SparseDice: Imitation Learning for Temporally Sparse Data via Regularization

Alberto Camacho [1] **Izzeddin Gur** [2] **Marcin Moczulski** [3] **Ofir Naschum** [2] **Aleksandra Faust** [2]

Abstract

Imitation learning learns how to act by observing the behavior of an expert demonstrator. We are concerned with a setting where the demonstrations comprise only a subset of state-action pairs (as opposed to the whole trajectories). Our setup reflects the limitations of real-world problems when accessing the expert data. For example, user logs may contain incomplete traces of behavior, or in robotics non-technical human demonstrators may describe trajectories using only a subset of all state-action pairs. A recent approach to imitation learning via distribution matching, ValueDice, tends to overfit when demonstrations are temporally sparse. We counter the overfitting by contributing regularization losses. Our empirical evaluation with Mujoco benchmarks shows that we can successfully learn from very sparse and scarce expert data. Moreover, (i) the quality of the learned policies is often comparable to those learned with full expert trajectories, and (ii) the number of training steps required to learn from sparse data is similar to the number of training steps when the agent has access to full expert trajectories.

1. Introduction

Sequential decision making is one of the central problems in computer science. We are concerned with *imitation learning*, an approach to sequential decision making in which the agent has to learn to imitate observed behavior (Abbeel & Ng, 2004). Imitation learning shares commonalities with *reinforcement learning* (RL), another popular approach to sequential decision making (Puterman, 1994; Russell & Norvig, 2016). In either model, the agent does not know the dynamics of the world and has to learn a policy by interaction. Imitation learning differs from RL in the way that the objective is specified. In RL, the agent has to maximize accrued reward over time. Designing a good reward function is often difficult, and is prone to errors that result in learning unintended behaviors. Imitation learning alleviates the burden of having to design reward functions. Some approaches to imitation learning first learn a reward function that is consistent with observed behavior, and then recast the problem as reinforcement learning (e.g., (Abbeel & Ng, 2004; Camacho et al., 2020a;b)).

Typical approaches to imitation learning make an implicit assumption that the behavior to imitate is demonstrated with dense execution traces. We relax such assumption (Section 3). Sometimes, the demonstrations are sparse—for example, when demonstrations are extracted from videos that were recorded with a lower bitrate than the frequency in which robot cameras and actuators operate. Moreover, generating expert data is usually time- and cost-expensive, especially if demonstrations are generated manually by humans. We are interested in addressing two questions: (i) Can we learn to imitate expert behavior with just a few data points? (ii) How shall we generate expert data if we had limited resources?

We observe that existing approaches to imitation learning, originally designed to operate with dense demonstrations, may not perform well when demonstrations are sparse. This is the case of ValueDice, a recent approach to imitation learning via distribution matching (Kostrikov et al., 2019). In Section 3.2 we show that ValueDice has a high risk to overfit to sparse data and learned policies do not replicate observed behavior.

We contribute a number of regularizers that stabilize the learning of ValueDice (Section 4). We refer to our approach as SparseDice. We evaluate the performance of SparseDice on four different domains for continuous control, extracted from the Mujoco suite, provided with sparse demonstrations (Section 5). SparseDice is more stable than ValueDice, and makes it still possible to learn from sparse demonstrations. We also examined the tradeoffs between the diversity of demonstration data (when examples come from different demonstration traces), and their sparsity. SparseDice can learn better-quality policies when there is a good balance between the diversity and sparsity of demonstration data. Our results open the door to future research on imitation learning from sparse demonstrations.

[1]Google Research, New York, USA [2]Google Research, Mountain View, USA [3]Google Research, Warsaw, Poland. Correspondence to: Alberto Camacho <albercm@google.com>.

In the Unsupervised Reinforcement Learning Workshop, collocated with ICML, 2021.

