Revisiting Successor Features for Inverse Reinforcement Learning

Arnav Kumar Jain [1 2] **Harley Wiltzer** [1 3] **Jesse Farebrother** [1 3] **Irina Rish** [1 2] **Glen Berseth** [1 2]
Sanjiban Choudhury [4]

Abstract

In inverse reinforcement learning (IRL), an agent seeks to replicate expert demonstrations through interactions with the environment. Traditionally, IRL is treated as an adversarial game, where an adversary searches over reward models, and a learner optimizes the reward through repeated RL procedures. This game-solving approach is both computationally expensive and difficult to stabilize. Instead, we embrace a more fundamental perspective of IRL as that of state-occupancy matching: by matching the cumulative state features encountered by the expert, the agent can match the returns of the expert under any reward function in a hypothesis class. We present a simple yet novel framework for IRL where a policy greedily matches successor features of the expert where successor features efficiently compute the expected features of successive states observed by the agent. Our non-adversarial method does not require learning a reward function and can be solved seamlessly with existing value-based reinforcement learning algorithms. Remarkably, our approach works in state-only settings without expert action labels, a setting which behavior cloning (BC) cannot solve. Empirical results demonstrate that our method learns from as few as a single expert demonstration and achieves comparable performance on various control tasks.

1. Introduction

The primary objective in imitation learning (Abbeel & Ng, 2004; Silver et al., 2016; Ziebart et al., 2008; Swamy et al., 2021; Ho & Ermon, 2016) is to allow agents to reproduce behaviors by observing demonstrations from an expert. Interactive approaches like Inverse Reinforcement Learning (IRL; Abbeel & Ng, 2004; Ziebart et al., 2008) have led to agents that are more robust at recovering from their own mistakes in the environment. These methods show promising results when learning with a few expert demonstrations, and have been deployed in various real-world applications like autonomous-driving (Bronstein et al., 2022; Vinitsky et al., 2022; Igl et al.).

The problem of imitating a demonstrator can be fundamentally viewed as that of matching the state occupancy distribution between the imitating agent and the expert. While, in theory, distribution matching can replicate the exact behavior regardless of the reward function (Abbeel & Ng, 2004), estimating the occupancy distribution of a policy is difficult in continuous or large state spaces. Instead, many works consider matching moments of the distributions (Swamy et al., 2021), where the moments are chosen from a class of all possible reward functions. This leads to the following modern formulation of many IRL methods as a min-max game between an adversary that learns a reward function to maximally differentiate between the agent and expert; and a reinforcement learning (RL) subroutine that maximizes the adversarial reward. However, all such methods encounter a set of well-documented challenges: (1) optimizing an adversarial game between the agent and the expert can be unstable, often requiring multiple tricks to stabilize training (Swamy et al., 2022), (2) the inner loop involves repeatedly solving a computationally expensive RL problem (Swamy et al., 2023), and (3) the reward function class must be specified in advance. Motivated by these challenges, this leads us to the following research question:

Can a non-adversarial approach for occupancy matching recover the expert's behavior?

To address this question, we revisit feature-matching — a generalization of the distribution-matching formulation of this problem (Ziebart et al., 2008; Abbeel & Ng, 2004; Syed & Schapire, 2007; Syed et al., 2008) — with two key insights. First, rather than estimating the expected cumulative sum of features through Monte Carlo rollouts, we achieve a low-variance, fully online algorithm by employing temporal-difference-based methods introduced with Successor Features (SF Barreto et al., 2017). Additionally, we leverage recent work on learning rich domain-independent base fea-

[1]Mila- Quebec AI Institute [2]Université de Montréal [3]McGill University [4]Cornell University. Correspondence to: Arnav <arnav-kumar.jain@mila.quebec>.

ICML 2024 Workshop on Models of Human Feedback for AI Alignment, Vienna, Austria.

tures (e.g., Touati & Ollivier, 2021; Farebrother et al., 2023; Park et al., 2024; Gomez et al., 2024) to simultaneously learn state features alongside their corresponding successor features, lifting one of the primary limitations of feature matching. ***These two insights enable the development of a novel non-adversarial policy-gradient method that directly minimizes the feature gap, i.e., the expected deviation in successor features of the imitation agent and the expert.*** Specifically, the imitation agent greedily updates its decision policy to minimize the feature gap from any source state experienced by the expert — notably only requiring access to states, not the expert's actions. Our method of Successor Feature Matching (SFM) performs on par with state-of-the-art adversarial methods eliminating the need to solve a bi-level optimization problem while being both conceptually simpler and easier to optimize.

To summarize, our work makes the following contributions:

1. We introduce Successor Feature Matching (SFM), a novel algorithm for imitating expert behavior by learning state features and their corresponding successor features for an imitation policy trained to match the empirical successor features of an expert demonstrator.

2. We present a novel architecture for SFM and also implemented a state-only version of MM with an updated RL subroutine to have a more competitive baseline.

3. We demonstrate the efficacy of SFM from as few as a single expert demonstration across various Mujoco environment (Todorov et al., 2012) using expert demonstrations from the D4RL dataset (Fu et al., 2020). Notably, SFM, a non-adversarial algorithm can match the state-of-the-art performance of moment-matching methods in this challenging benchmark.

2. Related Work

Inverse Reinforcement Learning (IRL) methods typically combine expected feature matching with adversarial game dynamics, assuming the base features are known upfront (Abbeel & Ng, 2004; Ziebart et al., 2008; Syed & Schapire, 2007; Syed et al., 2008). The advent of modern deep learning architectures led to methods (e.g. Ho & Ermon, 2016; Swamy et al., 2021; Fu et al., 2018) that do not estimate expected features, but instead learn a more expressive reward function that captures the differences between the expert and the the agent. The class of Moment Matching (MM; Swamy et al., 2021) methods offers a general framework that unifies existing algorithms through the concept of moment matching, or equivalently Integral Probability Metrics (IPM; Sun et al., 2019). In contrast to these methods, our approach is non-adversarial and focuses on directly addressing the problem of matching expected features. Furthermore, unlike prior methods in Apprenticeship Learning (AL; Abbeel & Ng, 2004) and Maximum Entropy IRL (Ziebart et al., 2008), our work *does not* assume the knowledge of base features. Instead, SFM leverages representation learning technique to extract relevant features from the raw observations. The method most similar to ours is IQ-Learn (Garg et al., 2021), a non-adversarial approach that utilizes an inverse Bellman operator to directly estimate the value function of the expert. Our method is also non-adversarial, but offers a significant advantage over IQ-Learn: it does not require knowledge of expert actions during training. Methods that solve this more challenging task of imitation without expert action labels are called state-only IL (Torabi et al., 2019) algorithms. However, many existing state-only methods also rely on adversarial approaches (Torabi et al., 2018; Zhu et al., 2020). For instance, GAILfO (Torabi et al., 2018) modifies the discriminator to account for state-only inputs. In our work, to ensure a fair comparison with SFM, we similarly imbue recent moment matching methods (Swamy et al., 2021) with a state-only discriminator along with further modifications to the RL inner loop, resulting in significantly improved performance.

Successor Features (SF; Barreto et al., 2017) generalize the idea of the successor representation (SR) (Dayan, 1993) by modeling the expected cumulative state features discounted according to the time of state visitation. Instead of employing SFs for tasks such as transfer learning (Barreto et al., 2017; Lehnert et al., 2017; Ma et al., 2018; Barreto et al., 2018; Borsa et al., 2019; Abdolshah et al., 2021), representation learning (Le Lan et al., 2022; 2023a; Farebrother et al., 2023; Ghosh et al., 2023; Le Lan et al., 2023b), exploration (Zhang et al., 2017; Machado et al., 2020; Jain et al., 2024), or zero-shot RL (Touati et al., 2023; Park et al., 2024), our approach harnesses SFs for Inverse Reinforcement Learning (IRL), aiming to match expected features of the expert. Within the body of work on imitation learning SFs have been used within the adversarial framing of IRL typically serving as the basis for estimating the value-function that best explains the expert (Lee et al., 2019; Filos et al., 2021; Abdulhai et al., 2022). In contrast, our work instead seeks to directly match SFs through a policy-gradient update rule without requiring any bi-level optimization.

3. Background

Reinforcement Learning (RL; Sutton & Barto, 2018) typically considers a Markov Decision Process (MDP) defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, P_0)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition kernel, $r : \mathcal{S} \times \mathcal{A} \to [-1, 1]$ is the reward function, γ is the discount factor, and P_0 is the initial state distribution. Starting from the initial state $s_0 \sim P_0$ an agent takes actions according to its policy

$\pi : \mathcal{S} \to \Delta(\mathcal{A})$ producing trajectories $\tau = \{s_0, a_1, s_1, \dots\}$. The value function and action-value are respectively defined as $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r(S_t, A_t) | S_0 = s]$ and $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r(S_t, A_t) | S_0 = s, A_0 = a]$ where $\gamma \in [0, 1)$ represents the discount factor. The performance is the expected return obtained by following policy π from the initial state, given by $J(\pi) = \mathbb{E}_{s_0 \sim P_0}[\mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r(S_t, A_t) \,|\, S_0 = s]]$, and can be rewritten as $J(\pi) = \mathbb{E}_{s_0 \sim P_0}[V^\pi(s_0)]$.

