the number of actions. The mellow Bellman operator, which we denote as $\widetilde{T}$, has several appealing properties: (i) it converges to the maximum as $\kappa \to \infty$, (ii) it has a unique fixed-point, and (iii) it is *differentiable*. Denoting by $\widetilde{B}_{\boldsymbol{w}} = \widetilde{T}Q_{\boldsymbol{w}} - Q_{\boldsymbol{w}}$ the Bellman residual w.r.t. the mellow Bellman operator $\widetilde{T}$, we have that the corresponding TD error, $||\widetilde{B}_{\boldsymbol{w}}||_D^2$, is now differentiable w.r.t. $\boldsymbol{w}$. Further considerations on how to better optimize it are given in Appendix B.1.

4 Theoretical Analysis

A first important question that we need to answer is whether replacing max with mellow-max in the Bellman operator constitutes a strong approximation or not. It has been proved [2] that the mellow Bellman operator is a non-expansion under the L_∞-norm and, thus, has a unique fixed-point. However, how such fixed-point differs from the one of the optimal Bellman operator remains an open question. Since mellow-max monotonically converges to max as $\kappa \to \infty$, it would be desirable if the fixed point of the corresponding operator also monotonically converged to the fixed point of the optimal one. We confirm that this property actually holds in the following theorem.

Theorem 1. *Let Q^* be the fixed-point of the optimal Bellman operator T. Define the action-gap function $g(s)$ as the difference between the value of the best action and the second best action at each state s. Let $\widetilde{Q}$ be the fixed-point of the mellow Bellman operator $\widetilde{T}$ with parameter $\kappa > 0$ and denote by $\beta_\kappa > 0$ the inverse temperature of the induced Boltzmann distribution (as in [2]). Then:*

$$\left\|Q^* - \widetilde{Q}\right\|_\infty \le \frac{2\gamma R_{max}}{(1-\gamma)^2}\left\|\frac{1}{1+\frac{1}{|\mathcal{A}|}e^{\beta_\kappa g}}\right\|_\infty . \tag{4}$$

The proof is provided in Appendix A.1. Notice that $\widetilde{Q}$ converges to Q^* exponentially fast as κ (equivalently, β_κ) increases and the action gaps are all larger than zero. Notice that this result is of interest even outside our specific settings.

The second question that we need to answer is whether we can provide any guarantee on our algorithm's performance when given limited data. To address this point, we consider the two variants of Algorithm 1 from Section 3.2 and 3.3 with linear approximators. We assume only a finite dataset is available and provide a finite-sample analysis bounding the expected (mellow) Bellman error under the variational distribution minimizing the objective (2). Due to space constraints, we provide only the result for mixtures of Gaussians, while referring the reader to Appendix A.2 for the Gaussian case.

Theorem 2. *Fix a target task $\mathcal{M}_\tau$ and let $\widetilde{Q}$ be the fixed-point of the corresponding mellow Bellman operator. Assume linearly parameterized value functions $Q_{\boldsymbol{w}}(s,a) = \boldsymbol{w}^T\boldsymbol{\phi}(s,a)$ with bounded weights $\|\boldsymbol{w}\|_2 \le w_{max}$ and uniformly bounded features $\|\boldsymbol{\phi}(s,a)\|_2 \le \phi_{\max}$. Consider the mixture version of Algorithm 1 using C components, source task weights $\mathcal{W}_s$, and bandwidth σ_p^2 for the prior. Denote by $\widehat{\boldsymbol{\xi}} = (\widehat{\boldsymbol{\mu}}_1, \dots, \widehat{\boldsymbol{\mu}}_C, \widehat{\boldsymbol{\Sigma}}_1, \dots, \widehat{\boldsymbol{\Sigma}}_C)$ the variational parameters minimizing the objective of Eq.* (2) *on a dataset D of N i.i.d. samples distributed according to τ and ν. Let $\boldsymbol{w}^* = \operatorname{arginf}_{\boldsymbol{w}} ||\widetilde{B}_{\boldsymbol{w}}||_\nu^2$ and define $\upsilon(\boldsymbol{w}^*) \triangleq \mathbb{E}_{\mathcal{N}(\boldsymbol{w}^*, \frac{1}{N}\boldsymbol{I})}\left[\nu(\boldsymbol{w})\right]$, with $v(\boldsymbol{w}) \triangleq \mathbb{E}_\nu\left[Var_{\mathcal{P}_\tau}\left[\widetilde{b}(\boldsymbol{w})\right]\right]$. Then, there exist constants c_1, c_2, c_3 such that, with probability at least $1-\delta$ over the choice of weights $\boldsymbol{w} \sim \frac{1}{C}\sum_i \mathcal{N}(\widehat{\boldsymbol{\mu}}_i, \widehat{\boldsymbol{\Sigma}}_i)$ and dataset D:*

