Tackling the Zero-Shot RL Loss Directly

Yann Ollivier

Abstract

Zero-shot reinforcement learning (RL) methods aim at instantly producing a behavior for an RL task in a given environment, from a description of the reward function. These methods are usually tested by evaluating their average performance on a series of downstream tasks. Yet they cannot be trained directly for that objective, unless the distribution of downstream tasks is known. Existing approaches either use other learning criteria [BBQ+18, TRO23, TO21, HDB+19], or explicitly set a prior on downstream tasks, such as reward functions given by a random neural network [FPAL24].

Here we prove that the zero-shot RL loss can be optimized directly, for a range of non-informative priors such as white noise rewards, temporally smooth rewards, "scattered" sparse rewards, or a combination of those.

Thus, it is possible to learn the optimal zero-shot features algorithmically, for a wide mixture of priors.

Surprisingly, the white noise prior leads to an objective almost identical to the one in VISR [HDB+19], via a different approach. This shows that some seemingly arbitrary choices in VISR, such as Von Mises–Fisher distributions, do maximize downstream performance. This also suggests more efficient ways to tackle the VISR objective.

Finally, we discuss some consequences and limitations of the zero-shot RL objective, such as its tendency to produce narrow optimal features if only using Gaussian dense reward priors.

1 Introduction

Zero-shot reinforcement learning (RL) methods aim at instantly producing a behavior for an RL task in a given environment, from a description of the reward function. This is done after an unsupervised training phase. Such methods include, for instance, universal successor features (SFs, [BBQ+18]) and the forward-backward framework (FB, [TRO23, TO21]).

Zero-shot RL is usually tested by reporting average performance on a series of downstream tasks: a reward function r is sampled from a distribution β_{test} of tasks, a reward representation $z = \Phi(r)$ is computed, [1] and a policy π_z

[1] A requirement of zero-shot RL is that this computation should be scalable, with z of reasonable size. Without a computational constraint, one could just pre-compute all optimal policies of all possible downstream tasks up to some degree of approximation.

is applied, starting at some initial state s_0. Thus, the reported performance is the expectation

$$\mathbb{E}_{r\sim\beta_{\text{test}}}\mathbb{E}_{s_0\sim\rho_0}V_r^{\pi_z}(s_0) \tag{1}$$

where the value function $V_r^{\pi_z}(s_0)$ is the performance of policy π_z on the reward function r when starting at s_0 (Section 2.1).

Yet zero-shot RL methods are usually not trained by maximizing the performance (1), because the distribution of downstream tasks β_{test} is unknown. Other training criteria have to be introduced, such as a finite-rank representation of long-term transition probabilities in FB [TO21], or an information criterion on policies π_z in VISR [HDB+19].

Alternatively, it is possible to explicitly set a prior β on downstream tasks, and optimize the criterion (1) using that prior instead of the true task distribution β_{test}. This follows the machine learning philosophy of "follow the gradient of what you are actually doing", rather than made-up criteria.

For instance, [FPAL24] use random neural networks as a prior for the downstream reward function r. This prior is parametric (it is parameterized by the weights of a network), and it is unclear how sensitive performance is to this choice.

Here:

- We show that the zero-shot RL performance can be maximized directly for a wide mixture of nonparametric, uninformative priors. This includes dense reward priors such as white noise rewards, temporally smooth rewards with a "Dirichlet norm" prior related to Laplacian eigenfunctions, and sparse priors such as mixtures of a number of target states with random weights.

  Arguably, a mixture of such uninformative priors has the best chance of covering the unknown test distribution β_{test}. Note that learning meaningful representations does not require informative priors on downstream tasks: environment dynamics lead to informative representations even with non-informative priors.

  This makes it possible to compute the best possible representations for zero-shot RL by following the gradient of the criterion (1). Notably, we can do this for dense reward priors without explicitly sampling a reward from the prior, which would not be possible for infinite-dimensional priors such as white noise.

- We clarify the implicit priors on rewards in SFs: the SF strategy implicitly relies on a *white noise prior* on rewards (Section 3.2).

  Doing so, we extend the SF framework to other priors, such as a prior based on the Dirichlet norm, which introduces temporal smoothness related to Laplacian eigenfunctions (Section 2.3).

- We show a surprising connection with VISR [HDB+19]: VISR "almost" computes the optimal zero-shot features for a white noise prior (Section 3.3). (The "almost" comes from a minor change in the way the features are normalized.)

  This is unexpected, as VISR was not defined to maximize downstream zero-shot performance. Instead, VISR was defined as a feedback loop between a diversity method [EGIL18] and successor features, by training a family of policies π_z that maximize the rewards $\varphi^\top z$ for some features φ, and learning φ in turn by increasing $\varphi^\top z$ at the places visited by π_z, thus creating specialization.

  This newfound connection between VISR and downstream performance for a white noise prior may justify some seemingly arbitrary choices in VISR, such as its use of Von Mises–Fisher distributions.

  The analysis also suggests more efficient ways to tackle the VISR objective, notably, relying on occupation measures rather than Monte Carlo sampling.

- We derive further theoretical properties of the zero-shot RL loss. Notably, the Bayesian viewpoint has no regularizing effect on the policies learned: these policies are "sharp" in that they are necessarily optimal policies for some particular task r (Proposition 2).

  This has some consequences for exploration in settings where the reward is not exactly known: indeed, the zero-shot RL setting assumes that the reward function is fully specified at test time (such as reaching a particular goal or maximizing a particular quantity).

  This can also produce unexpectedly narrow optimal features (Section 4.1) for some particular priors.

- We discuss some limitations of the zero-shot RL setting, and possible extensions.

2 Setup, Notation, and Some Reward Priors

2.1 General Notation

Markov decision process. We consider a reward-free Markov decision process (MDP) $\mathcal{M} = (S, A, P, \gamma)$ with state space S, action space A, transition probabilities $P(s'|s, a)$ from state s to s' given action a, and discount factor $0 < \gamma < 1$ [SB18]. A policy π is a function $\pi\colon S \to \mathrm{Prob}(A)$ mapping a state s to the probabilities of actions in A. Given $(s_0, a_0) \in S \times A$ and a policy π, we denote $\Pr(\cdot|s_0, a_0, \pi)$ and $\mathbb{E}[\cdot|s_0, a_0, \pi]$ the probabilities and expectations under state-action sequences $(s_t, a_t)_{t\geqslant 0}$ starting at (s_0, a_0) and following policy π in the environment, defined by sampling $s_t \sim P(s_t|s_{t-1}, a_{t-1})$ and

$a_t \sim \pi(a_t|s_t)$. Given any reward function $r\colon S \to \mathbb{R}$, the Q-function of π for r is $Q_r^\pi(s_0, a_0) := \sum_{t\geqslant 0} \gamma^t \mathbb{E}[r(s_t)|s_0, a_0, \pi]$. The *value function* of π for r is $V_r^\pi(s) := \sum_{t\geqslant 0} \gamma^t \mathbb{E}[r(s_t)|s_0, \pi]$.

We assume access to a dataset consisting of *reward-free* observed transitions (s_t, a_t, s_{t+1}) in the environment. We denote by ρ the distribution of states s_t in the training set.

Occupation measures. We let ρ_0 be some distribution of initial states in the environment; if no such distribution is available, we just take $\rho_0 := \rho$.

Occupation measures will pop up repeadtedly in our analysis. The occupation measure d_π of policy π is a probability distribution over S, defined for each $X \subset S$ as

$$d_\pi(X) := (1-\gamma)\mathbb{E}_{s_0\sim\rho_0} \sum_{t\geqslant 0} \gamma^t \Pr(s_t \in X|s_0, \pi). \tag{2}$$

In particular, by construction,

$$\mathbb{E}_{s\sim d_\pi} r(s) = (1-\gamma)\mathbb{E}_{s_0\sim\rho_0} V_r^\pi(s_0). \tag{3}$$

2.2 The Zero-Shot RL Objective: Optimize Expected Downstream Performance

Existing zero-shot RL procedures proceed as follows: after an unsupervised, reward-free pretraining phase, the agent is confronted with a reward r (either via reward samples or via an explicit reward formula), computes a task representation $z = \Phi(r)$ in a simple, fast way, then apply an existing policy π_z. The map Φ from reward to task representation, as well as the policies π_z, are learned during pretraining.

Such methods are evaluated by running the policies π_z on a number of downstream tasks, and reporting the cumulated reward. Thus, if β_{test} is the distribution of downstream tasks, the reported loss is, in expectation,

$$\ell_{\text{test}}(\Phi, \pi) = -\mathbb{E}_{r\sim\beta_{\text{test}}} \mathbb{E}_{s_0\sim\rho_0} V_r^{\pi_{\Phi(r)}}(s_0) \tag{4}$$

where ρ_0 is the distribution of initial states used for testing. This corresponds to sampling a downstream task $r \sim \beta_{\text{test}}$, computing $z = \Phi(r)$, and running π_z on reward r.

Usually the downstream task distribution β_{test} is unknown. Still, if we have a prior β on rewards, a natural objective for the pretraining phase is to minimize the loss

$$\ell_\beta(\Phi, \pi) := -\mathbb{E}_{r\sim\beta} \mathbb{E}_{s_0\sim\rho_0} V_r^{\pi_{\Phi(r)}}(s_0) \tag{5}$$

over Φ and π. The prior β should ideally encompass the unknown actual distribution β_{test} of downstream tasks.

