An overview of my thesis: Some Principled Methods for Deep Reinforcement Learning

Léonard Blier

Contents

In this document, I give an overview of my thesis. We give a summary of every chapter, highlighting our main contributions. In Section 1, corresponding to Chapter 3, we present an information theory viewpoint on the complexity of deep learning models. Then, in Section 2, corresponding to Part III, we present two published papers, both developing mathematical tools for robustness in deep RL. The first one (Chapter 4) describes *Alrao* (All learning rates at once) an optimization method, not specific to RL, but designed to work in many setting, including non-stationary RL problems, without hyperparameter tuning. The second one (Chapter 5) studies near continuous-time environments, and show how to design robust algorithms in that setting. Section 3 corresponds to Part IV (Chapters 6-12) and present our work on the successor state operator, for policy evaluation. Finally, Section 4 corresponds to Part V (Chapter 13-17) and present how we applied tools developed for the successor state operator to the setting of multi-goal reinforcement learning.

1 The description length of deep learning models

In Chapter 3, we present the following published paper:

Blier, L. and Ollivier, Y. (2018). The description length of deep learning models. In *Advances in Neural Information Processing Systems*

This was the first of this thesis. Its story is worth mentioning. As I was starting to work with Yann Ollivier, he asked me to read about the information theory viewpoint (Solomonoff inference, Minimum Description Length) for machine learning.

In information theory and Minimum Description Length (MDL), learning a good model of the data is recast as using the model to losslessly transmit the data in as few bits as possible. Consider an image classification setting with a dataset (for example CIFAR10) $\mathcal{D} = \{(x_1, y_1), ..., (x_N, y_N)\}$ where the x_i are the images and the y_i are the labels. The classification problem is recast as follows: Alice has the entire dataset $\mathcal{D}$, but Bob only has the images $\{x_1, ..., x_N\}$, and Alice wants to transmit the labels. The first bpossibility is to directly send a file containing the labels $\{y_1, ..., y_N\}$ without leveraging the fact that Bob has the images. But it is possible to do better, by sending a *model* predicting the labels from the images, together with the list of errors of the model. A more *complex* model might make less errors, hence compress the data more, but the model must be transmitted as well. The overall codelength can be understood as a combination of quality-of-fit of the model (compressed data length), together with the cost of encoding (transmitting) the model itself. The MDL viewpoint is that the *best model* is the model achieving the best trade-off between *complexity* and accuracy of the model, measured with compression bounds.

MDL is based on Occam's razor, and on Chaitin's hypothesis that "*comprehension is compression*" (Chaitin, 2007): any regularity in the data can be exploited both to compress it and to make predictions. This is ultimately rooted in Solomonoff's general theory of inference (Solomonoff, 1964), whose principle is to favor models that correspond to the "shortest program" to produce the training data, based on its Kolmogorov complexity (Li and Vitányi, 2008). If no structure is present in the data, no compression to a shorter program is possible.

After a few weeks reading about these topics, I came back to Yann. I told him that while this viewpoint is elegant, it is *obviously wrong*: indeed, the hypothesis is that the best model for prediction is also the best model for compression. But the success of deep learning *proves* that this is untrue. I took the classification problem introduced above, with the example of CIFAR10. In that case, the *baseline* for compression is encoding directly the labels of the dataset, without any model using the images, which costs $N \times \log_2(\text{Number of classes}) = 50,000 \times \log_2 10 = 166\,\text{kbits}$. This means that the MDL viewpoint would only select a model if it is able to encode the labels with *less* than 166 kbits. But the best model for prediction on that dataset are deep learning models with millions of parameters, and encoding the dataset with such a model requires taking into account the cost of encoding the model weights in a way. The conclusion was quite clear to me: it is impossible to encode a dataset of 166 kbits using a model with millions of parameters, hence: *The success of deep learning proves that the MDL viewpoint is wrong, and that the best model for prediction are not the best models for compression.* Yann did not agree with my conclusion, and told me that it was actually possible to compress such a dataset with a large model, even taking into account the cost of encoding the weights, using variational techniques introduced by (Hinton and Van Camp, 1993). I went back home, and tried to think of other ways to encode a dataset with a deep learning model.

It turned out we were *both* wrong (but to be honest, especially me): it is actually possible to compress the labels of a dataset like CIFAR10 in less than 166 kbits (we reached 35 kbits) with a state-of-the-art network (in our case with a VGG19 which has more than 10M parameters), but not with variational methods.

We now describe our contributions. First, we study variational methods for deep learning.

- We show that the traditional method to estimate MDL codelengths in deep learning, variational inference (Hinton and Van Camp, 1993), yields surprisingly inefficient codelengths for deep models, despite explicitly minimizing this criterion. This might explain why variational inference as a regularization method often does not reach optimal test performance.

This contribution applies for standard variational techniques used while we were working on this project (Blei et al., 2017), but could be different for more recent techniques such as normalizing flows (Rezende and Mohamed, 2015; Kobyzev et al., 2020).

Then, we introduce new practical ways to compute tight compression bounds in deep learning models, based on the MDL toolbox (Grünwald, 2007):

- We show that *prequential coding* on top of standard learning, yields much better codelengths than variational inference, correlating better with test set performance. Thus, despite their many parameters, deep learning models do compress the data well, even when accounting for the cost of describing the model.

The principle of prequential code is the following: First, Alice sends to Bob the description of a network architecture and an optimization algorithm (which is only a few lines of code), and a file containing the first 100 labels of the dataset. Then, Alice and Bob both train the network with these 100 labels. Alice can now use this trained network as a *model* (which will not be very accurate) to encode 100 next labels. Now they both have 200 labels and can train a better model, which will be used for the next 100 labels. As more labels are transmitted, the model becomes more accurate and sending more labels is less expensive. This compression scheme leverages the generalization ability of deep learning networks, even with very limited datasets. These introduced techniques lead to the main contribution of this work:

- Deep learning models, even with a large number of parameters, compress the data well: from an information theory point of view, *the number of parameters is not an obstacle to compression.* This is consistent with Chaitin's hypothesis that "*comprehension is compression*".

Finally, prequential codes also lead to a model selection strategy:

- With prequential codes, we obtain a model selection strategy, suitable for a deep learning framework, not using cross validation, hence useful for small datasets settings. This technique was especially used for probing in NLP (Voita and Titov, 2020).

2 Mathematical approaches towards robustness in Deep Reinforcement Learning

In this Section, we introduce two published papers developing mathematical tools for robustness in deep reinforcement learning. First, *All learning rates at once* (Alrao) is an optimization method, not specific to RL, but designed to work in many setting, including non-stationary RL problems, without hyperparameter tuning. The second one studies near continuous-time environments, such as continuous control environments, many video games, ... It first shows that standard approaches such as DQN, DQQPG fail to learn in that setting when the time discretization decreases (or equivalently the number of action/observation per second increases), and then show how to design robust algorithms in that setting.

2.1 Learning with all learning rates at once

In Chapter 4, we present the published paper, which is joint work with Pierre Wolinski and Yann Ollivier:

> Blier, L., Wolinski, P., and Ollivier, Y. (2019). Learning with Random Learning Rates. In *ECML PKDD 2019 - European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*

This project started from discussions on how to design a learning system able both to adapt very quickly to new observed patterns, but also to learn complex patterns which might require a slow learning. I was starting to work in Reinforcement Learning at that time. In RL, the learning setting is *non-stationary*: the distribution of observations can change while the agent is improving its policy, and the learning methods has to be able to adapt to all of these regimes. My co-author Pierre Wolinski was interested in topics in AutoML (Guyon et al., 2016). From this viewpoint, a learning system has to be able to adapt to multiple learning settings without any hyperparameter tuning, such as settings which would require small or large learning rates. Many methods were designed to directly set optimal per-parameter learning rates (Tieleman and Hinton, 2012; Kingma and Ba, 2015), such as the popular Adam optimizer, but they still require some hyperparameter tuning.

This discussion lead to the idea of a learning system which would be a mixture of *slow learning units*, and *fast learning units*.

- We implemented this idea directly in deep learning models, by using different learning rates for different *neurons*, sampled across multiple order of magnitudes, hence leveraging redundancy in the network. We call this method *All Learning Rates At Once* algorithm (Alrao).

Alrao departs from the usual philosophy of trying to find the "right" learning rates; instead we take advantage of the overparameterization of network-based models to produce a diversity of behaviors from which good network outputs can be built.

We then experiment Alrao in multiple settings. Experimentally, we were interested to see if Alrao was working *out-of-the-box*, without any hyperparameter tuning, and how it would compare to SGD with an optimally selected learning rate.

- Surprisingly, Alrao does manage to learn well over a range of problems from image classification, text prediction, and reinforcement learning. In our tests, Alrao's performance is always close to that of SGD with the optimal learning rate, without any tuning. Alrao combines performance with *robustness*: not a single run failed to learn with the default learning rate range we used. In contrast, our parameter-free baseline, Adam with default hyperparameters, is not reliable across the board.

2.2 Making Deep Q-learning methods robust to time discretization

In Chapter 5, we present the published paper:

> Tallec, C., Blier, L., and Ollivier, Y. (2019). Making Deep Q-learning methods robust to time discretization. In *ICML 2019 - Thirty-sixth International Conference on Machine Learning*

In this paper we study the sensitivity of Deep Reinforcement Learning (DRL) techniques to time discretization, such as what happens when an agent receives 50 observations and is expected to take 50 actions per second instead of 10. In principle, decreasing time discretization, or equivalently shortening reaction time, should only improve agent performance. Robustness to time discretization is especially relevant in *near-continuous* environments, which includes most continuous control environments, robotics, and many video games.

This work is a contribution to the problem of robustness of DRL techniques. Despite the impressive results of DRL techniques in a variety of domains (Silver et al., 2017; OpenAI, 2018b; Mnih et al., 2015; OpenAI, 2018a), these approaches are sensitive to a number of factors, including hyperparameterization, implementation details or small changes in the environment parameters (Henderson et al., 2017; Zhang et al., 2018). This sensitivity, along with sample inefficiency, largely prevents DRL from being applied in real world settings. Notably, high sensitivity to environment parameters prevents transfer from imperfect simulators to real world scenarios. In this work, the goal is to mitigate one of these sensitivity factors: the time discretization.