The Successor Representation (SR; Dayan, 1993) provides the expected occupancy of future states for a given policy. For tabular state spaces, temporal-difference learning can be employed to estimate the SR. Successor Features (SF) (Barreto et al., 2017) generalize the idea of successor representation (SR) (Dayan, 1993) by instead counting the discounted sum of state features $\boldsymbol{\psi}^\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t \phi(S_t, A_t) | S_0 = s, A_0 = a]$ after applying the feature mapping $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$. The SR is recovered when ϕ is the identity function with ϕ typically serving as a form of dimensionality reduction to learn SFs in continuous or large state spaces. For tasks where the reward function can be expressed as a combination of base features ϕ and a weight function $w \in \mathbb{R}^d$ such that $r(s,a) = \phi(s,a)^\top w$, the performance of a policy π can be rewritten as $Q^\pi(s,a) = \boldsymbol{\psi}^\pi(s,a)^\top w$ and $J(\pi) = \mathbb{E}_{s_0 \sim P_0, a \sim \pi(s_0)}[\boldsymbol{\psi}^\pi(s_0, a)]^\top w$ (Barreto et al., 2017).

Inverse Reinforcement Learning (IRL; Ng et al., 2000; Abbeel & Ng, 2004; Ziebart et al., 2008) is the task of deriving behaviors using demonstrations through interacting with the environment. In contrast to RL where the agent improves its performance using the earned reward, IRL deals with learning without access to the reward function. As highlighted in (Swamy et al., 2021), this corresponds to minimizing an Integral Probability Metric (IPM) (Sun et al., 2019) between the agent's state-visitation occupancy and the expert's which can be framed as a task to minimize the imitation gap given by:

$$J(\pi_E) - J(\pi) \leq \sup_{f \in \mathcal{F}_\phi} \mathop{\mathbb{E}}_{\tau \sim \pi} \sum_{t=0}^\infty \gamma^t f(s_t, a_t) - \mathop{\mathbb{E}}_{\tau \sim \pi_E} \sum_{t=0}^\infty \gamma^t f(s_t, a_t)$$

where $\mathcal{F}_r : \mathcal{S} \times \mathcal{A} \to [-1, 1]$ denotes the class of reward basis functions. Under this taxonomy, the agent being the minimization player selects a policy $\pi \in \Pi$ to compete with a discriminator that picks a reward moment function $f \in \mathcal{F}_r$ to maximize the imitation gap, and this min-max game is framed as $\min_\pi \max_{f \in \mathcal{F}} J(\pi_E) - J(\pi)$. Note that, we use the the reward moment matching framework because as they are achieve performance gap that is linear with the horizon length (Swamy et al., 2021).

By restricting the class of reward basis functions to be within span of some base-features ϕ of state-action pairs such that $\mathcal{F}_\phi \in \{f(s,a) = \phi(s,a)^\top w_f\}$, the imitation gap becomes:

$$J(\pi_E) - J(\pi) \tag{1}$$

$$\leq \sup_{f \in \mathcal{F}_\phi} \mathop{\mathbb{E}}_{\tau \sim \pi} \sum_{t=0}^\infty \gamma^t \phi(s_t, a_t)^\top w_f - \mathop{\mathbb{E}}_{\tau \sim \pi_E} \sum_{t=0}^\infty \gamma^t \phi(s_t, a_t)^\top w_f \tag{2}$$

$$= \sup_{f \in \mathcal{F}_\phi} \left[\mathop{\mathbb{E}}_{\tau \sim \pi} \sum_{t=0}^\infty \gamma^t \phi(s_t, a_t) - \mathop{\mathbb{E}}_{\tau \sim \pi} \sum_{t=0}^\infty \gamma^t \phi(s_t, a_t) \right]^\top w_f \tag{3}$$

$$= \sup_{f \in \mathcal{F}_\phi} \left(\mathop{\mathbb{E}}_{s \sim P_0, a \sim \pi} [\boldsymbol{\psi}^\pi(s,a)] - \mathop{\mathbb{E}}_{s \sim P_0, a \sim \pi_E} [\boldsymbol{\psi}^E(s,a)] \right)^\top w_f, \tag{4}$$

where $\boldsymbol{\psi}^E(s,a)$ denotes the successor features (SF) of the expert policy π_E for a given state s and action a. Under this assumption, the agent that matches the SF with the expert will minimize the performance gap across the class of restricted basis reward functions. Solving this objective of matching expected features between the agent and the expert has been studied in prior methods where prior methods have often resorted to an adversarial game (Ziebart et al., 2008; Abbeel & Ng, 2004; Syed & Schapire, 2007; Syed et al., 2008). In the following section, we introduce a non-adversarial approach that updates the policy greedily to align the SFs between the expert and the agent, rather than learning a reward function to capture their behavioral divergence.

4. Successor Feature Matching (SFM)

Consider the setting where the agent has access to the expert demonstrations and the online environment but cannot query the expert policy. To learn a policy, we need an algorithm that can match the expected features between the expert demonstrations and the agent rollouts. We observed in Equation 4 that the IRL task can be solved using the objective of matching SF conditioned on the initial state distribution. In this regard, we define a utility function defined as the Mean Squared Error (MSE) between the expected features of the expert and the agent, given by:

$$U_\psi = \|\hat{\boldsymbol{\psi}}^\pi - \hat{\boldsymbol{\psi}}^E\|_2^2, \tag{5}$$

where $\hat{\boldsymbol{\psi}}^\pi = \mathbb{E}_{s \sim P_0, a \sim \pi(s)}[\boldsymbol{\psi}^\pi(s,a)]$ and $\hat{\boldsymbol{\psi}}^E = \mathbb{E}_{s \sim P_0, a \sim \pi_E(s)}[\boldsymbol{\psi}^E(s,a)]$ represents the expected SF of agent and expert conditioned on the initial state distribution P_0.

Since the agent is provided with expert demonstrations and cannot query the expert policy, the SF for the expert with M demonstrations $\{\tau^i = \{s_0^i, a_0^i, ..., s_{T-1}^i, a_{T-1}^i\}\}_{i=1}^M$ is

obtained using:

$$\hat{\boldsymbol{\psi}}^E = \frac{1}{M}\sum_{i=1}^{M}\sum_{t=0}^{T-1}\gamma^t\phi(s_t^i, a_t^i), \tag{6}$$

where T represents the horizon length. For a given base feature function ϕ, the SF for the expert policy is fixed and the utility function defined in Equation 5 is a convex and non-linear function with respect to the SF of the agent (Zahavy et al., 2021). Note that, the utility function defined in Equation 5 is contingent on the SF which changes with the policy π during training. To learn policies without an adversarial game for this task, we propose to optimize this non-linear objective where our method leverages the prowess of the Deterministic Policy Gradient (DPG) (Silver et al., 2014) algorithm.

Taking inspiration from off-policy actor-critic methods for standard RL tasks (Fujimoto et al., 2018; 2023; Haarnoja et al., 2018), SFM maintains a deterministic actor and a network to estimate the SF for agent's policy. Here, instead of having a critic to estimate the expected returns, the agent has a network to predict the discounted expected features. The network to predict SF of the agent is a parameterized and differentiable function with parameters θ and is denoted by $\boldsymbol{\psi}_\theta$. To obtain actions for a given state, SFM maintains a deterministic actor through a parameterized function π_μ with parameters μ.

Given an approximation $\boldsymbol{\psi}_\theta$, the SF network is updated using 1-step temporal difference (TD)-error. Here, the target value is obtained using the base feature of the current state obtained via the base feature function ϕ and the SF of the action taken at the next state, which is denoted by:

$$\boldsymbol{y}(s,a) = \phi(s) + \gamma\boldsymbol{\psi}_{\bar{\theta}}(s', \pi_\mu(s')) \tag{7}$$

where $s, s' \in \mathcal{S}$ represent the current and next state, $\bar{\theta}$ are the parameters of the target SF network, and the SF at the the next state is obtained using the current policy π_μ. Notably, we assume here that the base feature function only depends on the state information and does not depend on the state-action pair. Later in this section, we will emphasize on this assumption in Proposition 4.2 and further demonstrate how this leads to a state-only IRL algorithm. The parameters of the SF network are optimized using the MSE loss between the current SF estimate and the target $\boldsymbol{y}$ to get:

$$\mathcal{L}_{SF} = \|\boldsymbol{\psi}_\theta(s,a) - \boldsymbol{y}(s,a)\|_2^2, \tag{8}$$

where $a \in \mathcal{A}$ and the gradients are not propagated through the target network. The SF network updates itself iteratively via bootstrapping and keeps track of this quantity for the current policy that changes during the training phase. To update the actor network π_μ, we first show how SFM estimates the SF of the current policy conditioned on the initial state distribution $\hat{\boldsymbol{\psi}}_\theta$.