2. Background on Imitation Learning

Imitation learning is the problem of learning a policy that replicates observed behavior (Abbeel & Ng, 2004). The problem is framed in the context of an MDP $\langle S, A, p_0, p, r, \gamma \rangle$ that the agent can interact with. The agent can observe the current state $s \in S$ and apply an action $a \in A$. As a result, the state of the MDP transitions to a state $s' \in S$ with probability $p(s'|s,a)$. Such interaction yields a sequence $\tau = \{(s_i, a_i, s_{i+1})\}_0^{h-1}$ that we call *execution*. In the scope of this paper we consider that executions have finite horizon h. At the end of the execution, the state of the MDP is reset to a state s with probability $p_0(s)$.

A policy for an MDP is a function π that maps each state $s \in S$ into a probability distribution over actions, $\pi(\cdot|s) : A \to [0,1]$. An execution of a policy π from state s_0 is a sequence $\tau = \{(s_i, a_i, s_{i+1})\}_0^{h-1}$ such that each a_i has been sampled from $\pi^{\text{exp}}(\cdot|s_i)$, and $s_{i+1} \sim p(\cdot|s_i, a_i)$. The value of a policy π in a state s_0, $V_\pi(s_0)$, is the expected cumulative discounted reward of a policy execution:

$$V_\pi(s_0) := \mathbb{E}_\pi \Sigma_0^{h-1} \gamma^i r(s_i, a_i)$$

where $a_i \sim \pi(\cdot|s)$ and $s_{i+1} \sim p(\cdot|s_i, a_i)$, $r : S \times A \to \mathbb{R}$ is the reward function, and $0 < \gamma \leq 1$ is the discount factor of the MDP. A policy π is optimal if it maximizes $V_\pi(s)$ in all states $s \in S$.

The agent has access to a set of *demonstrations*, $\mathcal{D}$. Formally, a demonstration is an execution drawn from an optimal policy π^{exp} that optimizes for the expected cumulative discounted reward of the MDP. Demonstrations showcase how an expert may act in the MDP.

The objective of the agent is to recover π^{exp}. In imitation learning the agent does not know the dynamics model of the MDP, and does not have access to the reward function either. Note that imitation learning makes the assumption that demonstrations are drawn from an expert policy that optimizes for some reward function. However, such a reward function does not need to be explicitly specified.

2.1. Imitation Learning Via Distribution Matching

ValueDice is a recent, promising approach for imitation learning, that approaches the problem through the lens of distribution matching (Kostrikov et al., 2019). In this setting, we want to find a policy π whose state-action distribution d^π matches with the distribution d^{exp} of an expert policy $\pi^{\texttt{exp}}$, which are unknown to the agent. Following (Puterman, 1994), there exists a one-to-one correspondence between a policy π and d^π, defined as:

$$d^\pi(s,a) = (1-\gamma)\Sigma_{t=0}^{\infty} \gamma^t p(s_t = s, a_t = a|s_0)$$

where $s_0 \sim p_0(\cdot)$, $s_t \sim p(\cdot|s_{t-1}, a_{t-1})$, and $a_t \sim \pi(\cdot|s_t)$. With this notion, we can presume that expert demonstrations $\tau = \{(s_i, a_i, s_{i+1})\}_0^{h-1}$ are sampled from a distribution $s_{i+1} \sim d^{\text{exp}}(\cdot|s_i, a_i)$ associated to some expert policy π^{exp}. In order to measure the discrepancy between d^π and d^{exp}, it is common to use (and minimize) the KL divergence D_{KL}:

$$D_{KL}(d^\pi || d^{\text{exp}}) = -\mathbb{E}_{(s,a)\sim d^\pi} \log \frac{d^{\text{exp}}(s,a)}{d^\pi(s,a)}$$

One of the practical problems of the above expression is that we cannot sample from the expert policy. The authors of ValueDice perform several manipulations to the KL divergence to derive a max-min objective function J_{DICE} that is more practical to optimize for.

$$\begin{aligned} &\max_\pi \min_{\nu: S\times A \to \mathbb{R}} J_{\text{DICE}}(\pi, \nu) \\ &J_{\text{DICE}}(\pi,\nu) = \log \mathbb{E}_{(s,a)\sim d^{\texttt{exp}}} e^{\nu(s,a) - B^\pi(s,a)} \\ &\qquad\qquad - (1-\gamma)\mathbb{E}_{(s,a)\sim d^\pi} \nu(s,a) \end{aligned}$$