$$\mathbb{E}_{q_{\widehat{\boldsymbol{\xi}}}}\left[\left\|\widetilde{B}_{\boldsymbol{w}}\right\|_\nu^2\right] \le 2\left\|\widetilde{B}_{\boldsymbol{w}^*}\right\|_\nu^2 + \upsilon(\boldsymbol{w}^*) + c_1\sqrt{\frac{\log\frac{2}{\delta}}{N}} + \frac{c_2 + \lambda d \log N + 2\lambda\varphi\left(\frac{1}{2\sigma_p^2}\|\boldsymbol{w}^* - \boldsymbol{w}_j\|\right)}{N} + \frac{c_3}{N^2}, \quad (5)$$

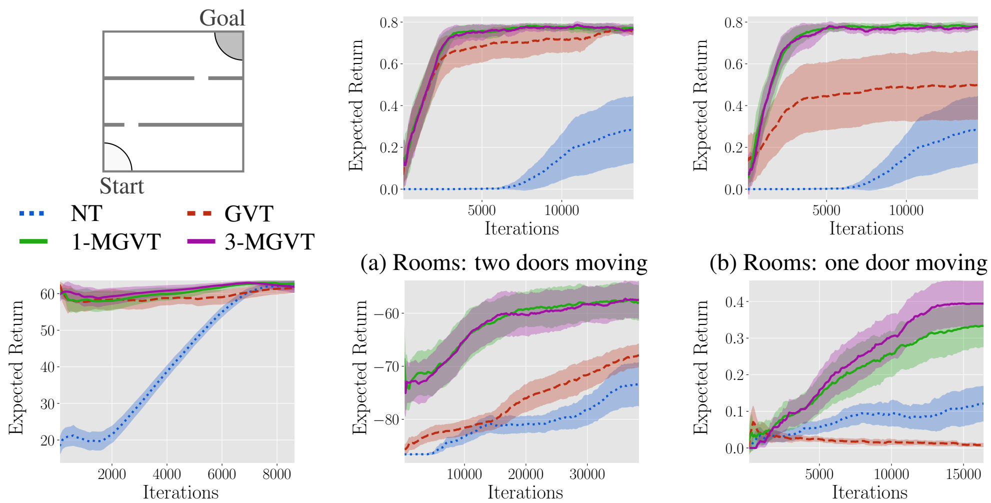
where, for a vector $\boldsymbol{x} = (x_1, \dots, x_d)$, $\varphi(x_j) \triangleq \sum_i \frac{e^{-x_i}}{\sum_j e^{-x_j}} x_i$ is the softmin function.