Without computational constraints, this problem is theoretically "easy" to solve: just precompute all optimal policies for all possible rewards. This corresponds to $\Phi = \mathrm{Id}$, namely, a reward function r is representated by $z = r$ itself, and then π_z should just be the optimal policy for r. If the state space is continuous, r and z are infinite-dimensional.

In practical methods, the task representation z will be finite-dimensional. This means some reward functions r are necessarily lumped together via Φ, and determining the best way to do this (e.g., for a fixed dimension of z) becomes a nontrivial mathematical question. This is what we address in the rest of the text.

2.3 Some Uninformative Priors on Reward Functions

We now introduce some priors on downstream tasks. Ideally, the prior should cover the true distribution of tasks at test time. Since this distribution is unknown, we try to consider the most uninformative priors we could handle, in the hope this results in more generic zero-shot performance.

We consider both dense and sparse reward priors. For dense rewards, we include white noise, and a Gaussian process based on the Dirichlet norm, which imposes more spatial smoothness on the rewards than white noise, related to Laplacian eigenfunctions. For sparse rewards, we consider random goal-reaching (reaching a target state specified at random), and mixtures of several goals with random weights.

These are some of the most agnostic models we can find on an arbitrary state equipped with an arbitrary probability distribution. All models are built to have well-defined continuous-space limits, and still make sense in an abstract state space equipped with a measure ρ. To avoid excessive technicality, we restrict ourselves to the finite case in this text.

Importantly, these priors rely on quantities that can be estimated from the dataset (such as expectations under ρ). This is why we use norms related to the dataset distribution ρ.

We will also use mixtures of these priors.

2.3.1 Dense Reward Priors

White noise prior. This is defined as

$$\beta(r) \propto \exp(-\|r\|_\rho^2/2) \tag{6}$$

where $\|f\|_\rho^2 := \mathbb{E}_{s\sim\rho} f(s)^2$.

This prior is very agnostic: the reward at every state is assumed to be independent from every other state.

Dirichlet prior. This is defined as

$$\beta(r) \propto \exp(-\|r\|_{\text{Dir}}^2/2) \tag{7}$$

where

$$\|f\|_{\text{Dir}}^2 := \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\rho}\,(f(s_t)-f(s_{t+1}))^2 + \alpha\,\|f\|_\rho^2 \tag{8}$$

where some $\alpha > 0$ is used because the first term vanishes on constant f.

Contrary to white noise, this prior enforces some smoothness over functions: the values at related states are closer.

The Dirichlet norm is directly related to Laplacian eigenfunctions. Indeed, when ρ is the invariant distribution of the policy in the dataset [2], one has

$$\|f\|_{\text{Dir}}^2 = 2\langle f, \Delta f\rangle_\rho + \alpha\,\|f\|_\rho^2 \tag{9}$$

where $\Delta := \text{Id} - P_0$ is the Laplace operator of the transition operator $P_0(s_{t+1}|s_t)$ of the policy implicitly defined by the dataset.

General Gaussian priors. To avoid proving the same results separately for white noise and Dirichlet priors, we will more generally use priors of the form

$$\beta(r) \propto \exp(-\|r\|_K^2) \tag{10}$$

where $\|f\|_K^2$ denotes an arbitrary symmetric positive-definite quadratic form on reward functions.

On a finite state space, this is equivalent to $\exp(-r^\top K r/2)$ for some p.s.d. matrix K of size $\#S \times \#S$. For instance, on a finite state space, the white noise prior corresponds to $K = \text{diag}(\rho)$, and the Dirichlet prior is given by the matrix $K = \mathbb{E}_{(s,s')\sim\rho}[(\mathbb{1}_s - \mathbb{1}_{s'})(\mathbb{1}_s - \mathbb{1}_{s'})^\top] + \alpha\mathbb{E}_{s\sim\rho}[\mathbb{1}_s\mathbb{1}_s^\top]$.

On infinite state spaces, this is an "infinite-dimensional Gaussian" whose formal definition involves having a consistent set of Gaussian distributions in every finite-dimensional projection.

We will also use the associated dot product $\langle f, g\rangle_K$. For instance,

$$\langle f, g\rangle_{\text{Dir}} = \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\rho}\,(f(s_t)-f(s_{t+1}))(g(s_t)-g(s_{t+1}))+\alpha f(s_t)g(s_t). \tag{11}$$

Like $\|f\|_{\text{Dir}}$, this can be estimated from the dataset.

Remark 1. In general, the optimal features are *not* directly related to the largest eigenvectors or singular vectors of K. For instance, the white noise prior corresponds to $K = \text{diag}(\rho)$, whose eigendecomposition is independent of the dynamics of the environment, while optimal features depend on the dynamics.

[2] More precisely, it is sufficient that the distributions of s_t and s_{t+1} under the distribution ρ in the dataset are the same. This does not require the existence of a specific policy that produced the dataset. For instance, if a dataset is a mixture of long trajectories from several policies, then the laws of s_t and of s_{t+1} in the dataset will be almost the same (up to neglecting the first and last state of each trajectory).

2.3.2 Sparse Reward Priors

Random goal-reaching prior. A *goal-reaching* reward is a reward that is nonzero only at a particular state, and 0 everywhere else.

If the prior β on downstream tasks only includes goal-reaching tasks (with some distribution of goals g), then arguably zero-shot RL is not needed: it is better to just do goal-reaching, namely, directly use $z = g$ as the task representation for goal g, and learn $Q(s, g)$ via algorithms such as HER [ACR+17].

But we want to mix goal-reaching with other priors, and find zero-shot RL methods that can work in a mixture of different priors, hence the interest of a general setup. So we formally define here a goal-reaching prior.

In this model, we first select a random state $s^\star \sim \rho$ in S. Then we put a reward $1/\rho(s^\star)$ at $s^\star$, and 0 everywhere else:

$$r(s) = \frac{1}{\rho(s^\star)} \mathbb{1}_{s=s^\star}. \tag{12}$$

The $1/\rho$ factor maintains $\int r \, \mathrm{d}\rho = 1$. Without this scaling, all Q-functions degenerate to 0 in continuous spaces, as discussed in [BO21]. Indeed, if we omit this factor, and just set the reward to be 1 at a given goal state $s^\star \in S$ in a continuous space S, the probability of exactly reaching that state with a stochastic policy is usually 0, and all Q-functions are 0. Thanks to the $1/\rho$ factor, the continuous limit is a *Dirac function* reward, infinitely sparse, corresponding to the limit of putting a reward 1 in a small ball $B(s^\star, \varepsilon)$ of radius $\varepsilon \to 0$ around $s^\star$, and rescaling by $1/\rho(B(s^\star, \varepsilon))$ to keep $\int r \, \mathrm{d}\rho = 1$. This produces meaningful, nonzero Q-functions in the continuous limit [BO21].

This model combines well with successor features or the FB framework: indeed, this model satisfies

$$\mathbb{E}_{s\sim\rho}[r(s)\varphi(s)] = \varphi(s^\star) \tag{13}$$

(both in finite spaces and in the continuous-space limit). This is useful in conjunction with the SF formulas such as (18) in Section 3.2.

Scattered random reward prior. We extend the random goal-reaching prior to rewards comprising several goals with various weights, where the weights may be random and may be positive or negative.

Generally speaking, we will call *scattered random reward prior* any prior which consists in first choosing an integer $k \geqslant 0$ according to some probability distribution, then choosing k goal states $(s_i^\star)_{1\leqslant i\leqslant k} \sim \rho$ and k random weights $w_i, \dots, w_k$ according to some fixed probability distribution on $\mathbb{R}$, and setting

$$r(s) = c_k \sum_{i=1}^{k} \frac{w_i}{\rho(s_i^\star)} \mathbb{1}_{s=s_i^\star} \tag{14}$$

namely, a sum of k goal-reaching rewards (12).

A suitable scaling factor c_k can sometimes produce more meaningful behavior for large k. For instance, if we take $w_i \sim N(0,1)$ and $c_k = 1/\sqrt{k}$, and let $k \to \infty$, then this prior tends to the white noise prior above.

Therefore, scattered random reward priors can be seen as interpolating between the pure goal-reaching and white noise priors.

3 Algorithmic Tractability of the Zero-Shot RL Loss

3.1 The Optimal Policies Given a Representation Φ

Here we work out half of the objective (5): what are the optimal policies π_z if the task representation Φ is known?

Proposition 2 (Policies must be optimal for the mean posterior reward knowing z). *For each z, define*

$$r_z := \mathbb{E}_{r|\Phi(r)=z}[r] \tag{15}$$

the mean reward function knowing $\Phi(r) = z$ under the prior β. Let also β_z be the distribution of $z = \Phi(r)$ when sampling $r \sim \beta$.

Then

$$\ell_\beta(\Phi, \pi) = -\mathbb{E}_{z\sim\beta_z}\, \mathbb{E}_{s_0\sim\rho_0} V_{r_z}^{\pi_z}(s_0). \tag{16}$$

Consequently, given the representation Φ, for every z, the best policy π_z is the optimal policy $\pi^\star_{r_z}$ for reward r_z.