To obtain the expression J_{DICE}, they start from an alternative representation of the KL divergence, known as the Donsker-Varadhan form (Donsker & Varadhan, 1983):

$$\begin{aligned} D_{KL}(d^\pi || d^{\text{exp}}) = &- \min_{x: S\times A \to \mathbb{R}} \log(\mathbb{E}_{(s,a)\sim d^{\texttt{exp}}} e^{x(s,a)}) \\ &+ \mathbb{E}_{(s,a)\sim d^\pi} x(s,a) \end{aligned}$$

Then, the objective J_{DICE} can be obtained by performing a change of variables $x(s,a) = \nu(s,a) - B^\pi(s,a)$, where $B^\pi(s,a) = \gamma \mathbb{E}_{s'\sim p(\cdot|s,a), a'\sim\pi(\cdot|s)} \nu(s', a')$ is the expected Bellman operator with respect to π and zero reward.

In practice, the expression for $J_{\text{DICE}}(\pi, \nu)$ can be adapted to perform mini-batch training with a mix of samples from the set of expert demonstrations and the tuples of experience (s, a, s') stored in the replay buffer, obtained by interaction with the environment. Namely,

$$\begin{aligned} J^{\texttt{mix}}_{\text{DICE}}(\pi,\nu) = &\log \mathbb{E}_{(s,a)\sim d^{\texttt{mix}}} e^{\nu(s,a) - B^\pi(s,a)} \\ &- (1-\alpha)(1-\gamma)\mathbb{E}_{\substack{s_0\sim p(\cdot) \\ a_0\sim\pi(\cdot|s_0)}} \nu(s_0, a_0) \\ &- \alpha \mathbb{E}_{(s,a)\sim d^{RB}} \nu(s_0, a_0) - B^\pi(s,a) \end{aligned}$$

where d^{RB} is the (uniform) distribution of state-action pairs in the replay buffer (RB), and $d^{\texttt{mix}}(s,a) = (1-\alpha) d^{\texttt{exp}}(s,a) + \alpha d^{RB}(s,a)$.

Relation with Q learning. It has been observed that ν is some sort of Q value function of the underlying MDP (Kostrikov et al., 2019). Such MDP has rewards $r(s,a) = -x(s,a)$. The function that optimizes the KL divergence is $x^* = \log \frac{d^\pi(s,a)}{d^{\text{exp}}(s,a)} + C$ for some constant C (see (Kostrikov et al., 2019)).

3. Learning from Sparse Demonstrations

The literature on imitation learning commonly makes an underlying assumption that the number of expert demonstrations available to the agent is scarce. This assumption reflects the limitations of many real-world learning systems for which generating expert data incurs a cost, there are limited time resources to collect such data, and the amount of storage is limited—e.g., generating expert demonstrations for a grasping robot is time- and cost-expensive (e.g., (Kalashnikov et al., 2018)). Perhaps surprisingly, there is also the *de facto* conception that each individual demonstration comprises *all* the state-action pairs that appear along the execution trajectory. We argue that this assumption does not accommodate many real-world systems in which collecting dense demonstrations is prohibitively costly or not even feasible. For example, we may use demonstration videos to teach a robot how to assemble furniture, but the bitrate of such videos may be lower than the frequency rate of robot cameras, sensors, and actuators.

We study imitation learning with *sparse* demonstrations. In this setting we relax the assumption that demonstrations are dense, and we consider that expert demonstrations constitute a number of *examples*—rather than dense trajectories. Formally, an example is a triplet (s, a, s'), where $a \sim \pi(\cdot|s)$ is drawn from an expert policy π^{exp} and $s' \sim p(\cdot|s, a)$. An sparse demonstration is a subsequence of examples $\{(s_i, a_i, s_{i+1})\}_J$ for some $J \subseteq [1, h-1]$, where $\tau = \{(s_i, a_i, s_{i+1})\}_0^{h-1}$ is a (dense) demonstration.

