Show me the Way: Intrinsic Motivation from Demonstrations

Léonard Hussenot, Robert Dadashi, Matthieu Geist, Olivier Pietquin

Show me the Way: Intrinsic Motivation from Demonstrations

Léonard Hussenot
Google Research, Brain Team
Univ. Lille, CNRS, Inria Scool, UMR 9189 CRIStAL

Robert Dadashi
Google Research, Brain Team

Matthieu Geist
Google Research, Brain Team

Olivier Pietquin
Google Research, Brain Team

ABSTRACT

The study of exploration in the domain of decision making has a long history but remains actively debated. From the vast literature that addressed this topic for decades under various points of view (*e.g.*, developmental psychology, experimental design, artificial intelligence), intrinsic motivation emerged as a concept that can practically be transferred to artificial agents. Especially, in the recent field of Deep Reinforcement Learning (RL), agents implement such a concept (mainly using a novelty argument) in the shape of an exploration bonus, added to the task reward, that encourages visiting the whole environment. This approach is supported by the large amount of theory on RL for which convergence to optimality assumes exhaustive exploration. Yet, Human Beings and mammals do not exhaustively explore the world and their motivation is not only based on novelty but also on various other factors (*e.g.*, curiosity, fun, style, pleasure, safety, competition, *etc.*). They optimize for life-long learning and train to learn transferable skills in playgrounds without obvious goals. They also apply innate or learned priors to save time and stay safe. For these reasons, we propose to learn an exploration bonus from demonstrations that could transfer these motivations to an artificial agent with little assumptions about their rationale. Using an inverse RL approach, we show that complex exploration behaviors, reflecting different motivations, can be learnt and efficiently used by RL agents to solve tasks for which exhaustive exploration is prohibitive.

1 INTRODUCTION

Intrinsic motivation [9] has emerged as one explanation for humans' and animals' impressive learning capabilities. Steered by the need to explore their environment (whether this need is satiable or not has been a fierce debate in the behavioral psychological community [17]), they are able to discover near-optimal strategies in very efficient ways. Designing artificial agents presenting such capabilities is a central goal of modern artificial intelligence and Reinforcement Learning (RL) is a popular candidate to do so. RL has addressed a variety of sequential-decision-making problems whether in games [24, 39, 46] or robotics [1, 2]. Nevertheless, some simple problems remain unsolved. Current state-of-the-art methods struggle to find good policies in environments (1) where constant negative rewards may discourage the agent to explore (*e.g.*, the *Pitfall!* game from Atari), (2) where the reward is so sparse that an agent does not find any (*e.g.*, the *Montezuma's Revenge* Atari game), (3) where state and action space are big (*e.g.*, text worlds). These tasks remain fairly easy for humans, though. In order to tackle these specific problems, the use of reward bonuses, inspired by animal curiosity, was proposed to steer the agent's exploration [40, 42]. Even though different intrinsic bonuses have been proposed, a large majority rely on the same principle: reward for *novelty*. These methods mostly differ in how they compute this notion of *newness*. Count-based methods do it by counting how often the agent has encountered a given state [42]. Pseudo-counts methods [4, 28] allow to approximate counts in large state spaces. Prediction error is also used to measure novelty, either by computing the agent's ability to predict the future [31] or random statistics about the current state [7]. Some restrict novelty to state-action pairs that have an impact on the agent [35] or derive empowerment metrics [25] using mutual information. All these methods naturally encourage the discovery of new states through exhaustive exploration. Yet, in most realistic environments, exhaustive exploration is (1) not feasible due to the size of the state-action space, (2) not desirable as most behaviors are unlikely to be relevant for the task at hand.

Nonetheless, human and more generally mammals exploration behaviors are governed by various motivations and constraints. Intelligent Beings do not have unlimited resources of time and energy. They optimize these resources to survive and reproduce but also to have fun [15], to help others [8] or to satisfy their curiosity. Oudeyer and Kaplan [29] make the difference between homeostatic motivations (that encourage to stay in the "comfort zone" and generally correspond to desires that can be satiated) and heterostatic motivations (that push organisms out of equilibrium but cannot be satiated). These many desires shape the way organisms interact with their environment, encouraging them to discover new things but also to protect themselves, avoiding over-surprising events with mechanisms like fear [22]. Berseth et al. [5] exemplified how to exploit such priors by implementing a "homeostasis" objective for RL, thereby showing how different from "novelty seeking" these priors can be. Eventually, the resource constraints stop organisms from exploring exhaustively their environment and push them to transfer knowledge from past experience. Hunt [17] developed the idea of optimal incongruity: high-novelty is not rewarded as much as intermediate-level novelty, suggesting how curiosity is tightly connected to fear. More recently, Kidd et al. [19] supported this hypothesis with experiments on children curiosity. Overall, novelty methods fail to model correctly human curiosity as they consider that "the newer, the better". This failure calls for a new way of definingour agent's intrinsic motivations.

In an arbitrary environment, exhaustive exploration is desirable and leads to convergence with theoretical guarantees [41]. But when the exploration presents some structure, one can transfer

skills and priors from similar environments. Dubey et al. [10] exemplified, in the case of simple video games, how humans priors help us to solve new problems. The authors enlighten how humans struggle to play the same underlying video game with change of the object semantics, physics modifications (*e.g.*, the gravity is rotated) or with visual similarities transformations. Overall, they show how much of the human's ability to solve a new game in a zero-shot manner is due to their prior on the environment.

In this paper, instead of hard coding what we think an agent's motivations should be (e.g. novelty), we propose to learn a bonus that captures these sources of motivation from demonstrations. By adopting this approach, we expect to learn a bonus that implicitly helps reproducing a structured exploration behavior (*i.e.* using priors from the demonstrations to reduce the search space), in lieu of an exhaustive one. We also argue that, to a certain extent at least, this can happen without the need of extra modelling inspired by cognitive or behavioral research. To do so, we cast this problem as an inverse RL problem with the difference that only some fraction of the reward optimized by an observed agent is hidden: the intrinsic motivation bonus. The task-related reward remains provided by the environment. We then build upon Klein et al. [21] to propose a method that allows us to recover the intrinsic motivation from demonstrations.

Therefore, our contributions are the following:

(1) a modelling that allows for disentangling the reward optimized by a demonstrator from its intrinsic motivation bonus;
(2) an architecture, that we call "*Show me the Way*" (SmtW), based on a cascade of supervised learning methods that extracts that exploration bonus from demonstrations;
(3) an empirical evaluation showing that SmtW is able to capture different exploration priors explained by various types of motivations.

