POLITECNICO DI MILANO

Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Master of Science in

Computer Science and Engineering

A Variational Approach to Transfer Value Functions in Reinforcement Learning

Supervisor:

PROF. MARCELLO RESTELLI

Assistant Supervisor:

DOTT. ANDREA TIRINZONI

Master Graduation Thesis by:

RAFAEL A. RODRIGUEZ S.
Student Id n. 874390

Academic Year 2017-2018

To my Mother, to my Father and to my Brother.

ACKNOWLEDGMENTS

First, I would like to sincerely thank my advisors Prof. Marcello Restelli and Andrea Tirinzoni for all the support during this work: your insightful comments and ideas were fundamental as they guided me through this work. This work made my interest in doing research even stronger. I am glad I took on this project with you.

I would like to thank, as well, the friends that gave me their unconditional support and words of encouragement that made my experience through this masters program better.

Finalmente, pero de ninguna manera menos importante, agradezco a mi familia porque su apoyo y enseñanzas han sido, y siempre serán, fundamentales para el desarrollo de mis proyectos.[1]

1 Finally, but never less important, I thank my family as their support and teachings have been, and will always be, fundamental in the development of my projects.

CONTENTS

LIST OF FIGURES

ACRONYMS

MDP	Markov Decision Process
GVT	Gaussian Variational Transfer
MGVT	Mixture of Gaussians Variational Transfer
RL	Reinforcement Learning
SGD	Stochastic Gradient Descent
TD	Time Difference
VT	Variational Transfer
DDQN	Double Deep Q-Network
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit
ELBO	Evidence Lower Bound
PAC	Probably Approximately Correct
KL	Kullback-Leibler
MTL	Multitask Learning
BNN	Bayesian Neural Network
LSVI	Least Squares Value Iteration
AI	Artificial Intelligence

ABSTRACT

Reinforcement Learning (RL) has shown promise to be a good framework for a sequential decision-making agent to learn by interaction; however, it could require a lot of time and experience samples to learn each individual task. *Transfer* in RL is of main importance to achieve efficient intelligent decision-making agents as it strives to exploit, as much as possible, previously learned tasks. We consider the transfer of knowledge, encoded by the optimal value functions, in the scenario in which a target task must be solved only knowing the optimal value functions of a set of related source tasks. We propose a general algorithm, called *Variational Transfer*, based on Variational Inference that works with parameterized value functions. This choice makes the algorithm applicable to the function regressors popular nowadays, such as neural networks, and by just constraining the distributions families to be parameterized, as well, it provides great flexibility to model the, possibly complex, distributions of the optimal value functions. In a more grounded perspective, we put forward two practical implementations: the first one based on the Multivariate Gaussian distribution family and, the second, to allow for more powerful representations, based on Mixtures of Gaussians. We evaluate both of these with numerical simulations to tackle different environments with increasing complexity, both to gain insight of the empirical behavior of our approaches and to measure their performance when compared to state-of-the-art RL algorithms.

SOMMARIO

L'apprendimento per rinforzo, Reinforcement Learning (RL) in inglese, ha dimostrato di essere un buon metodo per permettere ad un agente che prende decisioni sequenzialmente di imparare tramite l'interazione con l'ambiente, anche se di solito necessita molto tempo ed esperienza per apprendere ogni compito. L'uso di tecniche di trasferimento in RL è di centrale importanza per realizzare sistemi decisionali intelligenti che siano anche efficienti, dato che queste tecniche cercano di sfruttare, nella miglior maniera possibile, i compiti già appresi. In questa tesi consideriamo il trasferimento di conoscenza, rappresentata dalle funzioni di valore ottime, nello scenario in cui l'agente deve apprendere un compito obbiettivo quando conosce le funzioni di valore ottime di altri compiti simili. Proponiamo un algoritmo generale, chiamato *Variational Transfer*, basato su tecniche d'inferenza con metodi variazionali che opera con funzioni di valore parametrizzate. Questa scelta rende questo algoritmo applicabile a modelli di regressione molto utilizzati attualmente, come le reti neurali, richiedendo soltanto che le famiglie di distribuzione siano anch'esse parametrizzate, offre una buona flessibilità per modellare le, potenzialmente complesse, distribuzioni delle funzioni di valore ottime. Inoltre, introduciamo due implementazioni pratiche: la prima basata su distribuzioni gaussiane multivariate e, la seconda, permettendo rappresentazioni più potenti, basata su misture di distribuzioni gaussiane. Valutiamo gli entrambi algoritmi usando simulazioni numeriche con ambienti di diverse difficoltà per capire empiricamente il comportamento dei metodi e la loro performance in confronto con algoritmi dello stato dell'arte in RL.

1 INTRODUCTION

Reinforcement Learning (RL) is the framework that allows an artificially intelligent agent to learn how to act sequentially in stochastic environments in an optimal manner, by maximizing the accumulated reward that it receives from the environment over its interaction time (Sutton and Barto, 1998). The RL research field has had great advancements and breakthroughs in recent years thanks to the improvement of the algorithms and the increasing computational power available. Among these milestones, it is worth noting an agent that learned how to play optimally the Atari game suite (Mnih et al., 2013) by exploiting Deep Neural Networks to process complex visual inputs; the implementation of a system able to master the game of Go and win against professional players (Silver et al., 2016); robots able to learn complex controllers (Kober and Peters, 2009), robots able to act from raw visual inputs (Levine et al., 2016) and to learn in one shot by visually imitating a human (Yu et al., 2018).

Undoubtedly, these results show that the RL framework could achieve even greater results. Nevertheless, these methods still require a great amount of experience to be collected by the agent in order to learn complex controllers, when humans are able to learn new tasks with much less experience. Classic RL algorithms assume basically no prior knowledge each time the agent starts learning a new task, whereas humans are able to exploit previous experience without much additional effort and, thus, learn more efficiently. Is it possible to produce such behavior in artificial agents?

Transfer Learning in RL is an active research area that intends to exploit similarities among tasks solved by an agent, to try to solve new tasks in an efficient manner, much like humans do. This is a main challenge in Artificial Intelligence (AI), given that for an intelligent agent to be able to adapt to a continuously changing environment, it must be able to learn as efficiently as possible; it is simply not feasible for the agent to learn as if oblivious of its previous experience when it needs to adapt quickly. In a more short-term perspective, in many real systems such as robotics, experience samples are very costly and the access to the great quantity that current models require are, in many situations, unattainable, and even if we are far to achieve the goal of a completely autonomous intelligent agent, this efficient use of previous knowledge is of main importance to make RL scale to larger and more complex problems. Therefore, many methods have been proposed by now in which, by assuming specific similarities among the tasks to learn, some kind of knowledge is transferred. In this work, we will present the state of the art of approaches to tranfer, but surveys such as the one done by Taylor and Stone, 2009 offer broader views of the varied approaches taken to tackle *Transfer* in RL.

1.1 CONTRIBUTION

In the literature, some approaches to the *Transfer* problem have been from a Bayesian perspective, given that it is interesting to try to infer what would be the best solution to a new task given that the agent has solved some other similar tasks before. However, Bayesian techniques pose many challenges from the computational perspective and very restrictive assumptions are made in order to propose algorithms that are computationally tractable. In our present work, we propose a novel technique to transfer knowledge from previous tasks, also by taking a Bayesian view of the problem, and we propose to use Variational Inference techniques to avoid restrictive assumptions that would limit the effectiveness of transferring. We consider for this work the knowledge to transfer among the tasks to be in their optimal value functions given that an important set of state-of-the-art algorithms in RL are value-based—i.e. algorithms that exploit the dynamic programming properties of this kind of decision-making problems. Moreover, even though complex, the optimal value functions hold a lot of useful information for a decision-maker; successfully transferring these is of great interest for an agent to know what to expect from tasks. Additionally, value-based RL is, by design, efficient in terms of the samples required and, thus, it is important to research how to make these methods more efficient with the introduction of *Transfer*.

1.2 DOCUMENT OUTLINE

We start this document by introducing base definitions and results relevant to the design of our proposed algorithm in Chapter 2, including basic definitions, value-based RL results, Variational Inference basics and a PAC-Bayesian theory overview. Further, in Chapter 3, we extend the motivation of *Transfer Learning* in RL and provide an overview of various ways transfer has been attempted. In Chapter 4, we present state-of-the-art techniques based on Bayesian inference that are related to our work and other recent ideas that will motivate further our algorithm. The core of our contribution is in this general algorithm we call *Variational Transfer*, presented in Chapter 5, that intends to offer enough flexibility to be useful in the varied landscape of current RL applications. Furthermore, we present two practical implementations assuming that Multivariate Gaussians and Mixtures of Gaussians, respectively, are sufficient to model the underlying distribution of optimal value functions for the tasks. We also put forward a gradient-based optimization approach to search for optimal value functions based on a differentiable approximation of the optimal Bellman Operator that may be of separate interest. In Chapter 6, we present the empirical evaluation of these two implementations of *Variational Transfer* and use four environments of increasing degree of complexity to provide insights of the empirical behavior of the algorithm and the performances that can be obtained with respect to classic RL approaches. Finally, in Chapter 7, we present the conclusions of the research done and some future directions that could be of interest.

2 BACKGROUND

The present chapter is in order to introduce fundamental concepts that will be used during the presentation of our work. Moreover, it will help setting a base notation to be used consistently throughout the rest of the document.

Given that our work is developed within the context of Value-based RL, we start by introducing the concepts related to the main object of study, the Markov Decision Process (MDP), and continue to important definitions within the field of RL and the main results about Value functions. Finally, we present the basic concepts in *Variational Inference* methods for approximate *Bayesian Inference* and an important result in *PAC-Bayesian* learning theory that will set a cornerstone for the design of our approach.

2.1 REINFORCEMENT LEARNING AND MARKOV DECISION PROCESSES

RL considers the problem of an agent—a decision-maker—that interacts with an environment by taking actions sequentially. When the agent takes an action, it receives a reward and it understands, at least partially, the effect of such action by perceiving the change in its observation of the environment's state. In the learning context, the agent does not know about its environment and it must learn by interaction how to best maximize its *return*, that is, the sum of the rewards obtained during the period of interaction. Throughout the development of this document we will consider agents that interact with environments that are:

- *Stationary*, such that the dynamics and reward signals do not change as time passes;
- *Markovian*, such that the changes in the environment depend only on the current state of the environment and the action taken. More precisely, the effects of an action on the environment are completely defined by its current state;
- *Fully Observable*, such that the agent can completely understand the state in which the environment is.

The model used for the scenario described above is that of a discounted *Markov Decision Process* (MDP) as defined next.

Definition 2.1. An MDP $\mathcal{M}$ is defined as the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, p_0, \gamma \rangle$ such that:

- $\mathcal{S}$ is the (continuous) state space of the environment;
- $\mathcal{A}$ is a (finite) set of actions that the agent can take;
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the Markovian transition model such that $\mathcal{P}(\cdot|s,a)$ is the probability of the next state whenever action a is taken in state s;
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. We will consider hereafter that this function is uniformly bounded by a constant $R_{max} \in \mathbb{R}_+$;
- p_0 is a probability distribution over the initial state;
- $\gamma \in (0,1)$ is the discount factor that weighs the importance of the long term reward.

Now, we consider as a possible solution to such type of MDP a stationary deterministic control policy $\pi : \mathcal{S} \to \mathcal{A}$, a mapping from states to actions that will define the behavior of the agent. Moreover, we consider next the action-value function of a policy as:

Definition 2.2. Given a stationary deterministic policy π, we consider the *action-value* function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ to be the expected return when the agent takes action a in state s and follows the policy π from then on.

$$Q^\pi(s,a) := \mathcal{R}(s,a) + \mathop{\mathbb{E}}_{\mathcal{P}}\left[\sum_{t=1}^{\infty} \gamma^t \mathcal{R}(s', \pi(s'))\right]. \tag{2.1}$$

Additionally, the *value* function $V^\pi : \mathcal{S} \to \mathbb{R}$ is defined

$$V^\pi(s) := \mathcal{R}(s, \pi(s)) + \mathop{\mathbb{E}}_{\mathcal{P}}\left[\sum_{t=1}^{\infty} \gamma^t \mathcal{R}(s', \pi(s'))\right]. \tag{2.2}$$

Hereafter, we will refer to this action-value function as Q-function and define the *optimal* Q-function to be one such that

$$Q^*(s,a) = \max_\pi Q^\pi(s,a). \tag{2.3}$$

Further, the next theorem specify how to compute the Q-functions of a policy π and from there we are able to define important operators for the following of the results.

Theorem 2.1 (Puterman, 1994)**.** Given an MDP $\mathcal{M}$ as in Definition 2.1 and a deterministic policy π, we can compute its corresponding action-value function Q^π by solving the *Bellman Equation*

$$Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \mathcal{P}(\cdot|s,a)}\left[Q^\pi(s', \pi(s'))\right] \tag{2.4}$$

We can, then, define the *Bellman Operator* $\mathcal{T}^{\pi}$ such that,

$$\mathcal{T}^{\pi}Q(s,a) := \mathcal{R}(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \mathcal{P}(\cdot|s,a)} \left[Q(s', \pi(s')) \right]. \tag{2.5}$$

The following theorem allows to define analogously the *Optimal* Bellman Operator $\mathcal{T}^{*}$.

Theorem 2.2 (Puterman, 1994). Given an MDP $\mathcal{M}$ as in Definition 2.1, we can compute an optimal deterministic policy π^{*} by solving the *Optimal Bellman Equation*:

$$V^{*}(s) = \max_{a' \in \mathcal{A}} Q^{*}(s,a') \tag{2.6}$$

from which the optimal deterministic policy π^{*}, therefore, is:

$$\pi^{*}(s) \in arg\max_{a' \in \mathcal{A}} Q^{*}(s,a') \tag{2.7}$$

Then, by combining Theorem 2.1 and 2.2 we can solve for the *Optimal Q*-function by:

$$Q^{*}(s,a) = \mathcal{R}(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \mathcal{P}} \left[\max_{a' \in \mathcal{A}} Q^{*}(s',a') \right] \tag{2.8}$$

and, thus, define the *Optimal* Bellman Operator $\mathcal{T}^{*}$

$$\mathcal{T}^{*}Q(s,a) := \mathcal{R}(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim \mathcal{P}} \left[\max_{a' \in \mathcal{A}} Q(s',a') \right] \tag{2.9}$$

2.1.1 *Value-Based RL Fundamentals*

Next, we introduce the main theorem in Value-Based RL techniques and, consequently, define the measures that are basic for the value-based RL algorithms.

Theorem 2.3 (Puterman, 1994). Given an MDP $\mathcal{M}$ defined as in Definition 2.1, consider the Bellman Operator $\mathcal{T}^{\pi}$ as defined in Equation 2.5 and the Optimal Bellman Operator $\mathcal{T}^{*}$ as defined in Equation 2.9. We have that:

1. Given functions Q_1 and Q_2 we have that the Bellman Operator and the Optimal Bellman Operator are a contraction in the L_{∞}-norm such that:

$$||\mathcal{T}^{\pi}Q_1 - \mathcal{T}^{\pi}Q_2||_{\infty} \leq \gamma ||Q_1 - Q_2||_{\infty} \tag{2.10}$$

$$||\mathcal{T}^{*}Q_1 - \mathcal{T}^{*}Q_2||_{\infty} \leq \gamma ||Q_1 - Q_2||_{\infty} \tag{2.11}$$

2. Q^π and Q^* are, respectively, the unique fixed-point of the Bellman Operator $\mathcal{T}^\pi$ and the Optimal Bellman Operator $\mathcal{T}^*$.

