Learning One Representation to Optimize All Rewards

Ahmed Touati
Mila, Université de Montréal
ahmed.touati@umontreal.ca

Yann Ollivier
Facebook AI Research
yol@fb.com

Abstract

We introduce the *forward-backward* (FB) representation of the dynamics of a reward-free Markov decision process. It provides explicit near-optimal policies for any reward specified a posteriori. During an unsupervised phase, we use reward-free interactions with the environment to learn two representations via off-the-shelf deep learning methods and temporal difference (TD) learning. In the test phase, a reward representation is estimated either from observations or an explicit reward description (e.g., a target state). The optimal policy for that reward is directly obtained from these representations, with no planning.

The unsupervised FB loss is well-principled: if training is perfect, the policies obtained are provably optimal for any reward function. With imperfect training, the sub-optimality is proportional to the unsupervised approximation error. The FB representation learns long-range relationships between states and actions, via a predictive occupancy map, without having to synthesize states as in model-based approaches.

This is a step towards learning controllable agents in arbitrary black-box stochastic environments. This approach compares well to goal-oriented RL algorithms on discrete and continuous mazes, pixel-based Ms. Pacman, and the FetchReach virtual robot arm. We also illustrate how the agent can immediately adapt to new tasks beyond goal-oriented RL.

1 Introduction and Related Work

We consider one kind of unsupervised reinforcement learning problem: Given a Markov decision process (MDP) but no reward information, is it possible to learn and store a compact object that, for any reward function specified later, provides the optimal policy for that reward, with a minimal amount of additional computation? In a sense, such an object would encode in a compact form the solutions of all possible planning problems in the environment. This is a step towards building agents that are fully controllable after first exploring their environment in an unsupervised way.

Goal-oriented RL methods (Andrychowicz et al., 2017; Plappert et al., 2018) compute policies for a series of rewards specified in advance (such as reaching a set of target states), but cannot adapt in real time to new rewards, such as weighted combinations of target states or dense rewards.

Learning a model of the world is another possibility, but it still requires explicit planning for each new reward; moreover, synthesizing accurate trajectories of states over long time ranges has proven difficult (Talvitie, 2017; Ke et al., 2018).

Instead, we exhibit an object that is both simpler to learn than a model of the world, and contains the information to recover near-optimal policies for any reward provided a posteriori, without a planning phase.

Borsa et al. (2018) learn optimal policies for all rewards that are linear combinations of a finite number of feature functions provided in advance by the user. This limits applications: e.g., goal-oriented tasks would require one feature per goal state, thus using infinitely many features in continuous spaces. We reuse a policy parameterization from Borsa et al. (2018), but introduce a novel representation with better properties, based on state occupancy prediction instead of expected featurizations. We use theoretical advances on successor state learning from Blier et al. (2021). We obtain the following.

- We prove the existence of a learnable "summary" of a reward-free discrete or continuous MDP, that provides an explicit formula for optimal policies for any reward specified later. This takes the form of a pair of representations $F\colon S \times A \times Z \to Z$ and $B\colon S \times A \to Z$ from state-actions into a representation space $Z \simeq \mathbb{R}^d$, with policies $\pi_z(s) := \arg\max_a F(s,a,z)^\top z$. Once a reward is specified, a value of z is computed from reward values and B; then π_z is used. Rewards may be specified either explicitly as a function, or as target states, or by samples as in usual RL setups.
- We provide a well-principled unsupervised loss for F and B: if training is perfect, then the policies are provably optimal for all rewards (Theorem 1). In finite spaces, perfect training is possible with large enough dimension d (Proposition 3). With imperfect training, sub-optimality is proportional to the training error (Theorem 5).
- We provide a TD-like algorithm to train F and B to minimize this loss, with function approximation, adapted from recent methods for successor states Blier et al. (2021). As usual with TD, learning seeks a fixed point but the loss itself is not observable.
- We prove viability of the method on several environments from mazes to pixel-based MsPacman and a virtual robotic arm. For single-state rewards (learning to reach arbitrary states), we compare this method to goal-oriented methods such as HER. (Our method is not a substitute for HER: in principle they could be combined, with HER improving replay buffer management for our method.) For more general rewards, which cannot be tackled a posteriori by trained goal-oriented models, we provide qualitative examples.
- We also illustrate qualitatively the sub-optimalities (long-range behavior is preserved but local blurring of rewards occurs) and the representations learned.

2 Problem and Notation

We consider a reward-free Markov decision process (MDP) $\mathcal{M} = (S, A, P, \gamma)$ with state space S (discrete or continuous), action space A (discrete for simplicity, but this is not essential), transition probabilities $P(s'|s,a)$ from state s to s' given action a, and discount factor $0 < \gamma < 1$ Sutton & Barto (2018). If S is finite, $P(s'|s,a)$ can be viewed as a matrix; in general, for each $(s,a) \in S \times A$, $P(\mathrm{d}s'|s,a)$ is a probability measure on $s' \in S$. The notation $P(\mathrm{d}s'|s,a)$ covers all cases. All functions are assumed to be measurable.

Given $(s_0, a_0) \in S \times A$ and a policy $\pi\colon S \to \mathrm{Prob}(A)$, 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 $(s_t, a_t)_{t\geq 0}$ starting with (s_0, a_0) and following policy π in the environment, defined by sampling $s_t \sim P(\mathrm{d}s_t|s_{t-1}, a_{t-1})$ and $a_t \sim \pi(s_t)$.

Given a reward function $r\colon S \times A \to \mathbb{R}$, the Q-function of π for r is $Q^\pi_r(s_0, a_0) := \sum_{t\geq 0} \gamma^t\, \mathbb{E}[r(s_t, a_t)|s_0, a_0, \pi]$. We assume that rewards are bounded, so that all Q-functions are well-defined. We state the results for deterministic reward functions, but this is not essential. We abuse notation and write greedy policies as $\pi(s) = \arg\max_a Q(s,a)$ instead of $\pi(s) \in \arg\max_a Q(s,a)$. Ties may be broken any way.

We consider the following informal problem: Given a reward-free MDP (S, A, P, γ), can we compute a convenient object E such that, once a reward function $r\colon S \times A \to \mathbb{R}$ is specified, we can easily (with no planning) compute from E and r a policy π such that Q^π_r is close to maximal?

3 Outline of the Method

The method has three phases: an *unsupervised learning phase* where we learn a pair of representations F and B in a reward-free way, by observing state transitions in the environment; a *reward estimation phase* where we estimate a policy parameter z_R from reward observations, or directly set z_R if the reward is known (as in goal-oriented RL); and an *exploitation phase*, where we directly set a policy based on F, B, and z_R. The last two phases may be merged in some cases.

The unsupervised learning phase. We set a representation space $Z = \mathbb{R}^d$, used to represent both states and reward functions. We learn a pair of "forward" and "backward" representations

$$F\colon S \times A \times Z \to Z, \qquad B\colon S \times A \to Z \tag{1}$$

For each $z \in Z$, we define the policy $\pi_z(s) := \arg\max_a F(s, a, z)^\top z$.

No rewards are used in this phase, and no family of tasks has to be specified manually. F and B are trained such that, for any state-actions (s_0, a_0), (s', a'), and any $z \in Z$,

$$F(s_0, a_0, z)^\top B(s', a') \approx \sum_{t \geq 0} \gamma^t \Pr\left(s_t = s',\, a_t = a' \mid s_0,\, a_0,\, \pi_z\right) \tag{2}$$

(We actually learn probability densities, which make more sense in continuous spaces, see (7)–(8).) Since π_z depends on F, (2) is a fixed point equation, similarly to the dependency between the optimal Q-function and optimal policy in ordinary Q-learning.

Intuitively, F is a representation of the future of a state under a certain policy. B represents the past of a state, or the ways to reach that state (see also Appendix E.5). If $F^\top B$ is large, then it is possible to reach the second state from the first. This is akin to a model of the environment, without synthesizing state trajectories.

This unsupervised learning phase is fully principled: If learning of F and B is successful, then this is guaranteed to provide all optimal policies (Theorem 1). Namely, for any reward function $r(s, a)$, the optimal policy for r is the policy π_z with $z := \mathbb{E}_{(s,a)}[r(s, a)B(s, a)]$. For instance, $\pi_{B(s,a)}$ is the optimal policy to reach (s, a).

Approximate solutions still provide approximately optimal policies, with optimality gap directly proportional to the error on $F^\top B$ (Appendix, Theorem 5), so the guarantees are non-empty.

F and B can be learned off-policy from observed transitions in the environment, via the Bellman equation for (2),

$$F(s_0, a_0, z)^\top B(s', a') = \mathbb{1}_{s_0=s',\, a_0=a'} + \gamma\, \mathbb{E}_{s_1 \sim P(\mathrm{d}s_1|s_0,a_0)}[F(s_1, \pi_z(s_1), z)^\top B(s', a')] \tag{3}$$

in discrete spaces. We leverage efficient algorithms for *successor states* Blier et al. (2021), also valid in continuous spaces, and which learn without using the sparse reward $\mathbb{1}_{s_0=s',\, a_0=a'}$ (see Section 4).

In finite spaces, exact solutions F and B exist (Appendix, Prop. 3), provided the dimension d is large enough. In infinite spaces, arbitrarily good approximations can be obtained by increasing d, corresponding to a rank-d approximation of the pairwise cumulated transition probabilities in (2). The dimension d controls how many types of rewards can be optimized well; even a relatively small d can provide useful behaviors, see the experiments and Appendix E.2. The algorithm is linear in d, so d can be taken as large as the neural network models can handle.

For exploration in this phase, we use the policies being learned: the exploration policy chooses a random value of z from some distribution (e.g., Gaussian), and follows π_z for some time (Appendix, Algorithm 1). However, the algorithm can also work from an existing dataset of off-policy transitions.

The optimality guarantee and training algorithm are detailed in Section 4.

The reward estimation phase. Once rewards are available, we estimate a reward representation (policy parameter) z_R by weighing the representation B by the reward:

$$z_R := \mathbb{E}[r(s, a)B(s, a)] \tag{4}$$

where the expectation must be computed over the same distribution of state-actions (s, a) used to learn F and B. There are several scenarios.

If the reward is known explicitly, this phase is unnecessary. For instance, if the reward is to reach a target state-action (s_0, a_0) while avoiding some forbidden state-actions $(s_1, a_1), \ldots, (s_k, a_k)$, one may directly set

$$z_R = B(s_0, a_0) - \lambda \sum B(s_i, a_i) \tag{5}$$

where the constant λ adjusts the negative reward for visiting a forbidden state. This can be used for goal-oriented RL.

If the reward is known algebraically as a function $r(s, a)$, then z_R may be computed by averaging the function $r(s, a)B(s, a)$ over a replay buffer from the unsupervised training phase.

If the reward is black-box as in standard RL algorithms, then the exploration policy has to be run again for some time, and z_R is obtained by averaging $r(s, a)B(s, a)$ over the states visited.

If it is not possible to run the exploration policy again, one may learn a model $\hat{r}(s, a)$ of $r(s, a)$ on some reward observations from any source, then estimate $z_R = \mathbb{E}[\hat{r}(s, a)B(s, a)]$ by averaging the model over a replay buffer from the unsupervised training phase. In this case, reward identification may be done in parallel to exploitation.

An approximate value for z_R still provides an approximately optimal policy (Appendix, Prop. 6 and Thm. 8).

Since reward functions are represented by d-dimensional vectors z_R, loss of precision necessarily occurs. In practice, for small d we notice some blurring of rewards between nearby states (Fig. 3), for reasons discussed in Section 4.

The exploitation phase. Once the reward representation z_R has been estimated, the Q-function is estimated as

$$Q(s, a) = F(s, a, z_R)^\top z_R. \tag{6}$$

The corresponding policy $\pi_{z_R}(s) = \arg\max_a Q(s, a)$ is used for exploitation.

Fine-tuning was not needed in our experiments, but it is possible to fine-tune the Q-function using actual rewards, by setting $Q(s, a) = F(s, a, z_R)^\top z_R + q_\theta(s, a)$ where the fine-tuning model q_θ is initialized to 0 and learned via any standard Q-learning method.

4 The Forward-Backward Representation of an MDP Encodes the Optimal Policies

We now present in more detail the algorithm used to learn F and B, and the associated theoretical guarantee.

Background on successor states. We start with some definitions and an algorithm from Blier et al. (2021) for successor states of a single policy. The successor state operator M^π of a policy π is defined as follows. For each state-action (s_0, a_0), $M^\pi(s_0, a_0, \cdot)$ is a measure over the state-action space $S \times A$, representing the expected discounted time spent in each set $X \subset S \times A$, namely,

$$M^\pi(s_0, a_0, X) := \sum_{t \geq 0} \gamma^t \Pr\left((s_t, a_t) \in X \mid s_0,\, a_0,\, \pi\right) \tag{7}$$

for each $X \subset S \times A$. Viewing M as a measure deals with both discrete and continuous spaces.

In practice, M^π is represented by a function $m^\pi(s_0, a_0, s', a')$ taking a pair of state-actions and returning a number. Namely, assume access to a training set of transitions in the MDP (off-policy, reward-free). Let $\rho(\mathrm{d}s, \mathrm{d}a)$ be the probability distribution of state-actions in this training set; we assume $\rho > 0$ everywhere. The measure $M^\pi(s_0, a_0, \cdot)$ may be represented as a density with respect to ρ:

$$M^\pi(s_0, a_0, \mathrm{d}s', \mathrm{d}a') =: m^\pi(s_0, a_0, s', a')\, \rho(\mathrm{d}s', \mathrm{d}a') \tag{8}$$

where m^π is an ordinary function (or a distribution, see Appendix E.4).

Algorithms exist to train a parametric model $m^\pi_\theta(s_0, a_0, s', a')$ of m^π from a training set of off-policy observed transitions (s_0, a, s_1) in the MDP, based on a Bellman-like equation for M Blier et al. (2021). Namely, sample a transition (s_0, a, s_1) from the training set, generate an action $a_1 \sim \pi(a_1|s_1)$, sample a state-action (s', a') from the training set, independently from (s_0, a, s_1). Then update the parameter θ by $\theta \leftarrow \theta + \eta\, \delta\theta$ with learning rate η and

$$\delta\theta := \partial_\theta m^\pi_\theta(s_0, a, s_0, a) + \partial_\theta m^\pi_\theta(s_0, a, s', a') \times (\gamma\, m^\pi_\theta(s_1, a_1, s', a') - m^\pi_\theta(s_0, a, s', a')) \quad (9)$$

The true successor state density m^π is a fixed point of this update in expectation Blier et al. (2021) (it is the only fixed point in the tabular or overparameterized case). Variants exist, such as using a target network for $m^\pi_\theta(s_1, a_1, s', a')$ on the right-hand side, as in DQN.

The naive way to estimate successor states via a Bellman equation from (2) would involve a sparse reward term $\mathbb{1}_{s_0=s',\, a=a'}$, resulting in infinitely sparse rewards in continuous spaces. However, in expectation, this term's contribution is known algebraically: (9) avoids sparse rewards in a principled way thanks to the $\partial_\theta m^\pi_\theta(s_0, a, s_0, a)$ term Blier et al. (2021).

It is not necessary to know the training distribution ρ: all quantities are expressed as expectations under ρ, namely, expectations over the training set. When training under data distribution ρ, one learns the density m^π of M^π with respect to ρ; this will be useful below.

The forward-backward representation provides the optimal policies. The next theorem contains the core idea of our algorithms. In short, if we can learn two representations F and B of state-actions such that $F^\top B$ approximates the successor states of certain policies, then we can compute all optimal policies from F and B.

Theorem 1 (Forward-backward representation of an MDP)**.** *Consider an MDP with state space S and action space A. Let $Z = \mathbb{R}^d$ be some representation space. Let*

$$F\colon S \times A \times Z \to Z, \qquad B\colon S \times A \to Z \quad (10)$$

be two functions. For each $z \in Z$, define the policy

$$\pi_z(s) := \arg\max_a F(s, a, z)^\top z. \quad (11)$$

Let ρ be any probability distribution on $S \times A$ (e.g., the distribution of state-actions under some exploration scheme).

Assume that F and B have been chosen (trained) to satisfy the following: for any $z \in Z$, and any state-actions (s, a) and (s', a'), the quantity $F(s, a, z)^\top B(s', a')$ is equal to the successor state density $m^{\pi_z}(s, a, s', a')$ of policy π_z with respect to ρ.

Then, for any bounded reward function $r\colon S \times A \to \mathbb{R}$, the following holds. Set

$$z_R := \mathbb{E}_{(s,a)\sim\rho}\left[r(s, a)B(s, a)\right]. \quad (12)$$

Then π_{z_R} is an optimal policy for reward r in the MDP. Moreover, the optimal Q-function Q^* for reward r is

$$Q^*(s, a) = F(s, a, z_R)^\top z_R. \quad (13)$$

The theoretical guarantee extends to approximate training of F and B, with optimality gap proportional to the $F^\top B$ training error. This is important, as exact finite-d representations are often impossible in continuous spaces. Namely: if, for some reward r, the error $\left|F(s, a, z_R)^\top B(s', a') - m^{\pi_{z_R}}(s, a, s', a')\right|$ is at most ε on average over $(s', a') \sim \rho$ for every (s, a), then π_{z_R} is $3\varepsilon \left\|r\right\|_\infty / (1 - \gamma)$-optimal for r (Appendix, Theorem 5, with additional results on weaker norms).