Proposition 4.1. *Conditioned on a given initial state distribution P_0, the following holds:*

$$(1-\gamma)\mathbb{E}_{s\sim P_0}[\boldsymbol{\psi}^\pi(s,\pi(s))] \tag{9}$$
$$= \mathbb{E}_{s,s'\sim\mathcal{B}}[\boldsymbol{\psi}^\pi(s,\pi(s)) - \gamma\boldsymbol{\psi}^\pi(s',\pi(s'))], \tag{10}$$

where $\mathcal{B}$ is the replay buffer and π is a deterministic policy.

The proof of Proposition 4.1 is deferred to the Appendix A. Proposition 4.1 presents a way to estimate the SF for the agent conditioned on the initial state distribution. The proposed derivation can utilize samples coming from a different state-visitation distribution and uses an off-policy replay buffer in this work. Similar to standard off-policy RL algorithms (Fujimoto et al., 2018; 2023; Haarnoja et al., 2018), SFM maintains a replay buffer $\mathcal{B}$ to store the transitions and use it for sampling. Note that this proposition is applicable only when the base features depend on the state and not on the action sampled from the buffer.

Note that the utility function defined in Equation 5 depends only on the initial state distribution and does not specify a way of updating the actor for any other state. By plugging Equation 9 in Equation 5, we redefine the utility function to represent the gap between the expected features of the agent and the expert, which can depend on a state which is outside the support of initial state distribution. Thus, we define the loss for the actor called SF-Gap loss ($\mathcal{L}_G$) obtained using:

$$\mathcal{L}_G = \left\| \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{s,s'\sim\mathcal{B}}[\boldsymbol{\psi}_\theta^{\pi_\mu}(s) - \gamma\boldsymbol{\psi}_{\bar{\theta}}^{\pi_\mu}(s')] - \hat{\boldsymbol{\psi}}^E \right\|_2^2, \tag{11}$$

where $\boldsymbol{\psi}_\theta^\pi(s) = \boldsymbol{\psi}_\theta(s,\pi(s))$, and where we use the target network $\boldsymbol{\psi}_{\bar{\theta}}$ to get SF at the next state. We can see that Equation 11 approximates the utility function on states sampled from the replay buffer $\mathcal{B}$. To obtain the gradients with respect to the actor parameters μ, we propose using the Deterministic Policy Gradient (DPG) algorithm (Silver et al., 2014) that estimates the gradients by applying the chain-rule over Equation 11.

Proposition 4.2. *The gradients of the actor for a batch of sampled transitions from the replay buffer obtained by applying the DPG (Silver et al., 2014) algorithm to Equation 11 is:*

$$\nabla_\mu\mathcal{L}_G = \sum_{i=1}^{d} z_i \mathop{\mathbb{E}}_{s\sim\mathcal{B}}[\nabla_\mu\pi_\mu(s)\nabla_a\boldsymbol{\psi}_{\theta,i}(s,a)|_{a=\pi_\mu(s)}], \tag{12}$$

where $z_i = 2((1-\gamma)^{-2}\mathbb{E}_{s,s'\sim\mathcal{B}}[\boldsymbol{\psi}_{\theta,i}(s,\pi_\mu(s)) - \gamma\boldsymbol{\psi}_{\theta,i}(s',\pi_\mu(s'))] - \boldsymbol{\psi}_i^E)$ and $\boldsymbol{\psi}_{\theta,i}$ denotes the SF at the i-th dimension for the current policy and $\boldsymbol{\psi}_i^E$ is the i-th dimension of SF of expert policy.

We provide the details of this derivation in Appendix A. Proposition 4.2 provides a way to estimate the gradients for

the actor and optimize the objective defined in Equation 5. Note that, this is achieved using a given batch of state-action transitions which is sampled from the replay buffer.

To highlight, our mechanism of updating the actor network is different than existing actor-critic algorithm in standard RL where the utility function or the action-value function can be estimated for every state-action pair, and the agent is updated to maximize this function. This is because for the task of matching expected features, it is challenging to estimate this quantity for state-action pairs that are outside the support of expert's state-visitation distribution. However, with the assumption that the initial state distribution will not change, the agent can estimate the utility function, and we show how our method SFM uses Proposition 4.2 to derive an update rule for the agent's policy.

Since the agent has access to the expert demonstrations comprising of the states visited by the expert. We propose another loss function to utilize this information and update the actor to match the SF of the agent on states visited by the expert. To understand this, lets consider a state sampled from expert demonstration $s \sim \tau_E$, and compute the action from the current agent's policy as $a = \pi_\mu(s)$. Now, the SF network provides an estimate of $\boldsymbol{\psi}_\theta(s, a)$ for this state and conditioned on the agent's policy. Furthermore, the SF for the expert $\boldsymbol{\psi}^E(s)$ can be estimated using the discounted sum of base features of all successor states from the state s. We define the loss function, called SF-Cloning loss, which estimates the MSE between the SF of the agent $\boldsymbol{\psi}_\theta(s, a)$ and the expert $\boldsymbol{\psi}^E(s)$ across states sampled from the expert's distribution such that:

$$\mathcal{L}_C = \mathbb{E}_{s\sim\tau^E} \|\boldsymbol{\psi}_\theta(s, \pi_\mu(s)) - \boldsymbol{\psi}^E(s)\|_2^2, \tag{13}$$

where $s \sim \tau^E$ represents any state sampled from the trajectories in the demonstrations. The actor can further utilize the DPG algorithm to update itself for this objective. Interestingly, this loss is different from behavior cloning (BC) loss because firstly, it does not use the expert action information, and secondly, this objective updates the agent to match future state-visitation occupancy with that of the expert. Since SFM assumes that the base features are dependent only on the states, then the expected features encoded by the SF network would also depend only on the states. This explains that the loss function introduced in Equation 13 is based on the state-only information.

Overall, the total loss to update the actor network of the agent can be computed as the linear combination of losses defined in Equation 9 and Equation 13 denoted as:

$$\mathcal{L}_\pi = \lambda_G \mathcal{L}_G + \lambda_C \mathcal{L}_C, \tag{14}$$

where λ_G and λ_C represent the hyperparameters to scale the different loss terms. In Algorithm 1, we outline the training procedure for SFM. In the remainder of this section we will discuss the base feature function ϕ (Section 4.1) and the implementation details of SFM (Section 4.2).

4.1. Base Feature Function

We described in Section 3 that SF depends on a base feature function $\phi : \mathcal{S} \to \mathbb{R}^d$. In this work, SFM learns the base features jointly while learning the policy. We also experimented with multiple base feature functions including Random Features, Hilbert Representations (HR) (Park et al., 2024), Inverse Dynamics Model (Pathak et al., 2017) and Forward Dynamics Model (FDM). We describe them below and unless otherwise mentioned, we have used FDM base feature function as it performed superior to other methods (Figure 3). Below, we briefly outline the description of these base feature functions:

Random Features (Random): This denotes a function where the parameters of the base feature network ϕ are initialized randomly at the start of training and kept fixed.

Inverse Dynamics Model (IDM): The key idea is to learn a function that extracts the controllable aspects of the environment. This is attained by learning an Inverse Dynamics Model (IDM) $g : \mathbb{R}^d \times \mathbb{R}^d \to \mathcal{A}$ that predicts the action for given state and next state pair by optimizing:

$$\mathcal{L}_{g,\phi} = \mathbb{E}_{s,a,s'}[\|g(\phi(s), \phi(s')) - a\|_2^2], \tag{15}$$

where both g and ϕ are learned using this loss function.

Forward Dynamics Model (FDM): In the forward dynamics model (FDM), the key idea is to learn a one-step transition model to predict the next state for on a given state-action pair. For a base feature function $\phi : \mathcal{S} \to \mathbb{R}^d$, the forward dynamics model $f : \mathbb{R}^d \times \mathcal{A} \to \mathcal{S}$ is learned via:

$$\mathcal{L}_{f,\phi} = \mathbb{E}_{s,a,s'}[\|f(\phi(s), a) - \phi(s')\|_2^2]. \tag{16}$$

Hilbert Representations (HR): The premise of Hilbert Representations (HR) is that two states that are closer in the feature space should be temporally closer. HR (Park et al., 2024) utilizes the temporal distance between features as $d^*_\phi(s, g) = \|\phi(s) - \phi(g)\|$, where this metric represents that time step taken to reach a state g from s. Furthermore, they proposed learning these features by leveraging a goal-conditioned value-based approach to minimize the following temporal distance loss:

$$\mathcal{L}_\phi = \mathop{\mathbb{E}}_{s,s',g}[\ell^2_\tau(-\mathbb{1}(s \neq g) - \gamma d^*_{\bar{\phi}}(s', g) + d^*_\phi(s, g)] \tag{17}$$

where $\bar{\phi}$ is the target base feature network, γ is the discount factor and ℓ^2_τ computes the expectile loss with an asymmetric ℓ^2 loss (Newey & Powell, 1987) to estimate the $\max$ operator in the Bellman backup (Kostrikov et al., 2021).