To evaluate SmtW, we validate a set of hypotheses on a controlled environment. We notably find that our method can learn structures and styles, transfer useful priors and encourages long-term planning.

2 BACKGROUND

Markov Decision Processes. In Reinforcement Learning (RL), an agent learns to behave optimally through interactions with an environment. This is usually formalized as a Markov Decision Process (MDP) [34, 43], a tuple $(\mathrm{S}, \mathrm{A}, \mathrm{P}, \mathrm{R}, \gamma)$ with S the set of states, A the set of actions (assumed discrete here), $\mathrm{P} : \mathrm{S} \times \mathrm{A} \to P(\mathrm{S})$ the Markovian transition kernel defining the dynamic of the environment, $\mathrm{R} : \mathrm{S} \times \mathrm{A} \to \mathbb{R}$ a bounded reward function and $\gamma \in [0, 1[$ a discount factor. The agent interacts with the environment through a (here deterministic) policy $\pi : \mathrm{S} \to \mathrm{A}$. The quality of a given policy is quantified by the associated state-action value function, or Q-function. It is the expected discounted cumulative reward for starting from s, taking action a, and following π afterward: $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a]$, with $a_t = \pi(s_t)$, $r_t = \mathrm{R}(s_t, a_t)$ and $s_{t+1} \sim \mathrm{P}(.|s_t, a_t)$. By construction, it satisfies the Bellman equation: for any s, a, $Q^\pi(s, a) = \mathrm{R}(s, a) + \gamma \sum_{s'} \mathrm{P}(s'|s, a) Q^\pi(s', \pi(s'))$. An optimal policy π^* satisfies component-wise $Q^{\pi_*} \geq Q^\pi$, for any policy π. Let $Q^* = Q^{\pi_*}$ be the associated (unique) optimal Q-function, any deterministic optimal policy is greedy with respect to it: $\pi^*(s) \in \mathrm{argmax}_a Q^*(s, a)$.

Exploration Bonus. A common strategy to encourage exploration is to augment the reward function with a bonus. This bonus generally depends on past history. For example, a bonus rewarding novelty requires remembering what has been experienced so far. Write $h_t = (s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t)$ the history up to time t, and H the set of all histories. Generally speaking, we abstract a bonus as $\mathrm{B} : \mathrm{H} \times \mathrm{A} \to \mathbb{R}$, and use it for addressing the dilemma between exploration and exploitation, which thus amounts for the agent to optimize for $R(s_t, a_t) + B(h_t, a_t)$ instead of simply $R(s_t, a_t)$. This state-of-the-art exploration bonuses all rely on memory, *e.g.* by counting the state visitation [42] or through updates of a neural networks [7, 28]. These exploration bonuses are designed to express the prior that any source of novelty is good for exploration.

Such a prior on what is good for exploration is task-specific. Taıga et al. [44] showed that state-of-the-art bonuses were degrading performances in most Atari games.

3 SHOW ME THE WAY

Rather than handcrafting a bonus that encodes what we think intrinsic motivation should be (*e.g.* using novelty), we propose to learn it from demonstrations of exploratory behaviours.

We thus assume that the demonstrator learns to solve a task by exploring its environment and a simple solution would be to perform behavior cloning. Because the demonstrator is likely to use past interactions to make decisions (remembering what has been already tried so far), we could frame our problem as learning, in a supervised manner, a mapping from histories to actions. Yet, behavioral cloning suffers the behavioral drift [36], which would be exacerbated in the case of history dependent policies . Moreover, we would like to transfer this behavior to new environments and possibly to new tasks.

Imitation learning classically assumes that experts are optimizing an MDP with an unknown reward function $\mathrm{R}_E(s, a)$. Note that this introduces a modelling bias, *i.e.* a human performing a task is not necessarily explicitly solving an MDP. In this study, we do not tackle the standard problem of inferring an optimal behavior from demonstrations, but of estimating an exploratory behavior. Yet, the latter can be reduced to the former. Taking inspiration from RL, and especially from the body of work about the exploration-exploitation dilemma, we assume that our demonstrator is optimizing for an unknown reward function $\mathrm{R}_E(s, a)$, standing for the task objective, augmented with a trajectory-dependent intrinsic bonus $\mathrm{B}_E(h, a)$, standing for how the environment is explored. Making this bonus depend on past interactions is important as we can reasonably assume that exploration is based on memory (one would not try to always reproduce situations that were already seen). Then, we assume that the expert is optimal for the bonus-augmented reward $\mathrm{R}_E(s, a) + \mathrm{B}_E(h, a)$, in the augmented MDP $\{\mathrm{H}, A, P, \mathrm{R}_E + \mathrm{B}_E, \gamma\}$. That is the original MDP, with the state space being replaced by the set of all trajectories and the reward function being augmented with a bonus function. So, we reduced our original problem to Inverse Reinforcement Learning (IRL): learn a function $\widehat{RB} : \mathrm{H} \times \mathrm{A} \to \mathbb{R}$ such that the demonstrator is the (unique) optimal policy. By design, $\widehat{RB} = \mathrm{R}_E + \mathrm{B}_E$ is a solution to this problem (even if it is

not learnable exactly, as the optimal policy is invariant to many reward transformations [26]). Yet, we also assume that we know the task's reward R, or at least that we observe it in the demonstrations. Formally, it may be different from the reward R_E (even if we assume that it leads to a similar optimal behavior), but we can leverage it to disentangle the task contribution and the exploration one. For doing so, we propose to learn a bonus function $\hat{B} : \mathcal{H} \times A \to \mathbb{R}$ such that the demonstrator is optimal for $R + \hat{B}$. Notice that it does not change the problem, as it is just a reparameterization of the previous one (by setting $\hat{B} = \widehat{RB} - \mathrm{R}$).

Thus, our IRL problem has additional constraints. First, we want to recover the bonus, for transferring to new environments or new tasks, this preclude using imitation learning methods that do not explicitly recover rewards (IRL is mandatory). Second, our specific parameterization ($\mathrm{R} + \mathrm{B}$) precludes using IRL methods that would not allow using the observation of the task reward R along the expert trajectories. Third, the function we want to estimate is history-dependent which requires an IRL method able to use sequences as inputs.

Formalism. We assume to have access to demonstrations that are optimal according to the (known) reward of the environment *plus* an (unknown) intrinsic bonus. The environment being assumed Markovian, knowing the current state is enough to act optimally according to the task (optimizing for the environment's reward). Yet, the demonstrator also optimizes its exploration bonus, that depends on the past. To formalize things, we consider that the demonstrations are provided by a policy $\pi_E : \mathrm{H} \to \mathrm{A}$, and that the policy is optimal for the augmented MDP $(\mathrm{H}, \mathrm{A}, \mathrm{P}, \mathrm{R}_E + \mathrm{B}_E)$, where H replaces S and $\mathrm{R}_E + \mathrm{B}_E$ replaces R.