So, in this model, the optimal zero-shot policies have no induced stochasticity to account for uncertainties. This holds even if there is noise in the computation of z. The full point of zero-shot RL is to decide which rewards to lump together under the same policy.

This does not hold if one includes variance over r in the main loss (5) (Section B).

The value of r_z can be derived explicitly for some priors (Gaussian priors and linear Φ), which we now turn to. Other priors (goal-oriented or scattered random rewards) require a slightly different approach (Section 3.5).

3.2 Linear Task Representations Φ

We now emphasize the case of *linear* task representations Φ, because it corresponds to successor features and to the forward-backward framework, which are the most successful zero-shot RL approaches to date. (See Section 4.2 for nonlinear Φ.)

The easiest-to-compute task representations $z = \Phi(r)$ are linear functions of r. Any such finite-dimensional function Φ is given by integrating the reward against some features $\varphi = (\varphi_i(s))_{i=1,\dots,k}$:

$$z = (\operatorname{Cov} \varphi)^{-1} \mathbb{E}_{s\sim\rho}\, r(s)\varphi(s) \tag{17}$$

where we include a preconditioning by $(\operatorname{Cov}\varphi)^{-1}$ as in SFs. [3] Here all covariances are expressed with respect to the state distribution ρ: $\operatorname{Cov}\varphi := \mathbb{E}_{s\sim\rho}\,\varphi(s)\varphi(s)^\top$.

By Proposition 2, given the features φ, the best policies are the optimal policies for the rewards r_z. So we have to compute r_z. Then the policies can be learned, e.g., via Q-learning for each z.

The following result specifies the value of r_z and hence the optimal policies given the features, but does not yet say how to choose the features φ: this is covered in the next sections.

Proposition 3 (Linear task representations and white noise prior). *Assume the reward representation $z = \Phi(r)$ is given by the successor feature model*

$$z = (\operatorname{Cov} \varphi)^{-1} \mathbb{E}_{s\sim\rho}\, r(s)\varphi(s) \tag{18}$$

using some linearly independent features $\varphi\colon S \to \mathbb{R}^d$.

Then, for the white noise prior on rewards, the posterior mean reward r_z (15) is

$$r_z(s) = z^\top \varphi(s). \tag{19}$$

Therefore, by Proposition 2, for a given φ, the policies π_z that optimize the zero-shot RL loss (5) are the optimal policies for reward $z^\top\varphi$, for each z.

Moreover, under these assumptions, the distribution β_z of z is a centered Gaussian with covariance matrix $(\operatorname{Cov}\varphi)^{-1}$.

This proposition gives a justification for part of the strategy behind successor features, namely, projecting the reward onto the features and applying the optimal policy for the projected reward. This is optimal on average under an implicit *white noise prior* on rewards.

In the forward-backward framework (FB), the task representation z is computed as $z = \mathbb{E}_{s\sim\rho} r(s)B(s)$ with features B. This is the same as (18) up to the change of variables by $(\operatorname{Cov}\varphi)^{-1}$. Therefore, this result strongly suggests to train policies π_z for the rewards $z^\top(\operatorname{Cov} B)^{-1}B$. This contrasts with the FB framework, in which the policies π_z are defined through the forward function F. The two coincide only if the training of F is perfect. [4]

[3] Including $(\operatorname{Cov}\varphi)^{-1}$ from the start (as opposed to $z = \mathbb{E}_{s\sim\rho}\, r(s)\varphi(s)$ as in FB) is more adapted to distribution shifts. Indeed, for rewards in the span of φ, then the reward representation z is independent of the distribution ρ of states used for the computation.

[4] because in that case, F contains the successor features of $(\operatorname{Cov} B)^{-1}B$, by one of the results in [TO21].

In general, the proposition is *not* true for other reward priors, such as random goal-reaching. [5] Still, (19) also holds for any Gaussian prior on rewards such that the components of r along φ and its $L^2(\rho)$-orthogonal are independent, namely, $r(s) = \theta_1^\top \varphi(s) + \theta_2^\top \xi(s)$ where ξ are any features such that $\mathbb{E}_{s\sim\rho}[\varphi(s)\xi(s)^\top] = 0$ and θ_1, θ_2 are independent Gaussian vectors with any covariance matrix. But this cannot be used to optimize the features φ, because this condition depends on φ itself so it does not represent a fixed prior for the loss (5).

This result extends to the more general case of arbitrary Gaussian priors given by a metric $\|\cdot\|_K$: we just have to compute z by a formula involving this norm, instead of the SF formula (18).

This is especially relevant if K can be computed from expectations over the dataset, as with the Dirichlet prior: this results in an SF-like approach, but relying on a different implicit prior instead of the white noise prior on rewards.

Proposition 4 (Linear representations with arbitrary Gaussian prior). *Assume that the prior on rewards is*

$$\beta(r) \propto \exp(-\tfrac{1}{2}\|r\|_K^2) \tag{20}$$

for some Euclidean norm $\|\cdot\|_K$ on the space of rewards.

Assume the reward representation $z = \Phi(r)$ is computed as

$$z = C^{-1}\langle r, \varphi\rangle_K \tag{21}$$

using some linearly independent features $\varphi\colon S \to \mathbb{R}^d$, where C is the $k \times k$ matrix with entries $C_{ij} = \langle\varphi_i, \varphi_j\rangle_K$. Namely, z contains the weights of the $L^2(\|\cdot\|_K)$-orthogonal projection of r onto the features φ.

Then the posterior mean reward r_z given z is

$$r_z(s) = z^\top \varphi(s) \tag{22}$$

Therefore, by Proposition 2, for a given φ, the policies π_z that optimize the zero-shot RL loss (5) are the optimal policies for reward $z^\top\varphi$, for each z.

Moreover, the distribution β_z of z is Gaussian with covariance matrix $(\langle\varphi, \varphi\rangle_K)^{-1}$.

For instance, with the Dirichlet prior, we have

$$\langle\varphi, r\rangle_{\mathrm{Dir}} = \mathbb{E}_{(s_t, s_{t+1})\sim\rho}\,(r(s_t) - r(s_{t+1}))(\varphi(s_t) - \varphi(s_{t+1})) + \alpha\mathbb{E}_{s\sim\rho}\, r(s)\varphi(s) \tag{23}$$

[5] For instance, take any set of features such that $\varphi\colon S \to \mathbb{R}^d$ is injective, such as $\varphi = \mathrm{Id}$. Take for β the goal-reaching prior. Then for reaching a goal g, the reward r is a Dirac at g so that $\mathbb{E}[r\varphi] = \varphi(g)$ and $z = C^{-1}g$ with C the covariance matrix. Since the map $g \to z$ is bijective, it conveys full knowledge of the task for this prior, and the posterior mean r_z is just the single reward for reaching g.

and

$$\langle \varphi, \varphi \rangle_{\mathrm{Dir}} = \mathbb{E}_{(s_t, s_{t+1}) \sim \rho} \left(\varphi(s_t) - \varphi(s_{t+1})\right)\left(\varphi(s_t) - \varphi(s_{t+1})\right)^\top + \alpha \mathbb{E}_{s \sim \rho} \varphi(s) \varphi(s)^\top \tag{24}$$

and so z can be estimated from samples. This gives rise to a Dirichlet-prior-based version of successor features. [6]

Remark 5. It is also possible to train a Gaussian $\exp(-\|r\|_K^2)$ prior while using features $z = (\mathbb{E}\varphi\varphi^\top)^{-1}\mathbb{E}r\varphi$ that do not use the K-norm. But in that case the expression for the posterior mean r_z is much more complex and requires inverting K.

3.3 The Zero-Shot Loss is Tractable for Linear Representations

These results for fixed φ pave the way to computing the gradient of the zero-shot loss (5) with respect to φ: putting together all the ingredients yields the following result.

Theorem 6 (Zero-shot RL loss for linear task representations). *Assume that the prior β on reward functions r is $\beta(r) \propto \exp(-\frac{1}{2}\|r\|_K^2)$ for some Euclidean norm $\|\cdot\|_K$. Assume that the reward representation $z = \Phi(r)$ is computed as in successor features (21) using the norm $\|\cdot\|_K$, namely,*

$$z = C^{-1}\langle r, \varphi \rangle_K \tag{25}$$

where $\varphi\colon S \to \mathbb{R}^d$ are linearly independent features, and where C is the matrix with entries $C_{ij} = \langle \varphi_i, \varphi_j \rangle_K$.

Then the zero-shot RL loss (5) is

$$\ell_\beta(\Phi, \pi) = -\frac{1}{1-\gamma}\mathbb{E}_{z \sim N(0, C^{-1})}\, \mathbb{E}_{s \sim d_{\pi_z}} \varphi(s)^\top z \tag{26}$$

where d_{π_z} is the occupation measure (2) of policy π_z.

Moreover, the optimal π_z given Φ is the optimal policy for reward $r_z(s) := \varphi(s)^\top z$.

Relationship with VISR [HDB+19]: VISR almost optimizes expected downstream performance under a white noise prior. Surprisingly, the loss (26) is very close to the loss optimized in VISR, although VISR was built in a different way with no formal connection to expected downstream task performance.

[6] Depending on how the reward is specified for zero-shot RL, in some situations, we might not have access to both $r(s_t)$ and $r(s_{t+1})$. And for goal-oriented tasks, we usually don't have access to $\varphi(s_{t+1})$, the state visited one step after reaching the goal.