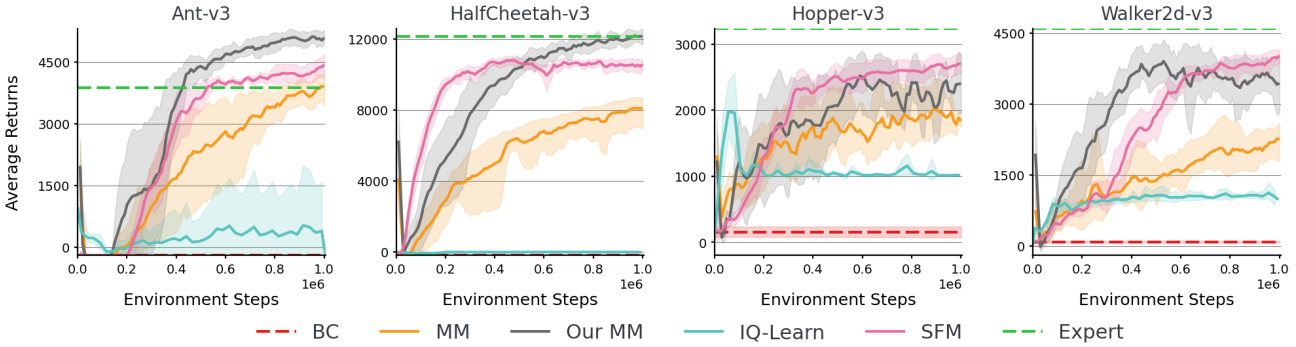

Figure 1: Comparison of the proposed method SFM with an offline method BC (Pomerleau, 1988), adversarial state-only IRL methods MM (Swamy et al., 2021), our implementation of state-only MM, a non-adversarial method IQ-Learn (Garg et al., 2021) with uses expert action labels, and Expert trained with the reward function on infinite horizon continuous control tasks. Average Returns are reported across 5 seeds with mean value and 95% confidence interval shading.

4.2. Network Architecture

Since SFM does not involve estimating a reward function and cannot leverage an off-the-shelf RL algorithm to learn a Q-funtion, we propose a novel architecture for our method. SFM is composed of 4 different networks- actor π_μ, SF network $\boldsymbol{\psi}_\theta$, base feature function ϕ and FDM f. Taking inspiration from state-of-the-art RL algorithms, we mantain target networks for both actor and the SF network. Since, SF network acts similarly to a critic in actor-critic like algorithms, SFM comprises of two networks to estimate the SF (Fujimoto et al., 2018). Here, instead taking a minimum over estimates of SF from these two networks, our method performed better with average over the two estimates of SF. To implement the networks of SFM, we incorporated several components from the TD7 (Fujimoto et al., 2023) algorithm. Moreover, unlike MM (Swamy et al., 2021), SFM did not require techniques like gradient penalty (Gulrajani et al., 2017), OAdam optimizer (Daskalakis et al., 2017) and a learning rate scheduler. In Appendix B and C, we provide the pseudocode of the SFM and describe the architecture used for the practical implementation of SFM.

5. Experiments

Through our experiments, we answer the following research questions to understand how effective our method is at using successor features for IRL: (1) can SFM learn an imitation policy with as little as a single demonstration, (2) what are the contributions of the different loss terms, and (3) what effects base feature function have on the learning progress?

5.1. Experimental Setup

We evaluate our method on the following continuous control environments- Ant, HalfCheetah, Hopper and Walker. Following the investigation in (Jena et al., 2020) which showed that the IRL algorithms are prone to biases in the learned reward function, we consider infinite horizon tasks where all episodes are truncated after 1000 steps in the environment. For each environment, the expert demonstrations are extracted from the widely used D4RL "expert-v2" dataset (Fu et al., 2020). For our experiments, the agent is provided with a single expert demonstration which is sampled at the start and kept fixed during the training phase. The agents are trained for 1M environment steps and we report the mean performance across 5 seeds with 95% confidence interval shading. We implemented SFM in Jax (Bradbury et al., 2018) and it takes about $\sim$2 hours to train one run on single NVIDIA A100 GPU. An important hyperparameter to tune was the discount factor γ for estimating the SF, where $\gamma = .99$ worked best for Ant task and $\gamma = .995$ was used for other environments. We provide more details about the implementation in Appendix C and hyperparameters in the Appendix D.

The baselines include behavior cloning (BC) (Pomerleau, 1988) which is a supervised learning based imitation learning method trained to match actions taken by the expert. To compare with a non-adversarial IRL method, we report results with IQ-Learn (Garg et al., 2021) which requires the knowledge of expert action labels. To compare with state-only IRL algorithms, we implemented a state-only version of MM (moment matching) (Swamy et al., 2021)– which is an adversarial IRL approach and used the version where the integral probability metric (IPM) is replaced with the Jenson-Shannon divergence (was shown to achieve better or comparable performance with GAIL (Swamy et al., 2022)). For the state-only MM baseline, we modified the discriminator network to depend only on the state and not on the actions. Lastly, to keep parity with the proposed method SFM, we replaced the the SAC (Haarnoja et al., 2018) with TD7 (Fujimoto et al., 2023) as the RL method.

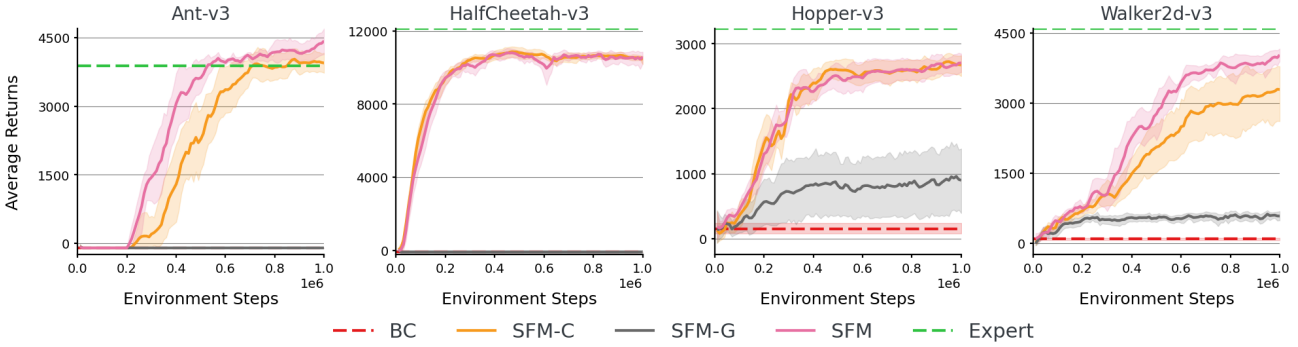

Figure 2: Study of different compositions of loss functions for updating the actor network. Here, SFM-M is the SFM model without $\mathcal{L}_G$ loss (Equation 9), SFM-C is the SFM model without $\mathcal{L}_C$ loss (Equation 13). We also report average returns of BC (Pomerleau, 1988) and the Expert trained with reward function in the environment. We observe that the

5.2. Results

Figure 1 presents the average returns across all environments. To highlight, our implementation of the state-only MM algorithm achieves superior performance and converges faster when compared with the original version. We observe that the proposed method SFM learns to solve the task with a single demonstration from the expert and significantly outperforms the adversarial baseline IQ-Learn, and without using the action label information in the demonstrations. Moreover, BC does fails in this regime of few expert demonstrations as the agent is unpredictable upon encountering states not in the expert dataset. Our implementation of MM performed better than SFM on Ant and HalfCheetah environments. However, SFM learned to perform better than the expert scores on the Ant task and reaches close to the expert score on the HalfCheetah task. Whereas on Hopper and Walker2d environments, no algorithms reached expert's performance. We believe these are more challenging tasks where the dynamical systems are less stable and the agent can reach states which are hard to recover from. On these task, SFM performed better than MM while being more stable at training.