We frame our problem as learning the bonus B_E from trajectories sampled from π_E.

Our approach. If we cannot naively apply any existing IRL algorithm to our problem, it can be a source of inspiration. Especially, one suits well our problem: the set-policy framework [32]. It shows that a formal bijection between supervised learning and IRL exists. Among the covered algorithms, the *Cascaded Supervised approach to IRL* (CSI) [21] is of particular interest to us. We refer the reader to these papers for more details and we explain in details here how the CSI paradigm can be readily applied to our setting.

The demonstrator's policy, π_E, is assumed optimal for $\mathrm{R}_E + \mathrm{B}_E$ (which is unknown), so

$$\pi_E = \pi^*_{\mathrm{R}_E + \mathrm{B}_E}.$$

Write $Q_E(h, a) = Q^*_{\mathrm{R}_E + \mathrm{B}_E}(h, a)$ the associated optimal Q-function. It satisfies the Bellman optimality equation (writing $h = (\dots, s)$, that is s the last state of the trajectory h and $h' = (h, a, s')$):

$$\begin{aligned} Q_E(h, a) &= \mathrm{R}_E(s, a) + \mathrm{B}_E(h, a) + \gamma \mathbb{E}_{s'|s,a}[\max_{a'} Q_E(h', a')] \\ &= \mathrm{R}_E(s, a) + \mathrm{B}_E(h, a) + \gamma \mathbb{E}_{s'|s,a}[Q_E(h', \pi_E(h'))]. \end{aligned}$$

Would the optimal policy and Q-function be known, we could use them to recover the optimized bonus-augmented reward using this Bellman equation:

$$\mathrm{R}_E(s, a) + \mathrm{B}_E(h, a) = Q_E(h, a) - \gamma \mathbb{E}_{s'|s,a}[Q_E(h', \pi_E(h'))].$$

Now, the quantities in the right hand side are unknown, but they can be estimated in an indirect way.

Assuming that the actions are discrete, we can learn the policy π_E by mapping histories to actions (using, for example, an LSTM network). Write $\hat{\pi} : \mathrm{H} \to \mathrm{S}$ the resulting policy, or classifier. If we train it by minimizing a cross-entropy loss, what we learn indeed are logits $\hat{Q}(h, a)$, and the classifier is $\hat{\pi}(h) = \mathrm{argmax}_a \hat{Q}(h, a)$. Said otherwise, $\hat{\pi}$ is greedy with respect to $\hat{Q}$, that can thus be interpreted as an optimal Q-function for an unknown reward. Using the Bellman equation, we can recover this reward:

$$\mathrm{R}(s, a) + \hat{\mathrm{B}}(h, a) = \hat{Q}(h, a) - \gamma \mathbb{E}_{s'|s,a}[\hat{Q}(h', \hat{\pi}(h'))]. \tag{1}$$

By Bellman, as $\hat{\pi}$ is greedy w.r.t. $\hat{Q}$, we have that $\hat{\pi}$ and $\hat{Q}$ are respectively the optimal policy and Q-function for the reward $\mathrm{R} + \hat{\mathrm{B}}$. We cannot use directly Eq. (1), the model being unknown, but we can sample the right hand side and estimate $\hat{\mathrm{B}}$ by solving a regression problem.

Therefore, we have reduced our initial problem to a sequence of supervised learning problem. Our algorithm is indeed CSI, up to the fact that we consider trajectories instead of states, and parameterize the bonus with the reward task. As such, the theoretical results of Klein et al. [21] applies to our setting. Notably, we would have that

$$0 \leq \mathbb{E}_{h \sim \pi_E}[\max_a Q^*_{R+\hat{B}}(h, a) - Q^{\pi_E}_{R+\hat{B}}(h, \pi_E(h))] \leq O\left(\frac{\epsilon_1 + \epsilon_2}{1 - \gamma}\right),$$

with ϵ_1 the classification error and ϵ_2 the regression error. This means that the demonstrator policy is close to optimal if these errors are small, for the learnt bonus function. One could argue that this bound trivially holds for $R + \hat{B} = 0$, when all behaviors are optimal. Yet, this is unlikely, as for having the classification error ϵ_1 small, we must have $\hat{Q}(h, \pi_E(h)) > \hat{Q}(h, a \neq \pi_E(h))$ w.h.p., and thus learn an informative bonus.

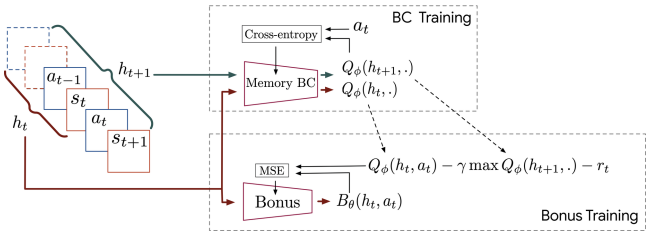


Figure 1: Trajectories $(s, a, \dots)$ are generated by a demonstrator exploring its environment. In order to recover a bonus that can explain its behavior, a BC policy parameterized with an LSTM is trained to predict the actions of the demonstrator from its trajectories of states, by minimizing $\mathcal{L}^{\mathbf{BC}}$. The policy's logits Q_ϕ are interpreted as optimal Q-values and used to compute a regression target. A bonus function B_θ, parameterized with an LSTM, is then trained to predict it, by minimizing $\mathcal{L}^{\mathbf{reg}}$.

Implementation. More concretely, we consider $\mathrm{softmax}(Q_\phi)$ to be a neural network classifier with LSTM [14] units, ϕ being the set of parameters and Q_ϕ being the logits. We train π_ϕ to do behavioral cloning, that is to predict the demonstrator actions a_E based on its past interactions h_E, by minimizing a cross-entropy loss:

$$\mathcal{L}^{\mathrm{BC}} = -\ln(\mathrm{softmax}(Q_\phi(h_E, a_E)),$$

with $Q_\phi(h, a)$ the a^{th} logit for input h. If the classifier learns correctly, the logits of the resulting network should satisfy $Q_\phi(h_E, a_E) > Q_\phi(h_E, a)$ for $a \neq a_E$, and the class predicted by the classifier will be $\pi_\phi(h) = \text{argmax}_a\, Q_\phi(h, a)$. Hence, as explained above, one can interpret Q_ϕ as an optimal Q-function (hence the notation), and π_ϕ as the associated optimal policy. Recall that these quantities can be related to the bonus-augmented reward through Bellman:

$$Q_\phi(h, a) = \mathrm{R}(s, a) + \mathrm{B}(h, a) + \mathbb{E}_{s'|s,a}[Q_\phi(h', \pi_\phi(h'))].$$