A similar statement holds if we use $\bar{m}(s, z, s', a') + F(s, a, z)^\top B(s', a')$ instead of just $F(s, a, z)^\top B(s', a')$ to learn successor states (Appendix, Theorem 2). Here $\bar{m}$ is any function that does not depend on a. Since $\bar{m}$ has no rank restriction, the finite rank approximation only applies to the advantage function.

Learning F and B. Theorem 1 suggests to choose a parametric model for the representations F and B, and then train F and B such that $F^\top B$ approximates the successor state density m, via any successor state learning algorithm.

This unfolds as follows (Appendix, Algorithm 1). Assume access to a training dataset of transitions (s_0, a, s_1) in the MDP, e.g., from some exploration policies. The samples (s_0, a) in this dataset follow some unknown distribution ρ; we make no assumptions on ρ. Then:

- In the unsupervised learning phase, train F and B so that $F^\top B$ approximates the successor state density m^{π_z} for every z. This can use any successor state learning algorithm, adapted to incorporate z. Here we use (9).
  For this, choose a parametric model F_θ, B_θ for the representations F and B. At each step, pick a z at random, pick a batch of transitions (s_0, a, s_1) from the training set, set the next actions $a_1 := \pi_z(s_1)$, and pick a batch of state-actions (s', a') from the training set, independently from z and (s_0, a, s_1). Then apply the successor state update (9) to the parametric model $m_\theta^{\pi_z}(s_0, a, s', a') = F_\theta(s_0, a, z)^\top B_\theta(s', a')$ (Algorithm 1).
  For sampling z, we use a fixed distribution (rescaled Gaussians, see Appendix G). Any number of values of z may be sampled: this does not use up training samples. We use a target network with soft updates (Polyak averaging) as in DDPG. For training we also replace the greedy policy $\pi_z = \arg\max_a F(s, a, z)^\top z$ with a regularized version $\pi_z = \text{softmax}(F(s, a, z)^\top z/\tau)$ with fixed temperature τ (Appendix G). Since there is unidentifiability between F and B (Appendix, Remark 4), we normalize B via an auxiliary loss in Algorithm 1.
- In the reward estimation phase, we estimate the reward representation z_R via (12), following one of the use cases from Section 3. The distribution ρ in (12) is the same as the one used to define the density m^π; thus, z_R must be estimated over the same distribution of state-actions used to train F and B.
  If the reward is known directly, z_R can be set by hand following (12). For instance, for the optimal policy to reach state-action (s, a), we directly set $z_R = B(s, a)$. (A unit reward at (s, a) corresponds to $z_R = B(s, a)/\rho(s, a)$ in (12), but ρ is unknown and scaling the reward yields the same optimal policy.) This extends to finite combinations of states. If r is provided as a function, then $\mathbb{E}_{(s,a)\sim\rho}[r(s, a)B(s, a)]$ may be estimated directly on (s, a) in a replay buffer. The case of r accessible only by samples has been discussed in Section 3.
  Stochastic rewards may be used to estimate z_R. Indeed, if $\mathbb{E}\,\tilde{r}(s, a) = r(s, a)$ then $\mathbb{E}[\tilde{r}(s, a)B(s, a)] = r(s, a)B(s, a)$.
- In the exploitation phase, the exploitation policy can be directly set to π_{z_R}, which selects the action $\arg\max_a F(s, a, z_R)^\top z_R$.

Incorporating prior information. We are often interested in rewards depending, not on the full state, but only on a part or some features of the state (e.g., a few components of the state, such as the position of an agent, or its neighbordhood, rather than the full environment). If this is known in advance, the representation B can be trained on that part of the state only, with the same theoretical guarantees (Appendix, Theorem 2). F still needs to use the full state as input. This way, the FB model (2) does not have to learn how often every (s', a') is reached, only the part of interest in (s', a'). More generally, if $\varphi \colon S \times A \to G$ is a feature map to some features $g = \varphi(s, a)$, and if we know that the reward will be a function $R(g)$, then the same theorem holds with $B(g)$ everywhere instead of $B(s, a)$, and with the successor density $m^\pi(s, a, g)$ instead of $m^\pi(s, a, s', a')$ (Appendix, Theorem 2). Learning the latter is done by replacing $\partial_\theta m_\theta^\pi(s_0, a, s_0, a)$ with $\partial_\theta m_\theta^\pi(s_0, a, \varphi(s_0, a))$ in the first term in (9) Blier et al. (2021). Rewards can be arbitrary functions of g, so this is more general than Borsa et al. (2018) which only considers rewards linear in g. For instance, in MsPacman below, we let g be the position of the agent, so we can optimize any reward function that depends on this position.

5 Experiments

We first consider the task of reaching arbitrary goal states. For this, we can make quantitative comparisons to existing goal-oriented baselines. Next, we illustrate qualitatively some tasks that cannot be tackled a posteriori by goal-oriented methods, such as introducing forbidden states. Finally, we illustrate some of the representations learned.

5.1 Environments and Experimental Setup

We run our experiments on a selection of environments that are diverse in term of state space dimensionality, stochasticity and dynamics.

- Discrete Maze is the classical gridworld with four rooms. States are represented by one-hot unit vectors.
- Continuous Maze is a two dimensional environment with impassable walls. States are represented by their Cartesian coordinates $(x, y) \in [0, 1]^2$. The execution of one of the actions moves the agent in the desired direction, but with normal random noise added to the position of the agent.
- FetchReach is a variant of the simulated robotic arm environment from (Plappert et al., 2018) using discrete actions instead of continuous actions. States are 10-dimensional vectors consisting of positions and velocities of robot joints.
- Ms. Pacman is a variant of the Atari 2600 game Ms. Pacman, where an episode ends when the agent is captured by a monster (Rauber et al., 2018). States are obtained by processing the raw visual input directly from the screen. Frames are preprocessed by cropping, conversion to grayscale and downsampling to 84×84 pixels. A state s_t is the concatenation of $(x_{t-12}, x_{t-8}, x_{t-4}, x_t)$ frames, i.e. an $84 \times 84 \times 4$ tensor. An action repeat of 12 is used. As Ms. Pacman is not originally a multi-goal domain, we define the goals as the 148 reachable coordinates (x, y) on the screen; these can be reached only by learning to avoid monsters.

For all environments, we run algorithms for 800 epochs. Each epoch consists of 25 cycles where we interleave between gathering some amount of transitions, to add to the replay buffer, and performing 40 steps of stochastic gradient descent on the model parameters. To collect transitions, we generate episodes using some behavior policy. For both mazes, we use a uniform policy while for FetchReach and Ms. Pacman, we use an ε-greedy policy with respect to the current approximation $F(s, a, z)^\top z$ for a sampled z. At evaluation time, ε-greedy policies are also used, with a smaller ε. More details are given in Appendix G.

5.2 Goal-Oriented Setting: Quantitative Comparisons

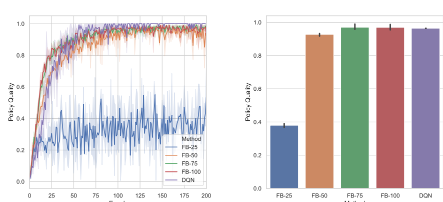

Figure 1: Comparative performance of FB for different dimensions and DQN in the discrete maze. **Left**: the policy quality averaged over 20 randomly selected goals as function of the the first 200 training epochs. **Right**: the policy quality averaged over the goal space after 800 training epochs.

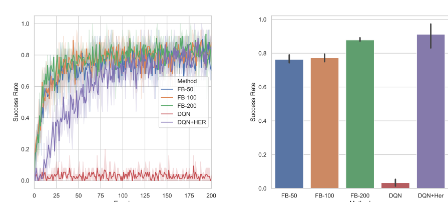

Figure 2: Comparative performance of FB for different dimensions and DQN in Ms. Pacman. **Left**: the success rate averaged over 20 randomly selected goals as function of the first 200 training epochs. **Right**: the success rate averaged over the goal space after 800 training epochs.

We investigate the FB representation over goal-reaching tasks and compare it to goal-oriented baselines: DQN[1], and DQN with HER when needed. We define sparse reward functions. For Discrete Maze, the reward function is equal to one when the agent's state is equal exactly to the goal state. For Discrete Maze, we measured the quality of the obtained policy to be the ratio between the true expected discounted reward of the policy for its goal and the true optimal value function, on average over all states. For the other environments, the reward function is equal to one when the distance of the agent's position and the goal position is

[1] Here DQN is short for goal-oriented DQN, $Q(s, a, g)$.

below some threshold, and zero otherwise. We assess policies by computing the average success rate, i.e the average number of times the agent successfully reaches its goal.

Figs. 1 and 2 show the comparative performance of FB for different dimensions d, and DQN and DQN+HER respectively in Discrete Maze and Ms. Pacman (similar results in FetchReach and Continuous Maze are provided in the Appendix). The performance of FB consistently increases with the dimension d and the best dimension matches the performance of the goal-oriented baseline. In Ms. Pacman, DQN totally fails to learn and we had to add HER to make it work.

In Discrete Maze, we observe a drop of performance for $d = 25$: this is due to the spatial smoothing induced by the small rank approximation and the reward being nonzero only if the agent is exactly at the goal. This spatial blurring is clear on heatmaps for $d = 25$ vs $d = 75$ (Fig. 3). With $d = 25$ the agent often stops right next to its goal.

To evaluate the sample efficiency of FB, after each epoch, we evaluate the agent on 20 randomly selected goals. Learning curves are reported in Figs. 1 and 2 (left). In all environments, we observe no loss in sample efficiency compared to the goal-oriented baseline. In Ms. Pacman, FB even learns faster than DQN+HER.

5.3 More Complex Rewards: Qualitative Results

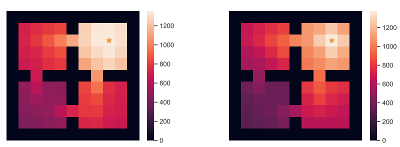

Figure 3: Heatmap of $\max_a F(s,a,z_R)^\top z_R$ for $z_R = B(☆)$ **Left**: $d = 25$. **Right**: $d = 75$.

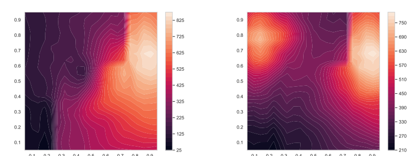

Figure 4: Contour plot of $\max_{a\in A} F(s,a,z_R)^\top z_R$ in Continuous Maze. **Left**: for the task of reaching a target while avoiding a forbidden region, **Right**: for two equally rewarding targets.

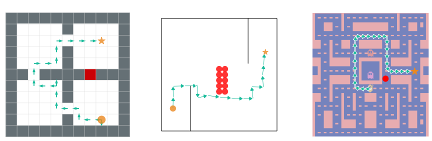

Figure 5: Trajectories generated by the $F^\top B$ policies for the task of reaching a target position (star shape ☆ while avoiding forbidden positions (red shape ●)

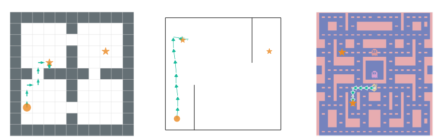

Figure 6: Trajectories generated by the $F^\top B$ policies for the task of reaching the closest among two equally rewarding positions (star shapes ☆). (Optimal Q-values are not linear over such mixtures.)

We now investigate FB's ability to generalize to new tasks that cannot be solved by an already trained goal-oriented model: reaching a goal with forbidden states imposed a posteriori, reaching the nearest of two goals, and choosing between a small, close reward and a large, distant one.

First, for the task of reaching a target position g_0 ☆ while avoiding some forbidden positions $g_1, \ldots g_k$ ●, we set $z_R = B(g_0) - \lambda \sum_{i=1}^{k} B(g_i)$ and run the corresponding ε-greedy policy defined by $F(s,a,z_R)^\top z_R$. Fig. 5 shows the resulting trajectories, which succeed at solving the task for the different domains. In Ms. Pacman, the path is suboptimal (though successful) due to the sudden appearance of a monster along the optimal path. (We only plot the initial frame; the full series of frames along the trajectory is in the Appendix.) Fig. 4 (left) provides a contour plot of $\max_{a\in A} F(s,a,z_R)^\top z_R$ for the continuous maze and shows the landscape shape around the forbidden regions.

Next, we consider the task of reaching the closest target among two equally rewarding positions g_0 and g_1, by setting $z_R = B(g_0) + B(g_1)$. The optimal Q-function is *not* a linear combination of the Q-functions for g_0 and g_1. Fig. 6 shows successful trajectories generated by the policy π_{z_R}. On the contour plot of $\max_{a \in A} F(s, a, z_R)^\top z_R$ in Fig. 4 (right), the two rewarding positions appear as basins of attraction. Similar results for a third task are shown in the Appendix: introducing a "distracting" small reward next to the initial position of the agent, with a larger reward further away. The Appendix includes embedding visualizations for different z and for Discrete Maze and Ms. Pacman.

6 Conclusion

The FB representation is a learnable mathematical object that "summarizes" a reward-free MDP. It provides near-optimal policies for any reward specified a posteriori, without planning. It is learned from black-box reward-free interactions with the environment. In practice, this unsupervised method performs comparably to goal-oriented methods for reaching arbitrary goals, but is also able to tackle more complex rewards in real time. The representations learned encode the MDP dynamics and may have broader interest.

References

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in atari. *arXiv preprint arXiv:1906.08226*, 2019.

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.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, David Silver, and Hado P van Hasselt. Successor features for transfer in reinforcement learning. In *NIPS*, 2017.

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

Vladimir I Bogachev. *Measure theory*. Springer, 2007.

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.

Pierre Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. 1999.

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

David Foster and Peter Dayan. Structure in the space of value functions. *Machine Learning*, 49(2):325–346, 2002.

Christopher Grimm, Irina Higgins, Andre Barreto, Denis Teplyashin, Markus Wulfmeier, Tim Hertweck, Raia Hadsell, and Satinder Singh. Disentangled cumulants help successor representations transfer to new tasks. *arXiv preprint arXiv:1911.10866*, 2019.

Charles Miller Grinstead and James Laurie Snell. *Introduction to probability*. American Mathematical Soc., 1997.

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.

Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-free exploration for reinforcement learning. In *International Conference on Machine Learning*, pp. 4870–4879. PMLR, 2020.

Nan Rosemary Ke, Amanpreet Singh, Ahmed Touati, Anirudh Goyal, Yoshua Bengio, Devi Parikh, and Dhruv Batra. Modeling the long term future in model-based reinforcement learning. In *International Conference on Learning Representations*, 2018.

J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Van Nostrand, New York, 1960.

Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(10), 2007.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Paulo Rauber, Avinash Ummadisingu, Filipe Mutz, and Jürgen Schmidhuber. Hindsight policy gradients. In *International Conference on Learning Representations*, 2018.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL `http://proceedings.mlr.press/v37/schaul15.html`.

Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. The hippocampus as a predictive map. *Nature neuroscience*, 20(11):1643, 2017.

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

Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768, 2011.

Erik Talvitie. Self-correcting models for model-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2371–2378. IEEE, 2017.

A Outline of the Supplementary Material

The Appendix is organized as follows.

- Appendix B provides an extended related work.
- Appendix C discusses how incoporate prior information as well as what kinod of representation is learned.
- Appendix D presents the pseudo-code of the unsupervised phase of FB algorithm.
- Appendix E provides extended theoretical results on approximate solutions and general goals:
  - Section E.1 formalizes the forward-backward representation with a goal or feature space.
  - Section E.2 establishes the existence of exact FB representations in finite spaces.
  - Section E.3 shows how approximate solutions provide approximately optimal policies.
  - Section E.4 presents a note of the measure M^π and its density m^π.
  - Section E.5 shows how F and B are successor and predecessor features of each other.
- Appendix F provides proofs of all theoretical results above.
- Appendix G provides additional information about our experiments:
  - Section G.1 describes the environments.
  - Section G.2 describes the different architectures used for FB as well as the goal-oriented DQN.
  - Section G.3 provides implementation details and hyperparameters.
  - Section G.4 provides additional experimental results.

B Extended Related Work

Borsa et al. (2018) learn optimal policies for rewards that are linear combinations of a finite number of feature functions provided in advance by the user. The approach of Borsa et al. (2018) cannot tackle generic rewards or goal-oriented RL: this would require introducing one feature per possible goal state, requiring infinitely many features in continuous spaces.