Straightforwardly, we say a Q-function to be optimal exactly when the *Bellman Error* (or *Bellman Residual*) $B_Q(s,a) := \mathcal{T}^* Q(s,a) - Q(s,a)$ is zero for every state-action pair. Additionally, from this point on we will only consider Q functions that belong to an uniformly bounded parameterized family $\mathcal{Q}$:

$$\mathcal{Q} := \Big\{ Q_w : \mathcal{S} \times \mathcal{A} \to \mathbb{R} \mid w \in \mathbb{R}^d \\ \wedge\ |Q_w(s,a)| \leq \frac{R_{max}}{1-\gamma}\ \forall (s,a) \in \mathcal{S} \times \mathcal{A} \Big\} \tag{2.12}$$

Hence, we can write the *Bellman Error* as $B_w(s,a) := \mathcal{T}^* Q_w(s,a) - Q_w(s,a)$. Assuming that there exists a probability measure ν over the state-action space $\mathcal{S} \times \mathcal{A}$ we can write its $L_{\nu,p}$-norm

$$\|B_w\|_{\nu,p}^p = \sum_{a \in \mathcal{A}} \int_{\mathcal{S}} |B_w(s,a)|^p \nu(ds,a), \tag{2.13}$$

with $p \in \mathbb{N}^+$, to be used as a measure of the optimality of a given Q function in a given MDP $\mathcal{M}$.

However, in the RL scenario, usually, we will not know about such distribution ν over the state-action space nor the specifics dynamics of the environment; thus, an empirical estimation will be necessary. We consider, then, to have a set of experience samples $D = \langle s_i, a_i, r_i, s'_i \rangle_{i=0}^N$ and we define the following empirical estimator of the Bellman Error:

$$\|\widetilde{B}_w\|_{D,p}^p := \frac{1}{N} \sum_{i=1}^N |b_i(w)|^p \tag{2.14}$$

$$b_i(w) := r_i + \gamma \max_{a' \in \mathcal{A}} Q_w(s'_i, a') - Q_w(s_i, a_i) \tag{2.15}$$

Nonetheless, the estimator in Equation (2.14) is biased and it is known as the average Time Difference (TD) error—given that the Equation (2.15) is known in the literature as the TD error.

2.2 VARIATIONAL INFERENCE

Since our proposed algorithms are based on Variational Inference methods, in this section we will present a summary of the principal concepts for Variational Inference.

2.2.1 *Bayesian Inference and Intractability*

The main motivation for Variational Inference is the need to tractably compute *posterior* distributions as prescribed by the Bayes theorem,

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} = \frac{P(X|Y)P(Y)}{\int_Y P(X|y)P(y)dy} \tag{2.16}$$

where X and Y are random variables and P corresponds to the density function of the indicated random variable.

The main problem is that we are constrained to compute the integral in the denominator of Equation 2.16, known as the *evidence* $P(X)$, when we do not deal with conjugate probability distributions—that is, distributions for the likelihood $P(X|Y)$ and the *prior* distribution $P(Y)$ such that the *posterior* distribution has the same functional form of that of the *prior*. In the presence of conjugacy of the distributions involved, there are analytical solutions that lessen the computational burden of the inference. However, for many useful distributions the direct computation of the inference is intractable and many real situations required distributions that are not conjugate. For instance, many current Neural Network models, such as those used in Computer Vision applications, consider instances with a very large number of parameters (in the order of millions) to be learned—due to the large dimension of the inputs and the many complicated layers being used to tackle such complex tasks. In our Bayesian inference case, Y would correspond to the model's parameters and X would correspond to the data and, hence, we would be constrained to integrate over very large space to compute the *evidence* $P(X)$.

Variational Inference methods are techniques based in the Calculus of Variations to perform approximate Bayesian Inference—hence, the name—that aims to transform such inference problem in an optimization problem.

2.2.2 *Evidence Lower Bound (ELBO)*

To solve the approximate inference problem, variational methods aim to search for the distribution within a, possibly parameterized, family of distributions that best approximates the *posterior*. In order to do this, we first define the *Kullback-Leibler Divergence*, a measure of the difference between probability distributions based on Information Theory.

Definition 2.3 (Kullback-Leibler Divergence)**.** Given probability distributions p and q of a random variable X, the *Kullback-Leibler Divergence* is:

$$KL(q\|p) = \mathop{\mathbb{E}}_{X\sim q}\left[\log\left(\frac{q(x)}{p(x)}\right)\right] \tag{2.17}$$

Next, consider the *posterior* distribution of our inference problem to be p and a parameterized family of distributions q_ξ, with $\xi \in \Xi$, where ξ are known as variational parameters. Therefore, using the Kullback-Leibler (KL) divergence, the solution of the following optimization problem is the Variational Inference approximation.

$$\min_{\xi \in \Xi} KL(q_\xi \| p). \tag{2.18}$$

However, we cannot optimize this directly. To solve this problem, the Evidence Lower Bound (ELBO) is used, which comes from the concavity of the logarithm and Jensen's inequality, that is a lower bound to the *evidence* $P(X)$ (Blei, Kucukelbir, and McAuliffe, 2017). This is a tight lower-bound to the divergence objective in 2.18 above. First, consider that we have an inference problem such as the one in Equation 2.16 with random variables X and Y, such that $p(Y|X)$ is the *posterior distribution*, $P(X|Y)$ is the *likelihood* and $P(Y)$ is the *prior distribution*; thus, the ELBO is defined as follows,

$$ELBO(q_\xi(Y), p(Y|X)) = \mathop{\mathbb{E}}_{y \sim q_\xi} [\log(p(x|y))] - KL(q_\xi(Y) \| p(Y)) \tag{2.19}$$

which can be proved to be equal to the negative KL in the optimization problem in 2.18 (Blei, Kucukelbir, and McAuliffe, 2017)—up to a constant. Therefore, reducing our inference problem to the following optimization problem,

$$\max_{\xi \in \Xi} ELBO(q_\xi(Y), p(Y|X)). \tag{2.20}$$

In this way, the problem of performing Bayesian Inference is reduced from an potentially intractable integration to an optimization problem in the parameter space of approximation distributions.

2.3 INDUCTIVE PAC-BAYESIAN LEARNING OVERVIEW

In this section we introduce an important result within the Probably Approximately Correct (PAC)-Bayesian framework, that will later on serve as a motivation to the design of our approach.

First, we go through some notation that will serve to the introduction of the main result. Let's consider a set of labeled samples $S = \langle x_i, y_i \rangle_{i=1}^N \in (\mathcal{X} \times \mathcal{Y})^N$. It is assumed that such samples are generated i.i.d. (independently and identically distributed) by a probability distribution D over $\mathcal{X} \times \mathcal{Y}$. Additionally, we also suppose a set $\mathcal{H}$ of hypotheses (predictors) $h : \mathcal{X} \to \mathcal{Y}$ and we define some loss function $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. Furthermore, we define the empirical risk of the sampled set S and the generalization error of a predictor h in Equation 2.21 and 2.22, respectively.

$$\widetilde{\mathcal{L}}^{\ell}_{S}(h) = \frac{1}{N}\sum_{i=1}^{N} \ell(h, x_i, y_i) \tag{2.21}$$

$$\mathcal{L}^{\ell}_{\mathsf{D}}(h) = \mathop{\mathbb{E}}_{(x,y)\sim \mathsf{D}} [\ell(h, x, y)] \tag{2.22}$$

In PAC-Bayesian, we are interested in studying bounds when there is a *posterior* probability measure q over the set of hypotheses $\mathcal{H}$ and we would like to infer information from the *empirical* loss with such distribution q, $\mathbb{E}_{h\sim q}\left[\widetilde{\mathcal{L}}^{\ell}_{S}\right]$, to estimate the expected *generalization* error $\mathbb{E}_{h\sim q}\left[\mathcal{L}^{\ell}_{\mathsf{D}}\right]$. Finally, we present the following theorem by Catoni, 2007 that holds for every *posterior* distribution q, that is normally obtained by a learning algorithm after receiving a sampled set S and a *prior* distribution p over the set of hypotheses $\mathcal{M}$, that is known before sampling S.

Theorem 2.4 (Catoni, 2007)**.** Given a distribution D over the $\mathcal{X} \times \mathcal{Y}$, a set of hypotheses $\mathcal{H}$, a loss function $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to [0, 1]$, a *prior* distribution p over $\mathcal{H}$, a real number $\delta \in [0, 1)$ and a real number $\beta > 0$ with probability at least $1 - \delta$ over the choice $(X, Y) \sim \mathsf{D}^N$, we have the following holds for every *posterior* distribution q over $\mathcal{H}$

$$\mathop{\mathbb{E}}_{h\sim q}\left[\mathcal{L}^{\ell}_{\mathsf{D}}(h)\right] \leq \frac{1}{1-e^{-\beta}}\left[1 - e^{-\beta \mathbb{E}_{h\sim q}\left[\widetilde{\mathcal{L}}^{\ell}_{(X,Y)}(h)\right] - \frac{1}{N}\left(KL(q\|p) + \ln\left(\frac{1}{\delta}\right)\right)}\right]$$

Although the bound in Theorem 2.4 is valid for loss functions with range in $[0, 1]$, through a straightforward scaling/translation it could be also applied to bounded loss functions. Furthermore, it also provide an interesting insight that the optimization of such bound could be done by managing a trade-off between the deviation of the *posterior* distribution q from the *prior* p in terms of the KL divergence and the expectation of the *empirical* loss given the *posterior*.

Moreover, it is also known from Catoni, 2007 that the *Gibbs posterior* q^* that optimizes such bound is given by

$$q^*(h) = \frac{e^{-\Lambda \widetilde{\mathcal{L}}^{\ell}_{(X,Y)}(h)}p(h)}{\int e^{-\Lambda \widetilde{\mathcal{L}}^{\ell}_{(X,Y)}(h')}p(dh')} \tag{2.23}$$

where $\Lambda \propto N$. It is interesting to notice that this *posterior* acts as a *softmax* giving more probability mass to hypotheses that have low empirical loss proportionally to the importance of such hypothesis in the given *prior* distribution. However, it is worth noting that, even though it is call PAC-*Bayesian*, this *Gibbs posterior* does not correspond to a *Bayesian* posterior as it is not the solution of a probabilistic inference, in general. Instead, the *Gibbs* posterior is just the best probability distribution to measure the goodness of the hypotheses, based on the empirical error measures, to optimize the expected generalization error.

3 TRANSFER LEARNING IN REINFORCEMENT LEARNING

This chapter intends to provide an overview of the Transfer Learning setting within RL with the main objective to place our work within the state of the art. First, we provide the main motivation in *Transfer Learning* research. Then, we provide the considerations needed to design Transfer algorithms. Finally, we present the derived fields as a way to place better our contribution.

3.1 MOTIVATION

While RL offers a great framework for an agent to learn by interacting in a trial-error basis, it still requires a great amount of experience samples to learn optimally how to solve single tasks. Recent accomplishments within the field—e.g. agents learning to play the Atari game suite (Mnih et al., 2013), robots learning by imitation (Yu et al., 2018) and learning sensorimotor policies (Levine et al., 2016)—show the potential of RL and push forward the interest to have more scalable techniques in terms of quantity of samples and time required.

To obtain scalable algorithms, different solutions have been proposed such as Hierarchical Reinforcement Learning to solve tasks by breaking them in subtasks to which a solution exists or can be learned, and Imitation Learning as a way to guide the search for the optimal policy using expert demonstrations. There exist, also, state-action generalizations that allow RL to be more efficient in dealing with large, potentially continuous, state-action spaces. State-of-the-art algorithms tackle the problem by generalizing the sampled experience to areas of the state-action space not seen before, but that are close enough, in some sense, to hold certain similarity. Thus, function approximators are used to represent the main functions to be learned by the agent—e.g. Q-functions, policies. Taking a step further, a similar motivation exists in the core of *Transfer Learning*. In classic RL, when the agent is presented with a new task it is normally assumed that it would have to learn from scratch, that is, not knowing anything about the dynamics of the environment nor what it is suppose to be doing. However, a new task could still hold similarities with previously learned ones; the new task may only have (relatively) small changes in the environment dynamics, the reward signal and/or the state-action space. *Transfer Learning* looks for efficient exploitation of such similarities among tasks.

Furthermore, artificially intelligent agents that are able to interact with an environment and learn from it cannot just assume along their lifetimes that they would need to learn every new thing as if they have not tried anything before. Besides the fact that it is not scalable, biological learning systems from which AI takes inspiration are far from learning that way. Humans, as intelligent beings, learn structures that efficiently and effectively introduce inductive bias for any future task to be learned and it is reasonable to bring such ability to AI systems. For instance, to deploy intelligent agents in real physical environments to execute tasks, as it would be done with robotic rovers to explore Mars, it is not feasible to predict all possible events beforehand. However, if we could allow an agent to build upon the knowledge it has to adapt quickly enough to more complex, unseen environments, we could maximize the probability of the rover to succeed in its exploration objectives.

Nonetheless, this field poses many complex challenges and it is still very young and from which different approaches have been taken to answer different questions. For instance, what knowledge should we try to transfer? Should we create different structures that naturally enable transfer? How can the agent manage its past knowledge? How can the agent learn different tasks simultaneously?

Transfer Learning, then, encompasses a set of techniques that try to exploit learned information at many different levels and in a wide range of situations: experience samples (Tirinzoni et al., 2018; Lazaric, Restelli, and Bonarini, 2008), action abstractions/options and transferring policies (Fernández and Veloso, 2006, Konidaris and Barto, 2007), value functions (Lazaric and Ghavamzadeh, 2010), features (Barreto et al., 2017), etc. Moreover, it has given way to interesting subfields that attempt to answer many of the open questions in RL and AI.

3.2 TRANSFER ALGORITHM DESIGN

In the more general setting, we consider a *Transfer Learning* problem to be the situation in which we would like to use a set of solved tasks—the *source* tasks—to solve efficiently a new, unseen task—the *target* task—while minimizing negative transfer, that is, inductive bias that hinders the learning process altogether. Here, a task is a learning problem and, in our RL setting, we consider it to be an MDP. Some variations of this setting give way to new subfields that find use in different scenarios. However, this formulation is very general and when considering to transfer learned knowledge some choices must be made in order to fit the particular situation we are trying to solve. The thorough survey by Taylor and Stone, 2009 offers insights to organize the techniques in Transfer Learning in RL based on these required assumptions. Here, we present a small overview of their main ideas.

First, consider the scenario in which we wish to transfer the learned policy that allows a differential robot to reach a specified position, to train a legged-robot to complete the same task. We encounter that a main difference is in the action space of the robots and this fact is important for the transfer of knowledge to succeed. Another possible situation would be a legged-robot that has learned to reach a given position and, in the target task, the same robot must reach another goal position given a change in the reward. In the first one, we have to consider that even though the high-level goal is the same the different state-action space of the second robot makes the problem non trivial; while in the second situation, the change in reward makes a previous optimal policy *suboptimal* for the *target*. Therefore, we must consider the *tasks differences* among the *source* tasks and the *target* to solve. This is of main importance to the design of a transfer algorithm and it is an intrinsic characteristic of the problem that is being solved.