Then, we can learn a network B_θ (parameterized by θ, with LSTM units) by minimizing a square-loss, the regression target being $Q_\phi(h_E, a_E) - \gamma Q_\phi(h_E{}', \pi_\phi(h_E{}')) - R(s_E, a_E)$, an unbiased sample of what would give the true Bellman equation. However, we only observe optimal actions (according to R + B), so this alone would hardly generalize to suboptimal ones. Therefore, we propose a heuristic, that consists in regressing for suboptimal actions towards B_{min}, a hyperparameter of the algorithm. For example, it could be set to $\min(Q_\phi(h_E, a_E) - \gamma Q_\phi(h_E{}', \pi_\phi(h_E{}')) - R(s_E, a_E))$, the minimum being over transitions in the dataset. This gives the following loss, for a transition $(h_E, a_E, h_E{}')$, and for $\overline{a_E}$ being sampled randomly in $\mathrm{A} \setminus \{a_E\}$:

$$\mathcal{L}^{\text{reg}} = \Big(Q_\phi(h_E, a_E) - \gamma Q_\phi(h_E{}', \pi_\phi(h_E{}')) - R(s_E, a_E) - \mathrm{B}_\theta(h_E, a_E)\Big)^2 + \Big(\mathrm{B}_{\text{min}} - \mathrm{B}_\theta(h_E, \overline{a_E})\Big)^2.$$

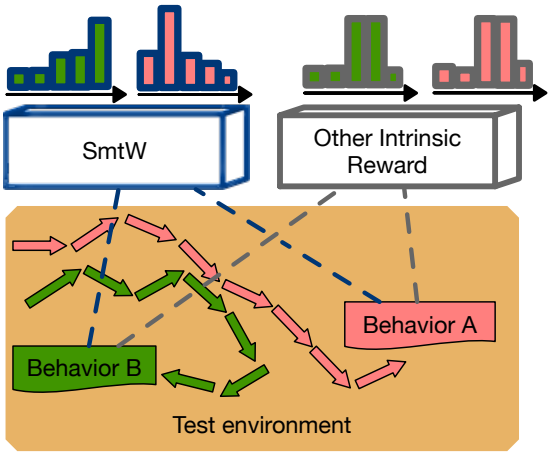
To sum up, we train a BC policy by minimizing $\mathcal{L}^{\text{BC}}$. The implicit resulting logits are considered optimal Q-values, that are in turn used to learn the bonus B_θ by minimizing the loss $\mathcal{L}^{\text{reg}}$ (Figure 1).

4 EXPERIMENTS

We aim at providing insights on what *priors* SmtW is able to extract from the demonstrations and specifically, we wish to verify that SmtW is able to encourage a *structured exploration* of the environment. In order to thoroughly study the method, we test it on a grid-world where we are able to design controllers with specific behaviors. As in IRL, studying the return of an agent trained with our bonus is only a proxy to evaluate SmtW's quality and is not informative on the priors the bonus conveys. We thus focus our experiments on analyzing the priors that were extracted from the demonstrations by the method. More specifically we wish to answer the following questions: (1) Is SmtW encouraging the demonstrator's behavior more than a random one? (2) Is SmtW capturing the demonstrator's style, its way of exploring the environment? (3) Does SmtW capture the skills required to solve the task? (4) Does SmtW encourage novelty seeking? (5) Does SmtW capture the constraints the demonstrator may be submitted to? To do so, we design controlled behaviors and study the bonus returned along these specific behaviors by SmtW, as described in Fig. 2. Given a behavior A and a behavior B, this allows to check if a given bonus encourages behavior A over B or vice versa or rewards them equivalently.

After addressing these questions, we also verify that a simple agent can benefit from SmtW to actually solve efficiently a task.

The environment. We introduce a specific environment to answer these. We require this environment to be procedurally-generated in order to test SmtW's ability to generalize to unseen



SmtW encourages behaviour B over behaviour A while the other intrinsic reward does not.

Figure 2: Comparing intrinsic rewards on what behavior they encourage or discourage on new unseen environments.

environments. We require the environment to be complex enough so that exhaustive exploration is prohibitively expensive. To achieve this, we introduce the KeysDoors grid-world of size NxN. It contains N keys and N doors, modeled by two different colors. The agent has a third color. The goal is to find the correct key and to open the correct door with it. As doors (resp. keys) are indistinguishable (except by their locations), an explorer has to try the different keys on the different doors. Actions available are {*go left, go right, go up, go down, take, open, wait*}. When an agent makes the action "take" on a key, it is then able to move with it. Actions "open" or "take" make the agent lose the key it was previously holding. To solve the task, the agent has to go to the correct key, take it, go to the correct door without doing action "take" or "open" on the way (so as not to lose the key), and then "open" the door. We need the environment to require *perseverance* so we made the reward function -1 for any actions but the *wait* action, that is rewarded 0. Opening the correct door with the correct key gives a reward of 100 and terminates the episode. It requires perseverance as a "lazy" policy would get a return of 0 whereas trying to find the 100 reward gives -1 at each step. This is a well known issue in RL that simple exploration leads to such lazy solutions.

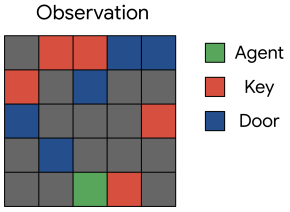


Figure 3: KeysDoors(N=5).

The KeysDoors environment is generated procedurally. For each column, locations for a door and a key are sampled uniformly without replacement. Thus, there is exactly one key and one door on each column and these cannot be at the same location. The "correct" key is then uniformly sampled among the keys and the "correct" door is sampled uniformly among the doors. The initial position of

the agent is sample uniformly on the grid. The environment gives both a ground-truth-state (an integer representing the current state), unused by SmtW as well as an RGB observation (as shown in Fig. 3), used by SmtW. Figure 4 shows a trajectory in one possible instance of the KeysDoors environment with $N = 5$. Every observation x (an $N \times N \times 3$ tensor) is normalized between 0 and 1 by dividing by 255.

The demonstrations. For a given instance of the environment, the demonstrator navigates between keys and doors and tries key/door pairs in a precise order. It takes the first key on the left and tries it on the first door on the left, then it tries the same key on the second door etc. Once it has tried the first key on every door, it repeats the operation with the second key and proceeds further this way. The episode ends when the demonstrator finds the right key/door pair and obtains the reward. Then it "exploits", taking the correct key and opening directly the correct door five consecutive times. Note that this also simulates the non-stationnarity happening in most goal-directed task solving processes. One first mainly explores and then exploits more and more.