Our approach does not require user-provided features describing the future tasks, thanks to using successor *states* Blier et al. (2021) where Borsa et al. (2018) use successor *features*. Schematically, and omitting actions, successor features start with user-provided features φ, then learn ψ such that $\psi(s_0) = \sum_{t\geq 0} \gamma^t \, \mathbb{E}[\varphi(s_t) \mid s_0]$. This limits applicability to rewards that are linear combinations of φ. Here we use successor *states*, namely, we learn two representations F and B such that $F(s_0)^\top B(s') = \sum_{t\geq 0} \gamma^t \Pr(s_t = s' \mid s_0)$. This does not require any user-provided input.

Thus we learn two representations instead of one. The learned backward representation B is absent from Borsa et al. (2018). B plays a different role than the user-provided features φ of Borsa et al. (2018): learning of F given B is not learning of ψ given φ. The features φ of Borsa et al. (2018) are split between our learned B and the functions φ we can use if the reward is known to depend on only part of the state.

We use a similar parameterization of policies by $F(s, a, z)^\top z$ as in Borsa et al. (2018), for similar reasons, although z encodes a different object.

Successor representations where first defined in Dayan (1993) for finite spaces, corresponding to an older object from Markov chains, the fundamental matrix Kemeny & Snell (1960); Brémaud (1999); Grinstead & Snell (1997). Stachenfeld et al. (2017) argue for their relevance for cognitive science. For successor representations in continuous spaces, a finite number of features φ are specified first; this can be used for generalization within a family of tasks, e.g., (Barreto et al., 2017; Zhang et al., 2017; Grimm et al., 2019; Hansen et al., 2019). Blier et al.

(2021) moves from successor features to successor states by providing pointwise occupancy map estimates even in continuous spaces, without using the sparse reward $\mathbb{1}_{s_t=s'}$. We adapt successor state learning algorithms from Blier et al. (2021), which also introduced simpler versions of F and B for a single, fixed policy.

There is a long literature on goal-oriented RL. For instance, Schaul et al. (2015) learn goal-dependent value functions, regularized via an explicit matrix factorization. Goal-dependent value functions have been investigated in earlier works such as (Foster & Dayan, 2002) and (Sutton et al., 2011). Hindsight experience replay (HER) (Andrychowicz et al., 2017) improve the sample efficiency of multiple goal learning with sparse rewards. A family of rewards has to be specified beforehand, such as reaching arbitrary target states. Specifying rewards a posteriori is not possible: for instance, learning to reach target states does not extend to reaching the nearest among several goals, reaching a goal while avoiding forbidden states, or maximizing any dense reward.

For finite state spaces, Jin et al. (2020) provide an algorithm to constitute a training set by reward-free interactions with a finite environment, such that any optimal policies later computed on this training set instead of the true environment are provably ε-optimal, for any reward. They prove tight bounds on the necessary set size. Planning still has to be done for each reward.

C Discussion

Incorporating prior information. Trying to plan in advance for all possible rewards in an arbitrary environment may be too generic and problem-agnostic, and become difficult in large environments, requiring long exploration and a large d to accomodate all rewards. Prior information can be incorporated in several ways into the FB method, by focussing on certain types of rewards or parts of the space.

First, F and B may be trained only on values of z_R corresponding to a category of rewards specified in advance. For instance, training only on $z_R = B(s)$ with s in some set $S' \subset S$ learns the optimal policies for reaching all states $s \in S'$. Any prior (e.g., Bayesian) on rewards may be incorporated this way. In our experiments, z was just sampled from a rescaled Gaussian.

Second, we are often interested in rewards depending, not on the full state, but only on a part or some features of the state (e.g., a few components of the state, such as the position of an agent, or its neighbordhood, rather than the full environment). If this is known in advance, the representation B can be trained on that part of the state only, with the same theoretical guarantees (Appendix, Theorem 2). F still needs to use the full state as input. This way, the FB model (2) does not have to learn how often every (s', a') is reached, only the part of interest in (s', a'). More generally, if $\varphi\colon S \times A \to G$ is a feature map to some features $g = \varphi(s, a)$, and if we know that the reward will be a function $R(g)$, then the same theorem holds with $B(g)$ everywhere instead of $B(s, a)$, and with the successor density $m^\pi(s, a, g)$ instead of $m^\pi(s, a, s', a')$ (Appendix, Theorem 2). Learning the latter is done by replacing $\partial_\theta m^\pi_\theta(s_0, a, s_0, a)$ with $\partial_\theta m^\pi_\theta(s_0, a, \varphi(s_0, a))$ in the first term in (9) Blier et al. (2021). Rewards can be arbitrary functions of g, so this is more general than Borsa et al. (2018) which only considers rewards linear in g. For instance, in MsPacman below, we let g be the position of the agent, so we can optimize any reward function that depends on this position.

Third, if we are interested only in part of the space, focusing the exploration policy on that part will provide representations F and B that minimize the loss over that part of the space. This could lead to optimal policies for rewards "around a typical set of behaviors", presumably a more realistic goal than fully agnostic unsupervised training in complex environments.

What kind of representation is learned? Which information on the reward gets lost? As reward functions are represented by a finite-dimensional object z_R, some approximation is incurred on rewards (unless fine-tuning is used at exploitation time). By (12), a reward is represented by 0 if and only if it is $L^2(\rho)$-orthogonal to all components of

B. So the representation B is not only part of a transition probability model, but also a reward regularizer.

The loss associated to (2), and minimized by a Bellman-like training, is

$$\mathbb{E}_{(s,a)\sim\rho,\,(s',a')\sim\rho,\,z\sim P_z}\left|F(s,a,z)^\top B(s',a')\ -\textstyle\sum_{t\geq 0}\gamma^t P_t(\mathrm{d}s',\mathrm{d}a'\mid s,a,\pi_z)/\rho(\mathrm{d}s',\mathrm{d}a')\right|^2 \tag{14}$$

where P_t is the distribution of states reached at step t by policy π_z starting at (s,a), and P_z is some user-chosen distribution, such as a Gaussian or a prior over tasks (see above).

Thus, F and B provide a rank-d approximation to the transition probabilities in the environment. For a single z, this loss is minimized when F and B are the SVD of the cumulated transition matrix $\sum_t \gamma^t P_t$, truncated to the largest d singular values. The SVD is computed in the $L^2(\rho)$ norm: the data distribution influences the representations learned by focusing on the part of the space covered by ρ.

For a fixed policy, on a finite space, $\sum_t \gamma^t P_t = (\mathrm{Id} - \gamma P)^{-1}$ as matrices. Thus the largest singular values of $\sum_t \gamma^t P_t$ correspond to the smallest singular values of the Markov chain Laplacian $\mathrm{Id} - \gamma P$. The associated singular vectors loosely correspond to long-range (low-frequency) behavior Mahadevan & Maggioni (2007): presumably, F and B will represent spatially smooth rewards first, while local detail may be lost (Fig. 3 and Section 5). Stachenfeld et al. (2017) argue for the cognitive relevance of low-dimensional approximations of successor representations.

The features learned in F and B may have broader interest.

D Algorithm

The unsupervised phase of the FB algorithm is described in Algorithm 1.

Algorithm 1 FB algorithm: Unsupervised Phase

Inputs: replay buffer $\mathcal{D}$, Polyak coefficient α , ν a probability distribution over $\mathbb{R}^d$, randomly initialized networks F_θ and B_ω, learning rate η, mini-batch size b, number of episodes E, number of gradient updates N, temperature τ and regularization coefficient λ.
for $m = 1, \dots$ **do**
  /* Collect E episodes
  for episode $e = 1, \dots E$ **do**
    Sample $z \sim \nu$
    Observe an initial state s_0
    for $t = 1, \dots$ **do**
      Select an action a_t according to some behaviour policy (e.g the ε-greedy with respect to $F_\theta(s_t, a, z)^\top z$)
      Observe next state s_{t+1}
      Store transition (s_t, a_t, s_{t+1}) in the replay buffer $\mathcal{D}$
    end for
  end for
  /* Perform N stochastic gradient descent updates
  for $n = 1 \dots N$ **do**
    Sample a mini-batch of transitions $\{(s_i, a_i, s_{i+1})\}_{i \in I} \subset \mathcal{D}$ of size $|I| = b$.
    Sample a mini-batch of target state-action pairs $\{(s'_i, a'_i)\}_{i \in I} \subset \mathcal{D}$ of size $|I| = b$.
    Sample a mini-batch of $\{z_i\}_{i \in I} \sim \nu$ of size $|I| = b$.
    Set $\pi_{z_i}(\cdot \mid s_{i+1}) = \texttt{softmax}(F_{\theta^-}(s_{i+1}, \cdot, z_i)^\top z_i / \tau)$
    $\mathscr{L}(\theta, \omega) = \frac{1}{2b^2} \sum_{i,j \in I^2} \left(F_\theta(s_i, a_i, z_i)^\top B_\omega(s'_j, a'_j) - \gamma \sum_{a \in A} \pi_{z_i}(a \mid s_{i+1}) \cdot F_{\theta^-}(s_{i+1}, a, z_i)^\top B_{\omega^-}(s'_j, a'_j) \right)^2 - \frac{1}{b} \sum_{i \in I} F_\theta(s_i, a_i, z_i)^\top B_\omega(s_i, a_i)$
    /* Compute orthonormality regularization loss
    $\mathscr{L}_{\texttt{reg}}(\omega) = \frac{1}{b^2} \sum_{i,j \in I^2} B_\omega(s_i, a_i)^\top \texttt{stop-gradient}(B_\omega(s'_j, a'_j)) \cdot \texttt{stop-gradient}(B_\omega(s_i, a_i)^\top B_\omega(s'_j, a'_j)) - \frac{1}{b} \sum_{i \in I} B_\omega(s_i, a_i)^\top \texttt{stop-gradient}(B_\omega(s_i, a_i))$
    Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathscr{L}(\theta, \omega)$ and $\omega \leftarrow \omega - \eta \nabla_\omega (\mathscr{L}(\theta, \omega) + \lambda \cdot \mathscr{L}_{\texttt{reg}}(\omega))$
  end for
  /* Update target network parameters
  $\theta^- \leftarrow \alpha \theta^- + (1 - \alpha)\theta$
  $\omega^- \leftarrow \alpha \omega^- + (1 - \alpha)\omega$
end for

E Extended Results: Approximate Solutions and General Goals

E.1 The Forward-Backward Representation With a Goal or Feature Space

Here we state a generalization of Theorem 1 covering the extensions mentioned in Section 4.

First, this covers rewards depending on the state-action (s, a) only via certain features $g = \varphi(s, a)$ where φ is a known function from state-actions to some goal state G (for instance, rewards depending only on some components of the state). Then it is enough to compute B as a function of the goal g. Theorem 1 corresponds to $\varphi = \mathrm{Id}$.

This also allows us to cover successor features as in Borsa et al. (2018), defined by user-provided features φ. Indeed, fixing B to Id and setting our φ to the φ of Borsa et al. (2018) (or fixing B to their φ and our φ to Id) will represent the same set of rewards and policies as in Borsa et al. (2018), although with a slightly different learning algorithm and up to a linear change of variables for F and z (given by the covariance of φ, see Appendix E.5), namely, optimal policies for rewards linear in φ. However, keeping the same φ but letting B free (with larger d) can provide optimal policies for rewards that are arbitrary functions of φ, linear or not.

For this, we extend successor state measures to values in goal spaces, representing the discounted time spent at each goal by the policy. Namely, given a policy π, let M^π be the the successor state measure of π over goals g:

$$M^\pi(s, a, \mathrm{d}g) := \sum_{t\geq 0} \gamma^t \Pr\left(\varphi(s_t, a_t) \in \mathrm{d}g \mid s_0 = s,\, a_0 = a,\, \pi\right) \tag{15}$$

for each state-action (s, a) and each measurable set $\mathrm{d}g \subset G$. This will be the object approximated by $F(s, a, z)^\top B(g)$.

Second, we use a more general model of successor states, $m \approx F^\top B + \bar{m}$ where $\bar{m}$ does not depend on the action, so that the $F^\top B$ part is enough to compute advantages; this lifts the constraint that the model of m has rank at most d, because there is no restriction on the rank on $\bar{m}$.

For simplicity we state the result with deterministic rewards, but this extends to stochastic rewards, because the expectation z_R will be the same.

Theorem 2 (Forward-backward representation of an MDP, with features as goals)**.** *Consider an MDP with state space S and action space A. Let $\varphi\colon S \times A \to G$ be a function from state-actions to some goal space $G = \mathbb{R}^k$.*

Let $Z = \mathbb{R}^d$ be some representation space. Let

$$F\colon S \times A \times Z \to Z, \qquad B\colon G \to Z, \qquad \bar{m}\colon S \times Z \times G \to \mathbb{R} \tag{16}$$

be three functions. For each $z \in Z$, define the policy

$$\pi_z(a|s) := \arg\max_a F(s, a, z)^\top z. \tag{17}$$

Let ρ be any probability distribution over the goal space G. For each $z \in Z$, let m^{π_z} be the density of the successor state measure of π_z with respect to ρ: $M^{\pi_z}(s, a, \mathrm{d}g) = m^{\pi_z}(s, a, g)\, \rho(\mathrm{d}g)$.

Assume that F and B have been chosen (trained) to satisfy the following: for any $z \in Z$, any state-actions (s, a), and any goal $g \in G$, one has

$$m^{\pi_z}(s, a, g) = F(s, a, z)^\top B(g) + \bar{m}(s, z, g). \tag{18}$$

Let $R\colon S \times A \to \mathbb{R}$ be any bounded reward function, and assume that this reward function depends only on $g = \varphi(s, a)$, namely, that there exists a function $r\colon G \to \mathbb{R}$ such that $R(s, a) = r(\varphi(s, a))$. Set

$$z_R := \mathbb{E}_{g\sim\rho}\left[r(g)B(g)\right]. \tag{19}$$

Then:

1. *π_{z_R} is an optimal policy for reward R in the MDP.*

2. *For any $z \in Z$, the Q-function of policy π_z for the reward R is equal to*

$$Q^{\pi_z}(s,a) = F(s,a,z)^\top z_R + \bar{V}^z(s) \tag{20}$$

and the optimal Q-function is obtained when $z = z_R$. The advantages $Q^{\pi_z}(s,a) - Q^{\pi_z}(s,a')$ do not depend on $\bar{V}$.

Here

$$\bar{V}^z(s) := \mathbb{E}_{g\sim\rho}[\bar{m}(s,z,g)r(g)]. \tag{21}$$

and in particular $\bar{V} = 0$ if $\bar{m} = 0$.

3. *If $\bar{m} = 0$, then for any state-action (s,a) one has*

$$F(s,a,z_R)^\top z_R = \sup_{z\in Z} F(s,a,z)^\top z_R. \tag{22}$$

(We do not claim that $\bar{V}$ is the value function and $F^\top z_R$ the advantage function, only that the sum is the Q-function. When $\bar{m} = 0$, the term $F^\top z_R$ is the whole Q-function.)

The last point of the theorem is a form of policy improvement. Indeed, by the second point, $F(s,a,z)^\top z_R$ is the estimated Q-function of policy π_z for rewards r. If z_R falls outside of the training distribution for F, the values of $F(s,a,z_R)$ may not be safe to use; in that case, it may be useful to use a finite set $Z' \subset Z$ of values of z closer to the training distribution, and use the estimate $\sup_{z\in Z'} F(s,a,z)^\top z_R$ instead of $F(s,a,z_R)^\top z_R$. This has been used, e.g., in Borsa et al. (2018), but in the end it was not necessary in our experiments.

E.2 Existence of Exact FB Representations in Finite Spaces

We now prove existence of an exact solution for finite spaces if the representation dimension d is at least $\#S \times \#A$. Solutions are never unique: one may always multiply F by an invertible matrix C and multiply B by $(C^\top)^{-1}$, see Remark 4 below (this allows us to impose orthonormality of B in the experiments).

The constraint $d \geq \#S \times \#A$ can be largely overestimated depending on the tasks of interest, though. For instance, we prove below that in an n-dimensional toric grid $S = \{1,\ldots,k\}^n$, $d = 2n$ is enough to obtain optimal policies for reaching every target state (a set of tasks smaller than optimizing all possible rewards).

Proposition 3 (Existence of an exact FB representation for finite state spaces)**.** *Assume that the state and action spaces S and A of an MDP are finite. Let $Z = \mathbb{R}^d$ with $d \geq \#S\times\#A$. Let ρ be any probability distribution on $S \times A$, with $\rho(s,a) > 0$ for any (s,a).*

Then there exists $F\colon S \times A \times Z \to Z$ and $B\colon S \times A \to Z$, such that $F^\top B$ is equal to the successor state density of π_z with respect to ρ:

$$F^\top(s,a,z)B(s',a') = \sum_{t\geq 0} \gamma^t \frac{\Pr((s_t,a_t) = (s',a') \mid s_0 = s,\ a_0 = a,\ \pi_z)}{\rho(s',a')} \tag{23}$$

for any $z \in Z$ and any state-actions (s,a) and (s',a'), where π_z is defined as in Theorem 1 by $\pi_z(s) = \arg\max_a F(s,a,z)^\top z$.