We refer the reader to Appendix A.2 for the proof and a specific definition of the constants. Four main terms constitute our bound: the approximation error due to the limited hypothesis space (first term), the variance (second and third terms), the distance to the prior (third term), and a constant term decaying as $\mathcal{O}(N^2)$. Our main result shows a remarkable property of using Alg. 1 with mixtures of Gaussians: in order to tighten the bound, it is enough to have at least one source task that is close to the optimal solution of the target task. In such case, the dominating error is due to the variance of the estimates, and, thus, the algorithm is expected to achieve good performance rather quickly, as new data is collected. Furthermore, as $N \to \infty$ the only error terms remaining are the irreducible approximation error due to the limited functional space and the variance term $\upsilon(\boldsymbol{w}^*)$. The latter is due to the fact that we minimize a biased estimator of the Bellman error and can be removed in cases where double sampling of the next state is possible (e.g., in simulation). Finally, we want to point out that the only difference between the bound of Theorem 2 and the one for the Gaussian version of Alg. 1 (Theorem 3 in Appendix A.2) is, as expected, in the term bounding the distance to the prior. While in Theorem 2 we have the (smoothened) minimum distance to the source tasks' weights, in the Gaussian case we have the distance to the mean of such weights, $\|\boldsymbol{w}^* - \boldsymbol{\mu}_p\|_{\boldsymbol{\Sigma}_p^{-1}}$. This proves a remarkable advantage of using mixtures. In fact, the Gaussian version requires the source tasks to be, on average, similar to the target task in order to perform well, while the mixture version only requires this property for one of them. We verify this consideration from an empirical perspective in the next section.

5 Experiments

In this section, we provide an experimental evaluation of our approach in four different domains with increasing level of difficulty. In all experiments, we compare our Gaussian variational transfer algorithm (GVT) and the version using a c-component mixture of Gaussians (c-MGVT) to plain no-transfer RL (NT) with ϵ-greedy exploration. To the best of our knowledge, no existing transfer algorithm is directly comparable to our approach from an experimental perspective. A comparative discussion of related works motivating this statement is provided in the next section.

The Rooms Problem We consider an agent navigating in the environment depicted in Figure 1. The agent starts in the bottom-left corner and must move from one room to another to reach the goal position in the top-right corner. The rooms are connected by small doors whose locations are unknown to the agent. The state-space is modeled as a 10×10 continuous grid, while the action-space is the set of 4 movement directions (up, right, down, left). After each action, the agent moves by 1 in the chosen direction and the final position is corrupted by Gaussian noise $\mathcal{N}(0, 0.2)$. In case the agent hits a wall, its position remains unchanged. The reward is 1 when reaching the goal (after which the process terminates) and 0 otherwise, while the discount factor is $\gamma = 0.99$. In this experiment, we consider linearly parameterized Q-functions with 121 equally-spaced radial basis features.
We generate a set of 50 source tasks for the three-room environment of Figure 1 by sampling both door locations uniformly in the allowed space, and solve all of them by directly minimizing the



(c) Cartpole (d) Mountain Car (e) Maze Navigation

Figure 1: Expected return as a function of the number of iterations averaged over 20 independent runs. 95% confidence intervals are shown.

TD error as presented in Section 3.4. Then, we use our algorithms to transfer from 10 source tasks sampled from the previously generated set. The average return over the last 50 learning episodes as a function of the number of iterations is shown in Figure 1a. Each curve is the result of 20 independent runs, each one resampling the target and source tasks, with 95% confidence intervals. Further details on the parameters adopted in this experiment are given in Appendix C.1. As expected, the no-transfer (NT) algorithm fails at learning the task in so few iterations due to the limited exploration provided by an ϵ-greedy policy. On the other hand, all our algorithms achieve a significant speed-up and converge to the optimal performance in few iterations, with GVT being slightly slower. Interestingly, we notice that there is no advantage in adopting more than 1 component for the posterior in MGVT. This result is intuitive since, as soon as the algorithm figures out which is the target task, all the components move towards the same region.

To better understand the differences between GVT and MGVT, we now consider transferring from a slightly different distribution than the one from which target tasks are drawn. We generate 50 source tasks again but this time with the bottom door fixed at the center and the other one moving. Then, we repeat the previous experiment, allowing both doors to move when sampling target tasks. The results are shown in Figure 1b. Interestingly, MGVT seems almost unaffected by this change, proving that it has sufficient representation power to generalize to slightly different task distributions. The same does not hold for GVT, which now is not able to solve many of the sampled target tasks, as can be noticed from the higher variance. This result proves again that assuming Gaussian distributions can pose severe limitations in our transfer settings.