Our first contribution is to show that standard approaches based on estimation of state-action value functions, such as *Deep Q-learning* (DQN, Mnih et al. 2015) and *Deep deterministic policy gradient* (DDPG, Lillicrap et al. 2015) are not at all robust to changes in time discretization:

- We show experimentally that when the time discretization decreases in standard environment, DQN and DDPG are unable to learn at all.

Intuitively, as the discretization timestep decreases, the effect of individual actions on the total return decreases too: $Q^*(s, a)$ is the value of playing action a then playing optimally, and if a is only maintained for a very short time its advantage over other actions will be accordingly small. If the discretization timestep becomes infinitesimal, the effect of every individual action vanishes. Hence, the Q-function $Q(s, a))$ collapses to the value function $V(s)$ and does not bear any information on the ranking of actions. We say that there is no continuous-time Q-function.

- Building on (Baird, 1994), we formalize these statement, in the framework of continuous-time RL, and show that there is no continuous-time Q-function, hence the poor performance of Q-learning with small time steps (Theorem 5.2). More precisely, Thus that standard Q-learning is ill-behaved in this setting.

We then looked for an algorithm that would be as invariant as possible to changing the discretization timestep. Such an algorithm should remain viable when this timestep is small, and in particular admit a continuous-time limit when the discretization timestep goes to 0. This leads to the algorithm *Deep Advantage Updating* (Algorithm 4). Our first contributions are to show that while there is no continuous Q-function, there is a continuous *advantage* function, which is possible to learn.:

- First, we formally show that while the Q-function $Q_{\delta t}$ collapses when the time discretization $\delta t \to 0$, the rescaled advantage:

$$A_{\delta t}(s, a) = \frac{Q_{\delta t}(s, a) - V_{\delta t}(s)}{\delta t} \tag{2.1}$$

  converge to a *continuous time limit advantage function* $A(s, a)$ (Theorem 5.3).

- We then learn together models of the value function $V(s)$ and the continuous time limit advantage function $A(s, a)$. We formally show that there are Bellman equations on V and A, and we use it for policy optimization, with the *Deep Advantage Updating* (DAU) algorithm.

In order to define a time-discretization invariant algorithm, it is also necessary to define an invariant exploration strategy:

- We formally show that an ε-greedy exploration strategy collapse to *no exploration at all* when the time-discretization goes to 0 (Theorem 5.4).

- We derive a time-discretization invariant exploration scheme, both for discrete and continuous actions.

The principle is the following, inspired by Lillicrap et al. (2016) in the case of continuous actions, and extended to deterministic actions. For continuous actions, if $a_t = \pi(s_t)$ is the action selected by the current deterministic policy π at step t, we actually perform action $\tilde{a}_t := a_t + z_t$, where z_t is a time-discretization invariant random process defined via an Ornstein-Uhlenbeck process (Uhlenbeck and Ornstein, 1930).

Finally, we also show how we can define a time-discretization invariant optimization procedure:

- We show that with a SGD algorithm, the learning rate needs to be of order $O(\delta t)$. If it is *larger*, the algorithm will diverge, if it is *smaller*, the parameters stay at their initial values (Theorem 5.5).

We then provide experiments comparing DQN or DPPG to DAU:

- We empirically show that standard Q-learning methods are not robust to changes in time discretization in continuous control environments (Brockman et al., 2016), exhibiting degraded performance, while our algorithm demonstrates substantial robustness.

3 Policy evaluation via the successor states operator

In Part IV, we present our work on policy evaluation via the successor states operator. This part is mainly based on the following preprint, with improved and additional results:

> Blier, L., Tallec, C., and Ollivier, Y. (2021). Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*

In an environment with a very sparse reward, learning the value function as described in Section 1.4.2 is a hard problem. At the beginning of training, no learning will occur until a reward is observed. This highlight the fact that not all the observed information is used. Leveraging this information might lead to better sample efficiency.

The *successor state operator* of a Markov reward process is an object that expresses the value functions of all possible reward functions for a given, fixed policy. Here we offer a formal treatment of these objects in both finite and continuous spaces. We present several learning algorithms and associated results. There are multiple motivations for this approach:

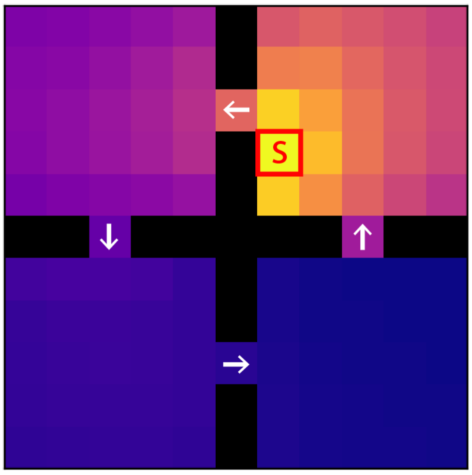

Figure 1: A learned successor states with function approximation in a maze. We represent the successor measure $M(s, s')$, where the state s marked in red is the starting state. The arrows represent doors which can only be passed in one direction.

First, learning the successor state operator is done without reward signals and extracts information from every observed transition, illustrating an *unsupervised reinforcement learning* approach. Successor state lie in between model-free and model-based reinforcement learning approaches, providing a representation of the future of a state without having to synthesize future states or unrolling synthetic trajectories.

Then, successor states exploit multiple relationships between how to reach different states. Similarly to the value function, it satisfies a Bellman equation. Additionally, it also satisfies two other fixed point equations: a *backward* Bellman equation and a *Bellman–Newton* equation, expressing path compositionality in the Markov process. These new relation allow us to generalize from observed trajectories in several ways, potentially leading to more sample efficiency.

Finally, the study of the successor states operator and its algorithms allow us to derive, in Section 4, unbiased methods in the setting of multi-goal RL, dealing with the issue of extremely sparse rewards.

3.1 The successor states operator

The *successor state operator* $M^\pi(s, \mathrm{d}s')$ of a Markov reward process is an object that linearly *transforms* a reward function into the corresponding value function. In particular, it expresses the value functions of all possible reward functions for a given, fixed policy. For finite spaces, the entries $M_{ss'}$ of the successor state matrix describe the expected discounted time spent in state s' by a trajectory starting at s Dayan (1993) (see Figure 1):

$$M^\pi_{ss'} = \sum_{t\geqslant 0} \gamma^t \mathbb{P}(S_t = s' | S_0 = s) \tag{3.1}$$

Equivalently:

$$M^\pi_{ss'} = \mathbb{E}_{a_t\sim\pi(a|s_t), s_{t+1}\sim P(s'|s_t,a_t)} \left[\sum_{t\geqslant 0} \gamma^t \mathbb{1}_{s_t=s'} | s_0 = s \right]. \tag{3.2}$$

The entry $M_{ss'}$ is also the value function at s if the reward is 1 at s' and 0 everywhere else. As such, M contains the information about *reaching* every state in the environment, not just those states providing a reward.

For a fixed policy π, the value function V^π depends linearly on the reward: in a finite state space, for any reward function, represented as a vector R over states, its associated value function is

$$V^\pi(s) = (M^\pi R)(s) = \sum_{s_2} M^\pi_{ss_2} R_{s_2}. \tag{3.3}$$

This equation will allow us to derive policy evaluation algorithms via successor states. First, we will estimate models of the successor state operator M^π. As M^π does not depend on the reward, we can learn it in an *unsupervised* way, without observing any reward. Contrary to standard value function learning algorithms, the algorithm can start to learn before observing any reward. Then, we can use the successor states model to *compute* a model of the value function once the reward function is observed, using (3.3).

In the following, we study how to learn a model of the successor state operator and how to use it. In finite environments with tabular models, this was studied by Dayan (1993), and we will extend this principle to general continuous state spaces, with function approximators. In Chapter 6, we introduce the successor states operator, and its proper definitions of in continuous state space:

- We formally define the successor states operator in general state spaces (Theorem 6.1), extending the discrete case of Dayan (1993). For continuous states, this involves some measure theory: the successor state operator defines for every state s a measure over successor states s':

  $$M^\pi(s, \mathrm{d}s') = (\mathrm{Id} - \gamma P)^{-1}(s, \mathrm{d}s') = \sum_{t\geqslant 0} \gamma^t (P^\pi)^t (s, \mathrm{d}s') \tag{3.4}$$

  We relate the value function V^π to the successor states operator M^π and the reward R in general state spaces (Proposition 6.3) via:

  $$V^\pi(s) = (M^\pi \cdot R)(s) = \int_{s_2} M^\pi(s, \mathrm{d}s_2) R(s_2) \tag{3.5}$$

We then describe how to represent the successor operator $M^\pi(s, \mathrm{d}s')$ with function approximators, as a density $m_\theta(s, s')$ with respect to a reference measure $\varphi(\mathrm{d}s')$:

$$M_\theta(s, \mathrm{d}s') := m_\theta(s, s')\rho(\mathrm{d}s') \tag{3.6}$$

The reference measure ρ is a probability distribution, such that we are able to sample $s \sim \rho$. We do not require more knowledge on ρ, such as knowing its probability density function, ... Typically, ρ can be the distribution of states observed along trajectories sampled with π with an initial state $s_0 \sim \rho_0$. Unless specified, there are no hypothesis on ρ. For $m_\theta(s_1, s_2)$, we can use any parametric family of functions. In practice, we will use deep learning models.

The successor states operator is related but not equivalent to *successor features* (Kulkarni et al., 2016; Borsa et al., 2018; Barreto et al., 2018; Zhang et al., 2017; Hansen et al., 2020). Given a *feature* function φ over the state space $\mathcal{S}$, the *successor feature* is the expectation of the cumulated, discounted future values of φ given the starting point s_0 of a trajectory (s_t) is

$$\mathbb{E}\left[\sum_{t\geqslant 0} \gamma^t \varphi(s_t)\right] = \sum_{t\geqslant 0} \gamma^t (P^t \varphi)(s_0) = (M\varphi)(s_0). \tag{3.7}$$

Thus, the successor representation of a state s is obtained by applying M to some user-chosen function φ: $M^\pi \cdot \varphi$. In practice the function φ is learned together with the successor feature. Still, in order to prevent convergence to the trivial solution $\varphi = 0$, an additional loss (such as pixel reconstruction) has to be added. On the contrary, the successor states operator does not depend on a given function φ, and can be learned without adding any information independent of the dynamic.