Second, the transfer algorithm must consider how the *source tasks selection* is done. It could be that a human designer chooses the set of sources in an optimal way to maximize the information transferred to solve that target. However, a more general situation is when the agent is allowed to select the best sources from which to transfer. This is a complex situation and the agent could incur in negative transfer, as bad choices could lead to performances worse than learning from scratch or even lead to the inability to learn to solve the target task.

Third, depending on the relationship between sources and target, there may be a need for *Tasks Mappings* to effectively transfer. In RL there may be a change between the action and state spaces among the tasks, such as in the example above, that require to specify a mapping that allow to *translate* the source solution to the *target*'s different state-action space. These mappings can be manually designed or learned. This *inter-task* transfer aims to distill high-level decision-making.

Finally, what *knowledge to transfer* is a choice that changes among many of the methods. Transfer can occur from very low levels, e.g. transferring experience samples (Tirinzoni et al., 2018; Lazaric, Restelli, and Bonarini, 2008), to higher levels solutions such as value function (Lazaric and Ghavamzadeh, 2010), policies and even the transfer heuristics in the form of actions abstractions (Fernández and Veloso, 2006, Konidaris and Barto, 2007) to solve particular situations. Deciding what to transfer is intrinsically related to how the source tasks and target tasks hold their similarities.

Lastly, it is worth noting that these design choices for *Transfer* algorithms are of main importance when analyzing new techniques and their applicability. Moreover, when proposing novel solutions, clearly defining these assumptions allows to understand better how to use them and the tasks to which the solution is useful. By the end of this chapter, then, we will specify the choices that guided the design of our approach.

3.3 RELATED FIELDS

Although the method that we present in this thesis is concerned with the simpler *Transfer Learning* scenario described before, it is worth noting the related, and even more complex, fields that hold in their core the interest to enable learning agents to exploit past knowledge efficiently and effectively. Hence, we present next: Multitask Learning (MTL), the Lifelong Learning and the Learning-to-Learn paradigms in a very succinct description. In this way, we aim to make clearer the particular situation our contributions focus on.

3.3.1 *Multitask Learning*

MTL is a particular setting of Transfer Learning in which an agent will be exposed to a set of tasks and it must learn them all. In this case, the assumption of having a set of already solved *sources* is relaxed and instead the agent must learn to solve a set of tasks.

It is common to assume that the tasks are drawn from a distribution, potentially unknown. This distribution implicitly encodes similarities among them. The agent must learn, then, how to solve efficiently the tasks that it will experience by using the commonalities and differences among those. Further, in this scenario, task selection is alleviated and the agent must automatically learn how to generalize among the tasks at hand.

Moreover, the learning procedure is normally expected to allow the agent to experiment in a given task for a period of time. During the learning process of an individual task, the agent should exploit the knowledge gotten so far from all the tasks in the set. In this way, if done efficiently and robustly to negative transfer, learning the whole set simultaneously would allow to generalize across the task more effectively and to learn them all together should be more efficient than learning them separately from scratch.

MTL, then, tackles the problem of transfer by efficiently managing correctly the past knowledge to use it, even when the solutions are still suboptimal, based on the common structure among the tasks. However, there are no restriction on what knowledge to transfer and, hence, algorithms in MTL exploit many objects such as value functions, policies and MDPs. Some examples will be seen in Chapter 4.

3.3.2 *Learning to Learn*

The main intuition behind the Learning-to-Learn paradigm is, again, to learn a *family* of tasks by assuming the existence of a probability distribution that characterizes it. However, the approach taken to solve this problem is different than the one of MTL. While MTL can assume the existence of the aforementioned distribution, the method might only focus on learning a finite set of tasks simultaneously. Instead Learning-to-Learn concentrates its efforts in avoiding the hand-design of an algorithm that effectively transfer the knowledge of tasks experienced before, the Learning-to-Learn approach aims to *learn an algorithm* to ensure fast learning of any task in the family.

When learning a single task, the agent is not able to learn how to generalize given that it learns behavior that optimally solve the one task and, intuitively, when we would try to directly use the found solution in a new sampled task, it would probably fail. This resembles an over-fitting effect, in general machine learning terms. Learning-to-learn, though, aims at generalizing, much as in supervised learning. Analogously, by supposing a distribution over tasks from which it is possible to require samples and an RL algorithm that "labels" the data points—the tasks and/or experience samples—with their solutions.

This last intuition coming from supervised learning is currently used in state-of-the-art approaches that, then, exploit extensively the knowledge of the distribution over tasks. They use sampled tasks to learn through some RL training algorithm an expressive representation of a general solution to the family of tasks by defining a higher-level objective to maximize the performance across the family and, thus, explicitly encourage generalization. When the agent is presented with a new task, the learned model is used to adapt quickly to any new task drawn from the distribution. In the literature state-of-the-art algorithms of this paradigm are, mainly, known as *Meta-Learning* such as in the works of Duan et al., 2016 and Finn, Abbeel, and Levine, 2017.

Learning-to-learn is an on-going research field and, mostly, assumes the possibility to sample many tasks in order to accurately learn a general solution. Although the core objective is the same as Transfer Learning, this field departs from the general transfer setting as described before. However, it still pushes towards answering *how to transfer* related knowledge and *how to efficiently manage knowledge* about the family of tasks.

3.3.3 *Lifelong Learning*

Lifelong Learning is a more complex paradigm in which an agent is supposed to learn many different tasks during their lifetime. In this situation, the MTL paradigm must be extended to consider more pressing issues needed to support a long-term commitment to continual learning. The main motivation for Lifelong Learning is to allow artificially intelligent agents to be put in an environment and continually build a knowledge base that is useful for all future tasks while experiencing different tasks/goals in a sequential manner.

Given that for practical implementations it is simply not possible to retain all the information collected, new structures must be devised in order to continually improve the generalization along the agent's lifetime. Therefore, this field aims to have Lifelong Learning agents that can distill learned information from particular tasks that allows to construct a shared knowledge base that effectively introduces the inductive bias for future learning and, even, improve performance in related past tasks.

This is a very complex setting that supersedes the Transfer Learning field and it aims to answer, in a more general way, *how to reuse past knowledge* effectively by investigating efficient structures to distill information useful for many different tasks that may hold their similarities at many different levels. The assumptions on what the agent might experience are relaxed given that during a lifetime many different family of tasks may be encountered. Further motivations and considerations for the Lifelong Learning paradigm can be found in works such as the one by Silver, Yang, and Li, 2013.

3.4 OUR SETTING

In this work, we focus our efforts in scenarios in which the past knowledge is limited but still we want to maximize the transferred information in order to reduce the sample complexity while being robust to negative learning. For which we set our research scope to the general Transfer Learning setting in RL. Although the related fields presented before are ambitious in what they aim to achieve, our main motivation and contribution is on achieving as much transfer as possible from a limited set of prior information; a situation that may hinder the application of more complex models such as Meta-learning and Lifelong Learning.

We consider, then, a set of *source tasks* that were solved optimally before and a *target* task. We assume the existence of a distribution $\mathcal{D}$ over tasks—more precisely, MDPs $\mathcal{M}_\tau$—though unknown to the agent. These $\mathcal{M}_\tau \sim \mathcal{D}$ maintain the same state-action space—hence, there is not need for inter-task mappings—letting the dynamics and reward signals from the environment to change such that a sampled MDP could be written as $\mathcal{M}_\tau = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}_\tau, \mathcal{R}_\tau, p_0, \gamma \rangle$.

Further, we consider the case in which we solve the source tasks using value-based RL methods and we are interested in transferring information about the optimal value functions. However, further access to experience the source tasks is not possible and only the knowledge of these optimal Q-function is available to use. These should provide a naturally sample-efficient approach to RL and they hold the required information to derive optimal decision-making behavior. In order to do this, we will consider the probability distribution over *optimal* Q-functions induced by the distribution $\mathcal{D}$ over tasks to model the transferable information. We defer further discussion to Chapter 5.

4 RELATED WORKS

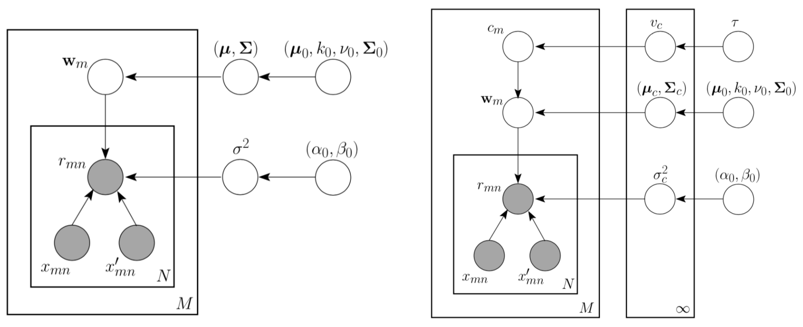
In this chapter, we present succinct descriptions of works done in the MTL field—as relevant ideas exploiting Bayesian techniques are presented in such works—and a final pair of techniques that will motivate further the design of our proposed algorithm.

4.1 HIERARCHICAL BAYESIAN MULTITASK RL

In order to solve the problem of MTL in RL from the Bayesian perspective, two main solutions take a Hierarchical Bayesian approach (Lazaric and Ghavamzadeh, 2010; Wilson et al., 2007). The authors, then, propose inference and sampling procedures to learn the priors that govern the relationship among the tasks. Additionally, they build a higher level within the hierarchy to account for the existence of different classes of tasks and leverage the Dirichlet Process formalities to learn through experience these different classes.

The method proposed by Lazaric and Ghavamzadeh, 2010 is the most related to the algorithm we present in this work. They focus on transferring *value functions* that are linearly parameterized using a set of basis functions and whose parameters are governed by a Multivariate Normal Distribution. They build their method on the Value Function decomposition proposed by Engel, Mannor, and Meir, 2005 in their work on Gaussian Processes for RL: $D(s) := V(s) + \Delta V(s)$ where $V(s)$ is a mean value function and $\Delta V(s)$ is a zero-mean residual. When combined with the Bellman Equation in Equation (2.7) and a fixed policy π they obtain that $R(s) = V(s) - \gamma V(s') + \epsilon(s,s')$ where $\epsilon(s,s') := \Delta V(s) - \gamma \Delta V(s')$ and $s' \sim P^{\pi}(\cdot|s)$, the transition kernel of the Markov Chain induced by following policy π. Then, they use such equations to learn from the experience samples obtained from the agent interaction with the environment.

Furthermore, they propose two algorithms: a single class MTL algorithm and a multiple class MTL algorithm. In Figure 4.1a, we see their hierarchical model for the single class case in which they consider the parameters of the class to be (μ, Σ, σ^2) for Gaussian distributions that govern the parameters $\mathbf{w}$ of the value function and the disturbance ϵ of the reward equation above. Using this hierarchy they propose the hyper-prior distributions over the parameters of those the Gaussians to be their classical conjugates—i.e. normal-inverse-Wishart and inverse-Gamma—to procure a tractable inference process using an Expectation-Maximization procedure.


$\mathbf{w}_m$
$(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
$(\boldsymbol{\mu}_0, k_0, \nu_0, \boldsymbol{\Sigma}_0)$
r_{mn}
σ^2
(α_0, β_0)
x_{mn}
x'_{mn}
N
M
c_m
v_c
τ
$(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$
σ_c^2
∞

(a) Single Class Hierarchical Bayesian MTL

(b) Multiple Class Hierarchical Bayesian MTL

Figure 4.1: Hierachical Bayesian Models— diagrams taken from (Lazaric and Ghavamzadeh, 2010)

The extension to multiple classes is shown in Figure 4.1b, where they add an extra level in the hierarchy to account for classes c of tasks, that, in turn, determine the parameters of the normal distributions that model the class of tasks—analogously to the single class case. To automatically discover classes of tasks from data they use a Dirichlet Process with concentration parameters τ and base distribution G_0. Finally, to solve the inference process problem they propose a Gibbs sampling procedure.

These methods, even though powerful, they constrain the parameters to be normally distributed to allow for tractable analytical solutions for the Bayesian inference. This, in the transfer setting, may be too limiting when trying to tackle complex distributions over the parameters of the tasks' value functions. Their multiple class model, however, is interesting as they could capture complicated scenarios in which the agent must solve quite different tasks during its lifetime.

We will see, nonetheless, that our method relaxes the assumptions of the distribution of the value function parameters, and the existence of different classes of tasks may be captured depending of the choice of the distribution families to involve in the algorithm.

Wilson et al., 2007, also, propose a Hierarchical Bayesian Model to tackle MTL by using a model-based approach to RL. They only consider the case of multiple classes of tasks to be presented to the agent. Through a construction similar to that described above, they use a Dirichlet Process to discover classes of tasks directly from data. They group, then, classes of MDPs, as opposed to Value Functions as before, whose parameters are then determined by a prior distribution that depends on the given class. Additionally, they propose a Gibbs sampling procedure for their hierarchical model. They finally solve an MDP by sampling the distribution over tasks given the model parameters and the observations, and use the optimal policy of the sampled

MDP to collect new observations and update the belief of the task—and its class—being solved.

This last work, even though, it does not transfer value functions to solve the MTL problem, it still tries to discover structure by performing Bayesian inferences from the observations as we will do when presenting the motivation to our approach.

4.2 HIDDEN-PARAMETER MDPS

The work of Doshi-Velez and Konidaris, 2016 and its subsequent extension by Killian et al., 2017 propose the Hidden-Parameter MDP in which they extend the classical definition of an MDP by adding a set W of parameters and a probability distribution P_W, over the set W, to the tuple defining the MDP. In this way, a Hidden-Parameter MDP defines a *family* of tasks by supposing that a task is completely defined by the latent parameter $w \in W$, that parameterizes the transition probabilities of the MDP.

The work in Killian et al., 2017 uses this definition to design a model-based RL algorithm that efficiently allows to quickly adapt to a family of tasks defined by a Hidden-Parameter MDP. They use a Bayesian Neural Network (BNN) to learn an approximation to the transition dynamics $\widehat{T}(s,a,w)$ of the environment as a function of the state, the action and the latent parameter w of the task. Moreover, they assume a Gaussian distribution over the hidden parameters and a Gaussian perturbation in the transitions, i.e.

$$\begin{aligned} s' &\approx \widehat{T}^{(BNN)}(s,a,w) + \epsilon, \\ w &\sim \mathcal{N}(\mu_w, \Sigma_w), \\ \epsilon &\sim \mathcal{N}(0, \sigma_n^2). \end{aligned}$$

The use of BNN let the algorithm to represent complex dynamics and to perform efficient Bayesian Inference over the BNN parameters, in order to adapt quickly to a specific instance of the family of tasks. However, training a powerful model such as a BNN to represent intricate dynamics could require a great amount of samples and, in many situations, become impractical. Nonetheless, whenever the data is available and the correct model for the BNN is chosen, the Hidden-Parameter is a strong framework for generalizing among the tasks of the family.

4.3 RANDOMIZED VALUE FUNCTIONS

Thompson sampling, or posterior sampling, proposed in Thompson, 1933 is one of the heuristics in state-of-the-art solutions in managing the exploration-exploitation dilemma. This, roughly, consists on maintaining a probability distribution to measure the uncertainty about the quality of an action and manage the trade-off between exploration and exploitation by stochastically choosing actions proportionally to their expected return and their uncertainty. With each new sample, the procedure updates the belief—represented by the posterior distribution—of the quality of the actions taken. In RL, works such as Osband, Russo, and Van Roy, 2013, try to efficiently explore by leveraging the Thompson sampling by estimating a posterior distribution over MDPs.