Train vs. Test. The bonus is always used in new test environments, unseen in the demonstrations. SmtW's ability to generalize to new environments is thus tested in all the following experiments. Given the possible positions of the keys, of the doors and then of the correct key and the correct door, there are $(N-1)N^3$ possible instances of the environment. **The behaviors** that are designed to study what is actually encouraged or discouraged by SmtW are the following. Their associated bonus is always studied in test instances of the environment.

- The demonstrator behavior acts as described previously, sequentially trying key/door pairs.
- The *random* behavior takes random actions. Trajectories are limited to 1000 steps.
- The *demonstrator inverse* behavior is similar to the demonstrator as it navigates to a key, takes it, navigates to a door and opens it. However, the key/door pairs are tried in the reverse order to the demonstrations.
- The *demonstrator random* behavior is also similar but tries the key/door pairs in a random order.
- The *dummy demonstrator* behavior navigates exactly like the demonstrator but drops the key at a random time on the way to the door (uniformly sampled on the path to the door) by taking action *open*. The trajectories are limited to 1000 steps.
- The *standing still* behavior remains in its original position by only taking the *wait* action.
- The *waiting demonstrator* behavior acts like the demonstrator but has a probability 0.1 of waiting at each step.
- The *unsafe demonstrator* acts like the demonstrator but takes this action *take* each time it moves until it has a key. Taking action *take* somewhere else than on a key can be considered as breaking a safety constraint that the demonstrator respects strictly.

A trajectory of an agent moving to a key, taking it, moving to a door and trying to open it with the key is shown in Fig. 4

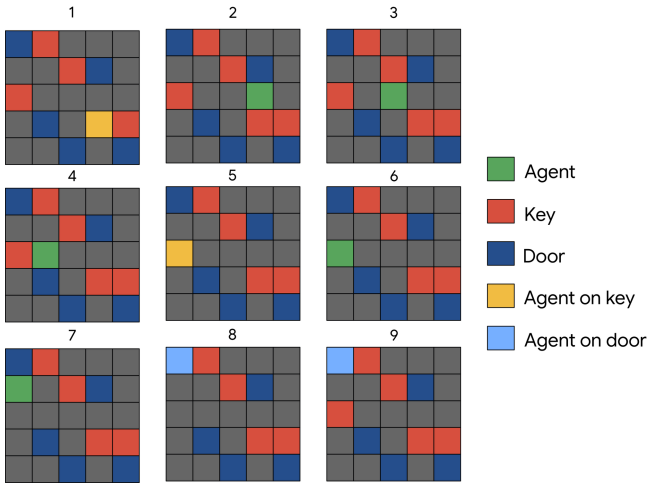


Figure 4: A trajectory of length 9 in an instance of the KeysDoors(N=5) environment.

4.1 Bonus analysis

We train the SmtW bonus on 200 KeysDoors(N=5) training environments with 10 demonstrations for each of them. The implementation choices are detailed in Sec. 4.3. In order to study the priors extracted from the demonstrations, we study the bonus given by SmtW along various trajectories following a given behavior. We thereafter plot the distribution of received bonus along the various controlled behaviors on 20 test environments, unseen during the training of SmtW. We compare the bonus given by SmtW along these trajectories to the one that would be given by a count-based [42] and a random network distillation bonus [7]. The very same trajectories are presented to each bonus.

Does SmtW encourage a structured exploration more than a random one? We compare in Figure 5 the distribution of bonus received along random trajectories to the ones obtained by the demonstrator's behavior. Recall that SmtW has been trained on similar environments but is here tested on different ones. It is thus provided with trajectories unseen during training.

As shown on Figure 5, the demonstrator's behavior (top) is more rewarded by SmtW than the random behavior (bottom). We can conclude that SmtW encourage the agent to follow a demonstrator-like behavior on new unseen environments more than a random behavior. The count-based bonus also rewards the demonstrator's behavior more than a random one as a random behavior explores the environment very locally. Surprisingly, RND rewards the random behavior more than the demonstrator's one. This might be explained by the fact that the demonstrator visits several times the same state in order to explore correctly. Indeed the demonstrator has to go several times to the same key to take it and try it on the several doors.

Does SmtW capture the demonstrator's style, his way of exploring the environment? We show in Figure 6 the distribution of bonus received along different behaviors: the *demonstrator* one, the *demonstrator inverse* one as well as the *demonstrator random* one. These three behaviors lead to the same outcome but we hope to

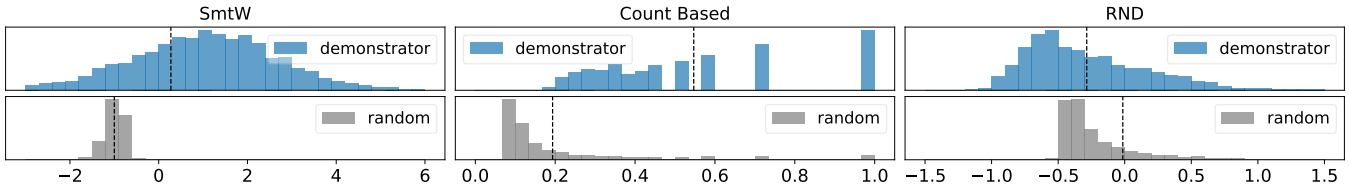


Figure 5: Bonus distribution received by an demonstrator's behavior (top) and a random behavior (bottom). We can say that a bonus encourages more a behavior A than a behavior B if the distribution of bonus along trajectories following A are globally higher than the one along trajectories following B. SmtW encourages the demonstrator behavior over the random behavior.

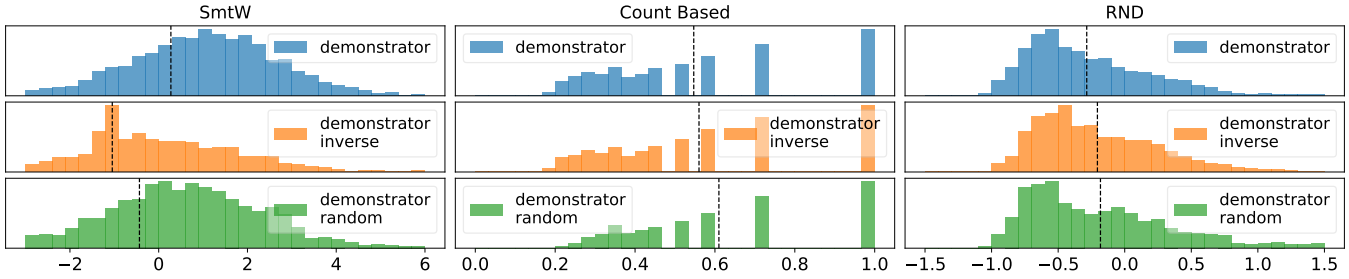


Figure 6: Bonus distribution received by the *demonstrator* behavior (top), the *inverse demonstrator* behavior (middle), and the *random demonstrator* behavior. (bottom).