The next steps are then to describe how this density model $m_\theta(s, s')$ can be learned, and used for policy evaluation.

3.2 TD algorithms for deep successor states

Once the successor states operator is properly defined, the goal is now to *learn* it. We consider a *model* $M_\theta(s_1, \mathrm{d}s_2) := m_\theta(s_1, s_2)\rho(\mathrm{d}s_2)$ parameterized by its density $m_\theta(s_1, s_2)$ with respect to a measure ρ, as introduced previously. In Chapter 7, we derive a temporal difference algorithm for learning $m_\theta(s_1, s_2)$. We extend the standard temporal difference approach with function approximators described in Section 1.4.2 to the successor state.

Following this strategy, our first contribution is define such an operator T for the successor state operator:

- We define the (forward) Bellman operator for successor operators:

$$T \cdot M := \mathrm{Id} + \gamma P \cdot M, \tag{3.8}$$

which corresponds to the standard Bellman equation for value functions, but for successor states.

We show that that the Bellman operator is γ-contractive, and that its unique fixed point is the true successor states operator M^{π} (Theorem 7.1 and Proposition 7.2).

Once such an operator is defined, we can derive a stochastic Temporal Difference estimate for successor states:

- We define the stochastic update $\widehat{\delta\theta}_{\text{F-TD}}(s, s', s_2)$, where we assume (s, s') is a transition observed in the Markov Process and s_2 is an independent state. The update is:

$$\widehat{\delta\theta}_{\text{F-TD}}(s, s', s_2) := \partial_\theta m_\theta(s, s) + \partial_\theta m_\theta(s, s_2)\left(\gamma m_\theta(s', s_2) - m_\theta(s, s_2)\right) \quad (3.9)$$

  We then define the corresponding Forward TD algorithm for successor states operator (Algorithm 8): informally, at step t, if our current parameter is θ_t, when observing a transition (s_t, s'_t) in the environment, we sample $s_2 \sim \rho$ and define:

$$\theta_{t+1} = \theta_t - \eta_t \widehat{\delta\theta}_{\text{F-TD}}(s_t, s'_t, s_2) \quad (3.10)$$

  We prove that $\widehat{\delta\theta}_{\text{F-TD}}$ is an unbiased estimate of the Bellman error (Theorem 7.5): if we define the target $M^{\text{tar}} := T \cdot M_\theta = \mathrm{Id} + \gamma P \cdot M_\theta$, we have:

$$\mathbb{E}_{s,s',s_2}\left[\widehat{\delta\theta}_{\text{F-TD}}(s, s', s_2)\right] = \frac{1}{2}\partial_\theta \|M_\theta - M^{\text{tar}}\|^2 \quad (3.11)$$

  where formal definition of the probability laws in the expectation, and of the norm $\|.\|$ are given in the corresponding sections.

The strategy used here is similar to standard policy evaluation for the value function with function approximations described in Section 1.4.2, and will be used to derive many algorithms in this thesis: First, we define a contractive *operator* T such that its unique fixed point is M^{π}. Hence, for any initialization $M_0(s, \mathrm{d}s')$, if we define the sequence $M_{t+1} := T \cdot M_t$, we have $M_t \to_{t\to\infty} M^{\pi}$. Then, we approximate the sequence M_t with function approximation. We define a model $M_\theta(s, \mathrm{d}s') = m_\theta(s, s')\rho(\mathrm{d}s')$ as introduced above. When observing a trajectory/observation/transition, we compute a stochastic update $\widehat{\delta\theta}$, such that it is an unbiased gradient step towards the target $M^{\text{tar}} := T \cdot M_{\theta_t}$: $\mathbb{E}[\widehat{\delta\theta}] = \frac{1}{2}\partial_\theta \|M_\theta - M^{\text{tar}}\|^2$, and we update θ with $\theta_{t+1} := \theta_t - \eta\widehat{\delta\theta}$.

As explained for V-function in Section 1.4.2, with such an algorithm, M^{π} is guaranteed to be a fixed point: if there is θ^* such that $M_{\theta^*} = M^{\pi}$, then $\mathbb{E}[\widehat{\delta\theta}] = 0$.

Moreover, if the parametric family M_θ is overparametrized then M^π is the *unique* fixed point of the algorithm.

Experimentally, we demonstrate that we are able to approximate the successor states operator in simple continuous environments with the Forward TD algorithm, with deep neural networks.

We know that several variants of TD are used in practice for learning the value function. We derive similar variants in the context of successor states operator:

- We show how to compute an unbiased forward temporal difference update with an additional target network (Theorem 7.6).
- We define a $TD(n)$ update $\widehat{\delta\theta}_{\mathrm{TD}(n)}$ for the successor states operator. Similarly to the forward TD update, the TD(n) update is an unbiased estimate of the n-step Bellman error (Theorem 7.7).
- We show how to define and learn the successor state-action operator $M^\pi(s, a, \mathrm{d}s')$, which takes into account the first action (similarly to the Q function) (Definition 7.13, Theorems 7.14 and 7.15).

Forward Temporal Difference for successor states corresponds to standard Temporal Difference for the value function, but on the successor states operator. We will now present new algorithms for learning the successor states operator, without any equivalent for the value function, leveraging the additional information contained in the object.

3.3 The Backward Temporal Difference Algorithm

Informally, the *forward* TD algorithm is using that, for every target state s_{tar}, if we observe a transition $s \to s'$ (hence s and s' are close to each other), then their value functions must be close. In this section, we present the *backward* temporal difference algorithm, which is using the opposite point of view: for every starting state s_{start}, if we observe a transition $s \to s'$, then the value function of s_{start} if the reward is localized in s must be close to the value function of the same state s_{start} if the reward is localized in s'.

Similarly to forward Temporal Difference, this relation can be formalized as a fixed point equation over the successor states operator:

- We define the backward Bellman operator for successor states operator as:

  $$M \mapsto \mathrm{Id} + \gamma M \cdot P \tag{3.12}$$

  We show that this operator is γ-contractive and that its unique fixed point is the true successor states operator M^π (Theorem 8.1 and Proposition 8.2).

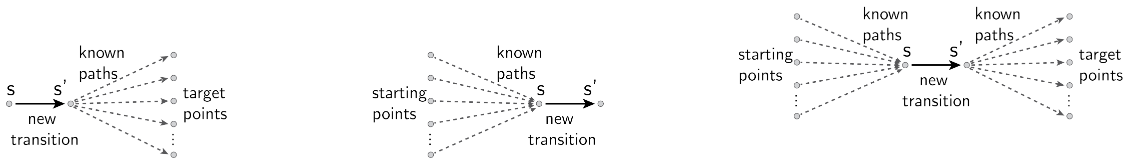

Figure 2: Combining paths: forward TD, backward TD, and path composition (Bellman–Newton).

Once this operator is defined, we obtain a algorithm for function approximators similarly to forward temporal difference:

- We define the stochastic update $\widehat{\delta\theta}_{\text{B-TD}}(s_2, s, s')$, where we assume (s, s') is a transition observed in the Markov Process and s_1 is a state sampled independently. We then define the corresponding Backward TD algorithm for successor states operator.

  We prove that $\widehat{\delta\theta}_{\text{B-TD}}$ is an unbiased estimate of the backward Bellman error (Theorem 8.3).

Finally, we analyze the backward TD algorithm from the *backward process* viewpoint. Informally, if P is the transition matrix of a Markov process, the *backward* process is the process corresponding to (infinite) trajectories $(..., s_{-2}, s_{-1}, s_0, s_1, s_2, ...)$ in the reversed order $(..., s_2, s_1, s_0, s_{-1}, s_{-2}, ...)$. Viewing the backward TD algorithm from this viewpoint lead to the following contribution:

- We show that the backward TD update on operators is equivalent to the forward TD update applied to the backward process (Theorem 8.5).

While the updates $\widehat{\delta\theta}_{\text{B-TD}}$ and $\widehat{\delta\theta}_{\text{F-TD}}$ are not equal, there is therefore a strong relation between them. Still, the forward TD update has an equivalent update on the value function while the backward TD update does not. In the next section, we introduce second-order methods for successor states operator learning.

3.4 Second-order methods for successor states

The Bellman–Newton operator and path concatenation In order to introduce our second-order approaches, let us first give an interpretation of M^π, forward and backward TD in terms of *paths* in the environment. In the finite environment case, we can express $M^\pi_{ss'}$ as a sum over all paths from s to s' in the environment

$$
\begin{aligned}
M^\pi_{ss'} &= (\mathrm{Id} - \gamma P^\pi)^{-1}_{s_1 s_2} = \sum_{t \geqslant 0} \gamma^t (P^\pi)^t_{s_1 s_2} \\
&= \sum_{t \geqslant 0} \gamma^t \sum_{\substack{\text{path}(s_0, s_1, \ldots, s_t) \\ \text{with } s_0 = s \text{ and } s_t = s'}} P^\pi_{s_0 s_1} P^\pi_{s_1 s_2} \ldots P^\pi_{s_{t-1} s_t}. = \sum_{\substack{p \ \text{path from} \\ s \text{ to } s'}} \gamma^{\texttt{length}(p)} \mathbb{P}(p)
\end{aligned}
$$

where $\mathbb{P}((s_0, ..., s_t)) = P^\pi_{s_0 s_1} ... P^\pi_{s_{t-1} s_t}$. Therefore, $M^\pi_{ss'}$ is the *sum* of all paths from s to s', discounted by their length, and weighted by their probabilities.

From this viewpoint, we can give an other interpretation of forward and backward TD: When a transition $s \to s'$ is observed, for every target state s^{tar} forward TD builds *new* paths from s to s^{tar} by concatenating the transition (s, s') to all known paths from s' to s^{tar} (Figure 2, left). On the contrary, when $s \to s'$ is observed, for every starting state s^{start}, backward TD builds new paths from s^{start} to s' by concatenating all known paths from s^{start} to s with the transition (s, s') (Figure 2, middle).