More recently, however, Osband, Van Roy, and Wen, 2014 propose to use Thompson sampling to randomize the value function. They extend Least Squares Value Iteration (LSVI) by using a Bayesian Linear Regressor. In this way, to take an action the parameters of the value function are sampled from the current estimation of the posterior distribution and, using this, a greedy action is taken. They prove that heuristics for exploration such as ϵ-greedy and Boltzmann softmax policies result in exponential regret, while they prove, for discrete MDPs though, that their Randomized LSVI attains regret $O\left(\sqrt{H^3SAT}\right)$ where H is the length of the episodes, S and A are the cardinalities of the state and action spaces, respectively; and T is the time elapsed during the learning algorithm.

Additionally, Azizzadenesheli, Brunskill, and Anandkumar, 2018 leverage on such work and extend Double Deep Q-Network (DDQN) (Van Hasselt, Guez, and Silver, 2016) by implementing the last layer of the network as a Bayesian Linear Regressor, much in the spirit of Randomized LSVI above, using the Deep Neural Network, before the last layer, to produce the features that will be linearly combined to produce the Q-values. They, then, keep a posterior distribution over the parameters of the last layer—that is updated when a predefined number of time-steps has passed. The agent, then, acts by behaving greedily from the Q-values obtained when the parameters are sampled from the estimation of the posterior distribution.

Both of these implementations of RL algorithms derived from the randomization of value functions suggest efficient exploration in the RL scenario and they show promising results thus far. However, these assume Gaussian distributions over the parameters of the regressors that make their inferences tractable; an assumption that, we argue, is too limiting in the transfer setting. In Chapter 5, when we present our approach, it will be clearer how value function randomization is in the core of our algorithms.

5 VARIATIONAL TRANSFER LEARNING

Across this chapter we present our proposed approach to *Transfer Learning* in RL. We start by presenting a model-agnostic version of the algorithm—i.e. without specifying specific forms for the objects involved, only requiring that they are *parameterized* and *differentiable*. We continue to provide the motivation for such design and, finally, we present our two main practical implementations based on Gaussian Distributions and Mixture of Gaussians.

5.1 VARIATIONAL TRANSFER ALGORITHM

In the *Transfer Learning* setting, recall Section 3.4, we suppose to have a distribution $\mathcal{D}$ over possible tasks—i.e. MDPs—such that the learner will solve tasks $\mathcal{M}_\tau \sim \mathcal{D}$. Additionally, we consider the situation in which the agent has already solved a set of such tasks optimally, the set of *source* tasks, by finding the parameters w of their optimal Q-function.

In our current setting, where we suppose to have such a distribution $\mathcal{D}$ over tasks, though unknown, it induces a distribution over optimal Q-functions, more precisely—in our setting—over the parameter space $p(w)$, given that for each task its optimal Q-function is unique—as seen in Section 2.1.1—and structured variations in the dynamics and reward of the MDPs must change the Q-function also in a structured manner. Leveraging the optimal Q-functions' parameters distribution $p(w)$, though it causes a loss of information w.r.t. learning about the MDPs directly, as done in previous works such as the ones presented in Section 4.1, it still provides enough information about the solutions of the tasks. Our aim is to efficiently exploit the knowledge from the *sources'* optimal Q-functions to induce behavior that is good and explorative in the *target* task to adapt the parameters to the optimal solution with a lower amount of experience samples.

From a Bayesian perspective, what we would like to infer is which Q-functions are more likely given that the agent knows $p(w)$ and it is given a *target* task to solve—drawn from the task distribution $\mathcal{D}$ mentioned above. Further, we suppose the agent to have a set of i.i.d. experience samples $D = \langle s_i, a_i, r_i, s'_i \rangle_{i=1}^{N}$ from the *target* task. What we would like to infer is, then, the *posterior* probability distribution over the optimal Q-function parameters $P(w|D) \propto P(D|w)P(w)$ using Bayes Theorem.

Nonetheless, this Bayesian Inference poses many challenges. Although Bayesian techniques are very appealing, the methods derived tend to be computationally expensive and they quickly become infeasible. For instance, state-of-the-art techniques are taking advantage of the representational power of deep Neural Network architectures to learn highly complex tasks in RL, such as the recent advancements mentioned in Section 3.1. However, the deeper these models get, the bigger the number of parameters they require and the more involved the functional relationship among parameters—in general—get. Therefore, current Bayesian techniques that constrain the prior and posterior distribution to get tractable densities such as Normal Distributions are of little use as they can only solve a very limited set of small problems. Moreover, the greater the number of parameters, when using prior/posterior density models that do not allow to have close formulas to compute the posterior density, the more expensive is to numerically compute the results to the point of becoming impractical.

In order to design a useful Bayesian method to transfer information of the optimal Q-functions, first, we discuss how to model the likelihood $P(D|w)$ and, second, how to tractably estimate the *posterior* and *prior* distributions from the source tasks and the experience samples from the *target* task without over-constraining the form of the densities concerned.

5.1.1 *Modeling the Likelihood*

The likelihood $P(D|w)$ of the dataset D of i.i.d. experience samples given parameters w should measure how likely is for such dataset to be generated by a task in which the parameters w correspond to the ones of its optimal Q-function.

As we know that the Bellman Residual $B_w(s,a)$ must be zero almost everywhere whenever the parameters w represent the optimal Q-function of the task, if we consider the empirical estimation $\left\|\widetilde{B}_w\right\|_{D,p}^{p}$ of the error, it is reasonable to require the likelihood of such a dataset D to be smaller whenever $\left\|\widetilde{B}_w\right\|_{D,p}^{p}$ is further from zero. However, as we would use a finite set of experience samples our empirical estimation of the Bellman Residual, apart from biased, would have some uncertainty given the limited number of samples. The likelihood, then, must also depend on the number N of samples as this should provide more certainty in our criterion to reject certain parameters w as non-optimal based on this TD error $\left\|\widetilde{B}_w\right\|_{D,p}^{p}$. That is, as N grows to infinity we should converge to an indicator function that tells whether the parameter w are optimal or not.

A natural choice for such probabilistic density would be to consider it proportional to a negative exponential of the form $P(D|w) \propto e^{-N\|B_w\|_{D,p}^{p}}$, in this way as N grows, the probability mass will concentrate (exponentially more) in the weights that corresponds to lower TD errors and it will converge with all the probability mass over the optimal parameters in the limit of infinite samples.

To more formally motivate a negative exponential form for the likelihood, we suppose, additionally, the existence of a probability measure ν over the state-action space $\mathcal{S} \times \mathcal{A}$ that weighs the relative importance of the state-action tuples in the task. Then, we consider the $L_{\nu,p}$-norm of the Bellman Error $\|B_w\|_{\nu,p}^p$ that would be zero when w represents the optimal Q-function for the target task.

Now, we consider the probability $P(w \in \mathcal{Q}_\epsilon^* | D)$ that a Q-function is in the set $\mathcal{Q}_\epsilon^* = \left\{ w \in \mathcal{Q} \mid \|B_w\|_{\nu,p}^p \leq \epsilon \right\}$ with $\epsilon > 0$, that is, the probability of a q function is ϵ-optimal given the dataset D. By applying Hoeffding's inequality we get

$$\begin{aligned} P(w \in \mathcal{Q}_\epsilon^* \mid D) &= P(\|B_w\|_{\nu,p}^p \leq \epsilon \mid D) \\ &= P\left(\left| \|B_w\|_{\nu,p}^p - \|\widetilde{B}_w\|_{D,p}^p \right| \leq \left| \epsilon - \|\widetilde{B}_w\|_{D,p}^p \right| \right) \\ &\leq e^{\mathcal{O}\left[-N \left| \|\widetilde{B}_w\|_{D,p}^p - \epsilon \right| \right]}, \end{aligned}$$

from which we obtain our proposed form when we let ϵ tend to zero.

However, a stronger argument comes from the PAC-Bayesian Theory. As explained in Section 2.3, we can infer information of the *generalization* error $\mathcal{L}_{\mathrm{D}}^{\ell}(h)$ of an hypothesis h from the empirical risk $\widetilde{\mathcal{L}}_{(X,Y)}^{\ell}$ when we consider that our algorithm outputs a *posterior* probability distribution q, over the hypotheses set, when given as input a dataset and a *prior* distribution over the hypotheses.

In our particular case, we consider our *generalization error* to be the Bellman Error $\|B_w\|_{\nu,p}^p$ and its empirical estimation to be the average TD error $\left\|\widetilde{B}_w\right\|_D^p$, the biased estimator defined at the end of Section 2.1.1. Moreover, we consider our set of hypotheses to be the set of parameterized Q-functions $\mathcal{Q}$. Drawing from the results presented in Section 2.3 we have that our *Optimal Gibbs Posterior* is given by

$$q^*(w) = \frac{e^{-\Lambda \|\widetilde{B}_w\|_{D,p}^p} p(w)}{\int e^{-\Lambda \|\widetilde{B}_{w'}\|_{D,p}^p} p(dw')}, \tag{5.1}$$

from which we can immediately notice the same exponential form for the likelihood intuited at the beginning of this section. However, drawing from PAC-Bayesian results, we are guaranteed that we are choosing a form that would optimize also the *generalization* error of our learning algorithm.

Finally, as suggested by the Theorem 2.4, we choose Λ to increase with the sample size $\Lambda = \lambda^{-1}N$, where the role of λ would be clearer later.

5.1.2 *Variational Approximation*

We recognize that the transfer settings are varied and, hence, the distributions that would govern the probability over the parameters of the optimal Q-functions that we aim to model can be of many complex forms. In order to avoid strong assumptions about the *prior* and *posterior* distributions that would limit the potential of our method, we propose to use Variational Inference methods to approximately compute our *posterior* distribution.

For this we consider that we have a parameterized family of distributions, such that its variational parameters are $\xi \in \Xi$ and, as presented in Section 2.2, we intend to find the best variational approximation to the actual *posterior* distribution by solving the optimization problem in Equation (5.2) by maximizing the ELBO, where we consider the *likelihood* to take the exponential form discussed before. From hereon, we will consider only the squared L_2-norm of the average TD error.

$$\max_{\xi\in\Xi} \mathop{\mathbb{E}}_{w\sim q_\xi}\left[-\frac{N}{\lambda}\|\widetilde{B}_w\|_D^2\right] - KL\left(q_\xi(w)\|p(w)\right) \quad \equiv$$

$$\min_{\xi\in\Xi} \mathcal{L}(\xi) = \mathop{\mathbb{E}}_{w\sim q_\xi}\left[\|\widetilde{B}_w\|_D^2\right] + \frac{\lambda}{N}KL\left(q_\xi(w)\|p(w)\right). \tag{5.2}$$

Equation (5.2) shows that our approach is based on managing a trade-off between our prior knowledge coming from the previously learned tasks and the reduction of the expected TD error—under the estimated posterior—given the experience drawn from the task being learned. Therefore, the *hyperparameter* λ is introduced as a way to tune such trade-off. Recall, also, that this follows both the intuitive idea that the importance of the prior knowledge is greater whenever the agent knows little about the current task, and the more formal idea coming from the bound on the *generalization* error from PAC-Bayesian theory, as it can be clearly seen in the objective $\mathcal{L}(\xi)$ in Equation (5.2).

5.1.3 *Algorithm*

In this section we present the algorithm designed from the problem in Equation (5.2). We start by explaining the procedure in a general way, such that we are able to highlight important characteristics of the learning algorithm while avoiding to constrain the approach with many assumptions. In this way, we maintain the design approach agnostic to the choice of models for the prior and posterior distributions and we leave the discussion of two practical implementations with specific models for the last two sections of this chapter.

To solve the variational optimization, we assume that the function approximators for the Q-functions are *differentiable* with respect to the