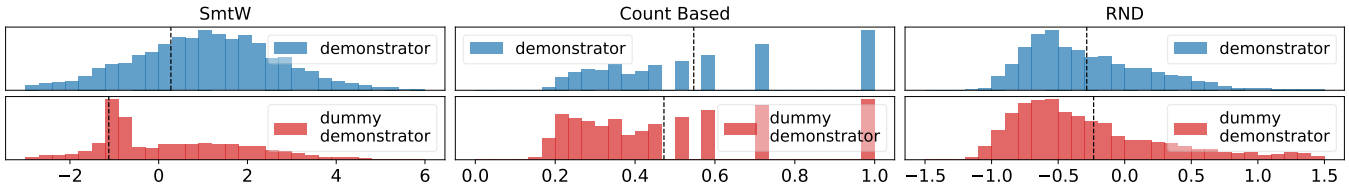


Figure 7: Bonus distribution received by the *demonstrator* behavior (top) and by the *dummy demonstrator* behavior, acting (almost) like the demonstrator but releasing the key on the way to the door (bottom).

capture the demonstrator's exploration bias and see if it encourages the behaviors that tries the key/door pair in the same order as in the demonstrations. As shown on Figure 6, the count-based bonus and RND reward similarly the three behaviors, as they lead to the same amount of novelty. Only the order in which the key/door pairs are tried is change. SmtW, on the contrary, encourages to reproduce the demonstrator bias. It rewards more the behavior trying the key/door pairs in the same order as in the demonstrations.

Does SmtW capture the priors useful to solve the task? Figure 7 shows the distribution of bonus received by the *demonstrator* behavior and compares the bonus received to the one received when following the *dummy demonstrator* one. As shown on Figure 7, the count-based bonus and RND reward equivalently these two behaviors as they bring the same amount of novelty (both in term of ground-truth-state and observations). SmtW does not reward the *dummy demonstrator* behavior as much as the expert one and we can interpret the lower distribution mode (SmtW-bottom) as the bonus obtained after loosing the key. We can argue that SmtW has somehow captured the prior that it is useful to navigate from the key to the door without loosing the key, as it rewards more the *demonstrator* behavior than the *dummy demonstrator* one.

Does SmtW encourage long-term exploration? As the environment gives a reward of -1 for taking any action but the *wait* action, an agent not exploring sufficiently would quickly converge to the policy only taking action *wait* to avoid negative rewards (verified in Figure 11). This same problem is visible in the *Pitfall!* game, where the best agents learn a policy obtaining 0 reward, while persevering humans get much higher scores. We show in Figure 8 the distribution of bonus obtained by the *standing still* behavior. As shown on Figure 8, SmtW rewards much less a behavior not seeking novelty. As expected the count based gives a bonus very close to 0 for such a behavior. Perhaps surprisingly, RND rewards negatively this behavior but not with an average bonus lower than the demonstrator's behavior. This might be also due to the designed bonus normalization that RND uses (zero-mean unit-variance).

Does SmtW capture the constraints the demonstrator may be submitted to? A demonstrator can be subject to time or energy constraints. In the demonstrations, the demonstrator tries to explore the environment as fast as possible and does not take action *wait* on his way to keys and doors. We compare the bonus distribution obtained by the *waiting demonstrator* behavior to the one obtained by the *demonstrator* one.

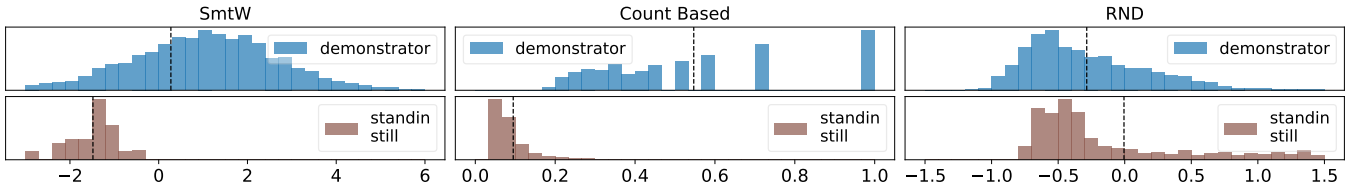


Figure 8: Bonus distribution received by the *demonstrator* behavior (top) and by the *standing still* behavior (bottom).

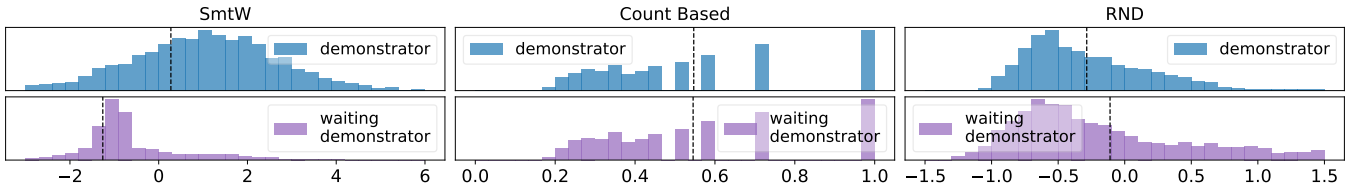


Figure 9: Bonus distribution received by the *demonstrator* behavior (top) and by the *waiting demonstrator* behavior (bottom).

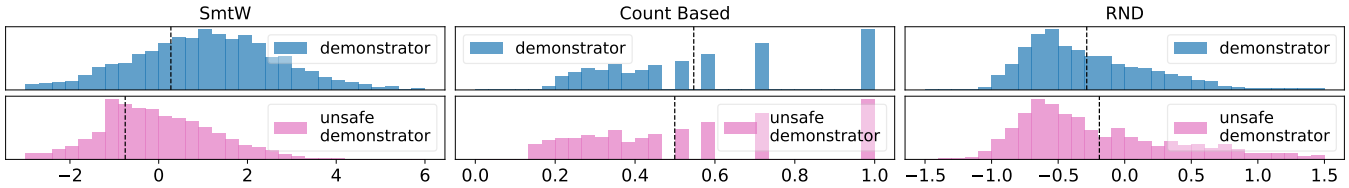


Figure 10: Bonus distribution received by the *demonstrator* behavior (top) vs. by the *unsafe demonstrator* behavior (bottom).