In practice, even a small d can be enough to get optimal policies for reaching arbitrary many states (as opposed to optimizing all possible rewards). Let us give an example with S a toric n-dimensional grid of size k.

Let us start with $n = 1$. Take $S = \{0,\ldots,k-1\}$ to be a length-k cycle with three actions $a \in \{-1,0,1\}$ (go left, stay in place, go right). Take $d = 2$, so that $Z = \mathbb{R}^2 \simeq \mathbb{C}$.

We consider the tasks of reaching an arbitrary target state s', for every $s' \in S$. Thus the goal state is $G = S$ in the notation of Theorem 2, and B only depends on s'. The policy for such a reward is $\pi_{z_R} = \pi_{B(s')}$.

For a state $s \in \{0, \dots, k-1\}$ and action $a \in \{-1, 0, 1\}$, define

$$F(s,a,z) := e^{2i\pi(s+a)/k}, \qquad B(s) := e^{2i\pi s/k}. \tag{24}$$

Then one checks that $\pi_{B(s')}$ is the optimal policy for reaching s', for every $s' \in S$. Indeed, $F(s,a,z_R)^\top z_R = \cos(2\pi(s+a-s')/k)$. This is maximized for the action a that brings s closer to s'.

So the policies will be optimal for reaching every target $s' \in S$, despite the dimension being only 2.

By taking the product of n copies of this example, this also works on the n-dimensional toric grid $S = \{0, \dots, k-1\}^n$ with $2n+1$ actions (add ± 1 in each direction or stay in place), with a representation of dimension $d = 2n$ in $\mathbb{C}^n$, namely, by taking $B(s)_j := e^{2i\pi s_j/k}$ for ecah direction j and likewise for F. Then $\pi_{B(s')}$ is the optimal policy for reaching s' for every $s' \in S$.

More generally, if one is only interested in the optimal policies for reaching states, then it is easy to show that there exist functions $F\colon S \times A \to Z$ and $B\colon S \to Z$ such that the policies π_z describe the optimal policies to reach each state: it is enough that B be injective (typically requiring $d = \dim(S)$). Indeed, for any state $s \in S$, let π^*_s be the optimal policy to reach s. We want π_z to be equal to π^*_s for $z = B(s)$ (the value of z_R for a reward located at s). This translates as $\arg\max_a F(s', a, B(s))^\top B(s) = \pi^*_s(s')$ for every other state s'. This is realized just by letting F be any function such that $F(s', \pi^*_s(s'), B(s)) := B(s)$ and $F(s', a, B(s)) := -B(s)$ for every other action a. As soon as B is injective, there exists such a function F. (Unfortunately, we are not able to show that the learning algorithm reaches such a solution.)

Let us turn to uniqueness of F and B.

Remark 4. *Let C be an invertible $d \times d$ matrix. Given F and B as in Theorem 1, define*

$$B'(s,a) := CB(s,a), \qquad F'(s,a,z) := (C^\top)^{-1} F(s,a,C^{-1}z) \tag{25}$$

together with the policies $\pi'_z(s) := \arg\max_a F'(s,a,z)^\top z$. For each reward r, define $z'_R := \mathbb{E}_{(s,a)\sim\rho}[r(s,a)B'(s,a)]$.

Then this operation does not change the policies or estimated Q-values: for any reward, we have $\pi'_{z'_R} = \pi_{z_R}$, and $F'(s,a,z'_R)^\top z'_R = F(s,a,z_R)^\top z_R$.

In particular, assume that the components of B are linearly independent. Then, taking $C = \left(\mathbb{E}_{(s,a)\sim\rho} B(s,a)B(s,a)^\top\right)^{-1/2}$, B' is $L^2(\rho)$-orthonormal. So up to reducing the dimension d to $\operatorname{rank}(B)$, we can always assume that B is $L^2(\rho)$-orthonormal.

Reduction to orthonormal B will be useful in some proofs below. Even after imposing that B be orthonormal, solutions are not unique, as one can still apply a rotation matrix on the variable z.

E.3 Approximate Solutions Provide Approximately Optimal Policies

Here we prove that the optimality in Theorems 1 and 2 is robust to approximation errors during training. We deal in turn with the approximation errors on (F, B) during unsupervised training, and on z_R during the reward estimation phase.

E.3.1 Influence of Approximate F and B

In continuous spaces, Theorems 1 and 2 are somewhat spurious: the equality $F^\top B = m$ will never hold exactly with finite representation dimension d. Instead, $F^\top B$ will only be a rank-d approximation of m. Even in finite spaces, since F and B are learned by a neural network, we can only expect that $F^\top B \approx m$ in general.

We now prove that approximate solutions still provide approximately optimal policies. We start with the error from learning F and B in the unsupervised learning phase. Then we turn to the error from estimating z_R in the reward estimation phase (in case the reward is not known explicitly).

We provide this result for different notions of approximate solutions for F and B: first, in sup norm over (s, a) but in expectation over (s', a') (so that a perfect model of the world is not necessary); second, for the weak topology on measures (this is the most relevant in continuous spaces: for instance, a Dirac measure can be approached by a continuous model in the weak topology).

F and B are trained such that $F(s, a, z)^\top B(s', a')$ approximates the successor state density $m^{\pi_z}(s, a, s', a')$. In the simplest case, we prove that if for some reward R,

$$\mathbb{E}_{(s',a')\sim\rho} \left|F(s, a, z_R)^\top B(s', a') - m^{\pi_{z_R}}(s, a, s', a')\right| \leq \varepsilon \tag{26}$$

for every (s, a), then the optimality gap of policy π_{z_R} is at most $(3\varepsilon/(1-\gamma)) \sup |R|$ for that reward (Theorem 5, first case).

In continuous spaces, m^π is usually a distribution (Appendix E.4), so such an approximation will not hold, and it is better to work on the measures themselves rather than their densities, namely, to compare $F^\top B\rho$ to M^π. We prove that if $F^\top B\rho$ is close to M^π in the weak topology, then the resulting policies are optimal for any Lipschitz reward.[2]

Remember that a sequence of nonnegative measures μ_n converges weakly to μ if for any bounded, continuous function f, $\int f(x)\mu_n(dx)$ converges to $\int f(x)\mu(\mathrm{d}x)$ (Bogachev (2007), §8.1). The associated topology can be defined via the following *Kantorovich–Rubinstein* norm on nonnegative measures (Bogachev (2007), §8.3)

$$\|\mu - \mu'\|_{\mathrm{KR}} := \sup\left\{\left|\int f(x)\mu(\mathrm{d}x) - \int f(x)\mu'(\mathrm{d}x)\right| : f \text{ 1-Lipschitz function with } \sup|f| \leq 1\right\} \tag{27}$$

where we have equipped the state-action space with any metric compatible with its topology.[3]

The following theorem states that if $F^\top B$ approximates the successor state density of the policy π_z (for various sorts of approximations), then π_z is approximately optimal. Given a reward function r on state-actions, we denote

$$\|r\|_\infty := \sup_{(s,a)\in S\times A} |r(s, a)| \tag{28}$$

and

$$\|r\|_{\mathrm{Lip}} := \sup_{(s,a)\neq(s',a')\in S\times A} \frac{r(s, a) - r(s', a')}{d((s, a), (s', a'))} \tag{29}$$

where we have chosen any metric on state-actions.

The first statement is for any bounded reward. The second statement only assumes an $F^\top B$ approximation in the weak topology but only applies to Lipschitz rewards. The third statement is more general and is how we prove the first two: weaker assumptions on $F^\top B$ work on a stricter class of rewards.

Theorem 5 (If F and B approximate successor states, then the policies π_z yield approximately optimal returns)**.** *Let $F\colon S \times A \times Z \to Z$ and $B\colon S \times A \to Z$ be any functions, and define the policy $\pi_z(s) = \arg\max_a F(s, a, z)^\top z$ for each $z \in Z$.*

Let ρ be any positive probability distribution on $S \times A$, and for each policy π, let m^π be the density of the successor state measure M^π of π with respect to ρ. Let

$$\hat{m}^z(s, a, s', a') := F(s, a, z)^\top B(s', a'), \qquad \hat{M}^z(s, a, \mathrm{d}s', \mathrm{d}a') := \hat{m}^z(s, a, s', a')\rho(\mathrm{d}s', \mathrm{d}a') \tag{30}$$

[2]This also holds for continuous rewards, but the Lipschitz assumption yields an explicit bound in Theorem 5.

[3]The Kantorovich–Rubinstein norm is closely related to the L^1 Wasserstein distance on probability distributions, but slightly more general as it does not require the distance functions to be integrable: the Wasserstein distance metrizes weak convergence among those probability measures such that $\mathbb{E}[d(x, x_0)] < \infty$.

be the estimates of m and M obtained via the model F and B.

Let $r\colon S\times A\to\mathbb{R}$ be any bounded reward function. Let V^* be the optimal value function for this reward r. Let $\hat V^{\pi_z}$ be the value function of policy π_z for this reward. Let $z_R = \mathbb{E}_{(s,a)\sim\rho}[r(s,a)B(s,a)]$.

Then:

1. *If $\mathbb{E}_{(s',a')\sim\rho}\left|\hat m^{z_R}(s,a,s',a') - m^{\pi_{z_R}}(s,a,s',a')\right| \le \varepsilon$ for any (s,a) in $S\times A$, then $\left\|\hat V^{\pi_{z_R}} - V^*\right\|_\infty \le 3\varepsilon\left\|r\right\|_\infty/(1-\gamma)$.*

2. *If r is Lipschitz and $\left\|\hat M^{z_R}(s,a,\cdot) - M^{\pi_{z_R}}(s,a,\cdot)\right\|_{\mathrm{KR}} \le \varepsilon$ for any $(s,a)\in S\times A$, then $\left\|\hat V^{\pi_{z_R}} - V^*\right\|_\infty \le 3\varepsilon\max(\left\|r\right\|_\infty, \left\|r\right\|_{\mathrm{Lip}})/(1-\gamma)$.*

3. *More generally, let $\left\|\cdot\right\|_A$ be a norm on functions and $\left\|\cdot\right\|_B$ a norm on measures, such that $\int f\,\mathrm{d}\mu \le \left\|f\right\|_A\left\|\mu\right\|_B$ for any function f and measure μ. Then for any reward function r such that $\left\|r\right\|_A<\infty$,*

$$\left\|\hat V^{\pi_{z_R}} - V^*\right\|_\infty \le \frac{3\left\|r\right\|_A}{1-\gamma}\sup_{s,a}\left\|\hat M^{z_R}(s,a,\cdot) - M^{\pi_{z_R}}(s,a,\cdot)\right\|_B. \tag{31}$$

   Moreover, the optimal Q-function is close to $F(s,a,z_R)^\top z_R$:

$$\sup_{s,a}\left|F(s,a,z_R)^\top z_R - Q^*(s,a)\right| \le \frac{2\left\|r\right\|_A}{1-\gamma}\sup_{s,a}\left\|\hat M^{z_R}(s,a,\cdot) - M^{\pi_{z_R}}(s,a,\cdot)\right\|_B. \tag{32}$$

E.3.2 An approximate z_R yields an approximately optimal policy

We now turn to the second source of approximation: computing z_R in the reward estimation phase. This is a problem only if the reward is not specified explicitly.

We deal in turn with the effect of using a model of the reward function, and the effect of estimating $z_R = \mathbb{E}_{(s,a)\sim\rho}\, r(s,a)B(s,a)$ via sampling.

Which of these options is better depends a lot on the situation. If training of F and B is perfect, by construction the policies are optimal for each z_R: thus, estimating z_R from rewards sampled at N states $(s_i,a_i)\sim\rho$ will produce the optimal policy for exactly that empirical reward, namely, a nonzero reward at each (s_i,a_i) but zero everywhere else, thus overfitting the reward function. Reducing the dimension d reduces this effect, since rewards are projected on the span of the features in B: B plays both the roles of a transition model and a reward regularizer. This appears as a $\sqrt{d/N}$ factor in Theorem 8 below.

Thus, if both the number of samples to train F and B and the number of reward samples are small, using a smaller d will regularize both the model of the environment and the model of the reward. However, if the number of samples to train F and B is large, yielding an excellent model of the environment, but the number of reward samples is small, then learning a model of the reward function will be a better option than direct empirical estimation of z_R.

The first result below states that reward misidentification comes on top of the approximation error of the $F^\top B$ model. This is relevant, for instance, if a reward model $\hat r$ is estimated by an external model using some reward values.

Proposition 6 (Influence of estimating z_R by an approximate reward)**.** *Let $\hat r\colon S\times A\to\mathbb{R}$ be any reward function. Let $\hat z_R = \mathbb{E}_{(s,a)\sim\rho}[\hat r(s,a)B(s,a)]$.*

Let ε_{FB} be the error attained by the $F^\top B$ model in Theorem 5 for reward $\hat r$; namely, assume that $\left\|\hat V^{\pi_{\hat z_R}} - \hat V^*\right\|_\infty \le \varepsilon_{FB}$ with $\hat V^*$ the optimal value function for $\hat r$.

Then the policy $\pi_{\hat z_R}(s) = \arg\max_a F(s,a,\hat z_R)^\top \hat z_R$ defined by the model $\hat r$ is $\left(\dfrac{\left\|r-\hat r\right\|_\infty}{(1-\gamma)} + \varepsilon_{FB}\right)$-optimal for reward r.

This still assumes that the expectation $\hat{z}_R = \mathbb{E}_{(s,a)\sim\rho}[\hat{r}(s,a)B(s,a)]$ is computed exactly for the model $\hat{r}$. If $\hat{r}$ is given by an explicit model, this expectation can in principle be computed on the whole replay buffer used to train F and B, so variance would be low. Nevertheless, we provide an additional statement which covers the influence of the variance of the estimator of z_R, whether this estimator uses an external model $\hat{r}$ or a direct empirical average reward observations $r(s,a)B(s,a)$.

Definition 7. *The* skewness $\zeta(B)$ *of B is defined as follows. Assume B is bounded. Let $B_1, \ldots, B_d \colon S \times A \to \mathbb{R}$ be the functions of (s,a) defined by each component of B. Let $\langle B \rangle$ be the linear span of the $(B_i)_{1\leq i\leq d}$ as functions on $S \times A$. Set*

$$\zeta(B) := \sup_{f\in\langle B\rangle,\, f\neq 0} \frac{\|f\|_\infty}{\|f\|_{L^2(\rho)}}. \tag{33}$$

Theorem 8 (Influence of estimating z_R by empirical averages)**.** *Assume that $z_R = \mathbb{E}_{(s,a)\sim\rho}[r(s,a)B(s,a)]$ is estimated via*

$$\hat{z}_R := \frac{1}{N}\sum_{i=1}^{N} \hat{r}_i B(s_i, a_i) \tag{34}$$

using N independent samples $(s_i, a_i) \sim \rho$, where the r_i are random variables such that $\mathbb{E}[\hat{r}_i|s_i,a_i] = r(s_i,a_i)$, $\mathrm{Var}[\hat{r}_i|s_i,a_i] \leq v$ for some $v \in \mathbb{R}$, and the $\hat{r}_i$ are mutually independent given $(s_i,a_i)_{i=1,\ldots,N}$.

Let V^* be the optimal value function for reward r, and let $\hat{V}$ be the value function of the estimated policy $\pi_{\hat{z}_R}$ for reward r.

Then, for any $\delta > 0$, with probability at least $1-\delta$,

$$\left\|\hat{V} - V^*\right\|_\infty \leq \varepsilon_{FB} + \frac{1}{1-\gamma}\sqrt{\frac{\zeta(B)\,d}{N\delta}\left(v + \|r(s,a) - \mathbb{E}_\rho r\|^2_{L^2(\rho)}\right)} \tag{35}$$

which is therefore the bound on the optimality gap of $\pi_{\hat{z}_R}$ for r. Here ε_{FB} is the error due to the $F^\top B$ model approximation, defined as in Proposition 6.

The proofs are not direct, because F is not continuous with respect to z. Contrary to Q-values, successor states are not continuous in the reward: if an action has reward 1 and the reward for another action changes from $1-\varepsilon$ to $1+\varepsilon$, the return values change by at most 2ε, but the actions and states visited by the optimal policy change a lot. So it is not possible to reason by continuity on each of the terms involved.

E.4 A Note on the Measure M^π and its Density m^π

In finite spaces, the definition of the successor state density m^π via

$$M^\pi(s,a,\mathrm{d}s',\mathrm{d}a') = m^\pi(s,a,s',a')\rho(\mathrm{d}s',\mathrm{d}a') \tag{36}$$

with respect to the data distribution ρ poses no problem, as long as the data distribution is positive everywhere.

In continuous spaces, this can be understood as the (Radon–Nikodym) density of M^π with respect to ρ, assuming M^π has no singular part with respect to ρ. However, this is *never* the case: in the definition (7) of the successor state measure M^π, the term $t=0$ produces a Dirac measure $\delta_{s,a}$. So M^π has a singular component due to $t=0$, and m^π is better thought of as a distribution.