The SFM agent updates the actor network to minimize the 1) *SF-Gap* loss ($\mathcal{L}_G$) that computes the difference between expected feature between expert and the agent when conditioned on the initial state distribution (Equation 9) and 2) *SF-Cloning* loss ($\mathcal{L}_C$) to match the SF between the agent and the expert at states visited by the expert's policy (Equation 13). Towards this end, we study the effect of different loss function in Figure 2. Here, SFM-G ($\lambda_G = 0, \lambda_C = 1$) and SFM-C ($\lambda_G = 1, \lambda_C = 0$) represent the variant of SFM without the SF-Gap and SF-Cloning loss. We observe that on Ant and HalfCheetah environments, the SFM-G agent was failing similarly to BC. However, on Hopper and Walker2d, the SFM-G variant performs better than BC but did not reach optimal behaviors. We believe that SF Cloning loss is interesting and will perform better with more expert transitions (as observed with BC) and leave this question for future research. The variant without the SF-Cloning loss was performing well and similar to SFM on HalfCheetah and Hopper environments with a slight drop in performance on Ant and Walker2d tasks. This demonstrates that SF-Gap loss (Equation 9) is a crucial component of SFM which, similar to IRL methods, updates the agent at any state visited in the environment and enables them to be more robust and recover from their mistakes.

In Figure 3, we present the comparison of different base features functions ϕ (descrived in Section 4.1). We experiment with Random Feature, Inverse Dynamics Model (IDM) (Pathak et al., 2017) which learns to predicts the action for a given state and next-state pair, Hilbert Representations (HR) (Park et al., 2024) which use the notion of temporal distance between states as a distance metric, Forward Dynamics Model (FDM) which is a one-step transition model to predict the next-state conditioned on a given state-action pair. We observe in Figure 3 that FDM achieves superior results when compared with other base feature functions. The closest to FDM was the IDM based features which attained similar performance to FDM on all environments except Walker2d and similar trends were observed for HR features (Park et al., 2024). Lastly, Random features was not doing well when compared with other base features. We believe our approach can leverage any representation learning technique and a potential avenue for future work would be to leverage pretrained features for more complex tasks and speed-up learning.

6. Limitations

One limitation of SFM is that the algorithm is currently tied with a particular choice of RL solvers, i.e. deterministic policy gradients. We believe our approach can be extended to a broader set of solvers that optimize both deterministic and stochastic policies. Secondly, our method only works

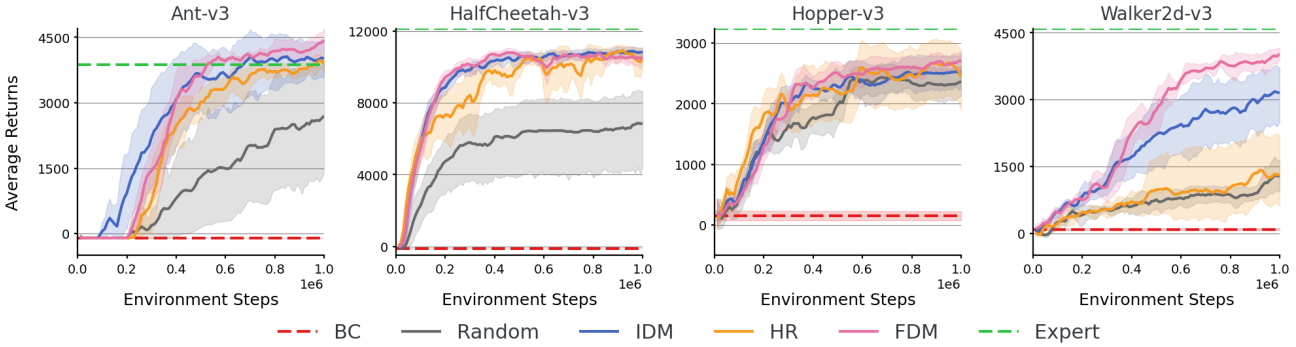

Figure 3: Effect of different base feature functions on the performance of the agent. Here, we compare with Random, Inverse Dynamics Model (IDM), Hilbert Representations (HR) (Park et al., 2024) and Forward Dynamics Models (FDM). Note that all these base feature functions where learned during the training phase.

with state-only base feature functions. We believe future work can lift this assumption by doing on-policy over a learned world model. Finally, while SFM is simpler than IRL methods, it still doesn't theoretically alleviate the exploration problem that IRL methods encounter. A promising direction of future work would be to combine SFM with mechanisms like reset distribution (Swamy et al., 2023) or hybrid IRL (Ren et al., 2024) to improve computational efficiency.

7. Discussion

We introduced SFM—a novel non-adversarial method for IRL that requires no expert action labels for training—via a reduction to a deterministic policy gradient algorithm. Through experiments in standard IRL benchmarks, SFM roughly matches the performance of the state of the art in IRL, and considerably exceeds the performance of existing non-adversarial methods. Of independent concern, our experiments illustrate that IRL methods are truly only as good as their RL subroutine: we achieve state-of-the-art in IRL by augmenting the existing MM method with the recent state-of-the-art TD7 algorithm(Fujimoto et al., 2023) for online RL, and achieve similar performance with our SFM approach built on TD7.

While SFM has not convincingly advanced the start-of-the-art in performance in the D4RL benchmark, we note that SFM has many desirable properties, which makes it a standout method among its competitors. Most notably, SFM is *simple*. It is relatively robust to feature learning methods, and is no less stable to train than online RL. This is not the case with adversarial approaches, which involve much more complex game dynamics in training, and generally require the implementation of several tricks to stabilize the optimization. This is similar to issues observed while training GAN (Goodfellow et al., 2014; Gulrajani et al., 2017) which needs tricks like gradient penalty and significant tuning of important hyperparameters.

In the context of Large Language Models (LLMs), learning to align the agent with human preferences is challenging. Towards this goal, existing methods like RLHF (Ouyang et al., 2022) involve learning a reward model of human preferences, leading to massively impactful LLM finetuning. NLHF (Munos et al., 2023) proposes an alternative preference model meant to capture the lack of total ordering of outcomes due to different preferences among a population, introducing another form of adversarial preference learning. More recently, alternatives such as DPO (Rafailov et al., 2024) and SPIN (Chen et al., 2024) demonstrated fantastic performance gains by circumventing the reward learning component of RLHF, similarly to how SFM circumvents the reward learning and game-theoretic optimization in IRL. We believe future work can look into harnessing the potential of SFM for a reward free method to align human preferences.

Current IRL algorithms have struggled to scale to high-dimensional inputs like images because due to the complex interaction between adversarial training and representation learning. In many real-world applications, the observations can be complex and interactions are expensive. However, standard RL algorithms have been successful at solving more challenging tasks with high-dimensional inputs (Hafner et al., 2023; Mnih et al., 2015; Yarats et al., 2021). We believe SFM can leverage the recent advancements in RL to unlock substantial performance gains in more complex IRL problems. Moreover, future works can pre-train the base features for SF using offline datasets.

Acknowledgements

The authors would like to thank Lucas Lehnert and Adriana Hugessen for their valuable feedback and discussions. The writing of the paper benefited from discussions with Darshan Patil, Mandana Samiei, Matthew Fortier, Zichao Li and anonymous reviewers. This work was supported

by Fonds de Recherche du Québec, National Sciences and Engineering Research Council of Canada (NSERC), Calcul Québec, Canada CIFAR AI Chair program, and Canada Excellence Research Chairs (CERC) program. The authors are also grateful to Mila (mila.quebec) IDT and Digital Research Alliance of Canada for computing resources. SC was supported by NSF NSF RI (2312956).

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Abdolshah, M., Le, H., George, T. K., Gupta, S., Rana, S., and Venkatesh, S. A new representation of successor features for transfer across dissimilar environments. In *International Conference on Machine Learning*, pp. 1–9. PMLR, 2021.

Abdulhai, M., Jaques, N., and Levine, S. Basis for intentions: Efficient inverse reinforcement learning using past experience. *CoRR*, abs/2208.04919, 2022.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D., Zidek, A., and Munos, R. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pp. 501–510. PMLR, 2018.

Borsa, D., Barreto, A., Quan, J., Mankowitz, D. J., van Hasselt, H., Munos, R., Silver, D., and Schaul, T. Universal successor features approximators. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1VWjiRcKX.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Bronstein, E., Palatucci, M., Notz, D., White, B., Kuefler, A., Lu, Y., Paul, S., Nikdel, P., Mougin, P., Chen, H., et al. Hierarchical model-based imitation learning for planning in autonomous driving. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8652–8659. IEEE, 2022.

Chen, Z., Deng, Y., Yuan, H., Ji, K., and Gu, Q. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

Daskalakis, C., Ilyas, A., Syrgkanis, V., and Zeng, H. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

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

Farebrother, J., Greaves, J., Agarwal, R., Lan, C. L., Goroshin, R., Castro, P. S., and Bellemare, M. G. Proto-value networks: Scaling representation learning with auxiliary tasks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=oGDKSt9JrZi.

Filos, A., Lyle, C., Gal, Y., Levine, S., Jaques, N., and Farquhar, G. PsiPhi-learning: Reinforcement learning with demonstrations using successor features and inverse temporal difference learning. In *International Conference on Machine Learning (ICML)*, 2021.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkHywl-A-.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 10–15 Jul 2018.