Imitation learning from sparse examples has been underexplored. One of the objectives of this paper is to evidence different trade-offs related to generating expert data and learning from it. Certainly, we want to better understand how we shall generate and process such expert data. Our contributions can be summarized as follows:

- We evidence that current methods for imitation learning are brittle when they are trained with sparse demonstrations, thereby motivating the need for new algorithms that can handle sparse demonstrations;
- We show that current methods for imitation learning can be made more robust to sparse demonstrations, thereby unveiling opportunities for future research on this topic;
- We study the performance of our methods relative to different distributions of sparse demonstration data, gaining insights on how we shall generate demonstrations when our resources to do so are limited.

3.1. Generating Datasets with Sparse Experience

One of the reasons that motivated the work presented in this paper was that we wanted to gain a better understanding on how to generate expert data for imitation learning. Given a fixed amount of resources to generate data, shall we generate a few dense demonstrations, or shall we generate a larger number of sparse demonstrations?

Existing datasets for imitation learning contain dense demonstrations. We design different methods to generate datasets of sparse demonstrations, that we detail below. Informally, we generate sparse data by sampling dense trajectories according to a fixed distribution. Such distributions are parameterized in a way that we can tune the degree of sparsity of data.

Uniform(p): Each state-action pair in every demonstration is sampled iid with probability p. On average, the number of state-action pairs is reduced by a factor p.

Periodic(t) A state-action pair is sampled at every t timesteps. The number of state-action pairs is reduced by a factor $1/t$.

Snippets($\ell; p$) Snippets of ℓ consecutive timesteps are sampled independently with probability p. Approximately and on average, the number of state-action pairs is reduced by a factor p.

3.2. Limitations of ValueDice

The first contribution of this paper is an empirical evaluation of the performance of ValueDice when demonstrations are sparse. We found that the quality of the policies learned by ValueDice can degrade significantly in such a setting. This is not an unexpected result, because ValueDice (as well as other algorithms for imitation learning) was originally designed to learn from dense demonstrations. The result, however, motivates the development of new algorithms for imitation learning with sparse demonstrations.

We evaluate our methods on four popular benchmarks for continous control extracted from the Mujoco suite: Ant, Hopper, Half Cheetah, and Walker 2D. To generate sparce demonstation datasets, we downsample dense demonstrations with the Uniform, Periodic, and Snippets methods described earlier in this section. We use the same set of dense demonstrations used by Kostrikov et al. (2020a), generated by executing policies trained using TRPO (Schulman et al., 2015). The TRPO agent was trained with a reward function, but our agents for imitation learning do not have access to such reward function.

To evaluate the performance of the policies learned by ValueDice, we measure the reward collected by policy rollouts, and averaged results over 10 runs. The reward function is the same used by the TRPO agent that generates expert demonstrations. However, the reward function is hidden to the ValueDice agent, which has to learn solely from (sparse) demonstrations.

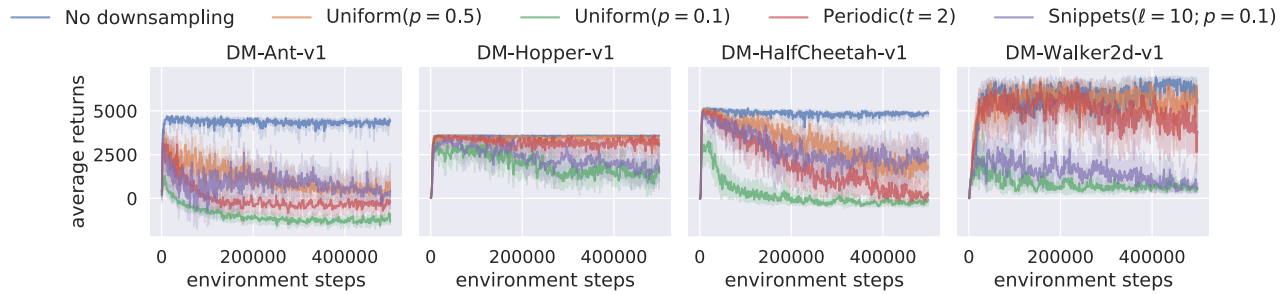


Figure 1: The quality of the policies learned with ValueDice drops significantly when demonstrations are downsampled. We used 10 demonstrations, that were downsampled with principled sampling methods (Uniform, Periodic, and Snippets). ValueDice can overfit data, even when trajectories are sampled periodically every other timestep.