Classic Control We now consider two well-known classic control environments: Cartpole and Mountain Car [35]. For both, we generate 20 source tasks by uniformly sampling their physical parameters (cart mass, pole mass, pole length for Cartpole and car speed for Mountain Car) and solve them by directly minimizing the TD error as in the previous experiment. We parameterize Q-functions using neural networks with one layer of 32 hidden units for Cartpole and 64 for Mountain Car. A better description of these two environments and their parameters is given in Appendix C.2. In this experiment, we use a Double Deep Q-Network (DDQN) [38] to provide a stronger no-transfer baseline for comparison. The results (same settings of Section 5) are shown in Figures 1c and 1d. For Cartpole (Figure 1c), all transfer algorithms are almost zero-shot. This result is expected since, although we vary the system parameters in a wide range, the optimal Q-values of states near the balanced position are similar for all tasks. On the contrary, in Mountain Car (Figure 1d) the optimal Q-functions become very different when changing the car speed. This phenomenon hinders the learning of GVT in the target task, while MGVT achieves a good jump-start and converges in fewer iterations.

Maze Navigation In our last experiment, we consider a robotic agent navigating mazes. At the beginning of each episode, the agent is dropped to a random position in a $10m^2$ maze and must reach a goal area in the smallest time possible. The robot is equipped with sensors detecting its absolute position, its orientation, the distance to any obstacle within $2m$ in 9 equally-spaced directions, and whether the goal is present in the same range. The only actions available are *move forward* with speed $0.5m/s$ or *rotate* (in either direction) with speed of $\pi/8\ rad/s$. Each time step corresponds to $1s$ of simulation. The reward is 1 for reaching the goal and 0 otherwise, while the discount factor is $\gamma = 0.99$. For this experiment, we design a set of 20 different mazes and solve them using a DDQN with two layers of 32 neurons and ReLU activations. Then, we fix a target maze and transfer from 5 source mazes uniformly sampled from such set (excluding the chosen target). To further assess the robustness of our method, we now consider transferring from the Q-functions learned by DDQNs instead of those obtained by minimizing the TD error as in the previous domains. From our considerations of Sections 3.4 and 4, the fixed-points of the two algorithms are different, which creates a further challenge for our method. We show the results for a fixed target maze in Figure 1e, while referring the reader to Appendix C.3 for the illustration of our mazes and additional results. Once again, MGVT achieves a remarkable speed-up over (no-transfer) DDQN. This time, using 3 components achieves slightly better performance than using only 1, which is likely due to the fact that the task distribution is much more complicated than in the previous domains. For the same reason, GVT shows negative transfer and performs even worse than DDQN.

6 Related Works

Our approach is mostly related to [20]. Although we both assume the tasks to share similarities in their value functions, [20] consider only linear approximators and adopt a hierarchical Bayesian model of the corresponding weights' distribution, which is assumed Gaussian. On the other hand, our variational approximation allows for more general distribution families and can be combined with non-linear approximators. Furthermore, [20] propose a Dirichlet process model for the case where weights cluster into different classes, which relates to our mixture formulation and proves the importance of capturing more complicated task distributions again. Finally, [20] considers the problem of jointly learning all given tasks, while we focus on transferring information from a set of source tasks to the target task. In [39], the authors propose a hierarchical Bayesian model for the distribution over MDPs. Unlike our approach and [20], they consider a distribution over transition probabilities and rewards, rather than value functions. In the same spirit of our method, they consider a Thompson sampling-based procedure which, at each iteration, samples a new task from the posterior and solves it. However, [39] consider only finite MDPs, which poses a severe limitation on the algorithm's applicability. On the contrary, our approach can handle high-dimensional tasks. In [10], the authors consider a family of tasks whose dynamics are governed by some hidden parameters and use Gaussian processes (GPs) to model such dynamics across tasks. Recently, [15] extended this approach by replacing GPs with Bayesian neural networks to obtain a more scalable approach. Both approaches result in a model-based algorithm that quickly adapts to new tasks by estimating their hidden parameters, while we propose a model-free method which does not require such assumptions.