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: $\boldsymbol{\xi} \leftarrow \operatorname{argmin}_{\xi} KL(q_{\boldsymbol{\xi}}||p)$
 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 $\nabla_{\xi} \mathcal{L}(\boldsymbol{\xi})$ using $D'$
13:     Update $\boldsymbol{\xi}$ in the direction of $-\nabla_{\xi} \mathcal{L}(\boldsymbol{\xi})$ using any stochastic
        optimizer (e.g., ADAM)
14:   end while
15: until forever
```

parameters w and the family of distributions to approximate the posterior are also *differentiable* in its variational parameters ξ. Further, we assume that the choices of function approximators and distribution families allow to tractably compute the gradients of the KL divergence and the expectations involved.

In Algorithm 1, we present the *Variational Transfer* algorithm. As a preamble to the main procedure we consider the estimation of the prior $p(w)$ from the set of source weights $\mathcal{W}_s$ and the estimation of the best approximation of such prior by minimizing the KL divergence. In this way, the specific forms of the *prior* and *posterior* are left to specific implementations of the algorithm. Moreover, we introduce a replay buffer to hold the samples obtained during the learning process and to allow to discard samples when they are old enough.

Next, the agent starts to act within the *target* task by leveraging its information from the distribution over optimal Q-functions. This is done using a *Thompson Sampling* approach, that is, we sample from the current estimation of the posterior distribution the parameters (line 7) that determine the Q-function that the agent will use to act greedily (line 8). In this way, we use our certainty estimation of the *good* Q-functions to guide the *exploration* of the target task. This is a core step of the algorithm; the agent is expected to sample *sufficiently good* Q-functions and, therefore, it should act, on average, better than when oblivious of the family of tasks that is facing. When too uncertain, namely at the beginning of the learning process, the intrinsic randomness will motivate the agent to take, most likely, actions that were good in the previous tasks—which will account for higher returns while learning—and take enough, perhaps, *not so good* actions to explore the target. However, as the learning process continues, the probability mass is expected to move towards the *optimal* Q-function for the target task and, hence, the amount of exploration will reduce naturally when the certainty of the agent improves.

After the agent takes its decision, it performs the action and it adds the new experience sample to the replay buffer and, finally, estimates the gradient of the optimization objective $\mathcal{L}(\xi)$ using a randomly sampled mini-batch of experience to update the variational parameters ξ using some stochastic optimizer (lines 9-13).

5.1.4 *TD error optimization*

Finally, another important point to notice is that by using Stochastic Gradient Descent (SGD) in this algorithm, we will be constrained to differentiate through the square TD error $\left\|\widetilde{B}_w\right\|_D^2$. However, recall that this is defined based on the optimal Bellman Operator $\mathcal{T}$ and, hence, it will require to differentiate through the max operator. This is clearly a problem that has been faced many times in the RL literature and many proposed solutions exist, being the most popular to use semi-gradient techniques. In these, the part of the TD error corresponding to the application of the empirical Bellman Operator, i.e. $r_i + \max_{a' \in \mathcal{A}} Q_w(s_i', a')$, are considered independent from w; thus, not differentiated. This, however, is known to lead to instabilities in the search which might hinder the potential advantages of our approach given that we intend to exploit already, probably, good Q-functions and such an instability may lead the algorithm to lose this knowledge.

To solve this issue, we propose to use a relaxation of the Optimal Bellman Operator that we introduce next, which is, among other characteristics, differentiable.

As seen in Section 2.1.1, to find optimal deterministic policies in Value-based RL, we depend on the use of the Optimal Bellman Operator $\mathcal{T}^*$ which has the property of having a *unique* fixed-point because of the non-expansion of the max operator.

Here, we introduce the softened max operator—defined by Asadi and Littman, 2017—coined *Mellowmax*.

Definition 5.1. Mellowmax

Consider a constant $\kappa \in \mathbb{R}_+$. Given a function $f : X \to \mathbb{R}$ where X is a finite set. The *Mellomax* operator is defined as,

$$\underset{x \in X}{\mathrm{mm}} f(x) := \frac{1}{\kappa} \log \left(\frac{1}{|X|} \sum_{x \in X} e^{\kappa f(x)} \right), \tag{5.3}$$

which is *differentiable* and the authors in Asadi and Littman, 2017 proved that is a non expansion and that when $\kappa \to \infty$ then mm $\to$ max. Moreover, they also proved that by substituting the max operator in the Optimal Bellman Operator $\mathcal{T}^*$ with the *Mellowmax* we obtain a new *softened* Optimal Bellman Operator $\widetilde{\mathcal{T}}^*$ that also has a *unique* fixed-point which we will call the *Mellow Bellman Operator*.

We will consider, hereafter, that when we mention the average TD error and Bellman Error, they will be the ones using the Mellow Bellman Operator. Nevertheless, Algorithm 1 remains the same independently on how we decide to optimize the expected square TD error of our objective.

Although, using the full gradients of the square TD error based on the Mellow Bellman Operator is ideal, we know that fitted RL algorithms with full gradients, even though stable, are to be slower than the ones based on semi-gradients used extensively in the literature. The authors in (Baird, 1995) propose to balance these two directions, i.e. the full gradient and the semi-gradient, by introducing a *hyper-parameter* to handle the trade-off between stability and convergence speed by controlling a convex combination of such directions and, thus, it is used to find the direction of fastest *guaranteed* convergence. We name this parameter $\psi \in [0,1]$ from which we write the gradient of the squared TD error as

$$\nabla_w \left\| \widetilde{B}_w \right\|_D^2 = \frac{2}{N} \sum_{i=1}^{N} \widetilde{b}_i(w) \left(\gamma \psi \nabla_w \operatorname*{mm}_{a' \in \mathcal{A}} Q_w(s_i, a') - \nabla_w Q_w(s_i, a_i) \right), \tag{5.4}$$

where $\widetilde{b}_i(w)$ is defined as the TD in Section 2.1.1 using the mellowmax operator instead: $\widetilde{b}_i(w) = r_i + \gamma \, \mathrm{mm}_{a' \in \mathcal{A}} Q_w(s_i, a') - Q_w(s_i, a_i)$. Notice that we recover the full gradient and the semi-gradient in the extremes of the range of ψ ($\psi = 1$ and $\psi = 0$, respectively).

5.2 PRACTICAL IMPLEMENTATIONS

In the following sections we propose two implementation of this algorithm by using Gaussian and Mixture of Gaussians as distribution families.

5.2.1 *Gaussian Variational Transfer*

The first practical implementation that we propose is based on Gaussians distributions as these are pervasive throughout the literature and, also, these provide closed-form solutions to many of the quantities involved and allow for an efficient implementation of the algorithm. We coin such implementation as Gaussian Variational Transfer

(GVT) when we set the variational family and the prior distribution to be *Multivariate Gaussians*.

Therefore, we have that the variational parameters are $\Xi = \{(\mu, \Sigma) \mid \mu \in \mathbb{R}^d \wedge \Sigma \in \mathbb{R}^{d \times d} \wedge \Sigma \succ 0\}$. As for the prior distribution $p(w) = \mathcal{N}(\mu_p, \Sigma_p)$ whose parameters we estimate from the set of source weights $\mathcal{W}_s$ by maximum likelihood estimation with Equation (5.5) and Equation (5.6). The dimension d is that of the parameters of the function approximator used.

$$\mu_p = \frac{1}{|\mathcal{W}_s|} \sum_{w \in \mathcal{W}_s} w \tag{5.5}$$

$$\Sigma_p = \frac{1}{|\mathcal{W}_s|} \sum_{w \in \mathcal{W}_s} (w - \mu_p)(w - \mu_p)^T \tag{5.6}$$

In this setting, we have that the KL divergence can be computed in closed-form by the well-known formula

$$KL\left(q_\xi \| p\right) = \frac{1}{2}\left(\log \frac{|\Sigma_p|}{|\Sigma|} + Tr(\Sigma_p^{-1}\Sigma) + (\mu - \mu_p)^T \Sigma_p^{-1}(\mu - \mu_p) - d\right) \tag{5.7}$$

and from which we derive the gradients—notice that we consider the Cholesky factors of the covariance matrices $\Sigma = LL^T$ instead, as to avoid learning non-positive definite matrices.

$$\nabla_\mu KL(q_\xi \| p) = \Sigma_p^{-1}(\mu - \mu_p) \tag{5.8}$$

$$\nabla_L KL(q_\xi \| p) = \Sigma_p^{-1} L - L^{-T} \tag{5.9}$$

To compute the gradient of the expected square TD error, we use the reparameterization trick (Kingma and Welling, 2013) to obtain the derivative w.r.t. the variational parameters by differentiating within the expectation w.r.t. the function approximator parameters as shown in the following equations.

$$\nabla_\mu \mathop{\mathbb{E}}_{w \sim \mathcal{N}(\mu, LL^T)} \left[\left\|\widetilde{B}_w\right\|_D^2\right] = \mathop{\mathbb{E}}_{v \sim \mathcal{N}(\bar{0}, I)} \left[\nabla_w \left\|\widetilde{B}_w\right\|_D^2\right], \tag{5.10}$$

$$\nabla_L \mathop{\mathbb{E}}_{w \sim \mathcal{N}(\mu, LL^T)} \left[\left\|\widetilde{B}_w\right\|_D^2\right] = \mathop{\mathbb{E}}_{v \sim \mathcal{N}(\bar{0}, I)} \left[\nabla_w \left\|\widetilde{B}_w\right\|_D^2 \cdot v^T\right], \tag{5.11}$$

where $w = Lv + \mu$ and the expectations are to be estimated by an empirical average with the v's sampled from an standard Gaussian distribution.

5.2.2 *Mixture of Gaussians Variational Transfer*

Although the Gaussian assumption of GVT is appealing for its simplicity, we argue that it is much limited when it ought to model a complex distribution as the one followed by the Optimal Q-functions of the tasks. Here, we propose to use a family of *Mixtures* of equally-weighted isotropic Gaussians for the prior distribution
$p(w) = \frac{1}{|\mathcal{W}_s|}\sum_{j=1}^{|\mathcal{W}_s|}\mathcal{N}(w|w_j, \sigma_p^2 I)$ with each component centered in a source weight w_j and a fixed variance σ_p^2. Notice that this prior form resembles the non-parametric kernel density estimator based on Gaussian kernels.

As for the variational family we propose a *Mixture* of equally weighted Gaussians with C components $q_\xi(w) = \frac{1}{C}\sum_{i=1}^{C}\mathcal{N}(w|\mu_i, \Sigma_i)$ where the variational parameters would be $\Xi = \{(\mu_1, ..., \mu_C, \Sigma_1, ..., \Sigma_C) \mid \mu_i \in \mathbb{R}^d \;\wedge\; \Sigma_i \in \mathbb{R}^{d\times d} \;\wedge\; \Sigma_i \succ 0 \quad \forall i = 1, ..., C\}$ where we let the normal components to have full covariances, in general, to be able to represent complex densities even when a small number of components C is used. When using these families of distributions, we refer to the algorithm as Mixture of Gaussians Variational Transfer (MGVT).

However, this choice poses in itself some additional implementation issues. With these complex families, we do not have closed-form formulas to compute the KL divergence between Mixtures of Gaussians. In order to solve this problem, we propose to use the following upper-bound on the KL divergence that computes, using variational methods, a tight upper-bound based on the pairwise KL divergences of the components as proposed by Hershey and Olsen, 2007. We present here their result,

Theorem 5.1 (Variational Upper-Bound for the KL divergence between Mixture of Gaussians from Hershey and Olsen, 2007)**.** Let $p = \sum_i c_i^{(p)} f_i^{(p)}$ and $q = \sum_j c_j^{(q)} f_j^{(q)}$ be two Mixtures of Gaussians where $f_i^{(p)} = \mathcal{N}(\mu_i^{(p)}, \Sigma_i^{(p)})$ is the i-th component and $c_i^{(p)}$ its corresponding weight—analogously, for the distribution q. Consider the variational parameters $\varphi_{i,j}$ and $\vartheta_{j,i}$ such that $c_i^{(p)} = \sum_j \vartheta_{j,i}$ and $c_j^{(q)} = \sum_i \varphi_{i,j}$, then:

$$KL(p\|q) \le KL(\vartheta\|\varphi) + \sum_{i,j}\vartheta_{j,i} KL\left(f_i^{(p)}\|f_j^{(q)}\right) \tag{5.12}$$

where ϑ and φ corresponds to the vectors of parameters $\vartheta_{j,i}$ and $\varphi_{i,j}$ and $KL(\vartheta\|\varphi) = \sum_{i,j}\vartheta_{j,i}\log\frac{\vartheta_{j,i}}{\varphi_{i,j}}$.

In this way, we replace the original KL with this upper-bound and compute its gradient, leveraging the linearity of the gradient, using the formula for the Gaussian case as in Equation (5.10) and Equation (5.11). Our upper-bound on the KL is, then,

$$KL\left(q_{\xi}\|p\right) \leq KL\left(\vartheta\|\varphi\right) + \sum_{i=1}^{C}\sum_{j=1}^{|\mathcal{W}_s|} \vartheta_{j,i} KL\left(\mathcal{N}\left(w|\mu_i,\Sigma_i\right)\|\mathcal{N}\left(w|w_j,\sigma_p^2 I\right)\right). \tag{5.13}$$

The authors in Hershey and Olsen, 2007, also propose a fixed-point optimization method to find the best variational parameters ϑ and φ by iteratively computing their values using the following equations—notice that, here, we also substitute the parameters by those of our distribution families—

$$\varphi_{i,j} = \frac{c_j^{(q)}\vartheta_{j,i}}{\sum_{i'}\vartheta_{j,i'}} = \frac{1}{|\mathcal{W}_s|}\frac{\vartheta_{j,i}}{\sum_{i'}\vartheta_{j,i'}}, \tag{5.14}$$