As shown on Figure 9, RND and the count-based bonus reward equivalently these two behaviors. On the other hand, SmtW rewards less the *waiting demonstrator* behavior. We argue it has somehow captured the prior resulting from the resource constraint that leads the demonstrator to try the key/door pairs as fast as possible. In other words, it favors behaviors that, as shown in the demonstrations, discard the *wait* action to simplify exploration of the MDP. What is more, a demonstrator might be subject to safety constraints. As example, it might be dangerous for a robot to try an action in an inappropriate place. The demonstrations minimize the number of time they use the action "take" and only do it when on keys. We can consider that the demonstrator's behavior complied with safety constraints. We show in Figure 10 the bonus distribution obtained by the demonstrator's behavior and compare it with the one obtained by the *unsafe demonstrator*. As shown on Figure 10, the RND and the count-based bonuses reward equivalently these two behaviors. This is expected as they bring the same amount of novelty. In contrast, SmtW rewards less the *unsafe demonstrator* behavior, capturing the safety prior the demonstrator have been subject to.

Overall, we argue that SmtW is able to recover some important bias and constraints inherent to the demonstrations. Hand-crafting a bonus expressing these motivations could be extremely complicated and we demonstrated that SmtW is able to generalize these motivations to unseen environments.

4.2 Training an agent on the bonus

We now wish to check that an agent can benefit from SmtW. We thus train a Q-learning agent with SmtW and compare the results with that of a simple ϵ-greedy (ϵ=0.1) exploration strategy and a count-based bonus with $B(s,a) = N(s,a)^{-1/2}$.

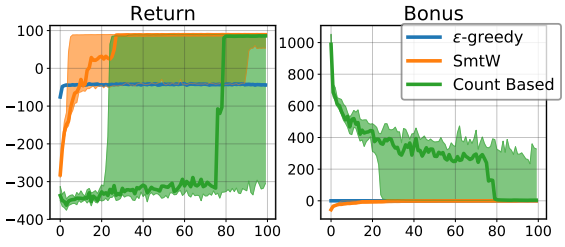


Figure 11: Median and min/max values of the return per episode (left) and of the total bonus per episode(right).

The results are averaged over 10 newly generated environments, unseen during SmtW training. For each of these environments, the experiment is repeated twice. We present, for each algorithm, the best result after a hyper-parameter search. The bonus given by our method is computed to capture the exploratory behavior of the demonstrator. In order for the agent not to keep exploring forever, our bonus is here divided by $\sqrt{k}$ with k the number of step of training.

As Figure 11 shows, the Q-learning with an ϵ-greedy exploration strategy quickly gets stuck in "waiting" at each timestep. SmtW

encourages the agent to visit its environment and solves the 10 new environments much faster than the count-based method that push for exhaustive exploration.

4.3 Implementation Details

Our method works directly with visual inputs, as shown in Fig. 3. The network used for the behavioral cloning policy π_ϕ has the following architecture: an LSTM with 64 units, a fully-connected layer with 512 units and relu activation and an output layer with as many units as there are actions available in the environment (7 for KeysDoors). It is trained with the Adam optimizer [20] with a learning rate of 10^{-3} and a batch size of 1. It uses the visual input from the environment and not the ground-truth state.

The network used for the regression of the bonus B_θ has the same architecture but an output layer with a single unit. It is trained with the Adam optimizer, a learning rate of 10^{-4} and a batch size of 1. The discount factor used in SmtW is set to $\gamma = 0.99$.

For experiment shown in Figure 11, the tabular Q-learning is trained on the 10 test environments twice and the figure shows the median and the min/max values. For each of the compared algorithms, we sweep over the agent learning rate over the following values: $[0.01, 0.1, 0.5, 0.7]$. Only the result of the learning rate with the highest median return over the 10×2 runs is shown for each algorithm. The ε-greedy strategy is used for all methods with $\varepsilon = 0.1$. Even though the agent is tabular, we recall that SmtW itself does not access the ground-truth state of the environment. It works from observations. The count-based bonus, on the contrary, counts ground-truth states.

5 RELATED WORK

Intrinsic Motivation. Intrinsic motivation is essential to mental development [30] and we can argue that this may, in consequence, be an essential component for computational learning. Oudeyer and Kaplan [29] argue that all humans respond to intrinsic motivations. Young infants motivations can be qualified as more chaotic as they push children to bite, throw, grasp or shout in order to learn. Adults, in contrast, have more structured intrinsic motivations, activated, for instance, when they play games, read novels or watch movies. Correctly using these numerous intrinsic motivations can be key to train agents that solve more and more difficult tasks. Instead of modeling such intrinsic motivations to mimic cognitive processes, we learn them from demonstrations.

Exploration. In order to provide an exploration signal to the agent, [42] proposed the very intuitive count-based method in order to measure novelty. Counting how many times the agent has been in a given state, it rewards less visited states. Several methods extended this idea to large state-space problems [4, 23, 28, 45], where it is not possible to count state occupancy. Intrinsic curiosity is also commonly computed as a prediction error, either trying to predict the environment's dynamics [31, 35] or random statistics about the current state [7]. Different methods try also to measure surprise as a prediction gain [16, 38]. Instead of designing such a bonus, we aim at learning one from demonstrations.

Learning from demonstrations. Imitation learning, the problem of learning from demonstrations, is typically folded into two different paradigms. (1) Behavioral cloning [3, 33, 36] tries to directly match the demonstrator's behavior, generally using supervised learning techniques. (2) Inverse Reinforcement Learning [27, 37] first tries to recover a reward explaining the demonstrator's behavior, before optimizing the reward for imitating the demonstrator. Some methods output an explicit reward [1, 21, 27, 47] while adversarial imitation learning can be seen as IRL with implicit reward recovery [11–13]. Overall these methods all assume that the near-optimality of the demonstrations. Some works try to relax this assumption and to learn from sub-optimal demonstrations [6, 18]. IRL methods typically control the quality of their algorithm through the proxy of the return obtained by an agent trained on the inferred reward.

Our methods differs from these methods it does not assume that demonstrations are optimal but rather try to answer the question: "In what way is the demonstrator's behavior deviating from an optimal policy?". Moreover, we do not seek to recover a reward as in IRL but rather to recover a bonus explaining which, added to the environment reward, explains the demonstrator's behavior. Facing the same problem that the usual proxy to control the algorithm quality (training an agent on the inferred bonus) is not informative, we decided to study our method through its response to various behaviors.