7 Conclusion

We presented a variational method for transferring value functions in RL. We showed our approach to be general, in the sense that it can be combined with several distributions and function approximators, while providing two practical algorithms based on Gaussians and mixtures of Gaussians, respectively. We analyzed both from a theoretical and empirical perspectives, proving that the Gaussian version has severe limitations, while the mixture one is much better for our transfer settings.

Since our algorithm effectively models the uncertainty over tasks, a relevant future work is to design an algorithm that explicitly explores the target task to reduce such uncertainty (e.g., [14]). Furthermore, our variational approach could be extended to model a distribution over optimal policies instead of value functions (e.g., [32, 24]), which might allow better transferred behavior.

References

[1] Pierre Alquier, James Ridgway, and Nicolas Chopin. On the properties of variational approximations of gibbs posteriors. *Journal of Machine Learning Research*, 17(239):1–41, 2016.

[2] Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pages 243–252, 2017.

[3] Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. *arXiv preprint arXiv:1802.04412*, 2018.

[4] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

[5] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

[6] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[7] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[8] Olivier Catoni. Pac-bayesian supervised classification: the thermodynamics of statistical learning. *arXiv preprint arXiv:0712.0248*, 2007.

[9] Vincent Cottet and Pierre Alquier. 1-bit matrix completion: Pac-bayesian analysis of a variational approximation. *Machine Learning*, 107(3):579–603, 2018.

[10] Finale Doshi-Velez and George Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *IJCAI: proceedings of the conference*, volume 2016, page 1432. NIH Public Access, 2016.

[11] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727. ACM, 2006.

[12] John R Hershey and Peder A Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–317. IEEE, 2007.

[13] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[14] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[15] Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. Robust and efficient transfer learning with hidden parameter markov decision processes. In *Advances in Neural Information Processing Systems*, pages 6250–6261, 2017.

[16] Jens Kober and Jan R Peters. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pages 849–856, 2009.

[17] George Konidaris and Andrew Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 489–496. ACM, 2006.

[18] George Konidaris and Andrew G Barto. Building portable options: Skill transfer in reinforcement learning.

[19] Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.

[20] Alessandro Lazaric and Mohammad Ghavamzadeh. Bayesian multi-task reinforcement learning. In *ICML-27th International Conference on Machine Learning*, pages 599–606. Omnipress, 2010.

[21] Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 544–551. ACM, 2008.

[22] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[23] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[24] Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*, 2017.

[25] Odalric-Ambrym Maillard, Rémi Munos, Alessandro Lazaric, and Mohammad Ghavamzadeh. Finite-sample analysis of bellman residual minimization. In *Proceedings of 2nd Asian Conference on Machine Learning*, pages 299–314, 2010.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[27] Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. *arXiv preprint arXiv:1402.0635*, 2014.

[28] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[29] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[30] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[31] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[32] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Policy gradients with parameter-based exploration for control. In *International Conference on Artificial Neural Networks*, pages 387–396. Springer, 2008.

[33] Yevgeny Seldin, François Laviolette, Nicolo Cesa-Bianchi, John Shawe-Taylor, and Peter Auer. Pac-bayesian inequalities for martingales. *IEEE Transactions on Information Theory*, 58(12):7086–7093, 2012.

[34] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[35] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[36] Matthew E Taylor, Nicholas K Jong, and Peter Stone. Transferring instances for model-based reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 488–505. Springer, 2008.

[37] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[38] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. 2016.

[39] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pages 1015–1022. ACM, 2007.