This discussion naturally leads to a *third* algorithm: when observing a transition (s, s'), it is possible to build new path from any starting point s^{start} to any target state s^{tar} by concatenating all paths from s^{start} to s, to the transition (s, s'), to all paths from s^{tar} (Figure 2, right). Informally, instead of increasing the length of all known paths by 1 at every step, this would double the length of all known paths at every step.

In Chapter 9, we define the Bellman-Newton equation and the corresponding algorithm, corresponding the path composition strategy defined above:

- We define the Bellman-Newton operator:

$$M \mapsto 2M - M \cdot (\mathrm{Id} - \gamma P^\pi) \cdot M \tag{3.13}$$

  and show that the true successor states operator M^π is a fixed point of the Bellman–Newton operator (Proposition 9.4).

  We show that the update defined via the Bellman–Newton equation $M_{t+1} := 2M_t - M_t \cdot (\mathrm{Id} - \gamma P^\pi) \cdot M_t$ corresponds to the path concatenation strategy described above (Theorem 9.6). It also corresponds to the Newton method for matrix inversion (Pan and Schreiber, 1991), which explains the name given to the Bellman Newton method method.

Using the Bellman–Newton operator, we then define a Bellman–Newton update for successor states with function approximation:

- We define the stochastic update $\widehat{\delta\theta}_{\text{BN}}(s_1, s, s', s_2)$, where we assume (s, s') is a transition observed in the Markov Process and s_1 and s_2 are states sampled independently. We then define the corresponding Bellman–Newton algorithm for successor states operator (Algorithm 9).

  We prove that $\widehat{\delta\theta}_{\text{BN}}$ is an unbiased estimate of the Bellman–Newton error (Theorem 9.5).

Experimentally, this update raises multiple issues. First, values of $m(s_1, s_2)$ can reach several order of magnitudes. Typically, when $s_1 \approx s_2$ the values can go to infinity, but be of order $O(1)$ in every other cases, and these two regimes needs to be learned accurately (this first issue is shared by every algorithm learning the successor state operator). Then, the update $\widehat{\delta\theta}_{\mathrm{BN}}(s_1, s, s', s_2)$ requires the sampling of one transition (s, s'), and two additional states s_1 and s_2. On the contrary, Forward TD and Backward TD only require the sampling of a single additional state. Hence, the Bellman–Newton update has a variance. Finally, it is known that Newton's methods can be numerically unstable. Hence, the high variance becomes important, as the method can diverge. To counterbalance this variance, we can use smaller learning rates, but this reduces the efficiency of the method.

These issues makes a vanilla implementation of the Bellman–Newton algorithm not efficient in continuous environments with function approximations. In Chapter 11 introduced below, we present a possible solution to this issue, via low-rank parametrization. We will see that this approach allow us to reduce variance, and still be close to a Bellman–Newton approach in some cases.

Still, we prove that in tabular cases, the Bellman–Newton algorithm approximates the *process estimation* algorithm (Theorem 9.2), defined as follows: in a tabular setting, we can learn an estimate $\hat{P}$ of the transition matrix P^π by keeping a *frequency matrix* of every transition starting from every state. Then, we can define an estimate of M^π as $\hat{M} := (\mathrm{Id} - \gamma\hat{P})^{-1}$. If we also learn $\hat{R}$ a model of the reward, we can define a model of the value function as $\hat{V} := \hat{M}\hat{R}$. This algorithm trivially converges to M^π and V^π when the number of samples goes to infinity, because the matrix inverse is continuous. It corresponds to the *Least Squares Temporal Difference* algorithm in the literature (Bradtke and Barto, 1996).

This strategy is only possible in the tabular setting, and has no direct equivalent in the function approximation setting. Indeed, in a continuous environment, if we learn a model $P_\theta(s_1, \mathrm{d}s_2)$ of the transition operator, we can't directly compute the inverse $(\mathrm{Id} - \gamma\hat{P})^{-1}$. Using a continuous model $P_\theta(s_1, \mathrm{d}s_2)$ would still be possible, for instance by sampling trajectories according to P_θ. This is related to model-based methods, and has known technical issues, discussed in Section 1.3.

- In the tabular setting, the Bellman–Newton algorithm approximates the *process estimation* algorithm (Theorem 9.2), while never estimating the process directly.

In the next Section, we study theoretically the convergence properties of the process estimation algorithm.

3.5 A non-asymptotic convergence bound for policy evaluation via process estimation

In order to better understand how much the Bellman–Newton algorithm can potentially improve the sample efficiency of policy evaluation, we study in more details in Chapter 10 the *tabular case*.

We therefore study the sample efficiency of policy evaluation with the *process estimation* algorithm, which corresponds to the LSTD algorithm (Bradtke and Barto, 1996) in its very specific tabular case. We consider the i.i.d data model: we assume the process is ergodic and has an invariant measure ρ. Then, we observe *independent* transitions (s_t, r_t, s'_t) of the environment such that $s_t \sim \rho$, $r_t \sim \mathcal{R}(.|s_t)$, $s'_t \sim P(.|s_t)$, and we consider the $L_1(\rho)$ norm on $\mathcal{S}$ defined as $\|f\|_{L_1(\rho)} = \sum_{s\in\mathcal{S}} \rho(s)|f(s)|$. We also assume that the reward r is bounded by $R_{\max}$.

We consider the process estimation algorithm defined in the previous section: we learn $\hat{P}_t$ as a frequency matrix: $\hat{P}_t = \frac{n_{ss'}}{n_s}$ where $n_{ss'}$ is the number of times the transition (s, s') was observed, and $n_s = \sum_{s_2} n_{ss_2}$. Similarly, we learn $\hat{R}_t$ as $\hat{R}_t(s) = \frac{1}{n_s}\sum_{k\leqslant t|s_k=s} r_k$. Then, we estimate the value function as $\hat{V}_t = (\mathrm{Id} - \gamma\hat{P}_t)^{-1}\hat{R}_t$. This estimate clearly converges to the true value function V^π, as $\hat{P}_t \to P^\pi$, $\hat{R}_t \to R$ almost surely, and the inverse is continuous. We are interested into measuring the sample efficiency of this approach.

We provide a convergence bound on $\hat{V}_t$ with the process estimation algorithm for the $L_1(\rho)$ norm. The most interesting feature this new convergence bound is that it does not depend on the number of states, or of the measure of infrequently visited states. Hence the result is non-vacuous even if some states are almost-never observed, or for a very large number of states, or even for a discrete infinite state space. Up to our knowledge, it is the first convergence bound for policy evaluation which shows that we can provably learn the value function in finite time, even with an arbitrarily large (or infinite) state space. This is a desirable result: if a state s is *almost never observed* for the measure ρ, an estimate $V(s)$ will clearly be inaccurate, but because we consider the $L_1(\rho)$ or $L_2(\rho)$ norms which weights the error in state s with ρ_s, this error is s should be controlled.

We use the following quantity $\Lambda_t(\rho P)$, introduced by Cohen et al. (2020) in the context of discrete distribution learning:

$$\Lambda_t(\rho P) := \sum_{(s,s')|\rho_s P_{ss'}<1/t} \rho_s P_{ss'} + \frac{1}{\sqrt{t}} \sum_{(s,s')|\rho_s P_{ss'}\geqslant 1/t} \sqrt{\rho_s P_{ss'}} \tag{3.14}$$

The sequence $\Lambda_t(\rho P)$ is decreasing when t increases. We easily have $\Lambda_t(\rho P) \geqslant \frac{1}{\sqrt{t}}$. Moreover, if $\mathcal{S}$ is finite, then we have $\Lambda_t(\rho P) \leqslant \sqrt{\frac{E}{t}}$ where E is the number of *edge* in the graph ((s, s') is an edge if $P^\pi_{ss'} > 0$). The equality corresponds to an environment in which the probability distribution $P(.|s)$ is uniform for every state s. More generally, the quantity $\Lambda_t(\rho P)$ is lower if when the distribution is short tail. Interestingly, the quantity $\Lambda_t(\rho P)$ is still well defined when $\mathcal{S}$ is infinite, and can handle a large number of states with low probability.

- We consider the process estimate $\hat{P}_t$ defined above, obtained by keeping a *frequency matrix* of every transition starting from every state, and similarly a tabular model $\hat{R}_t$ of the reward. We define the value estimate as $\hat{V}_t := (\mathrm{Id} - \gamma\hat{P})^{-1}\hat{R}_t$. Then, after t i.i.d. observations ($s \sim \rho$, $s' \sim P_{ss'}$), we have with probability $1-\delta$ (Theorem 10.2):

$$\|\hat{V}_t(s) - V(s)\|_{L_1(\rho)} \leqslant \frac{R_{\max}}{(1-\gamma)^2}\left(10\Lambda_t(\rho P) + 9\sqrt{\frac{\log\frac{4}{\delta}}{t}}\right) \tag{3.15}$$

For a finite environment, we can use that $\Lambda_t(\rho P) \leqslant \sqrt{\frac{E}{t}}$ and the bound corresponds to

$$\|\hat{V}_t(s) - V(s)\|_{L_1(\rho)} \leqslant \frac{R_{\max}}{(1-\gamma)^2\sqrt{t}}\left(10\sqrt{E} + 9\sqrt{\log\frac{4}{\delta}}\right). \tag{3.16}$$

We compare these convergence to known results for policy evaluation. In particular, we consider the results from Bhandari et al. (2018) for Temporal Difference for the norm $L_2(\rho)$ under the same i.i.d observation model, and the results from Pananjady and Wainwright (2019), for an algorithm equivalent to SSIPE (called the *plug-in* in their work), for the L_∞ norm, under the *synchronous* observation model (at every step, a transition from every state is observed).

From these comparisons, our bounds raises a few interesting properties. First, they are remarkably simple, and only depend on γ, $R_{\max}$, and the number of edges in the graph (or $\Lambda_t(\rho P)$). Then, it is, to our knowledge the first bound for policy evaluation, for i.i.d. (or trajectory) data, which is non-vacuous even when some states are hardly ever visited ($\rho(s)$ is very small), or when the number of states goes to infinity.