This contrasts with basic successor features, for which setting a goal state $s^\star$ just gives $z \propto (\mathrm{Cov}\, \varphi)^{-1}\varphi(s^\star)$.

VISR is a criterion to build features φ for successor features. It works with a set of features φ and policies π_z. Each policy π_z is the optimal policy for reward function $\varphi^\top z$. The features φ are chosen to maximize the mutual information between z and the states s visited by π_z; more exactly, the states s are assumed to be observed only through $\varphi(s)$, and the distribution of z knowing $\varphi(s)$ is assumed to follow a Von Mises–Fisher distribution $\exp(\varphi^\top z)$ (this is chosen for convenience so that the log-likelihood $\varphi^\top z$ matches with the reward). The features φ attempt to maximize the mutual information under this model of z given s; this mutual information is estimated via a variational lower bound. We refer to the VISR paper [HDB+19] for further details.

Yet it turns out Algorithm 1 in [HDB+19] optimizes the loss (26) above, except for a difference in the way z and φ are normalized. More precisely, the VISR algorithm consists in:

1. Sampling a hidden vector z (denoted w in [HDB+19]).
2. Training the policy π_z to optimize the reward $\varphi^\top z$. This is done in VISR via the computation of the successor features ψ of φ.
3. Running the policy π_z to get a sequence of states s_t, whose distribution is thus d_{π_z}.
4. Updating the features φ to minimize $-\varphi(s_t)^\top z$.

VISR "almost" optimizes the loss (26): the only difference between VISR and Theorem 6 lies in the normalization of z and φ. In Theorem 6, we sample z from $N(0, C^{-1})$ where C is the covariance matrix of φ, and we have no constraint on φ. In VISR, z is sampled from $N(0, \mathrm{Id})$ then normalized to unit length, and the features φ use a normalized output layer so that $\|\varphi(s)\| = 1$ for any state s.

Normalization is necessary in VISR: otherwise, the loss of φ can be brought to 0 by downscaling φ. On the other hand, in Theorem 6, if we downscale φ, the distribution $z \sim N(0, C^{-1})$ gets *up*scaled by the same factor so $\varphi^\top z$ is unchanged. This emphasizes the role of sampling z with covariance matrix C^{-1}. Also note that the normalization $\|\varphi(s)\| = 1$ in VISR does *not* imply that the covariance matrix of φ is $C = \mathrm{Id}$. So there is a slight mismatch between the VISR objective and the zero-shot RL loss.

Still, Theorem 6 proves that *VISR "almost" optimizes the expected downstream performance of π_z under a white noise prior on reward functions*, where the "almost" accounts for the difference in normalization and covariance of z. This is surprising, as expected downstream performance was not explicitly used to derive VISR.

3.4 Algorithms for Optimizing the Representation φ

A generic VISR-like algorithm to optimize the zero-shot RL loss (26) in Theorem 6 may have the following structure:

1. Sample a minibatch of z values.
2. Do a policy optimization step to bring π_z closer to the optimal policy for reward $\varphi(s)^\top z$.
3. Estimate the occupation measures d_{π_z} of π_z.
4. Do a gradient step on φ using the loss (26).
5. Iterate.

We present one possible such algorithm in Algorithm 1. It departs from VISR in three ways:

- Fixing normalization and influence of C: sampling z from $N(0, C^{-1})$. An extra complication occurs: since C depends on φ, it is necessary to estimate the gradients coming from $C^{-1} = (\langle \varphi, \varphi \rangle_K)^{-1}$ when taking gradients with respect to φ.

  This ensures we exactly optimize te zero-shot RL loss (26).

- Estimating a model of the occupation measures d_{π_z}. VISR obtains sample states $s \sim d_{\pi_z}$ by running trajectories of π_z and using a Monte Carlo estimate by averaging over these trajectories. This both suffers from high variance and limits applicability to the online RL setup, since interactions with the environment are needed during training.

  Instead, learning a model of d_{π_z} allows Algorithm 1 to run in an offline RL setting. It should also result in larger bias but smaller variance with respect to Monte Carlo sampling from d_{π_z}.

- Simplifying the learning of π_z: this can be done using any Q-learning algorithm with z-dependent Q-function $Q(s, a, z)$ for reward $\varphi(s)^\top z$. It does not have to use the successor features of φ as in VISR.

Let us further discuss two of these points (gradients coming from C, and estimating d_{π_z}). The exact derivations are included in Appendix A.

Learning the occupation measures d_{π_z}. Instead of explicitly running the policy π_z as in VISR, a number of techniques allow for direct estimation of the density of d_{π_z}.

Indeed, $d_{\pi_z}(s)$ is the average over $s_0 \sim \rho_0$ of the *successor measures* $M^{\pi_z}(s_0, a_0, s)$, multiplied by $(1 - \gamma)$. We refer to [BTO21] or to Appendix A

Algorithm 1 One possible algorithm to optimize the zero-shot RL loss (26)

Input:
Dataset of transitions (s_t, a_t, s_{t+1}) with distribution ρ.
Norm $\|\cdot\|_K$ on features (default: $\|\varphi\|_K^2 := \mathbb{E}_{s\sim\rho}|\varphi(s)|^2$), and associated dot product.
Weights $\lambda_C \in \{0, 1\}$, $\lambda_{\text{orth}} \geqslant 0$ for auxiliary losses.
Online EMA weights $\beta_t \in (0, 1)$ to estimate C.
Output:
Trained features $\varphi_1, \dots, \varphi_d$ with their covariance matrix C.
Trained policies π_z.
while not done **do**
  Update covariance matrix C via EMA: $C_{ij} \leftarrow \beta_t C_{ij} + (1 - \beta_t)\langle\varphi_i, \varphi_j\rangle_K$
  Sample a minibatch of values of z: $z \sim N(0, C^{-1})$
  Update a Q-function $Q(s, a, z)$ and policy $\pi_z(a|s)$ for reward $\varphi(s)^\top z$, using any RL algorithm
  Update the occupation measure model $d(s, z)$ via one step of Algorithm 2
  Sample a minibatch of states s from the dataset, and update φ with the loss
    $\mathcal{L}(\varphi) = -d(s, z)\varphi(s)^\top z + \lambda_C \mathcal{L}_C(\varphi, s, z) + \lambda_{\text{orth}} \mathcal{L}_{\text{orth}}(\varphi)$
  where $\mathcal{L}_C$ and $\mathcal{L}_{\text{orth}}$ are the auxiliary losses (30) and (27) respectively
end while
Deployment:
Once the reward function r is known:
Estimate $\langle r, \varphi_1\rangle_K, \dots \langle r, \varphi_d\rangle_K$
Set $z = C^{-1}\langle r, \varphi\rangle_K$
Apply policy π_z

for successor measures: intuitively, $M^{\pi_z}(s_0, a_0, s)$ encodes the expected amount of time spent at s if starting at (s_0, a_0) and running π_z.

Algorithm 2 first learns a model $m(s_0, a_0, s, z)$ of the successor measure, using one of the methods from [BTO21] (the measure-valued Bellman equation satisfied by successor measures). Then it averages the result over s_0 and a_0 to obtained the model $d(s, z)$ of the occupation measure d_{π_z}. The mathematical derivations are given in Appendix A. The model $m(s_0, a_0, s, z)$ may take any form; a particular case is a finite-rank approximation $m(s_0, a_0, s, z) = F(s_0, a_0, z)^\top B(s, z)$ similar to the forward-backward representation from [TO21], except that here B can be z-dependent. [7]

Dealing with the covariance matrix C. In the loss (26), the variable z is sampled from $z \sim N(0, C^{-1})$. Since C depends on φ, this produces extra terms when attempting to optimize the loss over φ.

Here a reparameterization trick $z \leftarrow C^{1/2}z$ is inconvenient, because it still requires computing the gradient of $C^{-1/2}$ with respect to C, and this

[7] A model $m(s_0, a_0, s, z) = F(s_0, a_0, z)^\top B(s)$, with B independent of z, would be too restrictive here: in this model, everything is projected onto the span of B, and the optimal φ is just B.

Algorithm 2 One possible algorithm to estimate occupation measures $d(s, z)$

Input: Dataset of transitions (s_t, a_t, s_{t+1}) with distribution ρ.
Distribution of initial states ρ_0 (default: $\rho_0 = \rho$).
Policies $\pi_z(a|s)$.
Covariance matrix C for sampling z.
Output: Trained occupation model $d(s, z)$.
while not done **do**
  Sample a minibatch of values of z: $z \sim N(0, C^{-1})$
  Sample a minibatch of transitions $(s_t, a_t, s_{t+1}) \sim \rho$
  Sample actions $a_{t+1} \sim \pi_z(a_{t+1}|s_{t+1})$
  Sample a minibatch of states $s' \sim \rho$
  Update the successor measure model $m(s_t, a_t, s', z)$ with the loss
    $\mathcal{L}(m) = \left(m(s_t, a_t, s', z) - \gamma \bar{m}(s_{t+1}, a_{t+1}, s', z)\right)^2 - 2m(s_t, a_t, s_t, z)$ with $\bar{m}$ a target network version of m (using EMA of parameters of m and a stop-grad)
  Sample a minibatch of initial states $s_0 \sim \rho_0$ and actions $a_0 \sim \pi_z(a_0|s_0)$
  Update the occupation measure model $d(s, z)$ with the loss
    $\mathcal{L}(d) = \left(d(s', z) - (1-\gamma) m(s_0, a_0, s', z)\right)^2$
end while

requires inverting a $d^2 \times d^2$ matrix, not just a $d \times d$ matrix.