When m^π is a distribution, a continuous parametric model m^π_θ learned by (9) can approximate m^π in the weak topology only: $m^\pi_\theta\rho$ approximates M^π for the weak convergence of measures. Thus, for the forward-backward representation, $F(s,a,z)^\top B(s',a')\rho(\mathrm{d}s',\mathrm{d}a')$ weakly approximates $M^{\pi_z}(s,a,\mathrm{d}s',\mathrm{d}a')$.

We have not found this to be a problem either in theory or practice, and Theorem 5 covers weak approximations.

Alternatively, one may just define successor states starting at $t = 1$ in (7). This only works if rewards $r(s, a)$ depend on the state s but not the action a (e.g., in goal-oriented settings). If starting the definition at $t = 1$, m^π is an ordinary function provided the transition kernels $P(\mathrm{d}s'|s, a)$ of the environment are non-singular, ρ has positive density, and $\pi(\mathrm{d}a|s)$ is non-singular as well. But starting at $t = 1$ induces some changes in the theorems:

- In the learning algorithm (9) for successor states, the term $\partial_\theta m^\pi_\theta(s, a, s, a)$ becomes $\gamma\, \partial_\theta m^\pi_\theta(s, a, s', a')$.
- The expression for the Q-function in Theorem 1 becomes $Q^*(s, a) = r(s, a) + F(s, a, z_R)^\top z_R$, and likewise in Theorem 2. The $r(s, a)$ term covers the immediate reward at a state, since we have excluded $t = 0$ from the definition of successor states.
- In general the expression for optimal policies becomes

$$\pi_z(s) := \arg\max_a \{r(s, a) + F(s, a, z)^\top z\} \tag{37}$$

  which cannot be computed from z and F alone in the unsupervised training phase. The algorithm only makes sense for rewards that depend on s but not on a (e.g., in goal-oriented settings): then the policy π_z is equal to $\pi_z(s) := \arg\max_a F(s, a, z)^\top z$ again.

E.5 F and B as Successor and Predecessor Features of Each Other

We now give a statement that specifies how F encodes the future of a state while B encodes the past of a state. Namely: if F and B minimize their unsupervised loss, then F is equal to the successor features from the dual features of B, and B is equal to the *predecessor* features from the dual features of F (Theorem 9).

This statement holds for a fixed z and the corresponding policy π_z. So, for the rest of this section, z is fixed.

By "dual" features we mean the following. Define the $d \times d$ covariance matrices

$$\operatorname{Cov} F := \mathbb{E}_{(s,a)\sim\rho}[F(s, a, z)F(s, a, z)^\top], \qquad \operatorname{Cov} B := \mathbb{E}_{(s,a)\sim\rho}[B(s, a)B(s, a)^\top]. \tag{38}$$

Then $(\operatorname{Cov} F)^{-1/2}F(s, a, z)$ is $L^2(\rho)$-orthonormal and likewise for B. The "dual" features are $(\operatorname{Cov} F)^{-1}F(s, a, z)$ and $(\operatorname{Cov} B)^{-1}B(s, a)$, without the square root: these are the least square solvers for F and B respectively, and these are the ones that appear below.

The unsupervised FB loss (14) for a fixed z is

$$\ell(F, B) := \int \left| F(s, a, z)^\top B(s', a') - \sum_{t\geq 0} \gamma^t \frac{P_t(\mathrm{d}s', \mathrm{d}a'|s, a, \pi_z)}{\rho(\mathrm{d}s', \mathrm{d}a')} \right|^2 \rho(\mathrm{d}s, \mathrm{d}a)\rho(\mathrm{d}s', \mathrm{d}a'). \tag{39}$$

Thus, minimizers in dimension d correspond to an SVD of the successor state density in $L^2(\rho)$, truncated to the largest d singular values.

Theorem 9. *Consider a smooth parametric model for F and B, and assume this model is overparameterized.* [4] *Also assume that the data distribution ρ has positive density everywhere.*

Let $z \in Z$. Assume that for this z, F and B lie in $L^2(\rho)$ and achieve a local extremum of $\ell(F, B)$ within this parametric model. Namely, the derivative $\partial_\theta \ell(F, B)$ of the loss with respect to the parameters θ of F is 0, and likewise for B.

[4] Intuitively, a parametric function f is overparameterized if every possible small change of f can be realized by a small change of the parameter. Formally, we say that a parametric family of functions $\theta \in \Theta \mapsto f_\theta \in L^2(X, \mathbb{R}^d)$ smoothly parameterized by θ, on some space X, is *overparameterized* if, for any θ, the differential $\partial_\theta f_\theta$ is surjective from Θ to $L^2(X, \mathbb{R}^d)$. For finite X, this implies that the dimension of θ is larger than $\#X$. For infinite X, this implies that $\dim(\theta)$ is infinite, such as parameterizing functions on $[0; 1]$ by their Fourier expansion.

Then F is equal to $(\operatorname{Cov} B)^{-1}$ times the successor features of B: for any $(s,a) \in S \times A$,

$$(\operatorname{Cov} B) F(s,a,z) = \sum_{t \geq 0} \gamma^t \int_{(s',a')} P_t(\mathrm{d}s', \mathrm{d}a' | s, a, \pi_z)\, B(s',a') \tag{40}$$

and B is equal to $(\operatorname{Cov} F)^{-1}$ times the predecessor features of F:

$$(\operatorname{Cov} F) B(s',a') = \sum_{t \geq 0} \gamma^t \int_{(s,a)} \frac{P_t(\mathrm{d}s', \mathrm{d}a' | s, a, \pi_z)}{\rho(\mathrm{d}s', \mathrm{d}a')} F(s,a,z)\, \rho(\mathrm{d}s, \mathrm{d}a) \tag{41}$$

ρ-almost everywhere. Here the covariances have been defined in (38), and $P_t(\cdot|s,a,\pi)$ denotes the law of (s_t, a_t) under trajectories starting at (s,a) and following policy π.

The same result holds when working with features $\varphi(s',a')$, just by applying it to $B \circ \varphi$.

Note that in the FB framework, we may normalize either F or B (Remark 4), but not both.

F Proofs

The first proposition is a direct consequence of the definition (15) of successor states with a goal space G.

Proposition 10. *Let $\varphi\colon S \times A \to G$ be a map to some goal space G.*

Let π be some policy, and let M^π be the successor state measure (15) of π in goal space G. Let m^π be the density of M^π with respect to some positive probability measure ρ on G.

Let $r\colon G \to \mathbb{R}$ be some function on G, and define the reward function $R(s,a) := r(\varphi(s,a))$ on $S \times A$.

Then the Q-function Q^π of policy π for reward R is

$$Q^\pi(s,a) = \int r(g)\, M^\pi(s,a,\mathrm{d}g) \tag{42}$$

$$= \mathbb{E}_{g\sim\rho}\left[r(g)\, m^\pi(s,a,g)\right]. \tag{43}$$

Proof of Proposition 10. For each time $t \geq 0$, let $P_t^\pi(s_0, a_0, \mathrm{d}g)$ be the probability distribution of $g = \varphi(s_t, a_t)$ over trajectories of the policy π starting at (s_0, a_0) in the MDP. Thus, by the definition (15),

$$M^\pi(s,a,\mathrm{d}g) = \sum_{t\geq 0} \gamma^t P_t^\pi(s,a,\mathrm{d}g). \tag{44}$$

The Q-function of π for the reward R is by definition (the sums and integrals are finite since R is bounded)

$$\begin{aligned}
Q^\pi(s,a) &= \sum_{t\geq 0} \gamma^t\, \mathbb{E}[R(s_t,a_t) \mid s_0 = s,\, a_0 = a,\, \pi] && (45)\\
&= \sum_{t\geq 0} \gamma^t\, \mathbb{E}[r(\varphi(s_t,a_t)) \mid s_0 = s,\, a_0 = a,\, \pi] && (46)\\
&= \sum_{t\geq 0} \gamma^t \int_g r(g) P_t^\pi(s,a,\mathrm{d}g) && (47)\\
&= \int_g r(g) M^\pi(s,a,\mathrm{d}g) && (48)\\
&= \int_g r(g) m^\pi(s,a,g)\rho(\mathrm{d}g) && (49)\\
&= \mathbb{E}_{g\sim\rho}[r(g) m^\pi(s,a,g)] && (50)
\end{aligned}$$

by definition of the density m^π. □

Proof of Theorems 1 and 2. Theorem 1 is a particular case of Theorem 2 ($\varphi = \mathrm{Id}$ and $\bar{m} = 0$), so we only prove the latter.

Let $R(s,a) = r(\varphi(s,a))$ be a reward function as in the theorem.

The Q-function of π for the reward R is, by Proposition 10,

$$Q^\pi(s,a) = \mathbb{E}_{g\sim\rho}[r(g) m^\pi(s,a,g)]. \tag{51}$$

The assumptions state that for any $z \in Z$, $m^{\pi_z}(s,a,g)$ is equal to $F(s,a,z)^\top B(g) + \bar{m}(s,z,g)$. Therefore, for any $z \in Z$ we have

$$\begin{aligned}
Q^{\pi_z}(s,a) &= \mathbb{E}_{g\sim\rho}\left[r(g) F(s,a,z)^\top B(g) + r(g)\bar{m}(s,z,g)\right] && (52)\\
&= F(s,a,z)^\top\, \mathbb{E}_{g\sim\rho}\left[r(g) B(g)\right] + \mathbb{E}_{g\sim\rho}[r(g)\bar{m}(s,z,g)] && (53)\\
&= F(s,a,z)^\top z_R + \bar{V}^z(s) && (54)
\end{aligned}$$

by definition of z_R and $\bar{V}$. This proves the claim (20) about Q-functions.

By definition, the policy π_z selects the action a that maximizes $F(s,a,z)^\top z$. Take $z = z_R$. Then

$$\pi_{z_R} = \arg\max_a F(s,a,z_R)^\top z_R \tag{55}$$

$$= \arg\max_a \left\{ F(s,a,z_R)^\top z_R + \bar{V}^z(s) \right\} \tag{56}$$

since the last term does not depend on a.

This quantity is equal to $Q^{\pi_{z_R}}(s,a)$. Therefore,

$$\pi_{z_R} = \arg\max_a Q^{\pi_{z_R}}(s,a) \tag{57}$$

and by the above, $Q^{\pi_{z_R}}(s,a)$ is indeed equal to the Q-function of policy π_{z_R} for the reward R. Therefore, π_{z_R} and $Q^{\pi_{z_R}}$ constitute an optimal Bellman pair for reward R. Since $Q^{\pi_{z_R}}(s,a)$ is the Q-function of π_{Z_R}, it satisfies the Bellman equation

$$Q^{\pi_{z_R}}(s,a) = R(s,a) + \gamma\, \mathbb{E}_{s'|(s,a)}\, Q^{\pi_{z_R}}(s', \pi_{z_R}(s')) \tag{58}$$

$$= R(s,a) + \gamma\, \mathbb{E}_{s'|(s,a)} \max_{a'} Q^{\pi_{z_R}}(s', a') \tag{59}$$

by (57). This is the optimal Bellman equation for R, and π_{z_R} is the optimal policy for R.

We still have to prove the last statement of Theorem 2. Since π_{z_R} is an optimal policy for R, for any other policy π_z and state-action (s,a) we have

$$Q^{\pi_{z_R}}(s,a) \geq Q^{\pi_z}(s,a). \tag{60}$$

Using the formulas above for Q^π, with $\bar{m} = 0$, this rewrites as

$$F(s,a,z_R)^\top z_R \geq F(s,a,z)^\top z_R \tag{61}$$

as needed. Thus $F(s,a,z_R)^\top z_R \geq \sup_{z\in Z} F(s,a,z)^\top z_R$, and equality occurs by taking $z = z_R$. This ends the proof of Theorem 2. ☐

Proof of Proposition 3. Assume $d = \#S \times \#A$; extra dimensions can just be ignored by setting the extra components of F and B to 0.

With $d = \#S \times \#A$, we can index the components of Z by pairs (s,a).

First, let us set $B(s,a) := \mathbb{1}_{s,a}$.

Let $r\colon S \times A \to \mathbb{R}$ be any reward function. Let $z_R \in \mathbb{R}^{\#S\times\#A}$ be defined as in Theorem 1, namely, $z_R = \mathbb{E}_{(s,a)\sim\rho}[r(s,a)B(s,a)]$. With our choice of B, the components of z_R are $(z_R)_{s,a} = r(s,a)\rho(s,a)$. Since $\rho > 0$, the correspondence $r \leftrightarrow z_R$ is bijective.

Let us now define F. Take $z \in Z$. Since $r \leftrightarrow z_R$ is bijective, this z is equal to z_R for some reward function r. Let π_z be an optimal policy for this reward r in the MDP. Let M^{π_r} be the successor state measure of policy π_z, namely:

$$M^{\pi_z}(s,a,s',a') = \sum_{t\geq 0} \gamma^t \Pr\left((s_t,a_t) = (s',a') \mid (s_0,a_0) = (s,a),\, \pi_z\right). \tag{62}$$

Now define $F(s,a,z)$ by setting its (s',a') component to $M^{\pi_z}(s,a,s',a')/\rho(s',a')$ for each (s',a'):

$$F(s,a,z)_{s',a'} := M^{\pi_z}(s,a,s',a')/\rho(s',a'). \tag{63}$$

Then we have

$$F(s,a,z)^\top B(s',a') = \sum_{s'',a''} F(s,a,z)_{s'',a''}\, B(s',a')_{s'',a''}$$

$$= F(s,a,z)_{s',a'} = M^{\pi_z}(s,a,s',a')/\rho(s',a') \tag{64}$$

because by our choice of B, $B(s',a')_{s''a''} = \mathbb{1}_{s'=s'',\,a'=a''}$.

Thus, $F(s,a,z)^\top B(s',a')$ is the density of the successor state measure M^{π_z} of policy π_z with respect to ρ, as needed.

We still have to check that π_z satisfies $\pi_z(s) = \arg\max F(s,a,z)^\top z$ (since this is not how it was defined). Since π_z was defined as an optimal policy for the reward r associated with z, it satisfies

$$\pi_z(s) = \arg\max_a Q^{\pi_z}(s,a) \tag{65}$$

with $Q^{\pi_z}(s,a)$ the Q-function of policy π_z for the reward r. This Q-function is equal to the cumulated expected reward

$$\begin{aligned}
Q^{\pi_z}(s,a) &= \sum_{t\geq 0} \gamma^t \,\mathbb{E}\left[r(s_t,a_t) \mid s_0 = s,\, a_0 = a,\, \pi_z\right] && (66)\\
&= \sum_{t\geq 0} \gamma^t \sum_{s',a'} r(s',a') \Pr\left((s_t,a_t) = (s',a') \mid s_0 = s,\, a_0 = a,\, \pi_z\right) && (67)\\
&= \sum_{s',a'} r(s',a') \sum_{t\geq 0} \gamma^t \Pr\left((s_t,a_t) = (s',a') \mid s_0 = s,\, a_0 = a,\, \pi_z\right) && (68)\\
&= \sum_{s',a'} r(s',a') M^{\pi_z}(s,a,s',a') && (69)\\
&= \sum_{s',a'} r(s',a') F(s,a,z)_{s',a'}\, \rho(s',a') && (70)\\
&= F(s,a,z)^\top \left(\sum_{s'a'} r(s',a')\rho(s',a') \mathbb{1}_{s'a'}\right) && (71)\\
&= F(s,a,z)^\top z && (72)
\end{aligned}$$

since z is equal to $\mathbb{E}_{(s',a')\sim\rho}[r(s',a')B(s',a')]$. This proves that $\pi_z(s) = \arg\max_a Q^{\pi_z(s,a)} = \arg\max_a F(s,a,z)^\top z$. So this choice of F and B satisfies all the properties claimed. □

We will rely on the following two basic results in Q-learning.

Proposition 11 ($r \mapsto Q^*$ is Lipschitz in sup-norm). *Let r_1, $r_2\colon S\times A \to \mathbb{R}$ be two bounded reward functions. Let Q_1^* and Q_2^* be the corresponding optimal Q-functions, and likewise for the V-functions. Then*

$$\sup_{S\times A} |Q_1^* - Q_2^*| \leq \frac{1}{1-\gamma} \sup_{S\times A} |r_1 - r_2| \tag{73}$$

and

$$\sup_{S} |V_1^* - V_2^*| \leq \frac{1}{1-\gamma} \sup_{S\times A} |r_1 - r_2|\,. \tag{74}$$

Proof. Assume $\sup_{S\times A} |r_1 - r_2| \leq \varepsilon$ for some $\varepsilon \geq 0$.