3.6 Matrix Factorization and the Forward-Backward parametrization

Finally, we get back to continuous environments, and study a specific parametric model for the successor state operator, in order to mitigate the variance issue of the Bellman–Newton method. We consider the model $M_\theta(s_1, \mathrm{d}s_2) = m_\theta(s_1, s_2)\rho(\mathrm{d}s_2)$ with the particular choice:

$$m_\theta(s_1, s_2) = \langle F_{\theta_F}(s_1), B_{\theta_B}(s_2)\rangle = \sum_{i=1}^{r}\left(F_{\theta_F}(s_1)\right)_i\left(B_{\theta_B}(s_2)\right)_i \tag{3.17}$$

where $F\colon S \to \mathbb{R}^r$ and $B\colon S \to \mathbb{R}^r$ are two learnable functions from the state space to some representation space $\mathbb{R}^r$, parameterized by $\theta = (\theta_F, \theta_B)$. This provides an approximation of M by a rank-r operator. Such a factorization is used for instance in (Schaul et al., 2015) for the goal-dependent Q-function (up to the factor ρ). Intuitively, F is a "*forward*" representation of states and B a "*backward*" representation: if the future of s_1 matches the past of s_2, then $M(s_1, \mathrm{d}s_2)$ is large.

The forward-TD and backward-TD algorithms introduced above can be directly applied to the FB parametrization, simply by considering $m_\theta(s_1, s_2)$ as any function approximator. Actually, it is also possible to *mix* the forward and backward updates: if we consider θ_B as fixed, we can use the forward or backward TD algorithms on the model $\theta_F \mapsto \langle F_{\theta_F}(.), B_{\theta_B}(.)\rangle$. This defines forward and backward updates for F_{θ_F}, and we can define similarly forward and backward updates for B_{θ_B}. We can then mix these updates and use independently a forward/backward update for F and a potentially different update for B_{θ_B}.

- We define the *mixed* algorithms *forward-forward*, *forward-backward*, *backward-forward* and *backward-backward* for the FB parametrization. We show that the true successor state operator M^π is a fixed point of every of these four algorithms (Theorem ref11.1).

We then study in more details the *forward-backward* algorithm. Indeed, this algorithm has two interesting properties. The first one is about variance. We learn online estimates $\hat{\Sigma}_F$ and $\hat{\Sigma}_B$ of the $r \times r$ covariance matrices Σ_F and Σ_B defined as $\Sigma_F := \mathbb{E}_{s_1 \sim \rho} F(s_1)F(s_1)^\top$ and $\Sigma_B := \mathbb{E}_{s_1 \sim \rho} B(s_1)B(s_1)^\top$, for example by computing a moving average of the matrices $F(s)F(s)^\top$ for every observed state. Then, we define a FB update with reduced variance:

- Knowing estimates $\hat{\Sigma}_F$ and $\hat{\Sigma}_B$ of Σ_F and Σ_B, we define an update $\widehat{\delta\theta}_{\text{fb-TD}}(s, s', \hat{\Sigma}_F, \hat{\Sigma}_B)$, where (s, s') is supposed to be an observed transition in the process (Algorithm 10).

  Contrary to the Forward or Backward TD algorithm defined for any model m_θ, this update does not require the sampling of an additional state s_2.

  We prove that, if we our estimates of the covariance matrices are correct ($\tilde{\Sigma}_F = \Sigma_F$ and $\tilde{\Sigma}_B = \Sigma_B$), the update $\widehat{\delta\theta}_{\text{fb-TD}}$ is an unbiased estimate of the Forward Bellman error gradient for F and of the Backward Bellman error for B, but with lower variance (Theorem 11.2).

The second interesting property is a relation between the fixed points of this method and the SVD. We know that the optimal low-rank approximation of an operator for the L_2 norm corresponds to a truncated SVD. We have the following result:

- We show that the fixed point of the forward-backward TD algorithm are truncated SVDs of rank r of the true successor states operator M^π for the norm $L_2(\rho)$ (Theorem 11.3).

This statement is necessary but not sufficient to show that the algorithm will converge to the optimal low-rank representation. In practice, we observe that this algorithm converges to the optimal low-rank representation of the successor state operator in simple environments.

Additionally, these representations might be useful for other purposes. Once state (or state-actions) representation are computed, they can be used directly as input of a more simple policy (Ha and Schmidhuber, 2018). They can also be used to derive a bonus for exploration (Machado et al., 2019). These FB representations only depend on the dynamics and not on other signals (such as pixels), which can be irrelevant for the task and biased representation learning toward ignoring the most important information.

Finally we show that the Forward-Backward algorithm is related to the Bellman–Newton update defined in the previous section:

- In a limited setting (tabular, in a reversible process in which the uniform measure is the invariant measure of the process), we show that for small learning rates, the Forward-Backward update is equivalent to Bellman–Newton update (Theorem 11.6).

This result is interesting, as it suggests that the FB algorithm might share with the Bellman–Newton algorithm the relation with implicit process estimation, hence its sample efficiency, without the variance issue of Bellman–Newton.

In the next section, we finally describe several methods to learn the value function V^π via the successor states operator.

3.7 Learning value functions via successor states models

In Chapter 12, we describe several methods to learn a model $V_\varphi(s)$ of the value function V^π once we are able to learn a model of the successor states operator. We mainly define two approaches: first, by using the equation $V^\pi(s) = (M^\pi \cdot R)(s)$ and estimating the integral $\int_{s_2} \rho(\mathrm{d}s_2) m_\theta(s, s_2) R(s_2)$. Then, by using m_θ as a way to propagate the Bellman error of Temporal Difference in the environment, similarly to TD(λ) with eligibility traces.

Estimating the value function via $V^\pi = M^\pi \cdot R$ We know that:

$$V^\pi(s) = (M^\pi \cdot R)(s) = \int_{s_2} M^\pi(s, \mathrm{d}s_2) R(s_2) \tag{3.18}$$

Therefore, after learning a model $m_\theta(s_1, s_2)$, we might want to use the model:

$$\hat{V}(s) := \int_{s_2} \rho(\mathrm{d}s_2) m_\theta(s, s_2) R(s_2) \tag{3.19}$$

. We introduce three cases in which we can estimate the integral (3.19):

- If the reward is sparse and located in a single known state s^{tar}, equation (3.19) corresponds to:
$$\hat{V}(s) = m_\theta(s, s^{\text{tar}}) \tag{3.20}$$
up to a multiplicative factor independent of s. Hence, we can directly use $m_\theta(s, s^{\text{tar}})$ as an estimate of the value function V^π.

- If the reward is dense, we consider a model $V_\varphi(s)$ (such as a neural network), and optimize the parameter φ such that $V_\varphi \approx \hat{V}$. We store a *buffer* of tuples (s, r_s) where s is an observed state and r_s the reward observed in s. Then, we can learn V_φ in a supervised way: by sampling (s, r_r) in the buffer and an additional independent state s_1, and reduce the empirical loss $(V_\varphi(s_1) - m_\theta(s_1, s)r_s)^2$

  This approach reduces the problem of policy evaluation to a supervised learning problem (once a model m_θ is learned)

- If the reward is dense and we additionally use the Forward-Backward parametrization described in the previous section ($m_\theta(s_1, s_2) := \langle F_{\theta_F}(s_1), B_{\theta_B}(s_2)\rangle$ where $F(s)$ and $B(s)$ are low rank representations in $\mathbb{R}^k$), we have: $\hat{V}(s) = \mathbb{E}_{s_2\sim\rho}\left[\langle F_{\theta_F}(s), B_{\theta_B}(s_2)\rangle\right] = \langle F_{\theta_F}(s), b\rangle$ where $b := \mathbb{E}_{s_2\sim\rho}\left[B_{\theta_B}(s_2)\right]$. In that case, we can estimate b with an online averaging $\hat{b} \in \mathbb{R}^k$ of state representations ($\hat{b} := \frac{1}{t}\sum_{i=1}^{t} B_{\theta_B}(s_i)$), and then use the following estimate for the value function: $\hat{V}(s) = \langle F_{\theta_F}(s), \hat{b}\rangle$. With this approach, there is no need to learn an additional parametric model V_φ of the value function.

Using M to propagate the Bellman error in the environment: expected value update via process estimation, and expected TD(λ) update We now consider an other approach for policy evaluation via the successor states operator, in which the successor state model m_θ is used to propagate the Bellman error in the environment, or in other words to improve the *credit assignment* when observing a transition (s, r, s').

First, we derive this method from the expected value update via the process estimation approach in the tabular case, introduced in Section 3.4 of this overview, defined as $\hat{V}_t := \hat{M}_t\hat{R}_t$, where $\hat{M}_t := (\text{Id} - \gamma\hat{P}_t)^{-1}$ and $\hat{P}_t$ is the frequency matrix of observed transitions.

- We show that, in expectation over the transition observed at step t (s_t, r, s_{t+1}), conditionally to the current estimates V_t, M_t, we have $\mathbb{E}_{s_t\sim\rho, s'_t\sim P(.|s_t), r\sim\mathcal{R}(.|s)}\left[V_{t+1}\right] = V_t + \frac{1}{t}\delta V + o(1/t)$, where $\delta V = M_t(R + \gamma P V_t - V_t)$ (Theorem 12.1).

Informally, this equation means that when observing a transition (s, r, s'), the bellman error $(r_s + \gamma \hat{V}_t(s') - \hat{V}_t(s))$ is propagated to every state s_1 with weight $\hat{M}_t(s_1, s)$. Hence, $M_t(s_1, s)$ propagates the *credit* in the entire environment. We then generalize this update for function approximations:

- Once we know a model $m_\theta(s_1, s_2)$ of the successor states density, we define the stochastic update $\widehat{\delta\varphi}_{\text{prop-TD}}(s, s', r, s_1)$ for the value function V_φ where we assume (s, r, s') is a transition observed in the Markov Process and s_1 is a state sampled independently, as: $\widehat{\delta\varphi}_{\text{prop-TD}}(s, s', r_s, s_1) = \partial_\varphi V_\varphi(s_1) m_\theta(s_1, s) \left(r_s + \gamma V_\varphi(s') - V_\varphi(s)\right)$ (Algorithm 13).

  We prove that $\widehat{\delta\varphi}_{\text{prop-TD}}$ is an unbiased estimate of $\|V_\varphi - V^{\text{tar}}\|^2_\rho$, where $V^{\text{tar}} := V_\varphi + \delta V$ and δV is defined as in the tabular expected value update via process estimation: $\delta V = M_\theta(R + \gamma P V_\varphi - V_\varphi)$ (Theorem 12.2).