Instead, two other strategies are possible:

1. Only work with orthonormal features, i.e., impose $C = \mathrm{Id}$ at all times. This is possible without loss of generality, because zero-shot RL with linear features only depends on the linear span of the features.

   In practice, this can be done by imposing a Lagrange multiplier for the constraint $C = \mathrm{Id}$. This means adding a loss term $\lambda_{\text{orth}} \mathcal{L}_{\text{orth}}(\varphi)$ in the algorithm, where λ_{orth} is a large weight, and where

   $$\mathcal{L}_{\text{orth}}(\varphi) := \|\langle \varphi, \varphi \rangle_K - \mathrm{Id}\|^2_{\text{Frobenius}} \tag{27}$$
   $$= -2 \sum_i \|\varphi_i\|^2_K + \sum_{ij} \left(\langle \varphi_i, \varphi_j \rangle_K\right)^2 + \text{cst} \tag{28}$$

   is the loss associated with violating the constraint $C = \mathrm{Id}$. This option corresponds to $\lambda_C = 0$ in Algorithm 1.

   With $\|\cdot\|_K = \|\cdot\|_\rho$, this loss simplifies to

   $$\mathcal{L}_{\text{orth}}(\varphi) = \mathbb{E}_{s \sim \rho,\, s' \sim \rho} \left[\left(\varphi(s)^\top \varphi(s') \right)^2 - \|\varphi(s)\|^2 - \|\varphi(s')\|^2 \right] + \text{cst} \tag{29}$$

   similarly to the orthonormalization loss for B in [TRO23].

   Even with a large weight λ_{orth}, the condition $C = \mathrm{Id}$ will be satisfied only approximately. Thus we still include C in the algorithm.

   When using the orthonormalization loss $\mathcal{L}_{\text{orth}}$ with a large weight, it is better if φ is initialized so that C is not too far from Id.

2. The second option provides an exact estimation of the gradient of C. This is carried out in Appendix A, and results in the following loss $\mathcal{L}_C$ included in Algorithm 1:

$$\mathcal{L}_C(\varphi, s, z) := \tfrac{1}{2} d(s, z) \left(\bar{\varphi}(s)^\top z\right) \sum_{ij} \left((\bar{C}^{-1})_{ij} - z_i z_j\right) \langle \varphi_i, \varphi_j \rangle_K \quad (30)$$

where $\bar{\varphi}$ and $\bar{C}$ are stop-grad versions of φ and C, respectively.

If $\|\cdot\|_K = \|\cdot\|_\rho$, this can be estimated as

$$\mathcal{L}_C(\varphi, s, z) = \tfrac{1}{2} d(s, z) \left(\bar{\varphi}(s)^\top z\right) \mathbb{E}_{s' \sim \rho} \left[\varphi(s')^\top \bar{C}^{-1} \varphi(s') - (\varphi(s')^\top z)^2\right] \quad (31)$$

Even if using the loss $\mathcal{L}_C$, we still recommend to include a loss $\mathcal{L}_{\text{orth}}$, for numerical reasons to keep φ within a reasonable numerical range. [8]

These results make it possible to optimize the features φ for a Gaussian prior on downstream tasks. We now turn to other priors.

3.5 Learning the Optimal Features for Sparse Reward Priors

We now turn to the sparse reward priors from Section 2.3.2. Since goal-reaching is a special case of scattered random rewards (with $k = 1$), we only deal with the latter. Namely, we consider sparse rewards of the form

$$r = c_k \sum_{i=1}^{k} w_i \, \delta_{s_i^\star} \quad (32)$$

where $\delta_{s^\star}(s) := \mathbb{1}_{s=s^\star}/\rho(s^\star)$ is the Dirac sparse reward [9] at $s^\star$ as defined in Section 2.3.2, k is an integer following some probability distribution, $(s_i^\star)_{1 \leqslant i \leqslant k}$ are goal states sampled from the data distribution ρ, the w_i are weights sampled from some distribution on $\mathbb{R}$, and c_k is a scaling factor. A typical example is $w_i \sim N(0, 1)$ and $c_k = 1/\sqrt{k}$.

Arguably, with such a model, we could just send the full reward description $(s_i^\star, w_i)_{1 \leqslant i \leqslant k}$ to a Q-function or policy model. However, our goal is to be able to mix several types of priors on rewards (Section 3.6): we want to find zero-shot RL methods that work both for dense and sparse rewards. Therefore, we describe a method whose structure is closer to that of the previous sections, by learning optimal features φ.

[8] When $\mathcal{L}_C$ is included, mathematically $\mathcal{L}_{\text{orth}}$ has no effect since everything only depends on the span of φ and not φ itself. But numerically it will be more convenient to keep φ well-conditioned.

[9] Dirac with respect to the measure ρ, namely, $\mathbb{E}_\rho[f.\delta_{s^\star}] = f(s^\star)$. In particular, $\mathbb{E}_\rho \delta_{s^\star} = 1$.

Proposition 7. *Let β be a prior on sparse rewards of the type (32), for some distribution of $(k, (w_i), c_k)$ and where each $(s_i^\star, a_i^\star)$ has distribution ρ.*

Assume that the reward representation $z = \Phi(r)$ is computed as in successor features (21) using the norm $\|\cdot\|_K$, namely,

$$z = C(\varphi)^{-1}\langle r, \varphi\rangle_K \tag{33}$$

where $\varphi\colon S \to \mathbb{R}^d$ are linearly independent features, and where $C(\varphi)$ is the matrix with entries $C_{ij} = \langle\varphi_i, \varphi_j\rangle_K$.

Then the zero-shot RL loss $\ell_\beta(\Phi, \pi)$ satisfies

$$\ell_\beta(\Phi, \pi) = -\frac{1}{1-\gamma}\mathbb{E}_{k,\, s_i^\star,\, w_i} \sum_{i=1}^{k} c_k w_i\, d\left(s_i^*, z(\varphi)\right) \tag{34}$$

where

$$z(\varphi) = \sum_j c_k w_j C(\varphi)^{-1}\langle\delta_{s_j^\star}, \varphi\rangle_K \tag{35}$$

and where $d(s, z)$ is the density of $d_{\pi_z}(s)$ with respect to the data distribution ρ.

The density $d(s, z)$ is the same as in Algorithm 1, and can be learned via Algorithm 2.

Algorithm 3 instantiates this result for the case where $\|\cdot\|_K = \|\cdot\|_\rho$. In that case, we have

$$\langle\delta_{s_j^\star}, \varphi\rangle_\rho = \varphi(s_j^\star) \tag{36}$$

which simplifies the expression for z.

Two points in Algorithm 3 are tricky. The first is how to compute the gradient of $z(\varphi)$ with respect to φ, and in particular the gradient of $C(\varphi)^{-1}$. In Algorithm 3, we have used that $C = \mathbb{E}_{s'}\varphi(s')\varphi(s')^\top$ when $\|\cdot\|_K = \|\cdot\|_\rho$. We have included an extra term

$$\bar{C}^{-1}\left(\bar{\varphi}(s')\bar{\varphi}(s')^\top - \varphi(s')\varphi(s')^\top\right)\bar{C}^{-1}\bar{\varphi}(s_i^\star) \tag{37}$$

which evaluates to 0 in the forward pass (since $\bar{\varphi} = \varphi$) but provides the correct gradients with respect to $C(\varphi)^{-1}$ in the backward pass.

The second tricky point is how to update the Q-function for the sparse reward. Here we have directly applied the results from [BO21] for Q-learning with Dirac rewards such as (32). This point is important when mixing different priors (Section 3.6): the Q-functions for different priors should be updated in a consistent way, (e.g., all updated using the Bellman loss for their respective rewards).