$$\begin{aligned}\vartheta_{j,i} &= \frac{c_i^{(p)}\varphi_{i,j} e^{-KL(f_i^{(p)}\|f_j^{(p)})}}{\sum_{j'}\varphi_{i,j'} e^{-KL(f_i^{(p)}\|f_{j'}^{(p)})}} \\ &= \frac{1}{C}\frac{\varphi_{i,j} e^{-KL\left(\mathcal{N}(w|\mu_i,\Sigma_i)\|\mathcal{N}\left(w|w_j,\sigma_p^2 I\right)\right)}}{\sum_{j'}\varphi_{i,j'} e^{-KL\left(\mathcal{N}(w|\mu_i,\Sigma_i)\|\mathcal{N}\left(w|w_j,\sigma_p^2 I\right)\right)}},\end{aligned} \tag{5.15}$$

from where we can observe that the new variational parameters of this upper-bound modify the actual weights of our mixtures by weighting their importance based on their pairwise KL divergences using a Boltzmann-like softmin.

Given this, we add new steps to the general Algorithm 1 to initialize the variational parameters of the upper-bound in the preamble and, just before estimating the gradient of our objective, we re-optimize these parameters to tight the upper-bound.

The generality of the procedure for MGVT allows to have a very powerful prior represented by the complete set $\mathcal{W}_s$ of source tasks and use different quantity of components for the posterior. The resemblance of the prior distribution to a kernel density estimator allows to model arbitrarily complex distributions over the optimal Q-functions and it should permit to successfully exploit much of the information required.

6 EMPIRICAL EVALUATION

In this chapter, we present the evaluation of our algorithms—as defined in Section 5.2.1 and Section 5.2.2.

We chose four different domains to evaluate the behavior and the performance of both of them. The domains are presented, hereafter, in increasing complexity as to leverage simpler scenarios to explain clearly the studied properties of the algorithms. Across this chapter when referencing the MGVT algorithm with c components for the posterior distribution family, we will use c-MGVT.

For all the experiments, we provide a comparison between the performance of an RL algorithm exploiting ϵ-greedy exploration and the performance of our transfer methods.

6.1 THE ROOMS PROBLEM

The first evaluation environment, shown in Figure 6.1, consists of an agent navigating a continuous space of dimension 10×10, starting at the bottom left corner of the room with the objective to arrive in the goal space—within a distance 1 from the top right corner. The environment is divided in rooms that are separated by walls with a door of width 1. The agent must navigate through the rooms to arrive to the goal position. It has a finite set of possible actions: UP, RIGHT, DOWN, LEFT. These make the agent move within the space by a distance of 1 corrupted by Gaussian White Noise $\mathcal{N}(0, 0.2)$ to simulate precision errors in the translation of the agent. Moreover, if the agent hits any of the walls, i.e. the walls separating the rooms or the frontiers of the environment, the position of the agent does not change. The agent is rewarded with 1 when it reaches the goal area and with 0 in any other case. Finally, the discount factor is $\gamma = 0.99$. As function

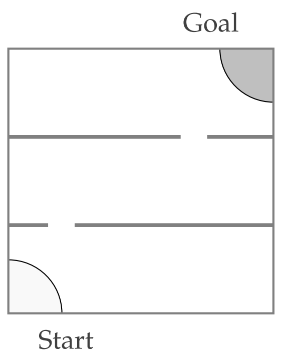


Figure 6.1: The Rooms Environment

approximator for the Q-function, we use a linear regressor with 121 radial bases as features, each centered in a integer coordinate of the environment. Each radial basis is an Isotropic Gaussian with a bandwidth of $1/9$ in order to allow a basis to reach the maximum when the contiguous bases are negligible and, thus, to ensure non-redundant covering of the state-action space.

As for the distribution over tasks, we sample tasks of this environment by choosing uniformly and independently the position of the doors in the range $[0.5, 9.5]$.

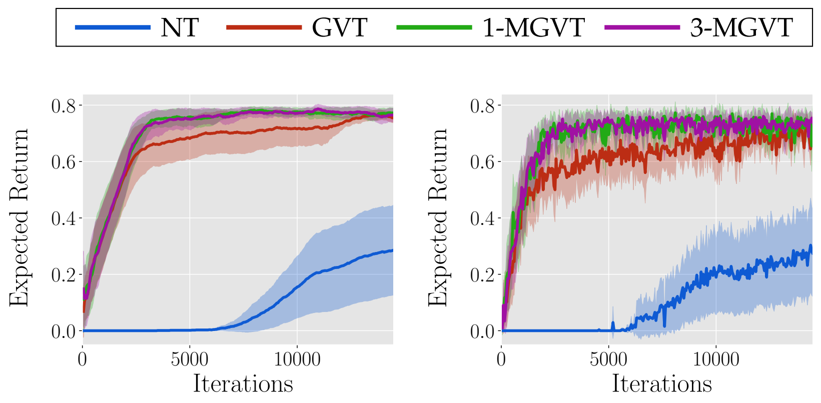
The Rooms environment, while simple, is one that allows for a particularly interesting study of transferring Q-functions. The presence of two different rooms, while keeping the goal fixed, provide the presence of different degree of variations of the Q-values when the position of the doors change: the top room has Q-values that remain invariant across the different tasks, whereas the middle and bottom rooms vary their values since different door positions change the distance—thus, time steps required—the agent has to traverse to arrive in the goal, with the middle room only depending on the position of the second door and the bottom room on the position of both doors. This, easily observable, structure in the Q-functions induced by the distribution over the tasks is what our transfer algorithms intend to exploit. Furthermore, as we design our algorithm to use the estimated distribution over optimal Q-functions to induce exploration, we also expect this environment to expose a well-structured exploration, that is, we expect that the agent, mostly, should look for the doors in order to go through towards the goal.

In this section we start by evaluating the performance of the different implementations of the *Variational Transfer* algorithm when compared to the no-transfer. Next, we investigate how it behaves when there are variations in the task distribution used to draw *target* tasks to solve. We empirically evaluate how the algorithms induce structured exploration that it is not possible with heuristics such as ϵ-greedy. Finally, we present how the algorithms are affected by the likelihood of the tasks and the different number of sources given.

6.1.1 *Evaluation*

To evaluate the GVT, the 1-MGVT and the 3-MGVT algorithms, first, we generate a set of 50 source tasks by sampling the position of both doors, as described before, and solve them by minimizing the TD error directly using SGD as described in Section 5.1.4. Next, we independently run the algorithms 20 times such that for each run we sample a subset of 10 source tasks—from the sources trained in advance. We evaluate the performance of the learning by averaging the return of the last 50 learning episodes as shown in Figure 6.2a and by evaluating the greedy policy induced by the expected Q-function as shown in Figure 6.2b—both plots show in shades a 95% confidence interval of the estimation.

In Figure 6.2, we can easily notice the obvious advantage in using the transfer algorithms—in terms of performance—w.r.t. the no-transfer (NT) performance: after 7K iteration, the NT is just starting to learn how to reach the goal while, approximately after 5K iterations, the MGVT performances are already optimal and the GVT algorithm has probably solved optimally the majority of the tasks. This behavior is as expected, since in the NT case the limited ϵ-greedy exploration makes it hard to find the goal in this little number of iterations. However, the transfer algorithms are able to effectively exploit the source tasks information and achieve the speed-up observed. Furthermore, MGVT is able to slightly surpass the performance of GVT, offering faster convergence seen in the reduced variance and the better mean return. Finally, a higher number of components in MGVT does not show any clear advantage; these components are supposed to converge to the optimal Q-function for the target task, for this environment they quickly converge to the same region and no significant difference is attained.



(a) Expected Return during Learning

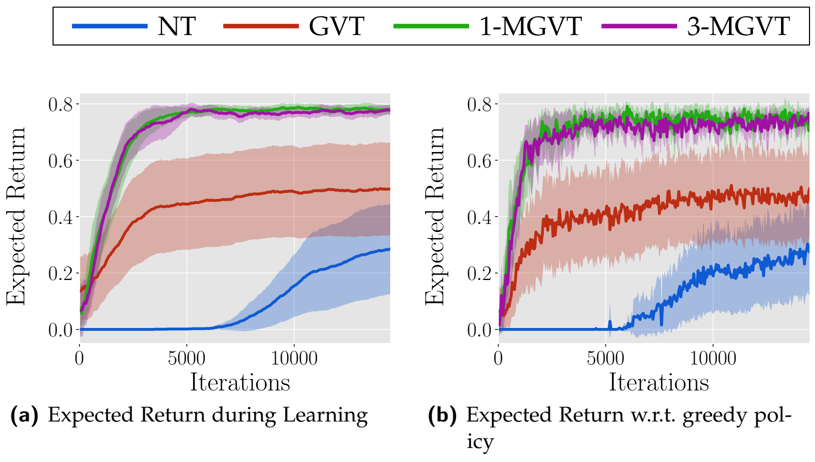
(b) Expected Return w.r.t. greedy policy

Figure 6.2: Rooms Environment Expected Return estimated with 20 independent runs. The 95% confidence intervals are also shown.

6.1.2 *Evaluation under a Distribution Change*

Further, we evaluate how different is the potential among the transfer algorithms GVT and MGVT. In order to do this, we propose to slightly complicate the setting by modifying the distribution from which the source tasks are drawn: we sample tasks by keeping the bottom door fixed in the middle and uniformly sample the position of the top door as before. We sample, then, 50 source tasks from this modified distribution, solve for their optimal Q-function as before and sample target tasks from the original distribution—both doors' positions sampled.

In Figure 6.3, we show the results of the experiments. As before, we show both the learning performance by averaging the last 50 episodes' return in Figure 6.3a and the performance obtained by evaluating the greedy policy derived from the mean Q-function in Figure 6.3b—the curves are the average of 20 independent runs and the shades are the corresponding 95% confidence intervals. These results clearly show the advantage of MGVT over GVT. The high variance and low mean reward of the latter show that as we deviate from the distribution of the tasks, the GVT fails to adapt, given that the Gaussian model over-constrains the admissible space of Q-functions to explore and, thus, discard the actual Q-functions required for solving the target tasks. On the other hand, MGVT is clearly more flexible, as it actually manages to achieve optimal performance. MGVT prior model is able to hold all the information of the source tasks and to exploit it without being overly constrained to the different distribution governing the source tasks of this experiment.



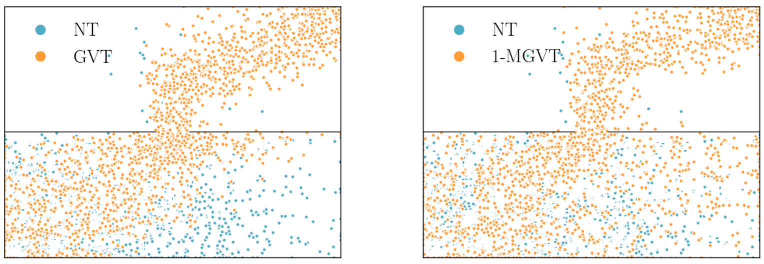
(a) Expected Return during Learning (b) Expected Return w.r.t. greedy policy

Figure 6.3: Evaluation under a distribution change in Rooms Environment: Expected Return—estimation with 20 independent runs with 95% confidence intervals.

6.1.3 *Induced Exploration*

In order to better understand how GVT and 1-MGVT induce exploration, we consider the following experiment. In a two room variant of the rooms environment, i.e. one wall in the middle of the room with a door in mid-position, we run GVT and MGVT for 2000 iterations and plot the states (positions) visited by the agent. The start and goal positions are kept as before, bottom left and top right, respectively, and 10 source tasks were used for the transfer algorithms. In Figure 6.4, we show the resulting plots.

First, in Figure 6.4a, we show the difference in the exploration obtained by the ϵ-greedy heuristic and the GVT algorithm. We clearly observe that in these few iterations the ϵ-greedy exploration is unable to allow the agent to reach the goal, as many of the states visited are scattered in the bottom room. GVT, instead, guides the exploration to states towards the middle of the wall. After crossing to the top room, the path to the goal is clearly defined as expected, since the values after the wall are expected to be the same among all the tasks. In Figure 6.4b, we compare the ϵ-greedy exploration and the 1-MGVT, which shows a similar behavior w.r.t. GVT, however, it is worth noting that MGVT has a sparser exploration of the bottom room when compare to GVT. It can be easily seen that MGVT is able to actually explore the right part of the bottom room. Thus, holding a more powerful model for the prior may allow MGVT to explore more effectively the environment, which in turn, would provide better adaptation to the target tasks as seen in the experiment discussed before in Section 6.1.2.



(a) ϵ-greedy vs. GVT **(b)** ϵ-greedy vs. MGVT

Figure 6.4: Exploration Comparison in a Two-Room variant of the Rooms Problem

6.1.4 *Performance vs. Task Likelihood*

GVT and MGVT were designed so that they hold different representational power, being the latter more powerful, to approximate a potentially complex distribution over optimal Q-function. Hence, it is of interest to understand in this simple scenario how does the likelihood of the possible tasks influence the learning performance of the algorithms.

In order to do this, we consider the simplest rooms problem: the two room (one door) variant. We modify the distribution over tasks to be, instead of a uniform distribution over the door position, a Normal distribution with mean 5 and standard deviation of 1.8. In such a way, we ensure that the extreme door positions are very unlikely. We sample 20 tasks from this Normal distribution and plot, in Figure 6.5, the expected return during learning—computed as the average return over the last 50 episodes—w.r.t. the likelihood of the task.

Figure 6.5 displays curves both for GVT and for 1-MGVT at 500 iterations, 1000 iterations and 4000 iterations to show the evolution of the

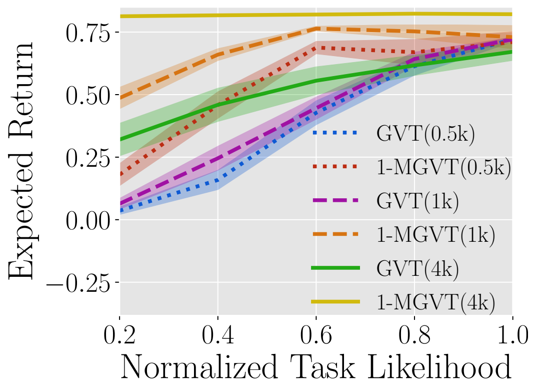
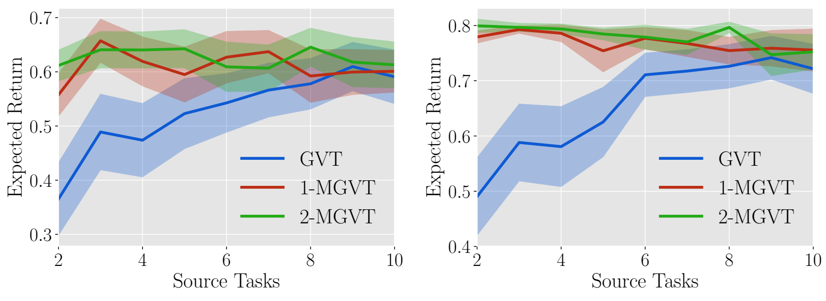


Figure 6.5: Expected return as a function of task likelihood

performance over time as well. From such plot, we can easily observe how GVT does not perform as well as MGVT when the tasks become very unlikely. Given that GVT overfits to the task distribution, it hinders the learning process for such unlikely tasks while MGVT is able to hold the information and to use it efficiently to quickly learn those same tasks. It is even possible to notice that at 4000 iterations 1-MGVT is already optimal for all tasks, no matter the likelihood. This further confirm the representational power of MGVT seen in the Generalization experiment presented in Section 6.1.2 as it shows that MGVT, besides holding a more informative prior, it also allows fast adaptation to unlikely tasks.

6.1.5 *Performance vs. Number of Sources*

Finally, as a higher number of source tasks must produce a better estimation of the distribution governing optimal Q-functions, we explore here the performance of GVT and MGVT with a varying number of source tasks.



(a) After 1000 iterations **(b)** After 1950 iterations

Figure 6.6: Expected Return vs. Number of source tasks for GVT

As in Section 6.1.4, we use the Two Rooms variant of the problem. In Figure 6.6 we present the expected learning return—the average return over the last 50 episodes—for GVT, 1-MGVT and 2-MGVT algorithms. The curves presented show the average over 20 independent runs and its 95% confidence interval.