Hence, this method can be seen as an approximation of the online update of the value function for the process estimation method, with function approximations.

We then show in Section 12.2.3 that this update corresponds to an estimate of the expected eligibility traces update in TD(λ). Eligibility traces, introduced in Section 1.4.2 are a way to improve credit assignment by propagating the Bellman error to the states recently visited in the current trajectory. We show that our approach is tackling credit assignment by propagating the Bellman error to all states which could have been visited from the current state s, according to the distribution of *predecessor* states, which is equivalent to the expected traces for a state s.

- We prove that the TD(λ) update with eligibility traces and the update $\widehat{\delta\varphi}_{\text{prop-TD}}$ estimating the value update via process estimation are both approximating the expected eligibility traces (Theorem 12.3).

This method is closely related to *expected eligibility traces* (van Hasselt et al., 2020), and *source traces* (Pitis, 2018), both discussed in Section 12.2.3.

One of our issues while working on this project was to find the proper experimental setup. As discussed in Section 3.4, learning a model $m_\theta(s_1, s_2)$ of the successor state operator raises multiple technical issues. Additionally, we cannot measure the *direct* efficiency of learning the successor states operator to improve our method: we first need to compute a value function, then to plug this estimate into an other RL algorithm such as actor critic, and observe the policy improvement. Hence, our measure of progress was very indirect.

We wanted to focus on a simple setup, but still with continuous state space. The simplest case for deriving the value function from the successor state operator, in a continuous state space, is when the reward is sparse and localized in a known target state s^{tar}. Unfortunately, this is not a frequent setting in standard environments. Still, this setup is quite similar to multi-goal RL, in which the reward is localized in a known *goal state* g. The main difference is that in multi-goal RL, the agent learns a *goal-dependent* policy $\pi(a|s,g)$, whose objective is to *reach* g, while in our setup, the policy was not goal-dependent $\pi(a|s)$.

We extended our approach for the successor state operator with a fixed policy to the multi-goal RL setting. In the next Section, we describe how this approach allows us to derive unbiased Q-learning methods and actor-critic methods for multi-goal RL, dealing with the issue of sparse rewards.

4 Unbiased methods for multi-goal RL

In Part V, we present our work on multi-goal reinforcement learning problems. This part is mainly based on the following preprint:

> Blier, L. and Ollivier, Y. (2021). Unbiased methods for multi-goal reinforcement learning. *arXiv preprint arXiv:2106.08863*

In the last part of the thesis, we study *Multi-goal* reinforcement learning is a specific setting of RL where the agent learns a *goal-dependent policy* $\pi(a|s,g)$, whose objective is to *reach* a goal g in the environment. In this introduction we will consider only states as goals ($g \in \mathcal{S}$), but in Part V the setting is more general.

This setting is not the same the one used in the previous part, for policy evaluation via the successor state operator. In the previous part, via M^π, we are estimating the value function of a *non-goal dependent* policy $\pi(.|s)$ for reaching any goal g. In this part, the policy is different for every goal. Still, we are able to translate some of the tools developed for the successor states for the goal oriented setting. Let $\mathcal{M}$ be a multi-goal environment with state space $\mathcal{S}$ and a goal dependent reward $R(s,g)$. Typically, in a discrete environment, the goal dependent reward is defined as $R(s,g) = \mathbb{1}_{s=g}$, which is the sparse reward, non-zero only if the goal is reached. We try to optimize a policy $\pi(.|s,g)$.

We define the *augmented* environment $\tilde{\mathcal{M}}$ with state space $\tilde{\mathcal{S}} := \mathcal{S} \times \mathcal{S}$, in which the current state $\tilde{s} \in \tilde{\mathcal{S}}$ is defined as $\tilde{s} := (s,g)$, the tuple containing the state in the *original* process s and the currently aimed goal g. In the *augmented* environment, the goal-dependent reward $R(s,g)$ becomes the non-goal oriented reward $\tilde{R}(\tilde{s}) := R(s,g)$, and similarly the goal-oriented policy $\pi(a|s,g)$ becomes the non-goal oriented policy $\tilde{\pi}(a|\tilde{s})$. In that setting, estimating the *multi-goal value function* $V^{\pi(.|.,g)}(s,g)$ is now equivalent to estimating the value function in the augmented environment $\tilde{V}^{\tilde{\pi}}(\tilde{s})$, if the reward is a sparse reward in g. Estimating the value function for every sparse goal is now very similar to estimating the successor states operator in the augmented environment $\tilde{\mathcal{M}}$.

The first known approach for multi-goal RL is with *Universal Value Function Approximators* (UVFA) (Schaul et al., 2015), which extend the classical Q-learning and Temporal Difference (TD) algorithms to the multi-goal setting. It learns the goal-conditioned value-function $V^{\pi}(s, g)$ or Q-function $Q^*(s, a, g)$ for every state-goal pair, with function approximation, via a TD algorithm.

Still, UVFA requires observing rewards, and no learning occurs until a reward is observed. In continuous state spaces, the reward is usually defined as $R_\varepsilon(s, g) = \mathbb{1}_{\|s-g\| \leqslant \varepsilon}$. When $\varepsilon \to 0$, the probability of reaching the reward with a stochastic policy goes to 0, and UVFA can't learn. In practice, UVFA fails in many high dimensional environments, when the probability of reaching the target goal is low and the agent almost never gets any learning signal. We call this phenomena the issue of *vanishing rewards.*

The most popular method in that setting is *Hindsight Experience Replay* (HER) (Andrychowicz et al., 2017). It leverages information between goals via the following principle: trajectories aiming at a goal g but reaching a goal g' can be used for learning exactly as if the trajectory had been aiming at g' from start. This strategy has proved successful in practice and removes the issue of *vanishing rewards*, but is known to be *biased* (Manela and Biess, 2021; Lanka and Wu, 2018).

In the following, we first study some of HER's theoretical properties. Then, we will derive a Q-learning algorithm and an actor-critic algorithm for multi-goal environments.

4.1 A study of Hindsight Experience Replay's bias

While HER has proved successful in practice, it is known to be *biased* (Manela and Biess, 2021; Lanka and Wu, 2018), which means it could converge in some settings to low-return policy. This bias corresponds to a well-known psychological bias (Fischhoff, 1975). In their request for research for robotic multi-goal environments, Plappert et al. (2018) list the necessity for an unbiased version of HER, as such bias can lead to low-return policies.

In chapter 14, we study HER's bias. First, we confirm theoretically that HER is biased:

- We define *counter-example* environments, such that it highlights HER's bias. Theoretically, we prove that HER cannot converge to the true optimal Q-function Q^* in these environments (Theorem 14.2). Empirically, we show that in such environments, HER converges to a low-return policy.

Our second contribution on HER is a *positive* result. We show that despite its bias in general settings, HER is mathematically well-grounded in deterministic environments:

- We show that HER is actually *unbiased* in deterministic environments (Theorem 14.1).

This result vindicates HER for deterministic environments: HER leverages the structure of multi-goal environments, is not vanishing when the rewards are sparse, and is mathematically well-grounded. This covers many usual environments such as robotic environments.

4.2 Multi-goal RL via infinitely sparse rewards

While HER is well-founded in deterministic environments, it is biased in the stochastic case and can learn low-return policies. We now introduce unbiased methods for multi-goal RL in the general setting, including stochastic environments, removing the issue of *vanishing rewards*. We first introduce the setting used to derive our algorithms.

In continuous state spaces, the goal-oriented reward is usually defined as:

$$R_\varepsilon(s,g) = \mathbb{1}_{\|s-g\|\leqslant\varepsilon}. \tag{4.1}$$

When $\varepsilon \to 0$, the probability of reaching the reward with a stochastic policy goes to 0, and for any stochastic policy, the value function $V^\pi_\varepsilon(s,g)$ converges to 0 as well. This is the *vanishing rewards* issue. To avoid this issue, we need a scaling factor, and consider the reward $\frac{1}{\lambda(\varepsilon)}R_\varepsilon(s,g)$, with $\lambda(\varepsilon)$ the volume of the ball of size ε in $\mathcal{S}$. When $\varepsilon \to 0$, this rescaled reward *converges* to the *Dirac reward*:

$$R(s,\mathrm{d}g) := \delta_s(\mathrm{d}g), \tag{4.2}$$

where δ_x is the Dirac measure at x. Intuitively, the Dirac reward $R(s,\mathrm{d}g)$ is infinite if the goal is reached ($s=g$) and 0 elsewhere. Formally, the reward is not a function but a *measure* on the goal space $\mathcal{G}$ parametrized by the state s.

However, even after such a scaling, the UVFA update still vanishes with high probability for small ε (this just scales things by $1/\lambda(\varepsilon)$). We will build algorithms that work directly in the limit $\varepsilon = 0$: replacing the sparse reward $R_\varepsilon(s,g)$ by the *infinitely sparse* reward $R(s,\mathrm{d}g) = \delta_s(\mathrm{d}g)$ will allow us to leverage the Dirac structure to remove the vanishing rewards issue.

The first step is to understand this setting mathematically. In Chapter 13, we formally define multi-goal RL with infinitely sparse rewards, and check that it *corresponds asymptotically* to the original problem with reward R_ε:

- We properly define the infinitely sparse reward via Dirac measures, and show that it corresponds to the *limit* of the reward R_ε.

  Under continuity assumptions, we define the corresponding *expected return with infinitely sparse rewards* $J(\pi)$, and show that its corresponds to the limit of the limit of the return $J_\varepsilon(\pi)$ with reward R_ε when $\varepsilon \to 0$: $J_\varepsilon(\pi) \to_{\varepsilon \to 0} J(\pi)$ (Theorem 13.8).

  Under these assumptions, we show that if $\pi_1(.|s,g)$ and $\pi_2(.|s,g)$ are two goal-conditioned policies, π_1 is *better* than π_2 with infinitely sparse rewards if and only if π_1 is *asymptotically better* than π_2 for reward R_ε when $\varepsilon \to 0$ (Theorem 13.7).