Algorithm 3 One possible algorithm to optimize the zero-shot RL loss with sparse rewards (32)

Input:
Dataset of transitions (s_t, a_t, s_{t+1}) with distribution ρ.
Online EMA weights $\beta_t \in (0, 1)$ to estimate C.
Probability distribution on $k \in \mathbb{N}$, the number of goals in the sparse rewards.
Probability distribution on weights w_i (default: $N(0, 1)$), scaling factor c_k (default: $1/\sqrt{k}$).
Output:
Trained features $\varphi_1, \dots, \varphi_d$ with their covariance matrix C.
Trained policies π_z.
while not done **do**
  Update covariance matrix C via EMA: $C_{ij} \leftarrow \beta_t C_{ij} + (1 - \beta_t)\mathbb{E}_{s\sim\rho}\varphi(s)\varphi(s)^\top$
  Sample a value of k. Sample k goal state-actions $(s_i^\star, a_i^\star)$ from the dataset distribution ρ. Sample weights w_i.
  Sample a state $s' \sim \rho$
  Compute
    $z(\varphi) = \sum_i c_k w_i \bar{C}^{-1}\varphi(s_i^\star) + c_k w_i \bar{C}^{-1}\left(\bar{\varphi}(s')\bar{\varphi}(s')^\top - \varphi(s')\varphi(s')^\top\right)\bar{C}^{-1}\bar{\varphi}(s_i^\star)$
  where $\bar{C}$ and $\bar{\varphi}$ are stop-grad versions of C and φ
  Update a Q-function $Q(s, a, z)$ at $z = z(\varphi)$ with the Bellman loss
    $\ell(Q) = Q(s_t, a_t, z)^2 - 2\sum_i c_k w_i Q(s_i^\star, a_i^\star, z) - 2\gamma Q(s_t, a_t, z)\bar{Q}(s_{t+1}, a_{t+1}, z)$
  where (s_t, a_t, s_{t+1}) is sampled from ρ, where a_{t+1} is sampled from $\pi_z(s_{t+1})$, and where $\bar{Q}$ is a target version of Q.
  Update a policy $\pi_z(a|s)$ based on $Q(s, a, z)$, using any RL policy algorithm
  Update the occupation measure model $d(s, z)$ via one step of Algorithm 2
  Update φ with the loss
    $\mathcal{L}(\varphi) = -\sum_i c_k w_i\, d(s_i^\star, z(\varphi))$
  where the gradients w.r.t. φ are backpropagated through d and z.
end while
Deployment:
Once the reward function r is known:
Estimate $\langle r, \varphi_1\rangle_K, \dots \langle r, \varphi_d\rangle_K$
Set $z = C^{-1}\langle r, \varphi\rangle_K$
Apply policy π_z

3.6 Mixing Priors

The zero-shot RL loss (5) is linear in the prior β. Therefore, if two priors β_1 and β_2 are amenable to gradient descent for this loss, one can deal with a mixture prior just by mixing the losses for β_1 and β_2, using a single set of features φ, Q-functions, and policies π_z.

In practice, this just means choosing at random, at each step, between doing an optimization steps for one of the priors, e.g., alternating between Algorithms 1 and 3.

Of course, this requires using consistent optimization methods for both priors: the same optimizer, but also similar Bellman losses and policy updates for β_1 and β_2. For instance, β_1 and β_2 may both use the standard Bellman

loss $\Big(Q(s_t, a_t, z) - r(s_t, z) - \gamma \bar{Q}(s_{t+1}, a_{t+1}, z)\Big)^2$ where $r(s, z)$ is the mean reward knowing z for a given prior. Then if β_1 has posterior mean reward $r_1(s, z)$ knowing z and likewise for β_2, optimizing the Q-function alternatively between β_1 and β_2 effectively optimizes for the mean posterior reward of the mixture.

4 Discussion

4.1 What Kind of Features are Learned? Skill Specialization and the Zero-Shot RL Loss

The features learned are influenced by the prior, and this is one reason why mixing priors may be appealing.

For a pure goal-oriented prior, it is enough to learn a feature that represents different goals by different values of z, so, it is enough for φ to be injective (e.g., with $\dim \varphi = \dim s$ and $\varphi = \mathrm{Id}$). On a discrete space, a one-dimensional φ may solve the problem just by sending every state to a different value. Of course, this will not work when mixed with other types of rewards.

For dense Gaussian priors, on the other hand, learning may produce narrow features φ, resulting in overspecialized skills. Indeed, conceptually, from Theorem 6, gradient descent of ℓ for π_z and φ amounts to:

- Learn π_z to optimize reward $\varphi^\top z$ for each z;
- Learn φ by increasing $\varphi^\top z$ at the states visited by π_z.

The above is related to diversity methods [EGIL18] and has a "rich-get-richer" dynamics: this is good for diversifying and specializing, but might overspecialize. We illustrate this phenomenon more precisely in the next paragraph.

Understanding overspecialization: Analysis with only one feature, and influence of the prior. This is best understood on a "bandit" case (we can jump directly to any state) and with only one feature. In this case, a full analysis can be done, and the optimal one-dimensional feature has only two non-zero values: a large positive value at a state s_1 and a large negative value at a state s_2. [10]

[10] Indeed, take a finite state space $S = \{1, \dots, n\}$ and assume that at any state, there's an action directly leading to any other state: this makes the MDP into a bandit problem. Take 1-dimensional φ. Then from Theorem 6, the gradient with respect to φ is $d_{\pi_z}.z$ where π_z is the policy to maximize reward $z.\varphi$. If $z > 0$ then π_z goes to the maximum of φ, and $d_{\pi_z} = (1-\gamma)U + \gamma \mathbb{1}_{\arg\max \varphi}$ where U is the uniform distribution. If $z < 0$ then $d_{\pi_z} = (1-\gamma)U + \gamma \mathbb{1}_{\arg\min \varphi}$. Since the distribution of z is symmetric, on average the gradient w.r.t. φ is proportional to $\mathbb{1}_{\arg\max \varphi} - \mathbb{1}_{\arg\min \varphi}$. Gradient ascent on φ will

Is this specific to the "bandit" case? If the environment has full reachability (the agent can reach any state and stay there), and if the discount factor γ is close to 1, then the problem is essentially a bandit problem. The transient dynamics before reaching a target state will contribute $O(1-\gamma)$ to occupation measures d_π, any the analysis done on the bandit case will hold up to $O(1-\gamma)$.

Using a smoother prior on rewards (such as the Dirichlet prior, which favors spatially smooth rewards) does not change this: this applies to any Gaussian prior including the Dirichlet prior. The prior will influence the location of the two states s_1 and s_2 at which the feature is nonzero. [11]

Yet such features *are* optimal for the zero-shot RL loss with a Gaussian prior. In an environment with full reachability and γ close to 1, *the optimal zero-shot behavior with one feature consists in measuring the reward at two states and going to whichever of those two states has the largest reward.* This applies to any Gaussian prior on rewards, including priors whose covariance matrix produces spatial smoothness on rewards.

So, if these features are considered undesirable, this reflects a mismatch between the prior β and the true distribution of test tasks in the test loss (1). This pushes towards mixing different types of priors, such as Gaussian and sparse reward priors.

Sparse reward priors such as goal-oriented (Dirac) rewards correspond to smoother features such as $\varphi = \mathrm{Id}$. This illustrates the mathematical duality between φ and r when estimating z via $\mathbb{E}[r.\varphi]$: smoother priors on r may lead to *less* smooth features φ (the dual of a space of smoother functions contains less smooth functions). Intuitively, with sparse rewards r, the features φ must be able to "catch" the location of the reward anywhere in the space, and cannot be zero almost-everywhere.

Does the Bayesian viewpoint regularize the optimal features? One might have expected that the Bayesian flavor of the zero-shot RL objective would result in regularized policies. But this is not the case: by Proposition 2, every policy π_z is a "sharp" policy, in the sense that it is optimal for some reward r_z. Uncertainty on the reward does not induce noise on the policy: if the maximum of r_z is reachable, π_z will go straight to it and stay there. This contrasts with the effect of regularizations such as an entropy regularization, which adds noise to the policy.

Yet this is the "correct" (optimal) answer given the zero-shot RL loss.

converges to a φ that has only two nonzero values, one positive and one negative. (The cases where there are ties between the values of φ at several states are numerically unstable.) This applies to any Gaussian prior on rewards.

[11] Intuitively, the feature φ only looks at the reward at states s_1 and s_2 before choosing and applying a policy. With a Dirichlet prior, nearby states have correlated rewards, so looking at the reward at s_2 does not bring much information if s_2 is close to s_1: it brings more information to measure the reward at distant states s_1 and s_2.

This overspecialization tendency has already been observed for diversity methods: for instance, [ESL21] also find that skills learned must be optimal for some particular downstream task, although they work from an information criterion and not from the zero-shot RL loss. This seems to be an intrinsic property of this general approach.

This illustrates the main assumption in the zero-shot RL framework: at test time, the reward function is fully known and one can compute $z = \Phi(r)$. This leaves no space for uncertainties on r, or any fine-tuning based on further reward observations. The model estimates z, then applies a policy that will maximize the mean posterior reward r_z, e.g., by going to the maximum of r_z and staying there if possible.

This optimal only if no uncertainty exists on r and no fine-tuning of the policies is possible.

Comparison with the Forward-Backward framework. The results in this text shows that forward-backward representations [TRO23, TO21] have no reason to be optimal: If the prior β on tasks is known, then one shold optimize the features for that prior.

However, the discussion above shows that the priors for which we can compute optimal features may not necessarily reflect the kind of features we expect to learn. Mixing different priors should mitigate that effect, but to what extent is currently unclear.

Forward-backward representations aim at learning features that can faithfully represent the long-term dynamics (successor measures) of many policies. This is a different kind of implicit prior, closer in spirit to a world model, and with no explicit distribution over downstream tasks.

4.2 Future Directions

Avoiding overspecialized skills. The zero-shot RL loss can lead to very narrow optimal features with a Gaussian prior, as we have seen. This is optimal for the loss (5), but not what we want in general, possibly reflecting a mismatch between a Gaussian prior and "interesting" rewards.

One possible solution is to mix different priors.

Another possible solution is to account for variance over downstream tasks in the zero-shot loss: we not only want the best expected performance, but we don't want performance to be very bad for some tasks. For the white noise prior, a full analysis is possible (Appendix B): incorporating variance is equivalent to penalizing the L^2 norm of the occupation measures d_π (this will minimize spatial variance, thus "spreading" d_π). However, it is not obvious how to exploit this algorithmically (since d_π is computed from π and not the other way around, adding a penalty on d_π will just make the computation of d_π wrong). Things are actually simpler if we add a downstream task variance penalty to the FB framework (Appendix B).