For any policy π, let Q_1^π be its Q-function for reward r_1, and likewise for r_2. Let π_1 and π_2 be optimal policies for r_1 and r_2, respectively. Then for any $(s,a) \in S\times A$,

$$\begin{aligned}
Q_1^*(s,a) &= Q_1^{\pi_1}(s,a) && (75)\\
&\geq Q_1^{\pi_2}(s,a) && (76)\\
&= \sum_{t\geq 0} \gamma^t \,\mathbb{E}\left[r_1(s_t,a_t) \mid \pi_2,\, (s_0,a_0) = (s,a)\right] && (77)\\
&\geq \sum_{t\geq 0} \gamma^t \,\mathbb{E}\left[r_2(s_t,a_t) - \varepsilon \mid \pi_2,\, (s_0,a_0) = (s,a)\right] && (78)\\
&= \sum_{t\geq 0} \gamma^t \,\mathbb{E}\left[r_2(s_t,a_t) \mid \pi_2,\, (s_0,a_0) = (s,a)\right] - \frac{\varepsilon}{1-\gamma} && (79)\\
&= Q_2^*(s,a) - \frac{\varepsilon}{1-\gamma} && (80)
\end{aligned}$$

and likewise in the other direction, which ends the proof for Q-functions. The case of V-functions follows by restricting to the optimal actions at each state s. □

Proposition 12. *Let $f\colon S \times A \to \mathbb{R}$ be any function, and define a policy π_f by $\pi_f(s) := \arg\max_a f(s,a)$. Let $r\colon S \times A \to \mathbb{R}$ be some bounded reward function. Let Q^* be its optimal Q-function, and let Q^{π_f} be the Q-function of π_f for reward r.*

Then

$$\sup_{S\times A} |f - Q^*| \leq \frac{2}{1-\gamma} \sup_{S\times A} |f - Q^{\pi_f}| \tag{81}$$

and

$$\sup_{S\times A} |Q^{\pi_f} - Q^*| \leq \frac{3}{1-\gamma} \sup_{S\times A} |f - Q^{\pi_f}| \tag{82}$$

Proof. Define $\varepsilon(s,a) := Q^{\pi_f}(s,a) - f(s,a)$.

The Q-function Q^{π_f} satisfies the Bellman equation

$$Q^{\pi_f}(s,a) = r(s,a) + \gamma\, \mathbb{E}_{s'|(s,a)}\, Q^{\pi_f}(s', \pi_f(s')) \tag{83}$$

for any $(s,a) \in S \times A$. Substituting $Q^{\pi_f} = f + \varepsilon$, this rewrites as

$$\begin{aligned} f(s,a) &= r(s,a) - \varepsilon(s,a) + \gamma\, \mathbb{E}_{s'|(s,a)} \left[f(s', \pi_f(s')) + \varepsilon(s', \pi_f(s')) \right] && (84)\\ &= r(s,a) - \varepsilon'(s,a) + \gamma\, \mathbb{E}_{s'|(s,a)}\, f(s', \pi_f(s')) && (85)\\ &= r(s,a) - \varepsilon'(s,a) + \gamma\, \mathbb{E}_{s'|(s,a)} \max_{a'} f(s', a') && (86) \end{aligned}$$

by definition of π_f, where we have set

$$\varepsilon'(s,a) := \varepsilon(s,a) - \gamma\, \mathbb{E}_{s'|(s,a)}\, \varepsilon(s', \pi_f(s')). \tag{87}$$

(86) is the optimal Bellman equation for the reward $r - \varepsilon'$. Therefore, f is the optimal Q-function for the reward $r - \varepsilon'$. Since Q^* is the optimal Q-function for reward r, by Proposition 11, we have

$$\sup_{S\times A} |f - Q^*| \leq \frac{1}{1-\gamma} \sup_{S\times A} |\varepsilon'| \tag{88}$$

By construction of ε', $\sup_{S\times A} |\varepsilon'| \leq 2 \sup_{S\times A} |\varepsilon| = 2 \sup_{S\times A} |f - Q^{\pi_f}|$. This proves the first claim.

The second claim follows by the triangle inequality $|Q^{\pi_f} - Q^*| \leq |Q^{\pi_f} - f| + |f - Q^*|$ and $\frac{2}{1-\gamma} + 1 \leq \frac{3}{1-\gamma}$. □

Proof of Theorem 5. By construction of the Kantorovich–Rubinstein norm, the second claim of Theorem 5 is a particular case of the third claim, with $\|f\|_A := \max(\|f\|_\infty, \|f\|_{\mathrm{Lip}})$ and $\|\mu\|_B := \|\mu\|_{\mathrm{KR}}$.

Likewise, since m is the density of M with respect to ρ, the first claim is an instance of the third, by taking $\|f\|_A := \|f\|_\infty$ and $\|\mu\|_B := \left\| \frac{\mathrm{d}\mu}{\mathrm{d}\rho} \right\|_{L^1(\rho)}$. Therefore, we only prove the third claim.

Let $z \in Z$ and let $r\colon S \times A \to \mathbb{R}$ be any reward function. By Proposition 10 with $G = S \times A$ and $\varphi = \mathrm{Id}$, the Q-function of policy Q^{π_z} for this reward is

$$Q^{\pi_z}(s,a) = \int r(s', a')\, M^{\pi_z}(s, a, \mathrm{d}s', \mathrm{d}a'). \tag{89}$$

Let $\varepsilon_z(s, a, \mathrm{d}s', \mathrm{d}a')$ be the difference of measures between the model $F^\top B \rho$ and M^{π_z}:

$$\begin{aligned} \varepsilon_z(s, a, \mathrm{d}s', \mathrm{d}a') &:= M^{\pi_z}(s, a, \mathrm{d}s', \mathrm{d}a') - \hat{M}^z(s, a, \mathrm{d}s', \mathrm{d}a') && (90)\\ &= M^{\pi_z}(s, a, \mathrm{d}s', \mathrm{d}a') - F(s,a,z)^\top B(s', a') \rho(\mathrm{d}s', \mathrm{d}a'). && (91) \end{aligned}$$

We want to control the optimality gap in terms of $\sup_{s,a} \|\varepsilon_z(s,a,\cdot)\|_B$.

By definition of ε_z,

$$Q^{\pi_z}(s,a) = \int r(s',a') F(s,a,z)^\top B(s',a') \rho(\mathrm{d}s',\mathrm{d}a') + \int r(s',a')\, \varepsilon_z(s,a,\mathrm{d}s',\mathrm{d}a') \tag{92}$$

$$= F(s,a,z)^\top z_R + \int r(s',a')\, \varepsilon_z(s,a,\mathrm{d}s',\mathrm{d}a') \tag{93}$$

since $z_R = \int r(s',a') B(s',a') \rho(\mathrm{d}s',\mathrm{d}a')$. Therefore,

$$\left|Q^{\pi_z}(s,a) - F(s,a,z)^\top z_R\right| = \left|\int r(s',a')\, \varepsilon_z(s,a,\mathrm{d}s',\mathrm{d}a')\right| \tag{94}$$

$$\leq \|r\|_A \|\varepsilon_z(s,a,\cdot)\|_B \tag{95}$$

for any reward r and any $z \in Z$ (not necessarily $z = z_R$).

Let Q^* be the optimal Q-function for reward r. Define $f(s,a) := F(s,a,z_R)^\top z_R$. By definition, the policy π_{z_R} is equal to $\arg\max_a f(s,a)$. Therefore, by Proposition 12,

$$\sup_{S\times A} |Q^{\pi_{z_R}} - Q^*| \leq \frac{3}{1-\gamma} \sup_{S\times A} |f - Q^{\pi_{z_R}}|\,. \tag{96}$$

and

$$\sup_{S\times A} |f - Q^*| \leq \frac{2}{1-\gamma} \sup_{S\times A} |f - Q^{\pi_{z_R}}|\,. \tag{97}$$

But by the above,

$$\sup_{S\times A} |f - Q^{\pi_{z_R}}| = \sup_{S\times A} \left|F(s,a,z_R)^\top z_R - Q^{\pi_{z_R}}(s,a)\right| \tag{98}$$

$$\leq \|r\|_A \sup_{S\times A} \|\varepsilon_{z_R}(s,a,\cdot)\|_B\,. \tag{99}$$

Therefore, for any reward function r,

$$\sup_{S\times A} |Q^{\pi_{z_R}} - Q^*| \leq \frac{3\|r\|_A}{1-\gamma} \sup_{S\times A} \|\varepsilon_{z_R}(s,a,\cdot)\|_B\,. \tag{100}$$

This inequality transfers to the value functions, hence the result. In addition, using again $f(s,a) = F(s,a,z_R)^\top z_R$, we obtain

$$\sup_{S\times A} \left|F(s,a,z_R)^\top z_R - Q^*(s,a)\right| \leq \frac{2\|r\|_A}{1-\gamma} \sup_{S\times A} \|\varepsilon_{z_R}(s,a,\cdot)\|_B\,. \tag{101}$$

□

Proof of Proposition 6. This is just a triangle inequality. By assumption, the difference between the value function of $\pi_{\hat{z}_R}$ and the optimal value function $V^*_{\hat{r}}$ is at most ε_{FB}. Then by Proposition 11, the difference between $V^*_{\hat{r}}$ and V^*_r is bounded by $\frac{1}{1-\gamma}\sup_{S\times A}|\hat{r} - r|$. □

Proof of Theorem 8. We proceed by building a reward function $\hat{r}$ corresponding to $\hat{z}_R$. Then we will bound $\hat{r} - r$ and apply Proposition 6.

First, by Remark 4, up to reducing d, we can assume that B is $L^2(\rho)$-orthonormal.

For any function $\varphi\colon (s,a) \to \mathbb{R}$, define $z_\varphi := \mathbb{E}_{(s,a)\sim\rho}[\varphi(s,a)B(s,a)]$. For each $z \in Z$, define φ_z via $\varphi_z(s,a) := B(s,a)^\top z$. Then, if B is $L^2(\rho)$-orthonormal, we have $z_{\varphi_z} = z$. (Indeed, $z_{\varphi_z} = \mathbb{E}[(B(s,a)^\top z)B(s,a)] = \mathbb{E}[B(s,a)(B(s,a)^\top z)] = \left(\mathbb{E}[B(s,a)B(s,a)^\top]\right) z$.)

Define the function

$$\hat{r} := r + \varphi_{\hat{z}_R - z_R} \tag{102}$$

using the functions φ_z defined above. By construction, $z_{\hat{r}} = z_R + z_{\varphi_{\hat{z}_R - z_R}} = \hat{z}_R$. Therefore, the policy $\pi_{\hat{z}_R}$ associated to $\hat{z}_R$ is the policy associated to the reward $\hat{r}$.

We will now apply Proposition 6 to r and $\hat{r}$. For this, we need to bound $\|\varphi_{\hat{z}_R - z_R}\|_\infty$.

Let $B_1, B_2, \ldots, B_d$ be the components of B as functions on $S \times A$. For any $z \in Z$, we have

$$\|\varphi_z\|^2_{L^2(\rho)} = \left\|\sum_i z_i B_i\right\|^2_{L^2(\rho)} = \sum_i z_i^2 = \|z\|^2 \tag{103}$$

since the B_i are $L^2(\rho)$-orthonormal. Moreover, by construction, φ_z lies in the linear span $\langle B \rangle$ of the functions (B_i). Therefore

$$\|\varphi_z\|_\infty \leq \zeta(B) \|\varphi_z\|_{L^2(\rho)} = \zeta(B) \|z\| \tag{104}$$

by the definition of $\zeta(B)$ (Definition 7).

Therefore,

$$\|\varphi_{\hat{z}_R - z_R}\|_\infty \leq \zeta(B) \|\hat{z}_R - z_R\|. \tag{105}$$

Let us now bound $\hat{z}_R - z_R$:

$$\mathbb{E}\left[\|\hat{z}_R - z_R\|^2\right] = \mathbb{E}\left[\mathbb{E}\left[\|\hat{z}_R - z_R\|^2 \mid (s_i, a_i)\right]\right] \tag{106}$$

$$= \mathbb{E}\left[\mathbb{E}\left[\|\hat{z}_R - \mathbb{E}[\hat{z}_R \mid (s_i, a_i)]\|^2 + \|\mathbb{E}[\hat{z}_R \mid (s_i, a_i)] - z_R\|^2 \mid (s_i, a_i)\right]\right] \tag{107}$$

$$= \mathbb{E}\left[\mathbb{E}\left[\left\|\frac{1}{N}\sum_i (\hat{r}_i - r(s_i, a_i))B(s_i, a_i)\right\|^2 + \left\|\frac{1}{N}\sum_i r(s_i, a_i)B(s_i, a_i) - z_R\right\|^2 \mid (s_i, a_i)\right]\right] \tag{108}$$

The first term satisfies

$$\mathbb{E}\left[\left\|\frac{1}{N}\sum_i (\hat{r}_i - r(s_i, a_i))B(s_i, a_i)\right\|^2 \mid (s_i, a_i)\right] = \frac{1}{N^2}\sum_i \mathbb{E}\left[(\hat{r}_i - r(s_i, a_i))^2 \|B(s_i, a_i)\|^2\right] \tag{109}$$

$$\leq \frac{1}{N^2}\sum_i v \|B(s_i, a_i)\|^2 \tag{110}$$

because the $\hat{r}_i$ are independent conditionally to (s_i, a_i), and because B is deterministic. The expectation of this over (s_i, a_i) is

$$\mathbb{E}\left[\frac{1}{N^2}\sum_i v \|B(s_i, a_i)\|^2\right] = \frac{v}{N}\,\mathbb{E}_{(s,a)\sim\rho} \|B(s, a)\|^2 = \frac{v}{N}\|B\|^2_{L^2(\rho)} \tag{111}$$

which is thus a bound on the first term.

The second term satisfies

$$\mathbb{E}\left[\left\|\frac{1}{N}\sum_i r(s_i, a_i)B(s_i, a_i) - z_R\right\|^2\right] = \frac{1}{N}\|r(s, a)B(s, a) - \mathbb{E}_\rho[r(s, a)B(s, a)]\|^2_{L^2(\rho)} \tag{112}$$

since the (s_i, a_i) are independent with distribution ρ. By the Cauchy–Schwarz inequality (applied to each component of B), this is at most

$$\frac{1}{N}\|r(s, a) - \mathbb{E}_\rho r\|^2_{L^2(\rho)} \|B\|^2_{L^2(\rho)}. \tag{113}$$

Therefore,

$$\mathbb{E}\left[\|\hat{z}_R - z_R\|^2\right] \leq \left(v + \|r(s, a) - \mathbb{E}_\rho r\|^2_{L^2(\rho)}\right)\frac{\|B\|^2_{L^2(\rho)}}{N}. \tag{114}$$

Since B is orthonormal in $L^2(\rho)$, we have $\|B\|^2_{L^2(\rho)} = d$. Putting everything together, we find

$$\mathbb{E}\left[\|\hat{r} - r\|^2_\infty\right] \leq \frac{\zeta(B)\,d}{N}\left(v + \|r(s,a) - \mathbb{E}_\rho\, r\|^2_{L^2(\rho)}\right). \tag{115}$$

Therefore, by the Markov inequality, for any $\delta > 0$, with probability $1 - \delta$,

$$\|\hat{r} - r\|_\infty \leq \sqrt{\frac{\zeta(B)\,d}{N\delta}\left(v + \|r(s,a) - \mathbb{E}_\rho\, r\|^2_{L^2(\rho)}\right)} \tag{116}$$

hence the conclusion by Proposition 6. □

Proof of Theorem 9. Let

$$m(s,a,s',a') := \sum_{t\geq 0} \gamma^t \frac{P_t(\mathrm{d}s',\mathrm{d}a'|s,a,\pi_z)}{\rho(\mathrm{d}s',\mathrm{d}a')} \tag{117}$$

so that

$$\ell(F,B) = \int \left|F(s,a,z)^\top B(s',a') - m(s,a,s',a')\right|^2 \rho(\mathrm{d}s,\mathrm{d}a)\rho(\mathrm{d}s',\mathrm{d}a'). \tag{118}$$

Let us first take the derivative with respect to the parameters of F. This is 0 by assumption, so we find

$$0 = \int \partial_\theta F(s,a,z)^\top B(s',a')\left(F(s,a,z)^\top B(s',a') - m(s,a,s',a')\right)\rho(\mathrm{d}s,\mathrm{d}a)\rho(\mathrm{d}s',\mathrm{d}a') \tag{119}$$

$$= \int \partial_\theta F(s,a,z)^\top G(s,a)\rho(\mathrm{d}s,\mathrm{d}a) \tag{120}$$

where

$$G(s,a) := \int B(s',a')\left(F(s,a,z)^\top B(s',a') - m(s,a,s',a')\right)\rho(\mathrm{d}s',\mathrm{d}a') \tag{121}$$

Since the model is overparameterized, we can realize any L^2 function $f(s,a)$ as the derivative $\partial_\theta F(s,a,z)$ for some direction θ. Therefore, the equation $0 = \int \partial_\theta F(s,a,z)^\top G(s,a)\rho(\mathrm{d}s,\mathrm{d}a)$ implies that $G(s,a)$ is $L^2(\rho)$-orthogonal to any function $f(s,a)$ in $L^2(\rho)$. Therefore, $G(s,a)$ vanishes ρ-almost everywhere, namely

$$\int B(s',a')F(s,a,z)^\top B(s',a')\rho(\mathrm{d}s',\mathrm{d}a') = \int B(s',a')m(s,a,s',a')\rho(\mathrm{d}s',\mathrm{d}a') \tag{122}$$

Now, since $F(s,a,z)^\top B(s',a')$ is a real number, $F(s,a,z)^\top B(s',a') = B(s',a')^\top F(s,a,z)$. Therefore, the right-hand-side above rewrites as

$$\int B(s',a')B(s',a')^\top F(s,a,z)\rho(\mathrm{d}s',\mathrm{d}a') = (\operatorname{Cov} B)F(s,a,z) \tag{123}$$

so that

$$(\operatorname{Cov} B)F(s,a,z) = \int B(s',a')m(s,a,s',a')\rho(\mathrm{d}s',\mathrm{d}a'). \tag{124}$$

Unfolding the definition of m yields the statement for F. The proof for B is similar. □

G Experimental Setup

In this section we provide additional information about our experiments.