There results allow us to work directly with *infinitely sparse rewards*, even for solving the original problem with reward R_ε, when ε is small. Counter-intuitively, replacing *sparse* rewards by *infinitely sparse* rewards solves the vanishing issue. Instead of waiting an observation of the reward, we can algebraically compute the reward contribution in the updates, leveraging our knowledge on the Dirac function, an obtain non-vanishing algorithms.

In the following sections, we describe how to design Q-learning and actor-critic methods, with no vanishing reward issue, via infinitely sparse rewards.

4.3 Unbiased actor-critic for multi-goal RL

In chapter 16, we describe actor critic methods for multi-goal RL via infinitely sparse rewards. We consider a parametric goal-conditioned policy $\pi_{\theta^\pi}(a|s,g)$. Our goal is to maximize $\theta^\pi \mapsto J(\pi_{\theta^\pi})$, by computing stochastic estimates of $\partial_{\theta^\pi} J(\pi_{\theta^\pi})$. Our goal is to adapt the standard policy gradient theorem stated in Proposition 1.3 in the case of goal-oriented environments: $\partial_{\theta^\pi} J(\pi_{\theta^\pi}) = \mathbb{E}_{s,a,s'}\left[\partial_{\theta^\pi} \log \pi_{\theta^\pi}(a|s)\,(r_s + \gamma V^\pi(s') - V^\pi(s))\right]$. This requires learning a model of the multi-goal value function.

Actually, we show that while the value function satisfies a Bellman equation, it is not directly possible to estimate an unbiased estimate of the Bellman error gradient on the value function. This can be explained because of the double dependency of the value function as a function of the goal: the goal both defines the policy and the reward. These two effects need to be separated in order to get an unbiased estimate of Bellman error gradient. We mitigate this issue by introducing the *successor goal operator* $M^\pi(s, g_1, \mathrm{d}g_2)$. This object is very similar to the successor states operator, and describes the expected discounted time spent in the goal g_2 if following the policy g_1. We have the following relation between the goal-conditioned value measure and the successor goal operator:

- Under continuity assumptions, we can compute the goal conditioned value measure from the successor goal operator, using:

$$V^\pi(s, \mathrm{d}g) = M^\pi(s, g, \mathrm{d}g) \tag{4.3}$$

Hence, if we are able to learn a model $M_\theta(s, g_1, \mathrm{d}g_2) = m_\theta(s, g_1, g_2)\rho(\mathrm{d}g_2)$ for the successor goal operator, we naturally obtain a model $V_\theta(s, \mathrm{d}g) = m_\theta(s, g, g)\rho(\mathrm{d}g)$ of the goal-conditioned value measure (Theorem 13.5).

Therefore, our objective is now to describe how to learn an unbiased estimate of the successor goal operator $M^\pi(s, g_1, \mathrm{d}g_2)$. This can be done by applying some of our results derived for the successor goals operator:

- We define a γ-contractive Bellman operator for the successor goals operator, show that its unique fixed point is the true successor goals operator M^π, and derive an unbiased estimate of the Bellman error's gradient for function approximators (Theorems 16.1 and 13.2). This algorithm removes the vanishing reward issue.

Finally, we are able to derive an unbiased actor-critic algorithm for goal-conditioned policy. This actor-critic update is an extension of the standard actor-critic update defined in Proposition 1.3, but for multi-goal environments:

- We define the actor-critic update $\widehat{\delta\theta}_{\delta\text{-AC}}(s, a, s', g)$

  $$\widehat{\delta\theta}_{\delta\text{-AC}}(s, a, s', g) := \partial_\theta \log \pi_\theta(a|s, g)\left(\gamma m_{\theta_M}(s', g, g) - m_{\theta_M}(s, g, g)\right) \tag{4.4}$$

  where we assume (s, a, s') is a transition observed while aiming for goal g.

  We show that with an accurate model m_{θ_M} of the successor goals operator, $\widehat{\delta\theta}_{\delta\text{-AC}}(s, a, s', g)$ is an unbiased estimate of the policy gradient $\partial_{\theta^\pi} J(\pi_{\theta^\pi})$ (Theorem 16.2).

4.4 Unbiased Q-learning for multi-goal RL

In Chapter 15, we derive an unbiased Q-learning algorithm with infinitely sparse rewards, solving the issue of *vanishing rewards*. Our approach is similar to the strategy described for the successor state operator: first, we define a contractive operator on the space of action-value measures such that its fixed point is the $Q^*(s, a, \mathrm{d}g)$, then we use this operator to define an unbiased Q-learning method with function approximators:

- We formally define the *optimal action-value measure* $Q^*(s, a, \mathrm{d}g)$, and the *optimal Bellman operator for action-value measure*:

  $$Q(s, a, .) \mapsto \delta_s(.) + \gamma \mathbb{E}_{s' \sim P(.|s,a)}\left[\sup_{a'} Q(s', a', .)\right], \tag{4.5}$$

We show that if we define the sequence $Q_{t+1} := T \cdot Q_t$, then $Q_t \to_{t\to\infty} Q^*$, similarly to standard result on the Q-function and the optimal Bellman operator.

Once we defined Q^* and the optimal Bellman operator, we can derive a Q-learning algorithm with function approximations, similarly to our approach for learning the successor state operator. We represent a model $Q_\theta(s, a, \mathrm{d}g) := q_\theta(s, a, g)\rho(\mathrm{d}g)$, where ρ is a reference measure on goals:

- We define the stochastic update $\widehat{\delta\theta}_{\delta\text{-DQN}}(s, a, s', g)$, where we assume (s, a, s') is a transition observed in the Markov Process and g is an independent goal:

  $$\widehat{\delta\theta}_{\delta\text{-DQN}}(s, a, s', g) := \partial_\theta q_\theta(s, a, s) + \partial_\theta q_\theta(s, a, g) \left(\gamma \max_{a'} q_\theta(s', a', g) - q_\theta(s, a, g)\right) \tag{4.6}$$

  We then define the corresponding Q-learning algorithm with infinitely sparse rewards (Algorithm 14). This update removes the issue of vanishing rewards.

  We prove that $\widehat{\delta\theta}_{\delta\text{-DQN}}$ is an unbiased estimate of the optimal Bellman error (Theorem 15.2).

This algorithm can be used with discrete actions such as DQN (Mnih et al., 2013) or with continuous actions such as DDPG (Lillicrap et al., 2015). It is off-policy, hence can be used with any exploration strategy. Informally, the first term is leveraging that, when observing the state s, we have an information on the Q-function on how to reach s. The second term propagates the Q-value on how to reach the target goal g.

Experimentally, we demonstrate that the algorithms using infinitely sparse rewards improves performance of the corresponding method (UVFA) using sparse reward R_ε. In environments designed to exhibit the HER bias issue, we show that HER is unable to learn while unbiased methods can learn the optimal policy. Still, these methods do not perform as well as HER in some standard environments, and are unable to learn at all in more complex environments.

One of the issues of these methods is variance. The Dirac rewards remove the *infinite* variance of vanishing rewards in UVFA when $\varepsilon \to 0$ (first term of (4.6)). But this does not change the way the reward is propagated to other states (second term of (4.6)). Selecting goals g more correlated to the state s as in HER could also be helpful, but this is not obvious to do without re-introducing HER-style bias.

To conclude, in Part V, we prove that there exist unbiased goal-oriented RL algorithms which do not vanish when rewards become sparse: it is possible to deal with sparse rewards in RL directly via the infinitely sparse reward limit, although this does not solve all variance issues. We also prove that another multi-goal method, HER, is unbiased and has the correct fixed point in all deterministic environments

5 Conclusion

In this thesis, we introduced several mathematical approaches and principled methods for deep learning and deep reinforcement learning. Our goal was to understand some crucial properties that principled methods should satisfy, in order to be robust, stable, efficient and work in many settings. Then, we tried to define methods which could satisfy these properties. Along that process, we tried to understand the mathematical objects at the core of our learning settings, how to define them, learn them, and their properties.

Our first project (Chapter 3, Blier and Ollivier (2018)) proposed an information theory viewpoint on the complexity of deep learning models. We proved empirically the ability of deep neural networks to compress the training data even when accounting for parameter encoding. Hence, deep learning models, even with a large number of parameters, compress the data well, and these approaches are principled from an information theory point of view: *the number of parameters is not an obstacle to compression.*

We then introduced *All Learning Rates At Once* (Alrao) (Chapter 4, Blier et al. (2019)), an optimization method for neural networks removing the burden of finding the optimal learning rate, by instead assigning to each unit or feature in the network its own learning rate sampled from a random distribution spanning several orders of magnitude. Surprisingly, Alrao performs close to SGD with an optimally tuned learning rate, for various architectures and problems. Alrao is a principled way to define robust optimization method with no hyperparameter tuning, leveraging the redundancy in the neural network architectures.

In Chapter 5, we studied RL in near continuous time environments (Tallec et al. (2019)), and highlighted empirically and theoretically the lack of robustness of Q-learning approaches to time discretization. We detail a principled way to build an off-policy RL algorithm that yields similar performances over a wide range of time discretizations, and confirm this robustness empirically.

After these projects, we focused our work on an in-depth study of the successor states operator (Part IV, Blier et al. (2021)). This mathematical object is at the core of Reinforcement Learning, as it contains the information on all possible value functions for every possible rewards for a fixed policy, or equivalently the information on all expected occupancy measure starting from any initial point, for a fixed policy. This object was studied in finite environments (Dayan, 1993), but the continuous state spaces raises more technical issues, especially because the operator is not a matrix or a function but a measure. While our first motivation for this project was to improve sample efficiency of policy evaluation, at the end, our contribution were especially a better theoretical understanding of this object, its properties and equations. This study then lead to Part V and methods for multi-goal RL. Our main contributions on the successor states operator are as follows.