Other solutions are to explicitly regularize the features (e.g., minimize their spatial variance, or their Dirichlet norm to impose temporal smoothness) or the policies (e.g., by entropy regularization). But since the overspecialized features actually optimize the zero-shot RL loss (for some priors), it is more principled to regularize the loss itself.

Nonlinear task representations. In this text, we have covered linear task representations, as these are the ones in the main zero-shot frameworks available (successor features and forward-backward). However, a linear task representation $r \mapsto z$ clearly limits the expressivity of zero-shot RL.

One way to get nonlinear reward representations, introduced in [CTO24], is to iterate linear reward representations in a hierarchical manner:

$$z_1 = \mathbb{E}\, r(s)\varphi_1(s), \qquad z_2 = \mathbb{E}\, r(s)\varphi_2(s, z_1), \qquad z_3 = \dots \tag{38}$$

namely, z_1 provides a rough first reward representation, which can be used to adjust features φ_2 more precisely to the reward function. [CTO24] prove that two such levels already provides full expressivity for the correspondence $r \mapsto z$. This is amenable to a similar analysis as the one performed in this text. The sparse reward case looks largely unchanged, but the case of Gaussian priors is more complex: the covariance matrix C now depends on z_1, so it would have to be represented via a learned model or estimated on a minibatch. We leave this for future work.

Another way to bypass the linearity of the task representation would be to kernelize the norm $\|r\|_K$ used in the definition of Gaussian reward priors.

Incorporating fine-tuning and reward uncertainty at test time. Finally, the analysis here relies the main hypothesis behind the zero-shot RL framework: that at test time, the reward function is instantly and exactly known. This is the case in some scenarios (eg, goal-reaching, or letting a user specify a precise task), but not all. In such situations, some fine-tuning of the policies will be necessary. Which features provide the best initial guess for real-time fine-tuning is out of the scope of this text. Zero-shot RL assumes the reward function is fully specified at test time: if it is not, then meta-RL approaches [BVL+23] probably provide a better solution.

5 Conclusions

The zero-shot RL loss is the expected policy performance of a zero-shot RL method on a distribution of downstream tasks. We have shown that this loss is algorithmically tractable for a number of uninformative priors on downward tasks, such as white noise, other Gaussian distributions favoring spatial smoothness, and sparse reward priors such as goal-reaching or random combinations of goals. We recover VISR as a particular case for the white

noise prior. We have also illustrated how dense Gaussian reward priors can lead to very narrow optimal features, which suggests that a mixture of different priors could work best.

A Additional Proofs

Proof of Proposition 2.
This is because Q-functions are linear in r for a given policy. Intuitively, all rewards represented by z will share policy π_z, and so the average return over rewards is the return of the average reward among those represented by z. More precisely, by definition of β_z, and by (3), the loss rewrites as

$$\begin{aligned}
\ell_\beta(\Phi, \pi) &= -\mathbb{E}_{z\sim\beta_z}\, \mathbb{E}_{r|\Phi(r)=z}\, \mathbb{E}_{s_0\sim\rho_0}\, V_r^{\pi_z}(s_0) && (39)\\
&= -\frac{1}{1-\gamma}\mathbb{E}_{z\sim\beta_z}\, \mathbb{E}_{r|\Phi(r)=z}\, \mathbb{E}_{s\sim d_{\pi_z}}\, r(s) && (40)\\
&= -\frac{1}{1-\gamma}\mathbb{E}_{z\sim\beta_z}\, \mathbb{E}_{s\sim d_{\pi_z}}\, \Big(\mathbb{E}_{r|\Phi(r)=z}\, r(s)\Big) && (41)\\
&= -\frac{1}{1-\gamma}\mathbb{E}_{z\sim\beta_z}\, \mathbb{E}_{s\sim d_{\pi_z}}\, r_z(s) && (42)\\
&= -\mathbb{E}_{z\sim\beta_z}\, \mathbb{E}_{s_0\sim\rho_0}\, V_{r_z}^{\pi_z}(s_0) && (43)
\end{aligned}$$

as needed. □

Proof of Propositions 3 and 4.
Since Proposition 3 is a particular case of Proposition 4, we only prove the latter.

By definition, the reward r is a centered Gaussian vector with probability density $\exp(-\|r\|_K^2/2)$.

The posterior mean reward r_z is the expectation of r knowing

$$z = C^{-1}\langle r, \varphi\rangle_K \tag{44}$$

where $\langle\cdot,\cdot\rangle_K$ is the dot product associated with the quadratic form $\|\cdot\|_K^2$, and $C = \langle\varphi,\varphi\rangle_K$ is the K-covariance matrix of the features φ,namely $C_{ij} = \langle\varphi_i,\varphi_j\rangle_K$.

Without loss of generality, by the change of variables $\varphi \leftarrow C^{-1/2}\varphi$ (which yields $z \leftarrow C^{1/2}z$), we can assume that $C = \mathrm{Id}$, namely, the features φ are K-orthonormal. So we must compute the mean of r knowing $z = \langle r, \varphi\rangle_K$.

Since φ is k-dimensional, this is a set of k constraints $\langle r, \varphi_1\rangle_K = z_1, \ldots, \langle r, \varphi_d\rangle_K = z_d$.

These k constraints define a codimension-k affine hyperplane in the space of reward functions. We have to compute the expectation of r conditioned to r lying on this hyperplane.

For any Euclidean norm $\|\cdot\|$, the restriction of a centered Gaussian distribution $\exp(-\|x\|^2/2)$ to an affine subspace is again a Gaussian distribution,

whose mean is equal to the point of smallest norm in the subspace. (This can be proved for instance by applying a rotation so the affine subspace aligns with coordinate planes, at which point the result is immediate.)

Therefore, the posterior mean r_z is the reward function that minimizes $\|r_z\|_K^2$ given the constraints $\langle r, \varphi_1\rangle_K = z_1, \dots, \langle r, \varphi_d\rangle_K = z_d$. Since the φ_i are K-orthonormal, this is easily seen to be $z_1\varphi_1 + \cdots + z_d\varphi_d$. This proves the claim about the posterior mean reward r_z.

For the claim about the distribution of z, assume again that the set of features φ is K-orthonormal ($C = \mathrm{Id}$). Completing φ into a K-orthonormal basis, the Gaussian prior $\exp(-\|r\|_K^2)$ means that all components of r onto this basis are one-dimensional standard Gaussian variables. So $z = \langle r, \varphi\rangle_K$ is a k-dimensional standard Gaussian. Undoing the change of variables with C, namely, $\varphi \leftarrow C^{1/2}\varphi$ and $z \leftarrow C^{-1/2}z$, results in z having covariance C^{-1}. □

Proof of Theorem 6.
Theorem 6 is a direct consequence of Proposition 4, Proposition 2, the definition of ℓ_β, and the expression (3) of V-functions using the occupation measures d_π. □

Derivation of Algorithm 2. The *successor measure* [BTO21] of a policy π is a measure over the state space S depending on an initial state-action pair (s_0, a_0). It encodes the expected total time spent in any part $X \subset S$, if starting at (s_0, a_0) and following π. The formal definition is

$$M^\pi(s_0, a_0, X) := \sum_{t \geqslant 0} \Pr(s_t \in X | s_0, a_0, \pi) \tag{45}$$

for each $X \subset S$.

By definition, the occupation measure (2) is the average of the successor measure over the initial state and action:

$$d_\pi(X) = (1-\gamma)\mathbb{E}_{s_0 \sim \rho_0,\, a_0 \sim \pi(a_0|s_0)} M^\pi(s_0, a_0, X). \tag{46}$$

A parametric model of M^π can be learned through various measure-valued Bellman equations satisfied by M. For instance, TD learning for M is equivalent to the following.

Represent M by its density with respect to the data distribution ρ, namely

$$M^\pi(s_0, a_0, \mathrm{d}s) = m^\pi(s_0, a_0, s)\rho(\mathrm{d}s) \tag{47}$$

where we want to learn $m^\pi(s_0, a_0, s)$. This can be done using one of the methods from [BTO21]. For instance, M^π satisfies a measure-valued Bellman equation, which gives rise to a Bellman-style loss on m^π with loss

$$\mathcal{L}_m := \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \rho,\, a_{t+1} \sim \pi_z(s_{t+1}),\, s' \sim \rho} \Big[\big(m^\pi(s_t, a_t, s') - \gamma m^\pi(s_{t+1}, a_{t+1}, s')\big)^2 - 2m^\pi(s_t, a_t, s_t) \Big]. \tag{48}$$

This is the loss $\mathcal{L}(M)$ used in Algorithm 2, where an additional z parameter captures the dependency on π_z.

In turn, the relationship (46) between d and M can be used to learn a parametric model of d from a parametric model of M. Let us parameterize d by its density with respect to ρ as we did for M, namely

$$d_{\pi_z}(\mathrm{d}s) = d(s,z)\rho(\mathrm{d}s) \tag{49}$$

we find

$$d(s,z) = (1-\gamma)\mathbb{E}_{s_0\sim\rho_0,\, a_0\sim\pi(a_0|s_0)} m(s_0,a_0,s,z) \tag{50}$$

which provides the loss for d in Algorithm 2.