6 CONCLUSION

In this work, we present a novel method for extracting an intrinsic bonus from the demonstrations. The method we introduce is offline and does not require environment interactions to recover the bonus, unlike recent adversarial imitation methods who need numerous interaction in order to recover a reward function. Anyway, those methods could not be readily applied to our problem, as they do not explicitly compute a reward function. Moreover, to the best of our knowledge, this is one of the very first method to recover some kind of reward that is history-dependent. We show how this bonus generalizes to unseen environments and is able to convey long-term priors. We exemplified the approach on a simple yet didactic and challenging example. Yet, testing the method on a larger-scale environment would require human exploratory demonstrations. Gathering such a dataset is costly and very few are already available, none of them really covering our setting. Even though the given example is simple, this novel approach of capturing the demonstrator's bias could potentially lead to new lines of work in RL. For instance, one could use our method to implement *behavioral style-transfer* in RL and show to an agent a specific way to solve the task thanks to demonstrations. Combining a reward and biases extracted from demonstrations may also help for robotic tasks, where some aspects of the task are easily programmable with a reward but some expectations on how to solve the task may be easier to transmit thanks to demonstrations. This could also lead to some advances in tackling mispecified rewards. Using both a reward, that would contain information on the task to solve but not fully describe the constraints of the problem and demonstrations to correct the reward can be key to train sequential controllers in complex dynamics.

REFERENCES

[1] Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*.

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. 2020. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research* (2020).

[3] JA Bagnell, Joel Chestnutt, David M Bradley, and Nathan D Ratliff. 2007. Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems*.

[4] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*. 1471–1479.

[5] Glen Berseth, Daniel Geng, Coline Devin, Chelsea Finn, Dinesh Jayaraman, and Sergey Levine. 2019. SMiRL: Surprise Minimizing RL in Dynamic Environments. *arXiv preprint arXiv:1912.05510* (2019).

[6] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. 2019. Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations. In *International Conference on Machine Learning*.

[7] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2018. Exploration by random network distillation. *International Conference on Learning Representations (ICLR)* (2018).

[8] Richard W Byrne and Andrew Whiten. 1989. *Machiavellian intelligence: social expertise and the evolution of intellect in monkeys, apes, and humans*. Clarendon Press.

[9] Edward L Deci and Richard M Ryan. 2010. Intrinsic motivation. *The corsini encyclopedia of psychology* (2010), 1–2.

[10] Rachit Dubey, Pulkit Agrawal, Deepak Pathak, Thomas L Griffiths, and Alexei A Efros. 2018. Investigating human priors for playing video games. *arXiv preprint arXiv:1802.10217* (2018).

[11] Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*.

[12] Justin Fu, Katie Luo, and Sergey Levine. 2018. Learning robust rewards with adversarial inverse reinforcement learning. *International Conference on Learning Representations* (2018).

[13] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*.

[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[15] Sarah L Holloway and Gill Valentine. 2004. *Children's geographies: Playing, living, learning*. Routledge.

[16] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2016. Variational information maximizing exploration. *Advances in Neural Information Processing Systems (NIPS)* (2016).

[17] JMcVLevine Hunt. 1965. Intrinsic motivation and its role in psychological development. In *Nebraska symposium on motivation*, Vol. 13. University of Nebraska Press, 189–282.

[18] Alexis Jacq, Matthieu Geist, Ana Paiva, and Olivier Pietquin. 2019. Learning from a Learner. In *International Conference on Machine Learning*.

[19] Celeste Kidd, Steven T Piantadosi, and Richard N Aslin. 2012. The Goldilocks effect: Human infants allocate attention to visual sequences that are neither too simple nor too complex. *PloS one* 7, 5 (2012), e36399.

[20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[21] Edouard Klein, Bilal Piot, Matthieu Geist, and Olivier Pietquin. 2013. A cascaded supervised learning approach to inverse reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 1–16.

[22] Peter J Lang, Michael Davis, and Arne Öhman. 2000. Fear and anxiety: animal models and human cognitive psychophysiology. *Journal of affective disorders* 61, 3 (2000), 137–159.

[23] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. 2018. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research* 61 (2018), 523–562.

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

[25] Shakir Mohamed and Danilo Jimenez Rezende. 2015. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*. 2125–2133.

[26] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, Vol. 99. 278–287.

[27] Andrew Y Ng, Stuart J Russell, et al. 2000. Algorithms for inverse reinforcement learning.. In *International Conference on Machine Learning*.

[28] Georg Ostrovski, Marc G Bellemare, Aäron van den Oord, and Rémi Munos. 2017. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2721–2730.

[29] Pierre-Yves Oudeyer and Frederic Kaplan. 2009. What is intrinsic motivation? A typology of computational approaches. *Frontiers in neurorobotics* 1 (2009), 6.

[30] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. 2007. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation* 11, 2 (2007), 265–286.

[31] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 16–17.

[32] Bilal Piot, Matthieu Geist, and Olivier Pietquin. 2016. Bridging the gap between imitation learning and inverse reinforcement learning. *IEEE transactions on neural networks and learning systems* 28, 8 (2016), 1814–1826.

[33] Dean A Pomerleau. 1991. Efficient training of artificial neural networks for autonomous navigation. *Neural computation* (1991).

[34] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[35] Roberta Raileanu and Tim Rocktäschel. 2020. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. *arXiv preprint arXiv:2002.12292* (2020).

[36] Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*.

[37] Stuart Russell. 1998. Learning agents for uncertain environments. In *Conference on Computational learning theory*.

[38] Jürgen Schmidhuber. 1991. Curious model-building control systems. In *Proc. international joint conference on neural networks*. 1458–1463.

[39] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* (2016).

[40] Özgür Şimşek and Andrew G Barto. 2006. An intrinsic reward mechanism for efficient exploration. In *Proceedings of the 23rd international conference on Machine learning*. 833–840.

[41] Alexander L Strehl, Lihong Li, and Michael L Littman. 2009. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research* 10, Nov (2009), 2413–2444.

[42] Alexander L Strehl and Michael L Littman. 2008. An analysis of model-based interval estimation for Markov decision processes. *J. Comput. System Sci.* 74, 8 (2008), 1309–1331.

[43] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[44] Adrien Ali Taıga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. 2020. ON BONUS-BASED EXPLORATION METHODS IN THE ARCADE LEARNING ENVIRONMENT. *International Conference on Learning Representations (ICLR)* (2020).

[45] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. 2017. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*. 2753–2762.

[46] Gerald Tesauro. 1995. Temporal difference learning and TD-Gammon. *Commun. ACM* (1995).

[47] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning.. In *AAAI Conference on Artificial Intelligence*.