In Chapter 6, we properly define the successor states operator in continuous state spaces and show the relation between the value function and the successor states operator. In Chapter 7, we derive a Bellman equation and a temporal difference algorithm for the successor states operator with function approximators and show the relation with standard TD on the value function. Then in Chapter 8 we show that the successor states operator also satisfies a backward Bellman equation, which has no equivalent for the value function, and lead to a backward temporal difference algorithm. We show a relation between forward and backward TD, as backward TD corresponds in expectation to forward TD on the time reversed process. Then in Chapter 9, we introduce second order methods for policy evaluation. These approaches were our first reason to study the successor states operator at the beginning of the project. We prove that the successor states operator satisfies a Bellman–Newton equation on the successor states operator, which also has no equivalent on the value function. We give three interpretations of this new Bellman–Newton equation : first, as a new way to combine paths in the environments, then, as the Newton method for policy evaluation, and finally as a expected exact model based update in the tabular case. This equation leads to a new Bellman–Newton algorithm, with function approximators, in continuous environments. We study this approach in the simpler tabular case, show empirically that it is more sample efficient than TD or TD(λ) in finite environments, and provide non asymptotic convergence bounds for this method in the tabular case (Chapter 10), which interestingly are independent of the number of states, hence are non-vacuous even for very large or infinite environments. This proves that it is possible to learn the value function in finite time, with theoretical guarantees, even in an infinite environment. Still, one of the main practical issues of the Bellman–Newton algorithm with function approximators is its variance, which makes it highly unstable in practice. To overcome this issue, we consider low rank FB representations for the successor states (Chapter 11). Using this FB representation allow us to define updates with lower variance. Additionally, we prove theoretically that the fixed point of these methods are truncated SVDs of the successor states operator, and that in practice it converges to its optimal low-rank representation. Finally in Chapter 12 we described several methods to learn the value function via a model of the successor states operator. In particular, we see that the successor states operator can be used as a model of the expected eligibility traces.

We then extended our approach to the multi-goal RL setting (Part V, Blier and Ollivier (2021)). We tackle the issues of *vanishing rewards* observed with methods as Universal Value Function Approximators (Schaul et al., 2015). While methods such as Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) tackle this issue and achieve great results in practice, they are known to be biased, and can converge to low-return policies in some cases. We first study the bias of HER, prove theoretically and empirically that it is unable to learn in some environments. On the other side, we prove that HER is unbiased and well-founded, in deterministic environments. We then introduce unbiased Q-learning and Actor-critic algorithms for multi-goal RL, in the general stochastic setting, by replacing standard sparse rewards of multi-goal RL by infinitely sparse Dirac rewards, and leveraging their structure.

My goal during this thesis was to improve performance of deep learning and deep RL with principled approaches. Trying to go from theoretical understanding to experimental results was quite a journey, and not always successful in the way we intended.

Some of our conclusion were negative results, impossible to translates in practical methods. In *The description Length of Deep Learning Models*, our first motivation was to use the information theory viewpoint to derive well-founded regularizers via compression bounds. At the end, our best compression bounds were intractable and useless to optimize deep learning models. But our main contribution was to show that standard deep RL approaches were already consistent with the information theory viewpoint, without the use of any additional regularizer.

We also sometimes struggled when comparing to standards methods which benefited from years of careful engineering tuning: in that case, a well-founded method might have nothing to improve. For example in *Making Deep Q-learning approaches robust to time discretization*, we thought that the lack of robustness of standard approaches in near continuous time would be a huge limitation in usual environments, and that our approach could improve these results. Actually, we found out that the time discretization of standard environments was set such that learning was possible, hence we observed almost no improvement in that regime, and had to look for smaller time discretization to highlight the practical issue we identified theoretically. But our contribution gave some interesting insights on some ways to design robust Q-learning agents.

Our study of the successor states operator happened to be even harder. We started this project with the hope of a significant practical impact on policy evaluation, with function approximators, thanks to the Bellman–Newton equation and its different exciting interpretations. In turned out we quickly encountered experimental obstacles, at first because of the variance of the method then for other reasons. We tried to switch to the multi-goal setting but did not succeed in reaching results comparable to those of HER in standard environments.

During this adventure, we understood a lot. On the successor states operator, which we believe is at the core of reinforcement learning. On the different ways to leverage gathered information for policy evaluation. And some of our most striking results were not in the direction we expected. For example, studying the tabular RL setting was not our main goal. But at some point we wondered what would be the sample efficiency of our method in that case, to understand how much gain we could hope in the general continuous case compared to temporal difference. This lead to a non-vacuous convergence bound for policy evaluation, even for very large or infinite sets. Similarly, in the multi-goal RL setting, one of our goal was to design an efficient and principled methods, solving the bias issue of HER. Surprisingly, we proved that HER is actually unbiased in the deterministic setting.

Many questions remain open for future work: Is there still a way to leverage the path composition of Bellman–Newton to accelerate policy evaluation, with function approximators, without the variance issue? Is policy evaluation still a bottleneck of RL, and how much can we improve it? How could we reliably make use the FB representation in practice? Is it possible to define an unbiased multi-goal algorithm, able to reach similar performance than HER? What are the most natural and efficient ways to generalize across goals in an environment? And many others. I hope that our theoretical insights will be of interest for other researchers in the field, who might tackle some of these open questions, or others I never imagined.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. (2017). Hindsight experience replay. In *Advances in neural information processing systems*, pages 5048–5058.

Baird, L. C. (1994). Reinforcement learning in continuous time: Advantage updating. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, volume 4, pages 2448–2453. IEEE.

Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D., Zidek, A., and Munos, R. (2018). Transfer in deep reinforcement learning using successor features and generalised policy improvement. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 501–510. PMLR.

Bhandari, J., Russo, D., and Singal, R. (2018). A finite time analysis of temporal difference learning with linear function approximation. In Bubeck, S., Perchet, V., and Rigollet, P., editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1691–1692. PMLR.

Blei, D. M., Kucukelbir, A., and Mcauliffe, J. D. (2017). Variational Inference: A Review for Statisticians.

Blier, L. and Ollivier, Y. (2018). The description length of deep learning models. In *Advances in Neural Information Processing Systems*.

Blier, L. and Ollivier, Y. (2021). Unbiased methods for multi-goal reinforcement learning. *arXiv preprint arXiv:2106.08863*.

Blier, L., Tallec, C., and Ollivier, Y. (2021). Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*.

Blier, L., Wolinski, P., and Ollivier, Y. (2019). Learning with Random Learning Rates. In *ECML PKDD 2019 - European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

Borsa, D., Barreto, A., Quan, J., Mankowitz, D., Munos, R., van Hasselt, H., Silver, D., and Schaul, T. (2018). Universal successor features approximators. *arXiv preprint arXiv:1812.07626*.

Bradtke, S. J. and Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine learning*, 22(1):33–57.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Chaitin, G. J. (2007). On the intelligibility of the universe and the notions of simplicity, complexity and irreducibility. In *Thinking about Godel and Turing: Essays on Complexity, 1970-2007*. World scientific.

Cohen, D., Kontorovich, A., and Wolfer, G. (2020). Learning discrete distributions with infinite support. *arXiv: Statistics Theory*.

Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624.

Fischhoff, B. (1975). Hindsight is not equal to foresight: The effect of outcome knowledge on judgment under uncertainty. *Journal of Experimental Psychology: Human perception and performance*, 1(3):288.

Grünwald, P. D. (2007). *The Minimum Description Length principle*. MIT press.

Guyon, I., Chaabane, I., Escalante, H. J., Escalera, S., Jajetic, D., Lloyd, J. R., Macià, N., Ray, B., Romaszko, L., Sebag, M., et al. (2016). A brief review of the ChaLearn AutoML challenge: any-time any-dataset learning without human intervention. In *Workshop on Automatic Machine Learning*, pages 21–30.

Ha, D. and Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Hansen, S. S., Dabney, W., Barreto, A., de Wiele, T. V., Warde-Farley, D., and Mnih, V. (2020). Fast task inference with variational intrinsic successor features. *ArXiv*, abs/1906.05030.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2017). Deep reinforcement learning that matters. *CoRR*, abs/1709.06560.

Hinton, G. E. and Van Camp, D. (1993). Keeping Neural Networks Simple by Minimizing the Description Length of the Weights. In *Proceedings of the sixth annual conference on Computational learning theory*. ACM.

Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

Kobyzev, I., Prince, S., and Brubaker, M. (2020). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Kulkarni, T. D., Saeedi, A., Gautam, S., and Gershman, S. J. (2016). Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*.

Lanka, S. and Wu, T. (2018). Archer: Aggressive rewards to counter bias in hindsight experience replay. *ArXiv*, abs/1809.02070.

Li, M. and Vitányi, P. (2008). *An introduction to Kolmogorov complexity*. Springer.

Lillicrap, T., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971.

Machado, M. C., Bellemare, M. G., and Bowling, M. (2019). Count-based exploration with the successor representation.

Manela, B. and Biess, A. (2021). Bias-reduced hindsight experience replay with virtual goal prioritization. *Neurocomputing*, 451:305–315.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. In *Neural Information Processing Systems*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

OpenAI (2018a). Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177.

OpenAI (2018b). Openai five. https://blog.openai.com/openai-five/.

Pan, V. and Schreiber, R. (1991). An improved newton iteration for the generalized inverse of a matrix, with applications. *SIAM Journal on Scientific and Statistical Computing*, 12(5):1109–1130.

Pananjady, A. and Wainwright, M. J. (2019). Value function estimation in Markov reward processes: Instance-dependent ℓ_∞-bounds for policy evaluation.

Pitis, S. (2018). Source traces for temporal difference learning. In *AAAI*.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. (2018). Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). Universal value function approximators. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, Lille, France. PMLR.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Solomonoff, R. (1964). A formal theory of inductive inference. *Information and control.*

Tallec, C., Blier, L., and Ollivier, Y. (2019). Making Deep Q-learning methods robust to time discretization. In *ICML 2019 - Thirty-sixth International Conference on Machine Learning.*

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Uhlenbeck, G. E. and Ornstein, L. S. (1930). On the theory of the Brownian motion. *Physical review*, 36(5):823.

van Hasselt, H., Madjiheurem, S., Hessel, M., Silver, D., Barreto, A., and Borsa, D. (2020). Expected eligibility traces. *arXiv preprint arXiv:2007.01839.*

Voita, E. and Titov, I. (2020). Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

Zhang, A., Wu, Y., and Pineau, J. (2018). Natural environment benchmarks for reinforcement learning. *CoRR*, abs/1811.06032.

Zhang, J., Springenberg, J. T., Boedecker, J., and Burgard, W. (2017). Deep reinforcement learning with successor features for navigation across similar environments. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2371–2378.