Derivation of $\mathcal{L}_C$. This is essentially an application of the log-trick $\partial_\theta \mathbb{E}_{z\sim p_\theta} f(z) = \mathbb{E}_{z\sim p_\theta}[f(z)\partial_\theta \ln p_\theta(z)]$, as follows.

Lemma 8. *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a bounded function and let C be a $d\times d$ matrix. Then the derivative with respect to C of $\mathbb{E}_{z\sim N(0,C^{-1})} f(z)$ satisfies*

$$\partial_C \mathbb{E}_{z\sim N(0,C^{-1})} f(z) = \tfrac{1}{2}\partial_C \mathbb{E}_{z\sim N(0,\bar{C}^{-1})}\left[f(z)\left(-z^\top C z + \mathrm{Tr}(\bar{C}^{-1}C)\right)\right] \tag{51}$$

taken at $\bar{C} = C$ (namely, $\bar{C}$ is a stop-grad version of C).

Applying this to $C = \langle\varphi,\varphi\rangle_K$ yields

$$\partial_\varphi \mathbb{E}_{z\sim N(0,C^{-1})} f(z) = \tfrac{1}{2}\partial_\varphi \mathbb{E}_{z\sim N(0,\bar{C}^{-1})}\left[f(z)\sum_{ij}\left((\bar{C}^{-1})_{ij} - z_i z_j\right)\langle\varphi_i,\varphi_j\rangle_K\right] \tag{52}$$

as needed for $\mathcal{L}_C$.

Proof of Lemma 8.
For any smooth parametric probability distribution p_θ and any bounded function f, one has the log-trick identity

$$\partial_\theta \mathbb{E}_{z\sim p_\theta} f(z) = \mathbb{E}_{z\sim p_\theta}[f(z)\partial_\theta \ln p_\theta(z)]. \tag{53}$$

Here we have $\theta = C$ and

$$\ln p_\theta(z) = -\frac{1}{2} z^\top C z + \frac{1}{2}\ln\det C + \mathrm{cst} \tag{54}$$

and Jacobi's formula for the derivative of the determinant states that

$$\partial_C \det C = \partial_C\left((\det\bar{C})\,\mathrm{Tr}(\bar{C}^{-1}C)\right) \tag{55}$$

evaluated at $\bar{C} = C$, hence

$$\partial_C \ln\det C = \partial_C \mathrm{Tr}(\bar{C}^{-1}C) \tag{56}$$

which implies the result. □

Proof of Proposition 7.
Given a reward function r and policy π, we have

$$V_r^\pi = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\pi} r(s) \tag{57}$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{s\sim\rho}[d(s,\pi)r(s)] \tag{58}$$

where $d(s,\pi)$ denotes the density of d_π with respect to ρ.

Applying this to a Dirac reward at $s^\star$, namely, $r(s) = \mathbb{1}_{s=s^\star}/\rho(s^\star)$, yields

$$V_r^\pi = \frac{1}{1-\gamma} d(s^\star, \pi). \tag{59}$$

The computation is the same when the reward is a sum of Dirac masses, $r = c_k \sum_k w_i \delta_{s_i^\star}$ yielding

$$V_r^\pi = \frac{1}{1-\gamma} \sum_i c_k w_i d(s_i^\star, \pi). \tag{60}$$

Proposition 7 then follows from the definition of the zero-shot loss ℓ_β. □

B Penalizing Variance over the Reward r is Equivalent to Spatial Regularization for the White Noise Prior

The loss (5) maximizes the expected performance over the reward r, but does not account for variance. This is one of the reason we might get overspecialized skills that take "risks" such as making a bet on the location of the best reward and going there.

Instead, let us consider a variance-penalized version of this loss,

$$\ell(\Phi,\pi) := -\mathbb{E}_{r\sim\beta}\,\mathbb{E}_{s_0\sim\rho_0}\, V_r^{\pi_{\Phi(r)}}(s_0) + \lambda\,\mathrm{Var}_{r\sim\beta}(\mathbb{E}_{s_0\sim\rho_0}\, V_r^{\pi_{\Phi(r)}}(s_0)) \tag{61}$$

where $\lambda \geqslant 0$ is the regularization parameter.

This is tractable, as follows. We only reproduce the main part of the computation, the second moment term in the variance. With notation as in Proposition 2, we have

$$\mathbb{E}_{r\sim\beta}(\mathbb{E}_{s_0\sim\rho_0}\, V_r^{\pi_{\Phi(r)}}(s_0))^2 = \mathbb{E}_{z\sim\beta_z}\mathbb{E}_{r|\Phi(r)=z}(\mathbb{E}_{s\sim\rho_0}\, V_r^{\pi_z}(s_0))^2 \tag{62}$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{z\sim\beta_z}\mathbb{E}_{r|\Phi(r)=z}(\mathbb{E}_{s\sim d_{\pi_z}} r(s))^2 \tag{63}$$

$$= \frac{1}{1-\gamma}\mathbb{E}_{z\sim\beta_z}\mathbb{E}_{r|\Phi(r)=z}\langle d(s,z), r(s)\rangle^2_{L^2(\rho)} \tag{64}$$

where as in Section 3.3, d is defined by

$$d_{\pi_z}(\mathrm{d}s) = d(s,z)\rho(\mathrm{d}s) \tag{65}$$

namely, $d(s,z)$ is the density of the occupation measure of policy π_z wrt the data distribution ρ.

For a white noise prior on r, we have

$$\mathbb{E}_r\langle d(\cdot,z), r\rangle^2_{L^2(\rho)} = \|d(\cdot,z)\|^2_{L^2(\rho)} \tag{66}$$

by the definition of white noise in general measure spaces.

But here we should not use $\mathbb{E}_r$ but $\mathbb{E}_{r|\Phi(r)=z}$, namely, we now what the rewar features are. With a white noise prior and with linear task representation, the distribution of r knowing $\Phi(r) = z$ is the $L^2(\rho)$-orthogonal projection of the white noise onto the orthogonal of the span of the features. Denoting $\Pi^\perp_\varphi$ this projector, we have $\mathbb{E}_{r|\Phi(r)=z}\langle d(\cdot,z), r\rangle^2_{L^2(\rho)} = \mathbb{E}_r\langle d(\cdot,z), \Pi^\perp_\varphi r\rangle^2_{L^2(\rho)} = \mathbb{E}_r\langle \Pi^\perp_\varphi d(\cdot,z), \Pi r\rangle^2_{L^2(\rho)}$. So, we have proved:

Proposition 9. *With linear features, penalizing the variance over r of the expected return is equivalent to penalizing the spatial variance (in $L^2(\rho)$-norm) of the projection onto the features of the occupation measure density $d(\cdot,z)$.*

Actually it might be safer just not to use the projection onto φ: it will overestimate variance, but results in simpler algorithms. Anyway, the estimation of $z = \mathbb{E}[\varphi.r]$ is itself subject to noise because we use a finite number of samples, so even knowing the empirical estimate of z, there is still variance in the direction of the span of φ.

Algorithmically, the applicability of this depends on the method. If the occupation measure d is just computed from π_z which is computed from φ, then adding a penalty on d will just throw off the computation of d without affecting the features φ. But in the VISR-like algorithm from Section 3.3, φ is in turn computed from d (the features $z^\top\varphi$ are increased on the part of the stte visited by π_z), so penalized the variance of d is more or less equivalent to penalizing the variance of φ.

A similar penalty over the variance of the policy performance can be incorporated in the FB framework. Things are a bit simpler because B is both the features and the successor measure d: we have $d(s,z) = \mathbb{E}_{s_0\sim\rho_0, a_0\sim\pi_z(s_0)}F(s_0,a_0,z)^\top B(s)$, so we might directly penalize the spatial variance of d, with loss

$$\mathbb{E}_{s\sim\rho}((\mathbb{E}_{s_0,a_0}F(s_0,a_0,z)^\top)B(s))^2 - (\mathbb{E}_{s\sim\rho}\mathbb{E}_{s_0,a_0}F(s_0,a_0,z)^\top B(s))^2 \tag{67}$$

This may be a sensible and principled way to avoid degenerate features in FB.

References

[ACR+17] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NIPS*, 2017.

[BBQ+18] Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado van Hasselt, David Silver, and Tom Schaul. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

[BO21] Léonard Blier and Yann Ollivier. Unbiased methods for multi-goal reinforcement learning. *arXiv preprint arXiv:2106.08863*, 2021.

[BTO21] Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*, 2021.

[BVL+23] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028*, 2023.

[CTO24] Edoardo Cetin, Ahmed Touati, and Yann Ollivier. Finer behavioral foundation models via auto-regressive features and advantage weighting. *arXiv preprint arXiv:2412.04368*, 2024.

[EGIL18] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2018.

[ESL21] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. The information geometry of unsupervised reinforcement learning. *arXiv preprint arXiv:2110.02719*, 2021.

[FPAL24] Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Unsupervised zero-shot reinforcement learning via functional reward encodings. *arXiv preprint arXiv:2402.17135*, 2024.

[HDB+19] Steven Hansen, Will Dabney, Andre Barreto, Tom Van de Wiele, David Warde-Farley, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*, 2019.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 2nd edition.

[TO21] Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. In *NeurIPS*, pages 13–23, 2021.

[TRO23] Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *ICLR*. OpenReview.net, 2023.