Fujimoto, S., Meger, D., and Precup, D. An equivalence between loss functions and non-uniform sampling in experience replay. *Advances in neural information processing systems*, 33:14219–14230, 2020.

Fujimoto, S., Chang, W.-D., Smith, E. J., Gu, S. S., Precup, D., and Meger, D. For SALE: State-action representation learning for deep reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xZvGrzRq17.

Garg, D., Chakraborty, S., Cundy, C., Song, J., and Ermon, S. IQ-learn: Inverse soft-q learning for imitation. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan,

J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Aeo-xqtb5p.

Ghosh, D., Bhateja, C. A., and Levine, S. Reinforcement learning from passive data via latent intentions. In *International Conference on Machine Learning (ICML)*, 2023.

Gomez, D., Bowling, M., and Machado, M. C. Proper laplacian representation learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Ho, J. and Ermon, S. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Igl, M., Kim, D., Kuefler, A., Mougin, P., Shah, P., Shiarlis, K., Anguelov, D., Palatucci, M., White, B., and Whiteson, S. Symphony: Learning realistic and diverse agents for autonomous driving simulation, 2022. *URL https://arxiv.org/abs/2205.03195*.

Jain, A. K., Lehnert, L., Rish, I., and Berseth, G. Maximum state entropy exploration using predecessor and successor representations. *Advances in Neural Information Processing Systems*, 36, 2024.

Jena, R., Agrawal, S., and Sycara, K. Addressing reward bias in adversarial imitation learning with neutral reward functions. *arXiv preprint arXiv:2009.09467*, 2020.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Le Lan, C., Tu, S., Oberman, A., Agarwal, R., and Bellemare, M. G. On the generalization of representations in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

Le Lan, C., Greaves, J., Farebrother, J., Rowland, M., Pedregosa, F., Agarwal, R., and Bellemare, M. G. A novel stochastic gradient descent algorithm for learning principal subspaces. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2023a.

Le Lan, C., Tu, S., Rowland, M., Harutyunyan, A., Agarwal, R., Bellemare, M. G., and Dabney, W. Bootstrapped representations in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023b.

Lee, D., Srinivasan, S., and Doshi-Velez, F. Truly batch apprenticeship learning with deep successor features. *arXiv preprint arXiv:1903.10077*, 2019.

Lehnert, L., Tellex, S., and Littman, M. L. Advantages and limitations of using successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1708.00102*, 2017.

Ma, C., Wen, J., and Bengio, Y. Universal successor representations for transfer reinforcement learning. *arXiv preprint arXiv:1804.03758*, 2018.

Machado, M. C., Bellemare, M. G., and Bowling, M. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5125–5133, 2020.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Munos, R., Valko, M., Calandriello, D., Azar, M. G., Rowland, M., Guo, Z. D., Tang, Y., Geist, M., Mesnard, T., Michi, A., et al. Nash learning from human feedback. *arXiv preprint arXiv:2312.00886*, 2023.

Newey, W. K. and Powell, J. L. Asymmetric least squares estimation and testing. *Econometrica: Journal of the Econometric Society*, pp. 819–847, 1987.

Ng, A. Y., Russell, S., et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Park, S., Kreiman, T., and Levine, S. Foundation policies with hilbert representations. *arXiv preprint arXiv:2402.15567*, 2024.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Ren, J., Swamy, G., Wu, Z. S., Bagnell, J. A., and Choudhury, S. Hybrid inverse reinforcement learning. *arXiv preprint arXiv:2402.08848*, 2024.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International conference on machine learning*, pp. 387–395. Pmlr, 2014.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Sun, W., Vemula, A., Boots, B., and Bagnell, D. Provably efficient imitation learning from observation alone. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6036–6045. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/sun19b.html.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Swamy, G., Choudhury, S., Bagnell, J. A., and Wu, S. Of moments and matching: A game-theoretic framework for closing the imitation gap. In *International Conference on Machine Learning*, pp. 10022–10032. PMLR, 2021.

Swamy, G., Rajaraman, N., Peng, M., Choudhury, S., Bagnell, D., Wu, S., Jiao, J., and Ramchandran, K. Minimax optimal online imitation learning via replay estimation. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=1mFfKXYMg5a.

Swamy, G., Wu, D., Choudhury, S., Bagnell, D., and Wu, S. Inverse reinforcement learning without reinforcement learning. In *International Conference on Machine Learning*, pp. 33299–33318. PMLR, 2023.

Syed, U. and Schapire, R. E. A game-theoretic approach to apprenticeship learning. *Advances in neural information processing systems*, 20, 2007.

Syed, U., Bowling, M., and Schapire, R. E. Apprenticeship learning using linear programming. In *Proceedings of the 25th international conference on Machine learning*, pp. 1032–1039, 2008.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.

Torabi, F., Warnell, G., and Stone, P. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

Torabi, F., Warnell, G., and Stone, P. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

Touati, A. and Ollivier, Y. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021.

Touati, A., Rapin, J., and Ollivier, Y. Does zero-shot reinforcement learning exist? In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MYEap_OcQI.

Vinitsky, E., Lichtlé, N., Yang, X., Amos, B., and Foerster, J. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. *Advances in Neural Information Processing Systems*, 35: 3962–3974, 2022.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

Zahavy, T., O'Donoghue, B., Desjardins, G., and Singh, S. Reward is enough for convex mdps. *Advances in Neural Information Processing Systems*, 34:25746–25759, 2021.

Zhang, J., Springenberg, J. T., Boedecker, J., and Burgard, W. 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.

Zhu, Z., Lin, K., Dai, B., and Zhou, J. Off-policy imitation learning from observations. *Advances in neural information processing systems*, 33:12402–12413, 2020.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., Dey, A. K., et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A. Proofs

Lemma A.1. *For a given initial state distribution P_0, a deterministic policy π and $\boldsymbol{\psi}^\pi$ defined as before, the following holds:*

$$(1-\gamma)\mathbb{E}_{s\sim P_0}[\boldsymbol{\psi}^\pi(s_1)] = \mathbb{E}_{s,a\sim\rho_\pi}[\boldsymbol{\psi}^\pi(s) - \gamma\boldsymbol{\psi}^\pi(s')], \tag{18}$$

where $\boldsymbol{\psi}^\pi(s) = \mathbb{E}_{a\sim\pi(s)}[\boldsymbol{\psi}^\pi(s,a)]$ and ρ_π is the state-action visitation distribution of policy π.

Proof. The state-visitation distribution over state-action pairs on the right-side of (18) can be expanded to form a telescoping sum. Suppose $\rho_t^\pi(s)$ be the marginal distribution of state at time t. Then,

$$\begin{aligned}
&\mathbb{E}_{s,a\sim\rho_\pi}[\boldsymbol{\psi}^\pi(s) - \gamma\boldsymbol{\psi}^\pi(s')] \\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s\sim\rho_t^\pi,a\sim\pi(s)}[\boldsymbol{\psi}^\pi(s) - \mathbb{E}_{s'\sim\mathcal{T}(.|s,a)}[\boldsymbol{\psi}^\pi(s')]] \\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s\sim\rho_t^\pi,a\sim\pi(s)}[\boldsymbol{\psi}^\pi(s)] - (1-\gamma)\sum_{t=0}^{\infty}\mathbb{E}_{s'\sim\rho_{t+1}^\pi}[\boldsymbol{\psi}^\pi(s')]] \\
&= (1-\gamma)\mathbb{E}_{s\sim P_0}[\boldsymbol{\psi}^\pi(s)].
\end{aligned}$$

For the last part, we replaced ρ_0^π with the initial state distribution P_0. This completes the proof. □

Proposition 4.1. *Conditioned on a given initial state distribution P_0, the following holds:*

$$(1-\gamma)\mathbb{E}_{s\sim P_0}[\boldsymbol{\psi}^\pi(s,\pi(s))] \tag{9}$$
$$= \mathbb{E}_{s,s'\sim\mathcal{B}}[\boldsymbol{\psi}^\pi(s,\pi(s)) - \gamma\boldsymbol{\psi}^\pi(s',\pi(s'))], \tag{10}$$

where $\mathcal{B}$ is the replay buffer and π is a deterministic policy.

Proof. Lemma A.1 presents a way to estimate SF for a given initial state distribution and the occupancy measure of the policy π. We begin the proof by first showing that Lemma A.1 holds for for any valid occupancy measure. Leveraging the fact that for any valid occupancy measure, there exists an unique policy that generates it (Ho & Ermon, 2016), we let the policy be $\beta(a|s)$ that generated the occupancy measure ρ^β.