In Figure 6.6a, we show the performance after 1000 iterations. We can observe that GVT with a small number of source tasks has a very low performance that, in the worst cases, is even half of that gotten from 1-MGVT and 2-MGVT for the same number of sources. The performance gap is more obvious in Figure 6.6b where the curves show the performance after 1950 iterations. GVT estimates a normal distribution over the parameters of the optimal Q-functions by Maximum Likelihood estimation and, thus, at 2 source tasks the estimation of the approximation to the distribution of optimal Q-function is very poor which it is reflected in its low performance. MGVT instead uses efficiently the little information by means of the non-parametric prior model that it uses. When the number of sources increases we can easily see that, for this scenario in which the distribution over optimal Q-functions is simple enough, the curves in Figure 6.6 behave similarly, as expected.

6.2 CLASSIC CONTROL

In this section, we focus on classic tasks within the RL research community: *Cartpole* and *Mountain Car*—representative diagrams shown in Figure 6.2.1 and Figure 6.2.2, as defined by Sutton and Barto, 1998. These two environments offer the possibility to explore the behavior of the Variational Transfer (VT) algorithms within canonical tasks. Moreover, we show how their performance compare to DDQN's when the latter is used as baseline for the NT setting. It is worth noting that DDQN has shown excellent performance in these classic control tasks and its use as a comparison baseline is intended to exhibit that our algorithm shows improvement w.r.t. state-of-the-art RL algorithms and that it is robust to negative transfer.

6.2.1 *Cartpole*

Cartpole is shown in Figure 6.8. Here, the objective is to maintain the pole in vertical position as long as possible. There are two possible actions that apply a force of a determined magnitude with direction *right* or *left*. The dynamics of the system are defined by the cart mass $\mathbf{m_{cart}}$, the pole length $\mathbf{l_{pole}}$ and the pole mass $\mathbf{m_{pole}}$.

We generate different tasks by uniformly sampling the cart physical parameters: the cart mass in the range $[0.5, 1.5]$, the pole mass in $[0.1, 0.3]$ and the pole length in $[0.2, 1.0]$. As reward we use the canonical one, with 1 for every action taken and a fixed time horizon of 100 time-steps. We set the discount factor $\gamma = 0.99$ and we use a Multilayer Perceptron (MLP) as function approximator for the

Q-function with a single hidden layer of 32 neurons with Rectified Linear Unit (ReLU) activations.

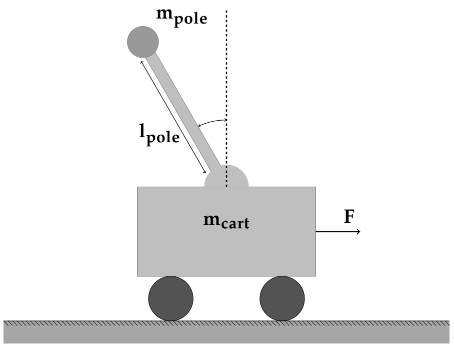
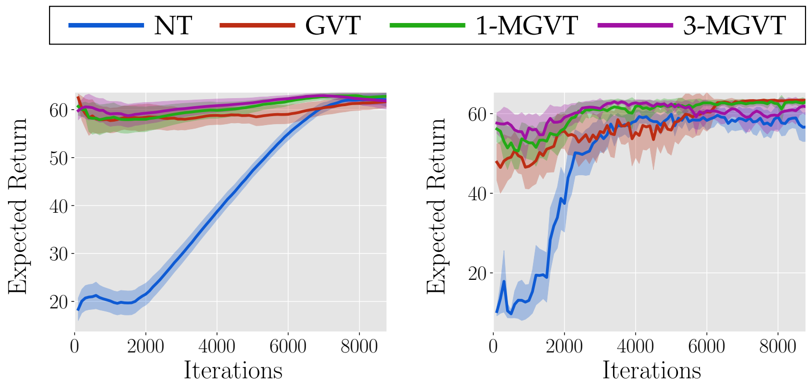


Figure 6.7: Cartpole Diagram. $\mathbf{m_{cart}}$ is the mass of the cart, $\mathbf{m_{pole}}$ is the mass of the pole and $\mathbf{l_{pole}}$ is the length of the pole. **F** is the force applied when one of the action (*left* or *right*) is taken.

We sample 100 source tasks and train to solve them optimally as described in Section 5.1.4. For the VT algorithms we run 20 independent runs by sampling a target task and selecting randomly 10 source tasks from which to transfer. The estimated expected returns are shown in Figure 6.8 as the average performance among the independent runs with a 95% confidence interval—in Figure 6.8a, we show the learning performance obtained as the average of the last 50 episodes' return and, in Figure 6.8b, the return of the greedy policy derived from the mean Q-function at the moment of evaluation. It is quite noticeable the almost zero-shot behavior of the transfer in this setting, all of the VT algorithms are able to start with near optimal behavior and converge quickly, even though the physical parameters vary within a relatively large range.



(a) Expected Return during Learning **(b)** Expected Return w.r.t. greedy policy

Figure 6.8: Cartpole's Expected Return estimated with 20 independent runs. The 95% confidence intervals are also shown.

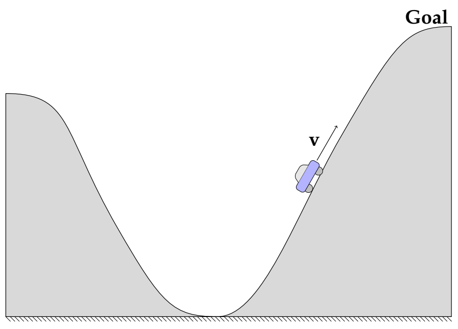


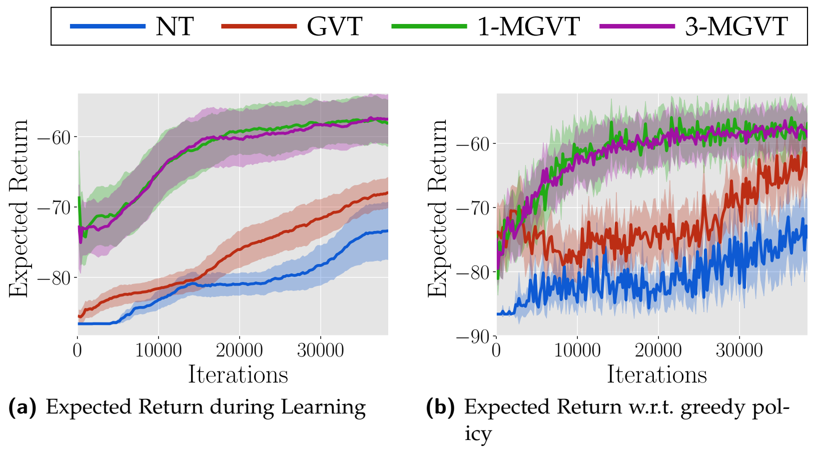
Figure 6.9: Mountain Car Diagram. **v** is the velocity applied by the taken action. The magnitude of this velocity defines the tasks and the direction corresponds to the *left* and *right* actions.

Cartpole's Q-values show an interesting structure for our algorithm. These are intrinsically linked to being able to keep the pole standing as long as possible—bounded by the time horizon—, hence, the optimal value functions vary little and starting from the expected Q-function of the prior distribution already induces a policy that is really close to being optimal. Thus, little experience is required to converge. Both the learning return in Figure 6.8a and the evaluation of the greedy policy induced by the posterior distribution in Figure 6.8b show the aforementioned behavior. Perhaps, the latter shows more clearly that: during the first 2000 iterations the changes of the expected Q-function of the posterior distribution are quite discernible.

6.2.2 *Mountain Car*

Mountain Car, being a more complex task than Cartpole, is worth studying within our transfer setting. This task consists of a controllable car in a valley, as seen in Figure 6.2.2, that has only three possible actions: to apply a velocity with determined magnitude to the *right*, to the *left* or not to apply it at all. The goal of the agent is to build enough momentum to reach the top of the hill in the right of Figure 6.2.2. To generate tasks, we uniformly sample the car base speed—used in its actions—within the range $[0.001, 0.0015]$ which ensures that the car is not able to trivially drive up the mountain. In this case the reward is -1 at every time step and 0 when it reaches the mountain top. The discount factor is $\gamma = 0.99$ and the task finishes when the car reaches the top. Finally, as in the case of Cartpole, the Q-function is parameterized by an MLP with a single hidden layer of 64 neurons.

Before discussing the results, it is worth analyzing the structure of Mountain Car's Q-values. In this environment, the faster the car reaches the mountain top the better and the car base speed, clearly, affect how much time would it take; the lower the speed, the higher the need for the car to build momentum by oscillating in the valley. Therefore, as opposed to Cartpole, the variation in the Q-functions is greater which, in turn, poses a more challenging situation for our VT algorithms.



(a) Expected Return during Learning **(b)** Expected Return w.r.t. greedy policy

Figure 6.10: Mountain Car's Expected Return estimated with 20 independent runs. The 95% confidence intervals are also shown.

To obtain the plots in Figure 6.10, we generate 50 source tasks, as described above. We execute 20 independent runs of the VT algorithms by sampling a target task and selecting 10 random sources from the previously constructed set. We show the expected return during learning—the average of the previous 50 episodes' return—in Figure 6.10a and the greedy evaluation of the expected Q-function in Figure 6.10b. The 95% confidence intervals are shown together with the average performance obtained for the 20 independent runs. Both plots show a very different behavior compared to Cartpole, as expected when considering the discussion on the Q-values in this environment. More clearly, this situation can be noticed in the behavior of GVT. In both plots, it fails to provide a significant advantage over DDQN. The Gaussian distribution hinders the learning of the target optimal Q-function as it over-constrains the algorithm to remain close to a mean Q-function that most probably cannot solve the target task optimally, given the variance of this environment's Q-functions; thus, GVT learns slowly. MGVT shows again its power, by converging to the expected optimal performance faster than GVT and DDQN, even if the jump-start shown is not great. Notice, that the variance seen is explained by the natural variance in the Q-values.

6.3 MAZE NAVIGATION

In the current section, we present a simulation of a more challenging scenario in order to evaluate how our VT algorithms cope with little information about the distribution of Q-functions induced by the distribution over tasks and when the transferable information is not quite as easy to observe.

We, then, define the navigation of a robotic agent through a maze. The robot starts its task in a random position within a maze of $10m^2$ with the possibility to rotate, left or right, with a speed of $\frac{\pi}{8} rad/s$ and to move forward with a speed of $0.5m/s$. The robot can sense its absolute position and orientation. Additionally, it is equipped with distance sensor that allows for the detection of obstacles—and the goal—in 9 equally-spaced directions within $2m$ of distance and a range 180 degrees. The agent receives a reward of 1 when it reaches the goal and 0 otherwise. The discount factor is $\gamma = 0.99$. The Q-function is hereafter considered to be parameterized by an MLP with two hidden layers of 32 neurons each and using ReLU activations.

Furthermore, we design a set of 20 different mazes, divided in 4 groups characterized by having the goal in one of the corner positions of the maze (identified by the green corner in the figures), in Figure 6.11 a sample of the set is shown. To further explore the performance of the VT algorithms, we use the DDQN algorithm, as opposed to the minimization of TD error used thus far, to solve optimally for the Q-functions and use these as source set for transferring. In this way, we intend to better understand the behavior of our proposed algorithm w.r.t. the use of other RL algorithms to solve the sources.

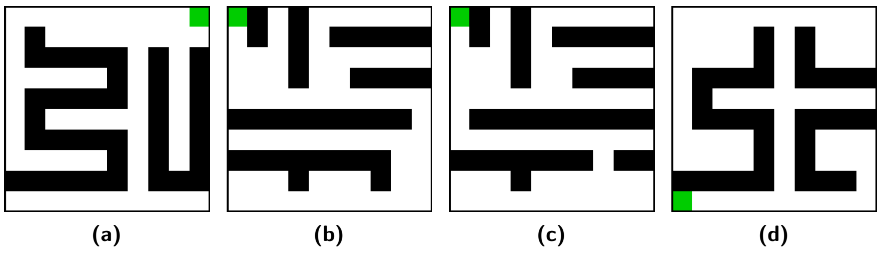


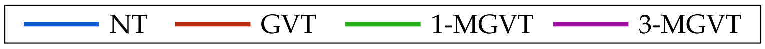
Figure 6.11: Sample of the set of mazes

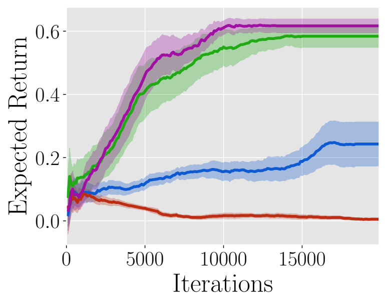
Here, we present the results found by considering each of the mazes shown in Figure 6.11 as a target task and transferring from a randomly sampled subset of 5 sources, in which we ensure not to include the target task. In Figure 6.12, we show the expected results during the learning process and the greedy evaluation of the expected Q-function. For each of those, we run 20 independent experiments and we show the mean performance and its 95% confidence interval. It is worth noting that we select these as they are representative of the set of mazes designed—which is shown in Appendix A.2.3—and these are the ones that pose more difficulties in this RL setting.

It is quite noticeable the utter failure of GVT to adapt in this context. We can observe clearly, in comparison with the DDQN (no transfer) baseline, that the Gaussian assumption in this environment is wrong.

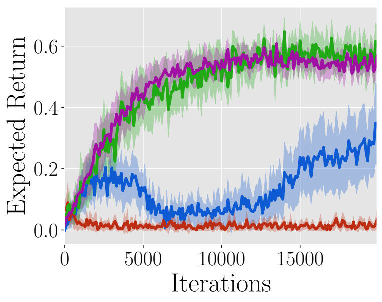
The Gaussian distribution loses much of the information from the sources and this, in turn, hinders the algorithm to effectively find the optimal function parameters. In fact, it shows negative transfer behavior as it can be noticed in Figure 6.12a: at the beginning of the training process, the agent is able—most likely—to reach the goal whenever it is close enough from its initial position but after the learning procedure continues, it also loses that behavior. 1-MGVT and 3-MGVT, instead, are able to exploit transferable characteristics of the sources' Q-values as shown in the performances in Figure 6.12. In fact, even though the expected Q-function of the prior is not enough for a good jump-start, it achieves a faster slope and, within this number of iterations, some converge to optimality when the DDQN baseline is unable to.

From these results, we can better appreciate the representational power MGVT is able to exploit from its prior. This environment is, in fact, challenging because the limited source of tasks makes it difficult to capture with much accuracy the distribution over Q-functions, however, our algorithms can use the little information available to guide the exploration in the parameter space to reach optimality. Furthermore, it is worth noticing that the base RL algorithm used to train the source tasks does not hinders the transfer process.

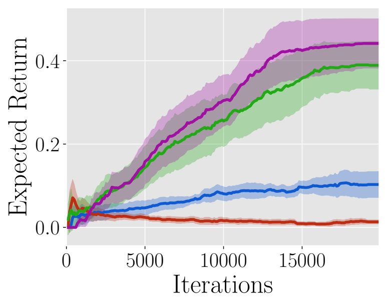




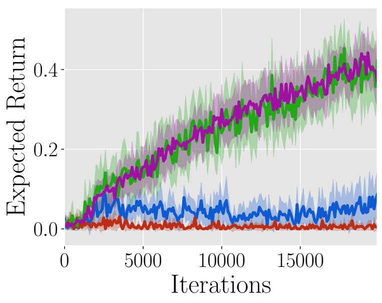
(a) Maze A.1a. Expected Return during Learning



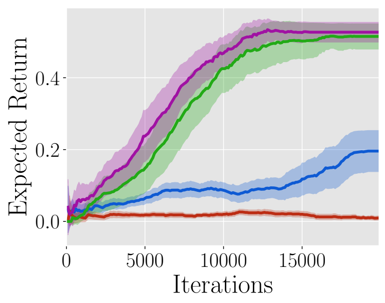
(b) Maze A.1a. Expected Return w.r.t. greedy policy



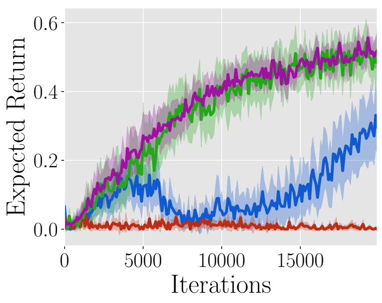
(c) Maze A.1g. Expected Return during Learning



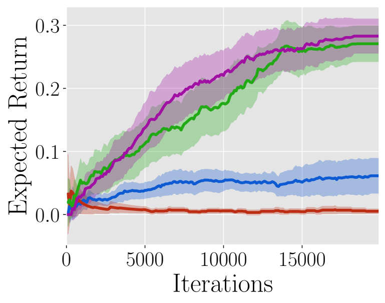
(d) Maze A.1g. Expected Return w.r.t. greedy policy



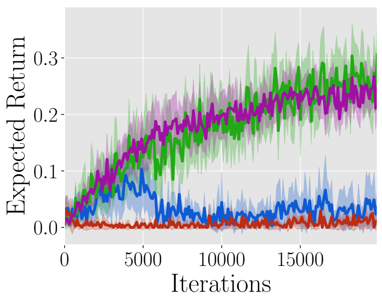
(e) Maze A.1i. Expected Return during Learning



(f) Maze A.1i. Expected Return w.r.t. greedy policy



(g) Maze A.1n. Expected Return during Learning



(h) Maze A.1n. Expected Return w.r.t. greedy policy

Figure 6.12: Maze's Expected Return estimated with 20 independent runs. The 95% confidence intervals are also shown.

7 CONCLUSIONS AND FUTURE WORKS

In this thesis, we presented the *Variational Transfer* algorithm to tackle the problem of transferring optimal value functions from a set of source tasks in order to solve a novel, related target task. As a main motivation of our design, we committed to provide as much generality as possible to allow for practical implementations of the method to be of use in the varied scenarios of current RL and, finally, we presented two interesting practical implementations of our approach based on Multivariate Gaussians and Mixture of Gaussians.