ValueDice is sensitive to data sparsity. We conducted a series of experiments in which we trained ValueDice with a diverse set of sparse demonstration datasets, for which we varied the degree of sparsity. We observed that the quality of the policies learned by ValueDice degrades when the demonstrations are sparse. Figure 1 summarizes the results of some of our experiments. We show the quality of the policies along a wide range of $500,000$ training steps. When the demonstrations are dense, the quality of the policies learned by ValueDice is high and stable. However, when the demonstrations are sparse the quality of the policies is lower and not stable, and it degrades significantly. It is very noticeable that ValueDice is unstable even when demonstrations are obtained by sampling *Periodically* every $t = 2$ timesteps. Such subsampling is not very aggressive and, a priori, we had expected that ValueDice was able to interpolate demonstration data. However, ValueDice overfitted data very easily, and their performance is very low also in that setting.

Batch Normalization in ValueDice. We conducted a series of experiments to determine whether the overfitting of ValueDice to sparse demonstrations can be solved via simple batch normalization. Unfortunately, this was not the case. Figure 2 summarizes the results of our evaluations. We augmented the network architecture used in ValueDice with batch normalization in the intermediate layers, and also in the final layer. The performance of these two configurations was very similar to the the performance of the original implementation of ValueDice. These ablations suggest that we need more sophisticated regularization methods.

4. SparseDice: Regularizing ValueDice

We propose SparseDice, a strongly regularized DICE algorithm, to overcome the overfitting of ValueDice and improve its stability (Section 3.2). The main objective, $J_{\text{DICE}}(\pi, \nu)$, is a functional that takes both π, and ν functions as inputs. We exploit this dependency and regularize the objective by ensuring that both π and ν stay close to the expert demonstrations without overfitting. In the following, we introduce a series of loss functions that we add to $J_{\text{DICE}}(\pi, \nu)$.

Notation. Recall from Section 2.1 that we denote by RB the *replay buffer*, i.e.., the set of tuples of experience (s, a, s') obtained by interaction with the environment. We denote by $\mathcal{D}$ the set of (sparse) demonstrations, that also comprise tuples of examples $(s^*, a^*, (s^*)')$. In the definition of the regularization losses, we denote with ν_t (resp., π_t) the *target network* that results from freezing the weights of ν (resp., π) to make them untrainable. Using target networks is a standard practice to make training more stable.

4.1. Regularizers for the Value Function

Expert contrastive loss ν can be interpreted as the value function of the underlying MDP where the reward is defined as $\log \frac{d^{\text{exp}}(s,a)}{d^{\pi}(s,a)}$. Based on this observation, we introduce contrastive learning losses to enforce ν to take high values on expert demonstrations and low values otherwise,

$$L(\nu) = \mathbb{E}_{(s^*,a^*)\sim d^{\text{exp}}} \max_a \nu(s^*, a) - \nu_t(s^*, a^*)$$

The action space is continuous and computing $\max_a \nu(s^*, a)$ is intractable. Instead, we replace $\max_a \nu(s^*, a)$ with $\nu(s^*, a')$, where a' is sampled randomly from the set of actions that appear in the expert demonstrations. Finally, we impose a non-negativity constraint as the above objective is always non-negative and use a mini-batch approximation,

$$L(\nu) = \Sigma_{\substack{(s^*,a^*)\in batch(\mathcal{D}) \\ a'\sim \text{Uniform}(A)}} \max(0, \nu(s^*, a') - \nu_t(s^*, a^*))$$

RB contrastive loss This loss enforces ν to take high values in the state-action pairs that are on-policy. For each pair $(s, \cdot) \in$ RB sampled from the replay buffer, we compute the value $\nu(s, a')$ with the action $a' \sim \pi(\cdot|s)$ returned by the current policy and enforce that it is higher than values with