Let ρ_t^β denote the marginal state distribution at time t. Then,

$$\begin{aligned}
&\mathbb{E}_{s,a\sim\rho_\beta}[\boldsymbol{\psi}^\pi(s) - \gamma\boldsymbol{\psi}^\pi(s')] \\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s\sim\rho_t^\beta,a\sim\beta(s)}[\boldsymbol{\psi}^\pi(s) - \mathbb{E}_{s'\sim\mathcal{T}(.|s,a)}[\boldsymbol{\psi}^\pi(s')]] \\
&= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{s\sim\rho_t^\beta,a\sim\beta(s)}[\boldsymbol{\psi}^\pi(s)] - (1-\gamma)\sum_{t=0}^{\infty}\mathbb{E}_{s'\sim\rho_{t+1}^\beta}[\boldsymbol{\psi}^\pi(s')]] \\
&= (1-\gamma)\mathbb{E}_{s\sim P_0}[\boldsymbol{\psi}^\pi(s)].
\end{aligned}$$

Notably, the proof relies on the fact that $\boldsymbol{\psi}^\pi$ is a function of the state only and does not depend on the action. If the policy π is deterministic, using $\boldsymbol{\psi}^\pi(s) = \boldsymbol{\psi}^\pi(s,\pi(s))$ gives:

$$(1-\gamma)\mathbb{E}_{s\sim P_0}[\boldsymbol{\psi}^\pi(s,\pi(s_1))] = \mathbb{E}_{s,s'\sim\mathcal{B}}[\boldsymbol{\psi}^\pi(s,\pi(s)) - \gamma\boldsymbol{\psi}^\pi(s',\pi(s'))],$$

where $\mathcal{B}$ is the replay buffer. This completes the proof. □

Proposition 4.2. *The gradients of the actor for a batch of sampled transitions from the replay buffer obtained by applying the DPG (Silver et al., 2014) algorithm to Equation 11 is:*

$$\nabla_\mu\mathcal{L}_G = \sum_{i=1}^{d} z_i \mathop{\mathbb{E}}_{s\sim\mathcal{B}}[\nabla_\mu\pi_\mu(s)\nabla_a\boldsymbol{\psi}_{\theta,i}(s,a)|_{a=\pi_\mu(s)}], \tag{12}$$

where $z_i = 2((1-\gamma)^{-2}\mathbb{E}_{s,s'\sim\mathcal{B}}[\boldsymbol{\psi}_{\theta,i}(s,\pi_\mu(s)) - \gamma\boldsymbol{\psi}_{\theta,i}(s',\pi_\mu(s'))] - \boldsymbol{\psi}_i^E)$ and $\boldsymbol{\psi}_{\theta,i}$ denotes the SF at the i-th dimension for the current policy and $\boldsymbol{\psi}_i^E$ is the i-th dimension of SF of expert policy.

Proof. For the loss function

$$\mathcal{L}_G = \|(1-\gamma)^{-1}\mathbb{E}_{s,s'\sim\mathcal{B}}[\boldsymbol{\psi}_\theta(s,\pi_\mu(s)) - \gamma\boldsymbol{\psi}_\theta(s',\pi_\mu(s'))] - \hat{\boldsymbol{\psi}}^E\|_2^2 \tag{19}$$

the gradient for the actor is given by:

$$\begin{aligned}
\nabla_\mu\mathcal{L}_G &= \nabla_\mu \sum_{i=1}^{d}\{(1-\gamma)^{-1}\mathbb{E}_{s,s'\sim\mathcal{B}}[\boldsymbol{\psi}_\theta(s,\pi_\mu(s)) - \gamma\boldsymbol{\psi}_\theta(s',\pi_\mu(s'))] - \hat{\boldsymbol{\psi}}^E\}^2 \\
&= \sum_{i=1}^{d} z_i \nabla_\mu\{(1-\gamma)^{-1}\mathbb{E}_{s,s'\sim\mathcal{B}}[\boldsymbol{\psi}_\theta(s,\pi_\mu(s)) - \gamma\boldsymbol{\psi}_\theta(s',\pi_\mu(s'))] - \hat{\boldsymbol{\psi}}^E\} \\
&= \sum_{i=1}^{d} z_i\{(1-\gamma)^{-1}\nabla_\mu\mathbb{E}_{s,s'\sim\mathcal{B}}[\boldsymbol{\psi}_\theta(s,\pi_\mu(s))\} \\
&= \sum_{i=1}^{d} z_i\{(1-\gamma)^{-1}\mathbb{E}_{s,s'\sim\mathcal{B}}[\nabla_\mu\boldsymbol{\psi}_\theta(s,\pi_\mu(s))\} \\
&= \sum_{i=1}^{d} z_i\{(1-\gamma)^{-1}\mathbb{E}_{s,s'\sim\mathcal{B}}[\nabla_\mu\pi_\mu(a)\nabla_a\boldsymbol{\psi}_\theta(s,a)|_{a=\pi_\mu(s)}\}
\end{aligned}$$

Here, we defined $z_i = 2\{(1-\gamma)^{-1}\mathbb{E}_{s,s'\sim\mathcal{B}}[\boldsymbol{\psi}_\theta(s,\pi_\mu(s)) - \gamma\boldsymbol{\psi}_\theta(s',\pi_\mu(s'))] - \hat{\boldsymbol{\psi}}^E\}$.

This completes the proof.

□

B. Algorithm

Algorithm 1 Successor Feature Matching (SFM)

Expert trajectory $\tau^E = \{s_0^i, a_0^i, ..., s_{T-1}^i, a_{T-1}^i\}_{i=1}^M$
Initialize actor and target actors: π_μ, $\pi_{\bar{\mu}}$, where $\bar{\mu} \leftarrow \mu$
Initialize SF and target SF networks: $\boldsymbol{\psi}_{\theta_1}, \boldsymbol{\psi}_{\theta_2}, \boldsymbol{\psi}_{\bar{\theta}_1}, \boldsymbol{\psi}_{\bar{\theta}_2}$ such that $\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$
Initialize base feature network ϕ and FDM network f
Initialize an empty replay buffer $\mathcal{B} = \{\}$
while Training **do**
  Observe state s and select action using $a = \mu(s) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$
  Execute a in environment and get next state s' and done d
  Append transition to replay buffer: $\mathcal{B} \leftarrow \mathcal{B} \cup (s, a, s', d)$
  Estimate SF of expert through demonstrations using

$$\hat{\boldsymbol{\psi}}^E = \frac{1}{M} \sum_{i=1}^{M} \sum_{t=0}^{T-1} \gamma^t \phi(s_t^i, a_t^i)$$

  if update_networks **then**
    Sample a mini-batch $\mathcal{D} = \{(s, a, s', d)\}$ from $\mathcal{B}$
    Get target actions: $a' = \pi_{\bar{\mu}}(s') + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$
    Compute target SF value for every transition tuple (s,a, s', d) using

$$\boldsymbol{y}(s, a) = \phi(s) + \gamma(1 - d)\frac{1}{2}\{\boldsymbol{\psi}_{\bar{\theta}_1}(s', a') + \boldsymbol{\psi}_{\bar{\theta}_2}(s', a')\}$$

    Update parameters θ_1 and θ_2 of SF network by taking gradients over

$$\mathcal{L}_{SF} = \frac{1}{|\mathcal{D}|} \sum_{(s,a,s') \sim \mathcal{D}} \|\boldsymbol{\psi}_{\theta_1}(s, a) - \boldsymbol{y}(s, a)\|_2^2 + \|\boldsymbol{\psi}_{\theta_2}(s, a) - \boldsymbol{y}(s, a)\|_2^2$$

    Update actor parameters μ using using the following loss

$$\mathcal{L}_G = \|(1 - \gamma)^{-1} \mathbb{E}_{s,s' \sim \mathcal{B}}[\boldsymbol{\psi}_\theta(s, \pi_\mu(s)) - \gamma \boldsymbol{\psi}_{\bar{\theta}}(s', \pi_{\bar{\mu}}(s'))] - \hat{\boldsymbol{\psi}}^E\|_2^2$$

    Sample states visited by the expert agent from the demonstrations $s^E \sim \tau^E$
    Update actor parameters μ using the following loss

$$\mathcal{L}_M = \mathbb{E}_{s^E \sim \tau^E} \|\boldsymbol{\psi}_\theta(s^E, \pi_\mu(s^E)) - \boldsymbol{\psi}^E(s^E)\|_2^2,$$

    where $\boldsymbol{\psi}^E(s)$ is the discounted expected sum of features of future states from s^E.
    Update base feature function and FDM f over $\mathcal{D}$ using $\mathcal{L}_{f,\phi} = \mathbb{E}_{s,a,s'}[\|f(\phi(s), a) - \phi(s')\|_2^2]$
    if update_targets **then**
      Update target parameters of actor using $\bar{\mu} \leftarrow \mu$
      Update target parameters of SF network using $\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$
    end if
  end if
end while=0

C. Implementation Details

The architecture used in this work is inspired from the TD7 (Fujimoto et al., 2023) algorithms for continuous control tasks (Pseudocode 2). We will describe the networks and sub-components used below:

- Two functions to estimate the SF ($\boldsymbol{\psi}_{\theta_1}, \boldsymbol{\psi}_{\theta_2}$)
- Two target functions to estimate the SF ($\boldsymbol{\psi}_{\bar{\theta}_1}, \boldsymbol{\psi}_{\bar{\theta}_2}$)
- A policy network π_μ
- A target policy network $\pi_{\bar{\mu}}$
- An encoder with sub-components f_ν, g_ν
- A target encoder with sub-components $f_{\bar{\nu}}, g_{\bar{\nu}}$
- A fixed target encoder with sub-components $f_{\bar{\bar{\nu}}}, g_{\bar{\bar{\nu}}}$
- A checkpoint policy π_c and the checkpoint encoder f_c
- A base feature function ϕ

Encoder: The encoder comprises of two sub-networks to output state and state-action embedding, such that $z^s = f_\nu(s)$ and $z^{sa} = g_\nu(z^s, a)$. The encoder is updated using the following loss:

$$\mathcal{L}(f_\nu, g_\nu) = \Big(g_\nu(f_\nu(s), a) - |f_\nu(s')|_\times\Big)^2 \tag{20}$$

$$= \Big(z^{sa} - |z^{s'}|_\times\Big)^2, \tag{21}$$

where s, a, s' denotes the sampled transitions and $|\,.\,|_\times$ is the stop-gradient operation. Also, we represent $\bar{z}^s = f_{\bar{\nu}}(s)$, $\bar{z}^{sa} = g_{\bar{\nu}}(\bar{z}^s, a)$, $\bar{\bar{z}}^s = f_{\bar{\bar{\nu}}}(s)$, and $\bar{\bar{z}}^{sa} = g_{\bar{\bar{\nu}}}(\bar{\bar{z}}^s, a)$.

SF network: Motivation by standard RL algorithms (Fujimoto et al., 2018; 2023), SFM uses a pair of networks to estimate the SF. The network to estimate SF are updated with the following loss:

$$\mathcal{L}(\boldsymbol{\psi}_{\theta_i}) = \|target - \boldsymbol{\psi}_{\theta_i}(\bar{z}^{sa}, \bar{z}^s, s, a)\|_2^2, \tag{22}$$

$$target = \phi(s) + \frac{1}{2}\gamma * clip([\boldsymbol{\psi}_{\bar{\theta}_1}(x) + \boldsymbol{\psi}_{\bar{\theta}_2}(x)], \boldsymbol{\psi}_{min}, \boldsymbol{\psi}_{max}), \tag{23}$$

$$x = [\bar{\bar{z}}^{s'a'}, \bar{\bar{z}}^{s'}, s', a'] \tag{24}$$

$$a' = \pi_{\bar{\mu}}(\bar{\bar{z}}^{s'}, s') + \epsilon, \text{where } \epsilon \sim clip(\mathcal{N}(0, \sigma^2), -c, c). \tag{25}$$

Here, instead of taking the minimum over the SF networks for bootstrapping at the next state(Fujimoto et al., 2018), the mean over the estimates of SF is used. The action at next state a' is samples similarly to TD3 (Fujimoto et al., 2018) and the same values of (z^s, z^{sa}) are used for each SF network. Moreover, the algorithm does clipping similar to TD7 (Fujimoto et al., 2023) on the predicted SF at the next state which is updated using $target$ ((23)) at each time step, given by:

$$\boldsymbol{\psi}_{min} \leftarrow min(\boldsymbol{\psi}_{min}, target) \tag{26}$$

$$\boldsymbol{\psi}_{max} \leftarrow min(\boldsymbol{\psi}_{max}, target) \tag{27}$$

Policy: SFM uses a single policy network which takes $[z^s, s]$ as input and is updated using the following loss function described in Section 4.

Upon every $target\_update\_frequency$ training steps, the target networks are updated by cloning the current network parameters and remains fixed:

$$(\theta_1, \theta_2) \leftarrow (\bar{\theta}_1, \bar{\theta}_2) \tag{28}$$
$$\mu \leftarrow \bar{\mu} \tag{29}$$
$$(\nu_1, \nu_2) \leftarrow (\bar{\nu}_1, \bar{\nu}_2) \tag{30}$$
$$(\bar{\nu}_1, \bar{\nu}_2) \leftarrow (\bar{\bar{\nu}}_1, \bar{\bar{\nu}}_2) \tag{31}$$
$$\tag{32}$$

Moreover, the agent maintains a checkpointed network similar to TD7 (Refer to Appendix F of TD7 (Fujimoto et al., 2023) paper). However, TD7 utilizes the returns obtained in the environment for checkpointing. Since average returns is absent in the IRL tasks, it is not clear how to checkpoint policies. Towards this end, we propose using the negative of MSE between the SF of trajectories generated by agent and the SF of demonstrations as a proxy of checkpointing. To highlight some differences with the TD7 (Fujimoto et al., 2023) algorithm, SFM does not utilize a LAP (Fujimoto et al., 2020) and Huber loss to update SF network, and we leave investigating them for future research.

Base Features: Since we use a base feature function ϕ, we have two networks- 1) To provide the embedding for the state, and 2) To predict the next state from the current state and action. Pseudocode 1 provides the description of the network architectures and the corresponding forward passes.

state-only MM: For the state-only MM method, we used the same architecture as TD7 for the RL subbroutine (Fujimoto et al., 2023). c Note that, SFM does not require any of these techniques for learning.

Pseudocode 1. Base Feature Network Details

Variables:

```
phi_dim = 128
```

Base Feature Network ϕ to encode state:

```
l0 = Linear(state_dim, 512)
l2 = Linear(512, 512)
l3 = Linear(512, phi_dim)
```

Base Feature ϕ Forward Pass:

```
input = state
x = Layernorm(l1(x))
x = tanh(x)
x = ReLU(x)
phi_s = L2Norm(l3(x))
```

FDM Network:

```
l0 = Linear(phi_dim + action_dim, 512)
l1 = Linear(512, 512)
l2 = Linear(512, action_dim)
```

FDM Network Forward Pass:

```
input = concatenate([phi_s, action])
x = ReLU(l0(x))
x = ReLU(l1(x))
action = tanh(l2(x))
```

Pseudocode 2. SFM Network Details

Variables:

```
phi_dim = 128
zs_dim = 256
```

Value SF Network:

▷ SFM uses two SF networks each with similar architechture and forward pass.

```
l0 = Linear(state_dim + action_dim, 256)
l1 = Linear(zs_dim * 2 + 256, 256)
l2 = Linear(256, 256)
l3 = Linear(256, phi_dim)
```

SF Network ψ_θ Forward Pass:

```
input = concatenate([state, action])
x = AvgL1Norm(l0(inuput))
x = concatenate([zsa, zs, x])
x = ELU(l1(x))
x = ELU(l2(x))
sf = l3(x)
```

Policy π Network:

```
l0 = Linear(state_dim, 256)
l1 = Linear(zs_dim + 256, 256)
l2 = Linear(256, 256)
l3 = Linear(256, action_dim)
```

Policy π Forward Pass:

```
input = state
x = AvgL1Norm(l0(input))
x = concatenate([zs, x])
x = ReLU(l1(x))
x = ReLU(l2(x))
action = tanh(l3(x))
```

State Encoder f Network:

```
l1 = Linear(state_dim, 256)
l2 = Linear(256, 256)
l3 = Linear(256, zs_dim)
```

State Encoder f Forward Pass:

```
input = state
x = ELU(l1(input))
x = ELU(l2(x))
zs = AvgL1Norm(l3(x))
```

State-Action Encoder g Network:

```
l1 = Linear(action_dim + zs_dim, 256)
l2 = Linear(256, 256)
l3 = Linear(256, zs_dim)
```

State-Action Encoder g Forward Pass:

```
input = concatenate([action, zs])
x = ELU(l1(input))
x = ELU(l2(x))
zsa = l3(x)
```

D. Hyperparameters

In Table 1, we provide the details of the hyperparameters used for learning. Many of our hyperparamters are similar to the TD7 (Fujimoto et al., 2023) algorithm. Important hyperparameters include the discount factor γ for the SF network and tuned it with values $\gamma = [0.98, 0.99, 0.995]$ and report the ones that worked best in the table. Rest, our method was robust to hyperparameters like learning rate and batch-size used during training.

Name	**Value**
Batch Size	1024
Discount factor γ for SF	.99 (Ant), .995 (Others)
Actor Learning Rate	5e-4
SF network Learning Rate	5e-4
Base feature function learning Rate	5e-4
FDM learning rate	5e-4
Network update interval	250
Target noise	.2
Target Noise Clip	.5
Number of layers	4
Number of units	400
Action noise	.2
Environments steps	1e6

Table 1: Hyper parameters used to train SFM.