G.1 Environments

- **Discrete maze:** is the 11×11 classical tabular gridworld with foor rooms. States are represented by one-hot unit vectors, $S = \{0, 1\}^{121}$. There are five available actions , $A = \{$`left`, `right`, `up`, `down`, `do nothing` $\}$. The dynamics are deterministic and the walls are impassable.
- **Continuous maze:** is a two dimensional environment with impassable walls. States are represented by their Cartesian coordinates $(x, y) \in S = [0, 1]^2$. There are five available actions, $A = \{$`left`, `right`, `up`, `down`, `do nothing` $\}$. The execution of one of the actions moves the agent 0.1 units in the desired direction, and normal random noise with zero mean and standard deviation 0.01 is added to the position of the agent (that is, a move along the x axis would be $x' = x \pm 0.1 + \mathcal{N}(0, 0.01)$, where $\mathcal{N}(0, 0.01)$ is a normal variable with mean 0 and standard deviation 0.01). If after a move the agent ends up outside of $[0, 1]^2$, the agent's position is clipped (e.g if $x < 0$ then we set $x = 0$). If a move make the agent cross an interior wall, this move is undone. For all algorithms, we convert a state $s = (x, y)$ into feature vector $\varphi(s) \in \mathbb{R}^{441}$ by computing the activations of a regular 21×21 grid of radial basis functions at the point (s, y). Especially, we use Gaussian functions: $\varphi(s) = \Big(\exp(-\frac{(x-x_i)^2+(y-y_i)^2}{\sigma}), \ldots, \exp(-\frac{(x-x_{441})^2+(y-y_{441})^2}{2\sigma^2})\Big)$ where (x_i, y_i) is the center of the i^{th} Gaussian and $\sigma = 0.05$.
- **FeatchReach:** is a variant of the simulated robotic arm environment from (Plappert et al., 2018) using discrete actions instead of continuous actions. States are 10-dimensional vectors consisting of positions and velocities of robot joints. We discretise the original 3-dimensional action space into 6 possible actions using action stepsize of 1 (The same way as in https://github.com/paulorauber/hpg, the implementation of hindsight policy gradient (Rauber et al., 2018)). The goal space is 3-dimensional space representing of the position of the object to reach.
- **Ms. Pacman:** is a variant of the Atari 2600 game Ms. Pacman, where an episode ends when the agent is captured by a monster (Rauber et al., 2018). States are obtained by processing the raw visual input directly from the screen. Frames are preprocessed by cropping, conversion to grayscale and downsampling to 84×84 pixels. A state s_t is the concatenation of $(x_{t-12}, x_{t-8}, x_{t-4}, x_t)$ frames, i.e. an $84 \times 84 \times 4$ tensor. An action repeat of 12 is used. As Ms. Pacman is not originally a multi-goal domain, we define the set of goals as the set of the 148 reachable coordinate pairs (x, y) on the screen; these can be reached only by learning to avoid monsters. In contrast with (Rauber et al., 2018), who use a heuristic to find the agent's position from the screen's pixels, we use the Atari annotated RAM interface wrapper (Anand et al., 2019).

G.2 Architectures

We use the same architecture for discrete maze, continuous maze and FeatchReach. Both forward and backward networks are represented by a feedforward neural network with three hidden layers, each with 256 ReLU units. The forward network receives a concatenation of a state and a z vector as input and has $|A| \times d$ as output dimension. The backward network receives a state as input (or gripper's position for FeatchReach) and has d as output dimension. For goal-oriented DQN, the Q-value network is also a feedforward neural network with three hidden layers, each with 256 ReLU units. It receives a concatenation of a state and a goal as input and has $|A|$ as output dimension.

For Ms. Pacman, the forward network is represented by a convolutional neural network given by a convolutional layer with 32 filters (8×8, stride 4); convolutional layer with 64 filters (4×4, stride 2); convolutional layer with 64 filters (3×3, stride 1); and three fully-connected layers, each with 256 units. We use ReLU as activation function. The z vector is concatenated with the output of the third convolutional layer. The output dimension of the final linear layer is $|A| \times d$. The backward network acts only on agent's position, a 2-dimensional input. It is represented by a feedforward neural network with three hidden layers, each with 256 ReLU units. The output dimension is d. For goal-oriented DQN, the Q-value network is represented by a convolutional neural network with the same architecture as the one of the forward network. The goal's position is concatenated with the output of the third convolutional layer. The output dimension of the final linear layer is $|A|$.

G.3 Implementation Details

For all environments, we run the algorithms for 800 epochs. Each epoch consists of 25 cycles where we interleave between gathering some amount of transitions, to add to the replay buffer $\mathcal{D}$ (old transitions are thrown when we reach the maximum of its size), and performing 40 steps of stochastic gradient descent on the model parameters. To collect transitions, we generate episodes using some behavior policy. For both mazes, we use a uniform policy while for FetchReach and Ms. Pacman, we use an ε-greedy policy ($\varepsilon = 0.2$) with respect to the current approximation $F(s, a, z)^\top z$ for a sampled z. At evaluation time, ε-greedy policies are also used, with a smaller $\varepsilon = 0.02$ for all envrionments except from discrete maze where we use Boltzmann policy with temperature $\tau = 1$. We train each models for three different random seeds.

For generality, we will keep using the notation $B(s, a)$ while in our experiments B acts only on $\varphi(s, a)$, a part of the state-action space. For discrete and continuous mazes, $\varphi(s, a) = s$, for FetchReach, $\varphi(s, a)$ the position of arm's gripper and for Ms. Pacman, $\varphi(s, a)$ is the 2-dimensional position (x, y) of the agent on the screen.

We denote by θ and ω the parameters of forward and backward networks respectively and θ^- and ω^- the parameters of their corresponding target networks. Both θ^- and ω^- are updated after each cycle using Polyak averaging; i.e $\theta^- \leftarrow \alpha\theta^- + (1-\alpha)\theta$ and $\omega^- \leftarrow \alpha\omega^- + (1-\alpha)\omega$ where $\alpha = 0.95$ is the Polyak coefficient.

During training, we sample z from a rescaled Gaussian that we denote ν. Especially, we sample a d-dimensional standard Gaussian variable $x \sim \mathcal{N}(0, \mathrm{Id}) \in \mathbb{R}^d$ and a scalar centered Cauchy variable $u \in \mathbb{R}$ of scale 0.5, then we set $z = \sqrt{d} u \frac{x}{\|x\|}$. We use a Cauchy distribution to ensure that the norm of z spans the non-negative real numbers space while having a heavy tail: with a pure Gaussian, the norm of z would be very concentrated around a single value. We also scale by $\sqrt{d}$ to ensure that each component of z has an order of magnitude of 1.

Before being fed to F, z is preprocessed by $z \leftarrow \frac{z}{\sqrt{1+\|z\|_2^2/d}}$; this way, z ranges over a bounded set in $\mathbb{R}^d$, and this takes advantage of optimal policies being equal for a reward R and for λR with $\lambda > 0$.

To update network parameters, we compute an empirical loss by sampling 3 mini-batches, each of size $b = 128$, of transitions $\{(s_i, a_i, s_{i+1})\}_{i \in I} \subset \mathcal{D}$, of target state-action pairs $\{(s'_i, a'_i)\}_{i \in I} \subset \mathcal{D}$ and of $\{z_i\}_{i \in I} \sim \nu$:

$$\begin{aligned}\mathscr{L}(\theta, \omega) = \frac{1}{2b^2} \sum_{i,j \in I^2} \left(F_\theta(s_i, a_i, z_i)^\top B_\omega(s'_j, a'_j) - \gamma \sum_{a \in A} \pi_{z_i}(a \mid s_{i+1}) \cdot F_{\theta^-}(s_{i+1}, a, z_i)^\top B_{\omega^-}(s'_j, a'_j) \right)^2 \\ - \frac{1}{b} \sum_{i \in I} F_\theta(s_i, a_i, z_i)^\top B_\omega(s_i, a_i) \qquad (125)\end{aligned}$$

where we use the Boltzmann policy $\pi_{z_i}(\cdot \mid s_{i+1}) = \texttt{softmax}(F_{\theta^-}(s_{i+1}, \cdot, z_i)^\top z_i / \tau)$ with fixed temperature $\tau = 200$ to avoid the instability and discontinuity caused by the argmax operator.

Since there is unidentifiability between F and B (Appendix, Remark 4), we include a gradient to make B closer to orthonormal, $\mathbb{E}_{(s,a)\sim\rho} B(s,a)B(s,a)^\top \approx \mathrm{Id}$:

$$\frac{1}{4}\partial_\omega \left\|\mathbb{E}_{(s,a)\sim\rho} B_\omega(s,a)B_\omega(s,a)^\top - \mathrm{Id}\right\|^2 = \\ \mathbb{E}_{(s,a)\sim\rho,(s',a')\sim\rho}\, \partial_\omega B_\omega(s,a)^\top \left(B_\omega(s,a)^\top B_\omega(s',a') \cdot B_\omega(s',a') - B(s,a)\right) \quad (126)$$

To compute an unbiased estimate of the latter gradient, we use the following auxiliary empirical loss:

$$\mathscr{L}_{\texttt{reg}}(\omega) = \frac{1}{b^2}\sum_{i,j\in I^2} B_\omega(s_i,a_i)^\top \texttt{stop-gradient}(B_\omega(s'_j,a'_j)) \cdot \texttt{stop-gradient}(B_\omega(s_i,a_i)^\top B_\omega(s'_j,a'_j)) \\ - \frac{1}{b}\sum_{i\in I} B_\omega(s_i,a_i)^\top \texttt{stop-gradient}(B_\omega(s_i,a_i)) \quad (127)$$

Finally, we use the Adam optimizer and we update θ and ω by taking a gradient step on $\mathscr{L}(\theta,\omega)$ and $\mathscr{L}(\theta,\omega) + \lambda \cdot \mathscr{L}_{\texttt{reg}}(\omega)$ respectively, where λ is a regularization coefficient that we set to 1 for all experiments.

We summarize the hyperparameters used for FB algorithm and goal-oriented DQN in table 1 and 2 respectively.

Hyperparameters	Discrete Maze	Continuous Maze	FetchReach	Ms. Pacman
number of cycles per epoch	25	25	25	25
number of episodes per cycles	4	4	2	2
number of timesteps per episode	50	30	50	50
number of updates per cycle	40	40	40	40
exploration ε	1	1	0.2	0.2
evaluation ε	Boltzman with $\tau = 1$	0.02	0.02	0.02
temperature τ	200	200	200	200
learning rate	0.001	0.0005	0.0005	0.0001 if $d = 100$, else 0.0005
mini-batch size	128	128	128	128
regularization coefficient λ	1	1	1	1
Polyak coefficient α	0.95	0.95	0.95	0.95
discount factor γ	0.99	0.99	0.9	0.9
replay buffer size	10^6	10^6	10^6	10^6

Table 1: Hyperparameters of the FB algorithm

Hyperparameters	Discrete Maze	Continuous Maze	FetchReach	Ms. Pacman
number of cycles per epoch	25	25	25	25
number of episodes per cycles	4	4	2	2
number of timesteps per episode	50	30	50	50
number of updates per cycle	40	40	40	40
exploration ε	0.2	0.2	0.2	0.2
evaluation ε	Boltzman with $\tau = 1$	0.02	0.02	0.02
learning rate	0.001	0.0005	0.0005	0.0005
mini-batch size	128	128	128	128
Polyak coefficient α	0.95	0.95	0.95	0.95
discount factor γ	0.99	0.99	0.9	0.9
replay buffer size	10^6	10^6	10^6	10^6
ratio of hindsight replay	-	-	-	0.8

Table 2: Hyperparameters of the goal-oriented DQN algorithm

G.4 Experimental results

In this section, we provide additional experimental results.

G.4.1 Goal-Oriented Setup: Quantitative Comparisons

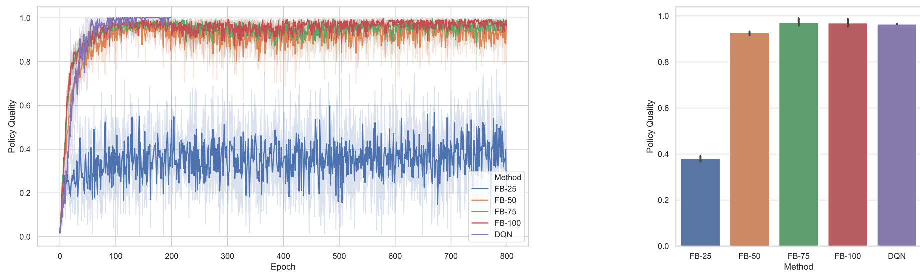

Figure 7: **Discrete maze**: Comparative performance of FB for different dimensions and DQN. **Left**: the policy quality averaged over 20 randomly selected goals as function of the training epochs. **Right**: the policy quality averaged over the goal space after 800 training epochs.

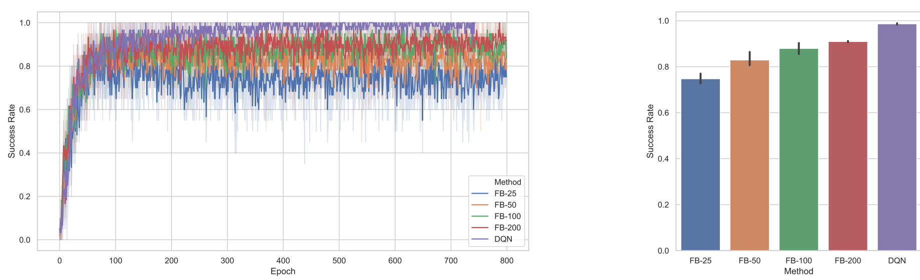

Figure 8: **Continuous maze:** Comparative performance of FB for different dimensions and DQN. **Left**: the success rate averaged over 20 randomly selected goals as function of the training epochs. **Right**: the success rate averaged over 1000 randomly sampled goals after 800 training epochs.

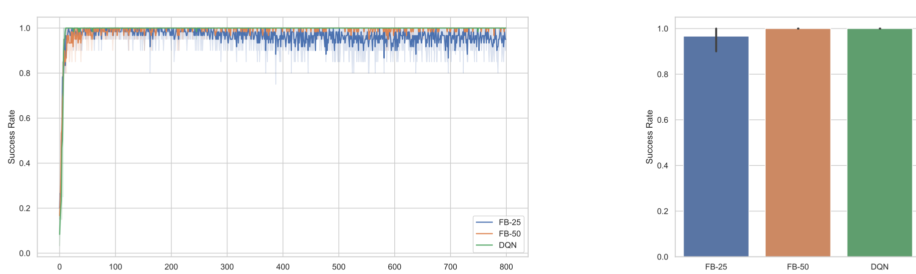

Figure 9: **FetchReach**: Comparative performance of FB for different dimensions and DQN. **Left**: the success rate averaged over 20 randomly selected goals as function of the training epochs. **Right**: the success rate averaged over 1000 randomly sampled goals after 800 training epochs.

G.4.2 More Complex Rewards: Qualitative Results

G.4.3 Embedding Visualization

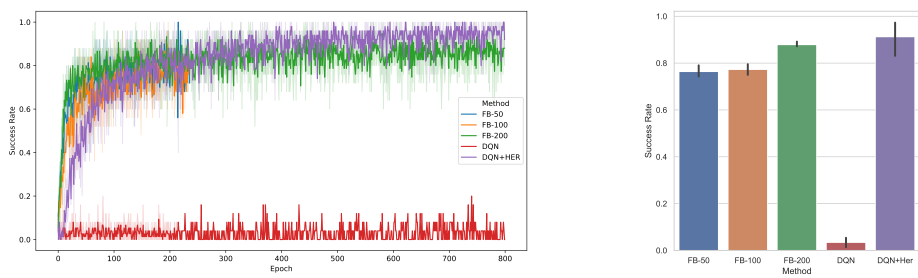

Figure 10: **Ms. Pacman**: Comparative performance of FB for different dimensions and DQN. **Left**: the success rate averaged over 20 randomly selected goals as function of the training epochs. **Right**: the success rate averaged over the 184 handcrafted goals after training epochs. Note that FB-50 and F-100 have been trained only for 200 epochs.