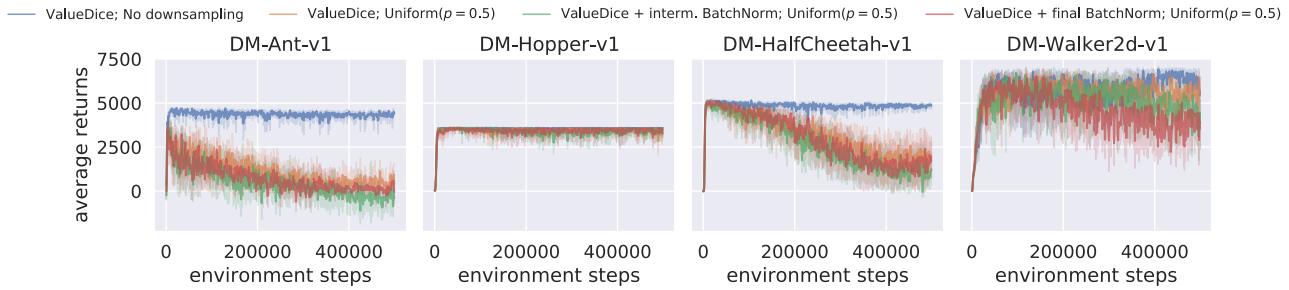


Figure 2: We augmented ValueDice with standard batch normalization methods used to stabilize learning. We experimented with adding batch normalization in the intermediate layers of ValueDice, and in the final layer. In our experiments, batch normalization does not improve the stability of ValueDice.

random actions. It is important to note that the parameters of the policy π are frozen and not made trainable.

$$L(\nu) = \mathbb{E}_{\substack{(s,\cdot)\sim d^{\text{RB}} \\ a'\sim\pi_t(\cdot|s)}} \max_a \nu(s,a) - \nu_t(s,a')$$

Similar to the expert contrastive loss, we replace $\max_a$ with random actions and use mini-batch approximation,

$$L(\nu) = \Sigma_{\substack{(s,\cdot)\in batch(\text{RB}) \\ a'\sim\pi_t(\cdot|s) \\ a''\sim\text{Uniform}(A)}} \max(0, \nu(s,a'') - \nu_t(s,a'))$$

ν RB expert contrastive loss We regularize ν to stay close to values of state and action pairs on expert demonstrations by minimizing the delta between on-policy and expert ν estimates,

$$L(\nu,\pi) = \mathbb{E}_{\substack{(s,\cdot)\in d^{\text{RB}} \\ a'\sim\pi(\cdot|s)}} \max(0, \nu(s,a') - \mu_{\texttt{exp}})$$

where $\mu_{\texttt{exp}} = \mathbb{E}_{(s^*,a^*)\sim d^{\text{exp}}} \nu_t(s^*,a^*)$ is the mean of ν values over the demonstration data. In practice we use,

$$L(\nu,\pi) = \Sigma_{\substack{(s,\cdot)\in batch(\text{RB}) \\ a'\sim\pi(\cdot|s)}} \max(0, \nu(s,a') - \mu_{\mathcal{D}})$$

where $\mu_{\mathcal{D}} = \frac{1}{|batch(\mathcal{D})|}\Sigma_{(s^*,a^*)\in batch(\mathcal{D})}\nu_t(s^*,a^*)$.

ν expert greater than mean loss We also regularize ν if it diverges too much from average expert values,

$$L(\nu) = \Sigma_{(s^*,a^*)\in\mathcal{D}}(\nu(s^*,a^*) - \mathbb{E}_{(s,a)\in\mathcal{D}}\nu_t(s,a))^2$$

In practice we use,

$$L(\nu) = \Sigma_{(s^*,a^*)\in batch(\mathcal{D})}(\nu(s^*,a^*) - \mu_{\mathcal{D}})^2$$

where $\mu_{\mathcal{D}} = \frac{1}{|batch(\mathcal{D})|}\Sigma_{(s^*,a^*)\in batch(\mathcal{D})}\nu_t(s^*,a^*)$.