Furthermore, in our evaluation scenarios we showed that we could apply our algorithm with different models of parameterized function models widely used in current Machine Learning such as the linear models with radial basis features and fully-connected neural networks (MLPs), that are the base of the current approaches in modern *Deep Learning*. We, empirically, showed that our algorithm brought value when compared with the classic, non transfer, RL methods in terms of faster adaptation to new tasks and better jump-starts in the learning settings. Interestingly, we provided insight on how limiting assuming Gaussian distributions for the Bayesian inferences in *Transfer*, such as in GVT, could be; and that our method based on Mixtures of Gaussians, MGVT, could overcome this by providing a more powerful representation and robustness to slight deviations from the original assumptions of the transfer process.

Moreover, while *Variational Transfer* is mainly trying to transfer efficiently the previous knowledge, it also explores the target tasks by guiding the exploration through the stochasticity introduced by the prior knowledge and the inference done. The trade-off of exploiting the sources' information and exploring the target effectively could lead to even better performances and robustness which future works could explore. Further, we also introduced some hyper-parameters of the algorithm to allow to control this trade-off between the importance of previous knowledge and newly-obtained experience. However, how to optimize these hyper-parameters in order to avoid early convergence of the posterior probability mass to sub-optimal solutions is an important future direction.

Another contribution that might be of interest from this work is the introduction of the learning procedure based on the *Mellow Bellman Operator* that we presented in Section 5.1.4. The direct minimization of the squared TD error based on this softened operator to learn an optimal Q-function is an interesting full residual RL algorithm. However, we encountered that it required some tuning to make it stable and, thus, further study would be necessary to understand its potential.

As the designed approach to transfer optimal Q-functions was general enough, our method could be extended to work with Policy Gradient methods, instead of value functions. Learning policies has proven to be very useful as it searches directly for the maximum discounted reward and, recently, these techniques has shown great successes. Transferring behavior could offer similar benefits and, hence, it is potentially a next step for the *Variational Transfer* method.

BIBLIOGRAPHY

Asadi, Kavosh and Michael L Littman (2017). “An Alternative Softmax Operator for Reinforcement Learning.” In: *International Conference on Machine Learning*, pp. 243–252 (cit. on pp. 26, 27).

Azizzadenesheli, Kamyar, Emma Brunskill, and Animashree Anandkumar (2018). “Efficient Exploration through Bayesian Deep Q Networks.” In: *arXiv preprint arXiv:1802.04412* (cit. on p. 20).

Baird, Leemon (1995). “Residual algorithms: Reinforcement learning with function approximation.” In: *Machine Learning Proceedings 1995*. Elsevier, pp. 30–37 (cit. on p. 27).

Barreto, André, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver (2017). “Successor features for transfer in reinforcement learning.” In: *Advances in neural information processing systems*, pp. 4055–4065 (cit. on p. 12).

Blei, David M, Alp Kucukelbir, and Jon D McAuliffe (2017). “Variational inference: A review for statisticians.” In: *Journal of the American Statistical Association* 112.518, pp. 859–877 (cit. on p. 9).

Catoni, Olivier (2007). “PAC-Bayesian supervised classification: the thermodynamics of statistical learning.” In: *arXiv:0712.0248* (cit. on p. 10).

Doshi-Velez, Finale and George Konidaris (2016). “Hidden parameter Markov decision processes: A semiparametric regression approach for discovering latent task parametrizations.” In: *IJCAI: proceedings of the conference*. Vol. 2016. NIH Public Access, p. 1432 (cit. on p. 19).

Duan, Yan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel (2016). “RL$ˆ2$: Fast Reinforcement Learning via Slow Reinforcement Learning.” In: *CoRR* abs/1611.02779. arXiv: 1611.02779 (cit. on p. 15).

Engel, Yaakov, Shie Mannor, and Ron Meir (2005). “Reinforcement Learning with Gaussian Processes.” In: *Proceedings of the 22Nd International Conference on Machine Learning*. ICML ’05. Bonn, Germany: ACM, pp. 201–208. ISBN: 1-59593-180-5 (cit. on p. 17).

Fernández, Fernando and Manuela Veloso (2006). “Probabilistic policy reuse in a reinforcement learning agent.” In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM, pp. 720–727 (cit. on pp. 12, 13).

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017). “Model Agnostic Meta-Learning for Fast Adaptation of Deep Networks.” In: *CoRR* abs/1703.03400. arXiv: 1703.03400 (cit. on p. 15).

Hershey, John R and Peder A Olsen (2007). “Approximating the Kullback Leibler divergence between Gaussian mixture models.” In:

Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on. Vol. 4. IEEE, pp. IV–317 (cit. on pp. 29, 30).

Killian, Taylor W, Samuel Daulton, George Konidaris, and Finale Doshi-Velez (2017). "Robust and Efficient Transfer Learning with Hidden Parameter Markov Decision Processes." In: *Advances in Neural Information Processing Systems*, pp. 6250–6261 (cit. on p. 19).

Kingma, D. P and M. Welling (2013). "Auto-Encoding Variational Bayes." In: *ArXiv e-prints*. arXiv: 1312.6114 [stat.ML] (cit. on p. 28).

Kober, Jens and Jan R Peters (2009). "Policy search for motor primitives in robotics." In: *Advances in neural information processing systems*, pp. 849–856 (cit. on p. 1).

Konidaris, George and Andrew Barto (2007). "Building Portable Options: Skill Transfer in Reinforcement Learning." In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. IJCAI'07. Hyderabad, India: Morgan Kaufmann Publishers Inc., pp. 895–900 (cit. on pp. 12, 13).

Lazaric, Alessandro and Mohammad Ghavamzadeh (2010). "Bayesian multi-task reinforcement learning." In: *ICML-27th International Conference on Machine Learning*. Omnipress, pp. 599–606 (cit. on pp. 12, 13, 17, 18).

Lazaric, Alessandro, Marcello Restelli, and Andrea Bonarini (2008). "Transfer of samples in batch reinforcement learning." In: *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 544–551 (cit. on pp. 12, 13).

Levine, Sergey, Chelsea Finn, Trevor Darrell, and Pieter Abbeel (2016). "End-to-end training of deep visuomotor policies." In: *The Journal of Machine Learning Research* 17.1, pp. 1334–1373 (cit. on pp. 1, 11).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller (2013). "Playing Atari with Deep Reinforcement Learning." In: *CoRR*. arXiv: 1312.5602 (cit. on pp. 1, 11).

Osband, I., D. Russo, and B. Van Roy (2013). "(More) Efficient Reinforcement Learning via Posterior Sampling." In: *ArXiv e-prints*. arXiv: 1306.0940 [stat.ML] (cit. on p. 20).

Osband, Ian, Benjamin Van Roy, and Zheng Wen (2014). "Generalization and exploration via randomized value functions." In: *arXiv preprint arXiv:1402.0635* (cit. on p. 20).

Puterman, Martin L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc. ISBN: 0471619779 (cit. on pp. 5, 6).

Silver, Daniel, Qiang Yang, and Lianghao Li (2013). "Lifelong Machine Learning Systems: Beyond Learning Algorithms." In: (cit. on p. 16).

Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, et al. (2016).

"Mastering the game of Go with deep neural networks and tree search." In: *nature* 529.7587, pp. 484–489 (cit. on p. 1).

Sutton, Richard S and Andrew G Barto (1998). *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge (cit. on pp. 1, 37).

Taylor, Matthew E and Peter Stone (2009). "Transfer learning for reinforcement learning domains: A survey." In: *Journal of Machine Learning Research* 10.Jul, pp. 1633–1685 (cit. on pp. 2, 12).

Thompson, William R (1933). "On The Likelihood That One Unknown Probability Exceeds Another in The View Of The Evidence of Two Samples." In: *Biometrika* 25.3-4, pp. 285–294. eprint: /oup/backfile/content_public/journal/biomet/25/3-4/10.1093/biomet/25.3-4.285/2/25-3-4-285.pdf (cit. on p. 19).

Tirinzoni, Andrea, Andrea Sessa, Matteo Pirotta, and Marcello Restelli (2018). "Importance Weighted Transfer of Samples in Reinforcement Learning." In: *arXiv preprint arXiv:1805.10886* (cit. on pp. 12, 13).

Van Hasselt, Hado, Arthur Guez, and David Silver (2016). "Deep Reinforcement Learning with Double Q-Learning." In: (cit. on p. 20).

Wilson, Aaron, Alan Fern, Soumya Ray, and Prasad Tadepalli (2007). "Multi-task reinforcement learning: a hierarchical Bayesian approach." In: *Proceedings of the 24th international conference on Machine learning*. ACM, pp. 1015–1022 (cit. on pp. 17, 18).

Yu, Tianhe, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine (2018). "One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning." In: *CoRR* abs/1802.01557. arXiv: 1802.01557 (cit. on pp. 1, 11).

A EXPERIMENTS DETAILS

In the present section we provide details on the parameters adopted in all experiments.

A.1 THE ROOMS PROBLEM

In order to train the source tasks, we directly minimize the TD error based on the *mellow* Bellman operator by stochastic gradient descent. We use a *batch size* of 50, a *buffer size* of 50000, $\psi = 0.5$ and a learning rate $\alpha = 0.001$. Additionally, we use an ϵ-greedy policy for exploration, with ϵ linearly decaying from 1 to 0.02 in a fraction of 0.7 the maximum number of iterations.

For the transfer algorithm GVT, we set a *batch size* of 50 and a *buffer size* of 10000. We use $\psi = 0.5$, $\lambda = 10^{-4}$ and 10 *weights* to estimate the expected TD error. For the learning rates, $\alpha_\mu = 0.001$ for the mean of the posterior and $\alpha_L = 0.1$ to learn its Cholesky factor L. Furthermore, we restrict the minimum value reachable by the eigenvalues of these factors to be $\sigma^2_{min} = 0.0001$. In the case of MGVT we use, instead, $\lambda = 10^{-6}$, $\alpha_\mu = 0.001$ and $\alpha_L = 0.1$. Finally, we use a bandwidth $\sigma^2_p = 10^{-5}$ for the prior.

A.2 CLASSIC CONTROL

A.2.1 *Cartpole*

For this environment we generate tasks by uniformly sampling the cart mass in the range $[0.5, 1.5]$, the pole mass in $[0.1, 0.3]$ and the pole length in $[0.2, 1.0]$.

During the training of the source tasks, we use a *batch size* of 150 and a *buffer size* of 50000. Specifically, for DDQN we use a *target update frequency* of 500,*exploration fraction* of 0.35 and a learning rate $\alpha = 0.001$. We use a Multilayer Perceptron (MLP) with ReLU as activation function and a single hidden layer of 32 neurons.

For the transfer experiments, we set the *batch size* to 500, the number of *weights* sampled to approximate the expected TD error to 5, $\lambda = 0.001$ and $\psi = 0.5$. We use $\alpha_\mu = 0.001$ as the learning rate for the mean of the Gaussian posterior. For its the Cholesky factor L we use $\alpha_L = 0.0001$ and set the limit that the minimum eigenvalue may reach to $\sigma^2_{min} = 0.0001$. Additionally, for MGVT we set the variance of the prior components $\sigma^2_p = 10^{-5}$ and leave the learning rates of the posterior components' means and Cholesky factor the same as GVT.

A.2.2 *Mountain Car*

We generate tasks sampling uniformly the base speed of the actions in the range $[0.001, 0.0015]$.

For the sources, we train the tasks using DDQN with a *target update frequency* of 500, a *batch size* of 32, a *buffer size* of 50000 and learning rate $\alpha = 0.001$. Moreover, we set the *exploration fraction* to 0.15. We use an MLP with single hidden layer of 64 neurons with ReLU activation function.

For the transfer experiments, we set the *batch size* to 500, and use 10 *weights* to approximate the expected TD error, $\lambda = 10^{-5}$ and $\psi = 0.5$. For the learning rates, we use $\alpha_\mu = 0.001$ for the means of the Gaussians. In the case of the Cholesky factors L, we use $\alpha_L = 0.0001$ and allow the eigenvalues to reach a minimum value of $\sigma^2_{min} = 0.0001$. In the case of MGVT, additionally, we set the prior covariance to be $\sigma^2_p = 10^{-5}$.

A.2.3 *Maze Navigation*

The mazes adopted in the experiments of Section 6.3 are shown in Fig. A.1. Our 20 mazes have varying degree of difficulty and are designed to hold few similarities that would be useful for transferring. Moreover, we ensure 4 groups of mazes that are characterized by their goal position.

For the experiments we use as an approximator an MLP with two hidden layers of 32 neurons with ReLU activation functions. For training the sources we use a DDQN with a *batch size* of 70, a *buffer size* of 10000 and a *target update frequency* of 100, setting the *exploration fraction* to 0.1 and learning rate to $\alpha = 0.001$.

In the transfer experiments we use $\psi = 0.5$, a *batch size* of 50, a *buffer size* of 50000 and use 10 sampled *weights* from the posterior to approximate the TD error. Moreover, we use $\lambda = 10^{-6}$. For GVT, in particular, we use $\alpha_\mu = 0.001$, $\alpha_L = 10^{-7}$, and set the minimum value reachable by its eigenvalues to be $\sigma_{min} = 0.0001$. In the case of MGVT, we set $\alpha_\mu = 0.001$ and $\alpha_L = 10^{-6}$. Finally, we use $\sigma^2_p = 10^{-5}$ as the prior bandwidth.

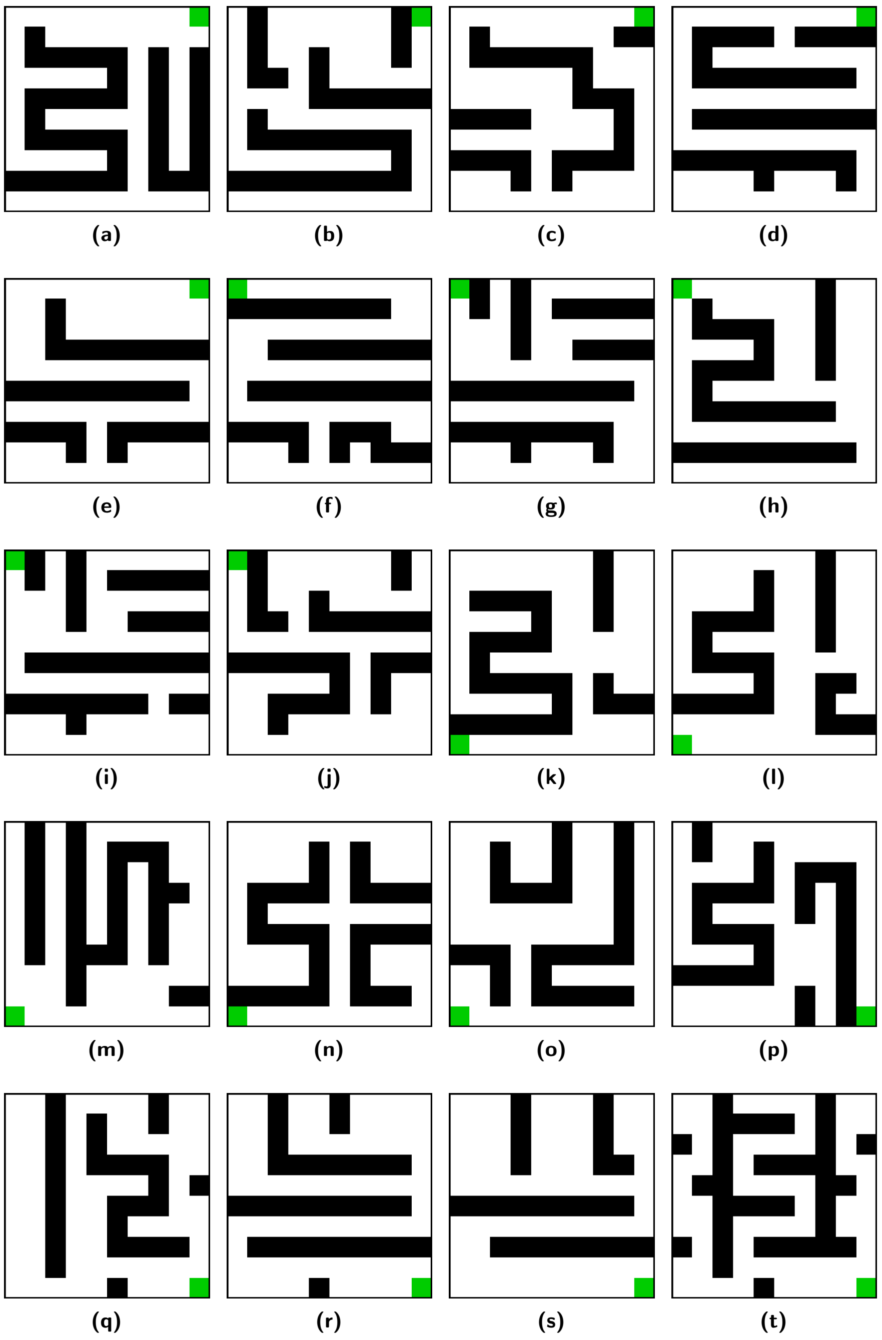


Figure A.1: Set of mazes used for experiments in Section 6.3