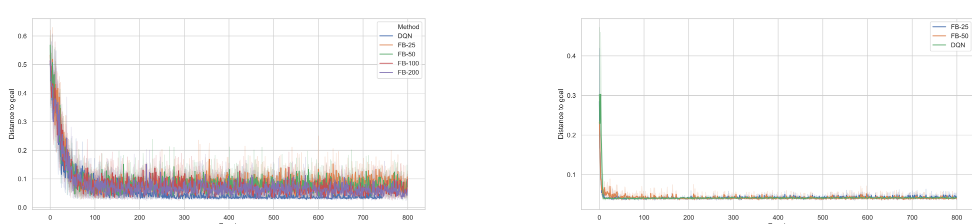

Figure 11: Distance to goal of FB for different dimensions and DQN as function of training epochs. **Left**: Continuous maze. **Right**: FetchReach.

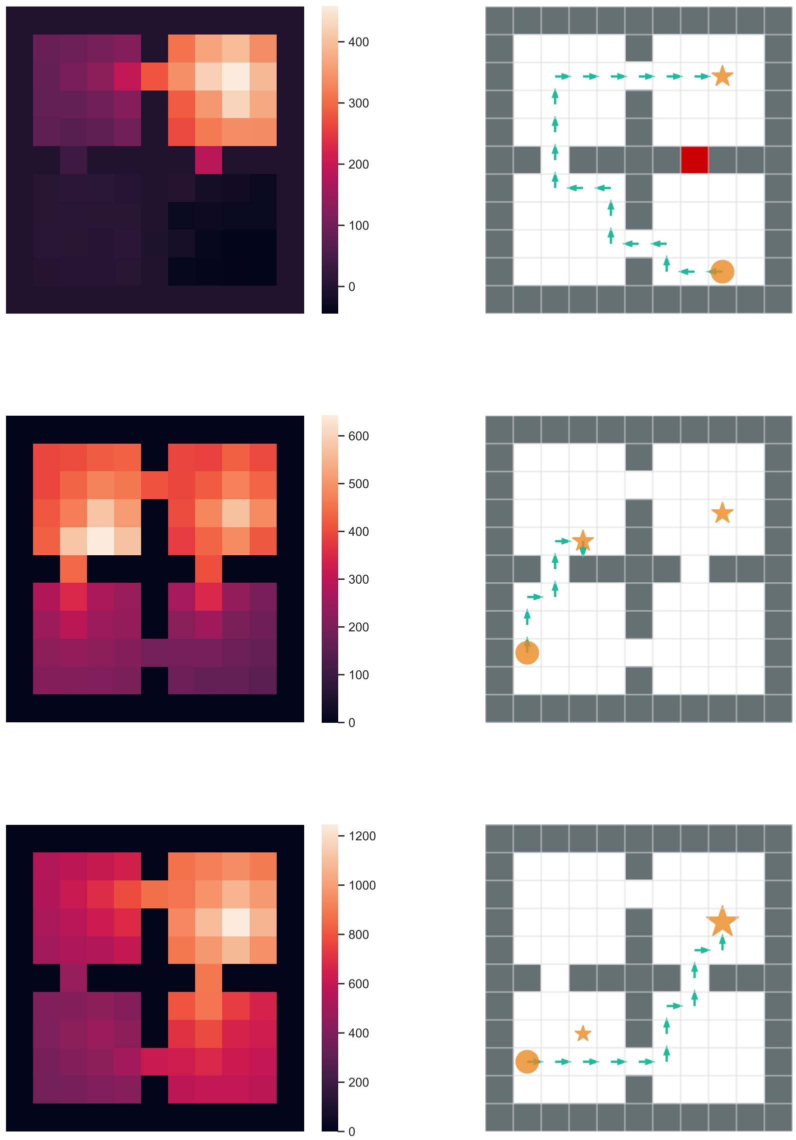

Figure 12: **Discrete Maze**: Heatmap plots of $\max_{a \in A} F(s,a,z_R)^\top z_R$ (left) and trajectories of the Boltzmann policy with respect to $F(s,a,z_R)^\top z_R$ with temperature $\tau = 1$ (right). **Top row**: for the task of reaching a target while avoiding a forbidden region, **Middle row**: for the task of reaching the closest goal among two equally rewarding positions, **Bottom row**: choosing between a small, close reward and a large, distant one.

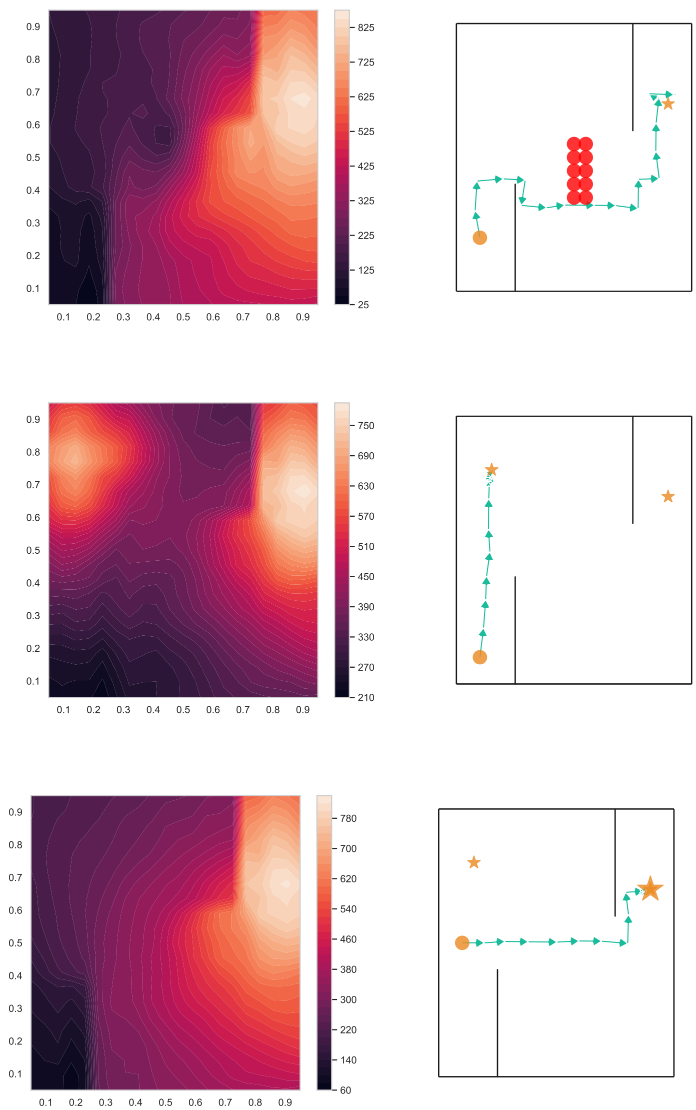

Figure 13: **Continuous Maze**: Contour plots plot of $\max_{a \in A} F(s,a,z_R)^\top z_R$ (left) and trajectories of the ε greedy policy with respect to $F(s,a,z_R)^\top z_R$ with $\varepsilon = 0.1$ (right). **Left**: for the task of reaching a target while avoiding a forbidden region, **Middle**: for the task of reaching the closest goal among two equally rewarding positions, **Right**: choosing between a small, close reward and a large, distant one..

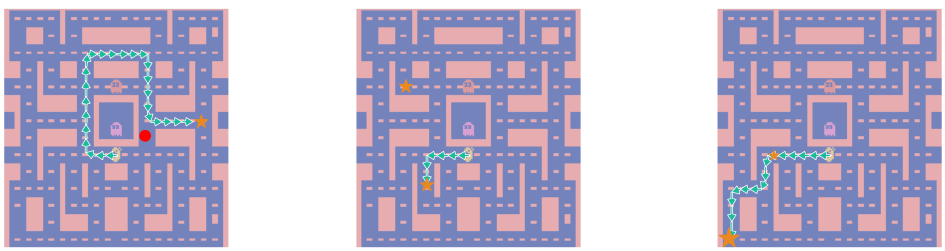

Figure 14: **Ms. Pacman**: Trajectories of the ε greedy policy with respect to $F(s, a, z_R)^\top z_R$ with $\varepsilon = 0.1$ (right). **Top row**: for the task of reaching a target while avoiding a forbidden region, **Middle row**: for the task of reaching the closest goal among two equally rewarding positions, **Bottom row**: choosing between a small, close reward and a large, distant one..

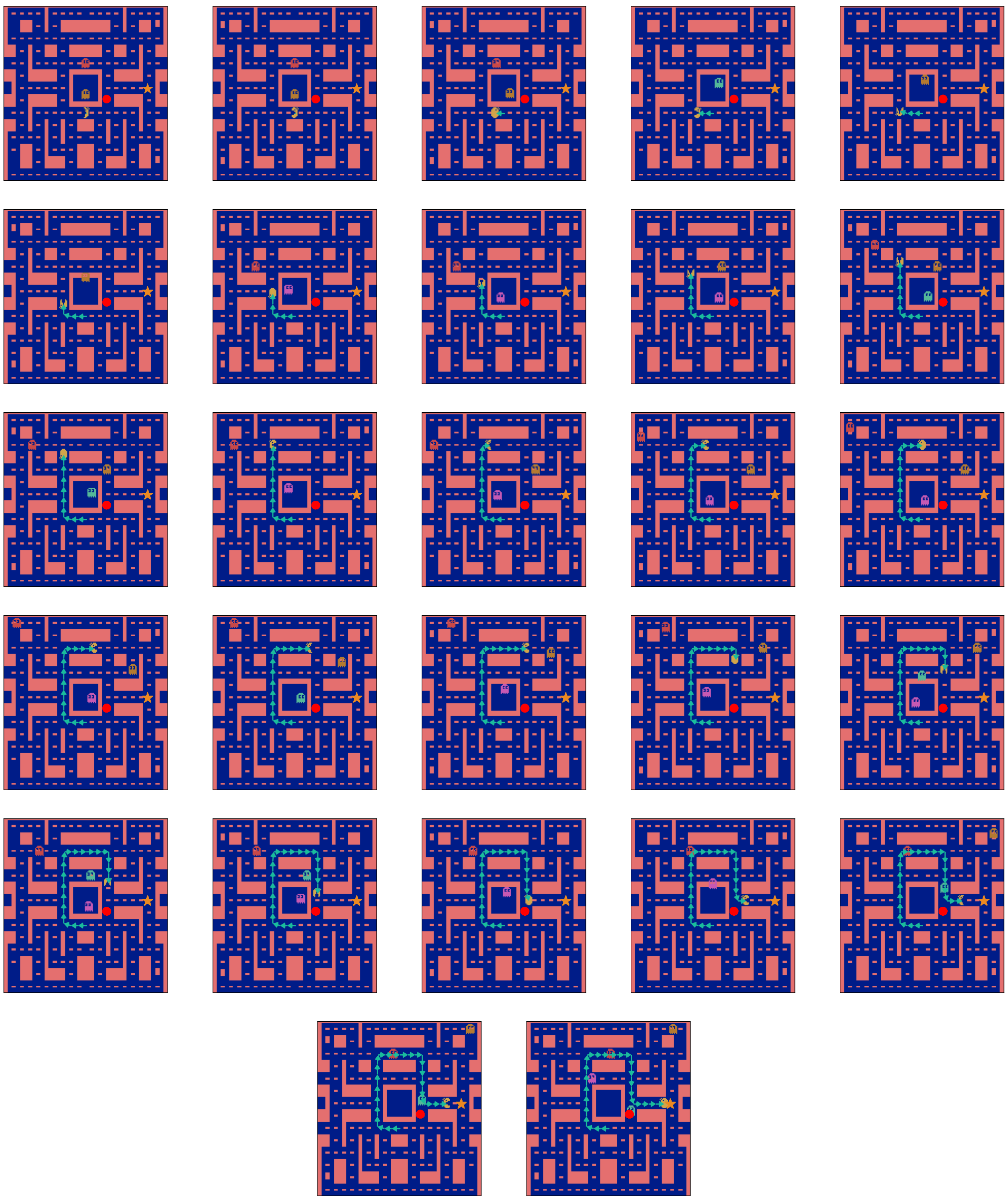

Figure 15: Full series of frames in Ms. Pacman along the trajectory generated by the $F^{\top}B$ policy for the task of reaching a target position (star shape ☆) while avoiding forbidden positions (red shape ●).

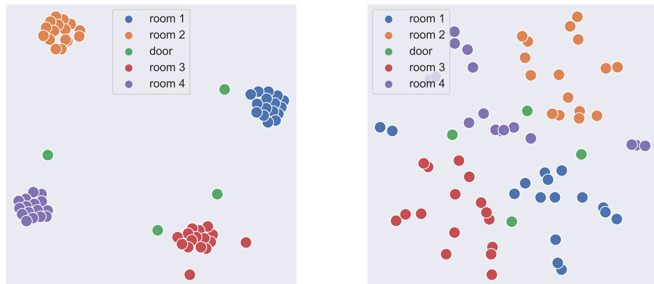

Figure 16: **Discrete maze**: Visualization of FB embedding vectors after projecting them in two-dimensional space with t-SNE. **Left**: the F embedding for $z = 0$. **Right**: the B embedding. Note how both embeddings recover the foor-room and door structure of the original environment. The spread of B embedding is due to the regularization that makes B closer to orthonormal.

Figure 17: **Continuous maze**: Visualization of FB embedding vectors after projecting them in two-dimensional space with t-SNE. **Left**: the states to be mapped. **Middle**: the F embedding. **Right**: the B embedding.

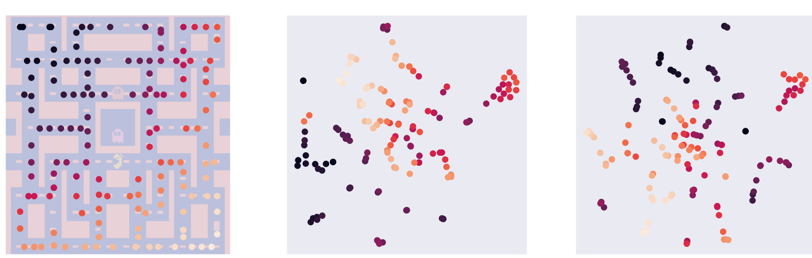

Figure 18: **Ms. Pacman**: Visualization of FB embedding vectors after projecting them in two-dimensional space with t-SNE. **Left**: the agent's position corresponding to the state to be mapped. **Middle**: the F embedding for $z = 0$. **Right**: the B embedding. Note how both embeddings recover the cycle structure of the environment. F acts on visual inputs and B acts on the agent's position.

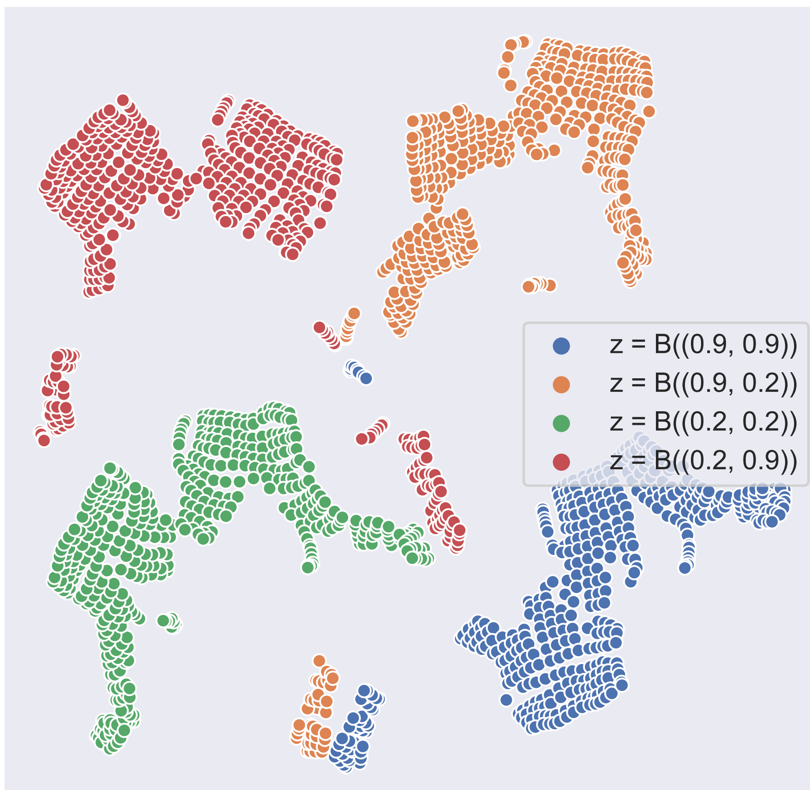

Figure 19: **Continuous maze**: visualization of F embedding vectors for different z vectors, after projecting them in two-dimensional space with t-SNE.