ν bellman update loss The value function ν can be interpreted as a Q value function that optimizes for the expected cumulative discounted reward $r(s,a) = -x(s,a)$ (Kostrikov et al., 2019). This is a consequence of reinterpreting the KL divergence as an MDP with rewards $-x(s,a)$. Furthermore, the optimal values $r^*(s,a) = -x^*(s,a) = \log\frac{d^{\text{exp}}(s,a)}{d^{\pi}(s,a)} + C$ are constant when π converges to π^{exp}. If we interpret ν as a Q value function, then we can apply Bellman updates to ν in the same manner that is done in SARSA—i.e., for a transition (s,a,s') and reward r, we update $Q(s,a)$ to $r + \gamma Q(s',\pi(s'))$ with some learning rate α. There is one missing piece: in imitation learning the agent does not have a reward function. However, we can estimate r from the values of the value function, v. Observe that ν, interpreted as a Q value, is the sum of discounted rewards $r + \gamma r + \gamma^2 r + \ldots = \frac{r}{1-\gamma}$, and we have the equivalence $r = \nu(1-\gamma)$. Finally, we can estimate a (constant) reward function r by sampling a batch of expert experience, and averaging over the values of ν: $r = (1-\gamma) * \frac{1}{|batch(\mathcal{D})|}\Sigma_{(s^*,a^*)\in batch(\mathcal{D})}\nu(s^*,a^*)$. We use this reward in conjunction with Q learning updates to derive the following loss,

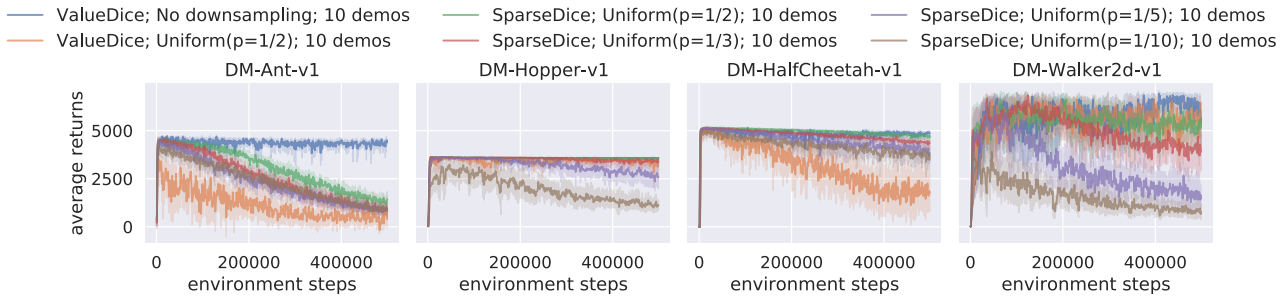
$$L(\nu) = \Sigma_{(s,a,s')\in\text{RB}}(\nu(s,a) - (r_t + \gamma * \nu_t(s',\pi(s')))^2$$

4.2. Regularizers for the Policy

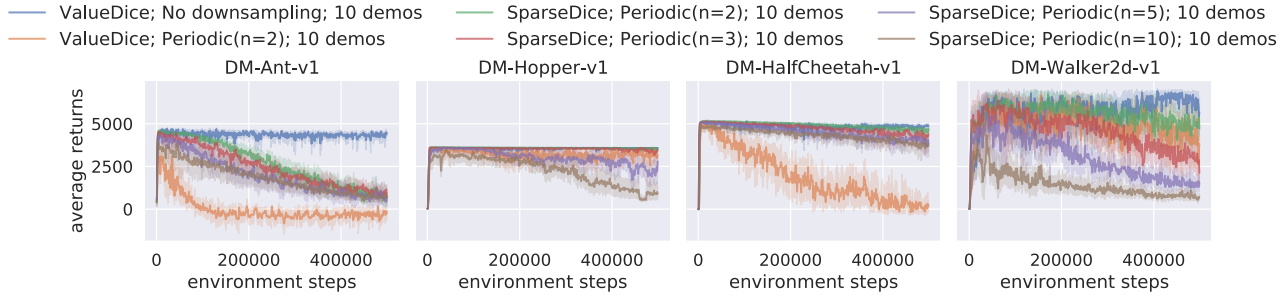
Policy expert imitation loss We enforce π to imitate the actions that appear in the expert demonstrations. More precisely, we use the squared L2 distance between an expert action a^* for state s^*, and the action $a' \sim \pi(\cdot|s^*)$ output by the policy.

$$L(\pi) = \Sigma_{\substack{(s^*,a^*)\in batch(\text{D}) \\ a'\sim\pi(\cdot|s^*)}}(a' - a^*))^2$$

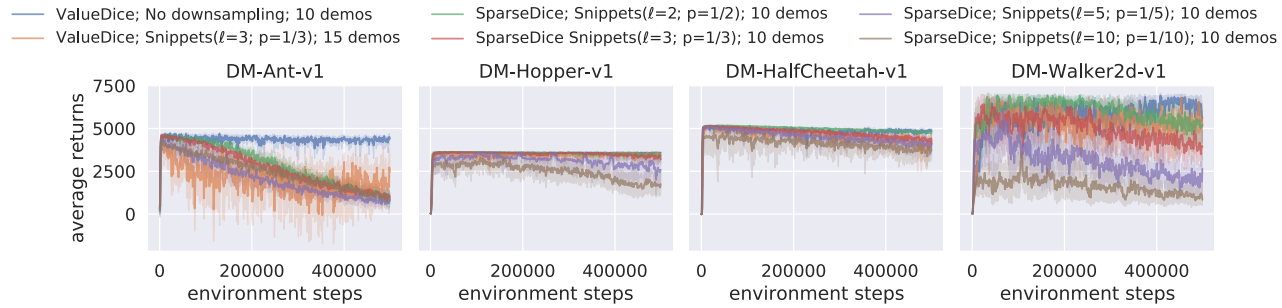
Policy self imitation loss Since the policy might overfit to expert demonstrations, we regularize the policy based on its distance to the replay buffer. More precisely, this loss samples state-action pairs (s,a) from the replay buffer, and computes the squared L2 distance between an action a for



(a) We used 10 demonstrations, that were sampled uniformly.



(b) We used 10 demonstrations, that were sampled periodically.



(c) We used 10 demonstrations, and we sampled snippets with uniform probability.

Figure 3: SparseDice is more robust than ValueDice when demonstrations are subsampled.

state s, and the action $a' \sim \pi(\cdot|s)$ output by the policy.

$$L(\pi) = \Sigma_{\substack{(s,a)\in batch(\text{RB}) \\ a'\sim\pi(\cdot|s)}}(a' - a))^2$$

5. Experimental Evaluation

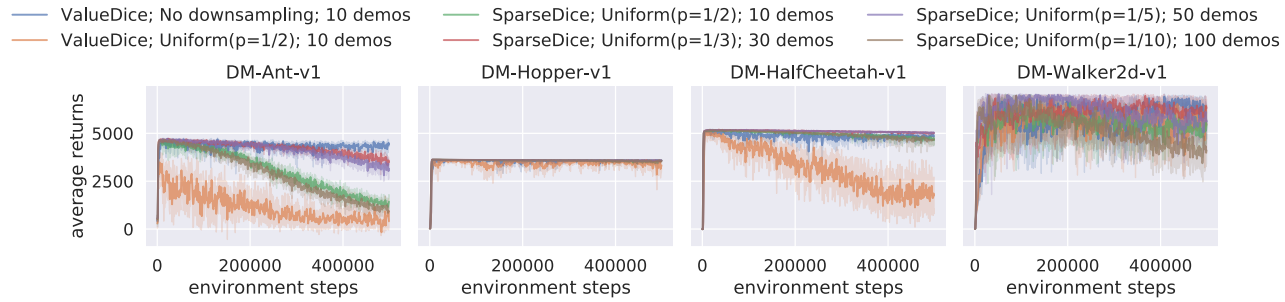
ValueDice can learn from a small set of dense demonstrations (Kostrikov et al., 2020a). However, as we saw in Section 3.2, one of its limitations is that demonstrations have to contain all the time steps in a trajectory. Otherwise, ValueDice has a high risk of overfitting when demonstrations are sparse. We saw that in such a case, the policies learned by ValueDice did not replicate expert behavior. Overfitting occured even when we sampled every other time step in the expert trajectories.

The purpose of our experimental evaluation is to address two practical questions:

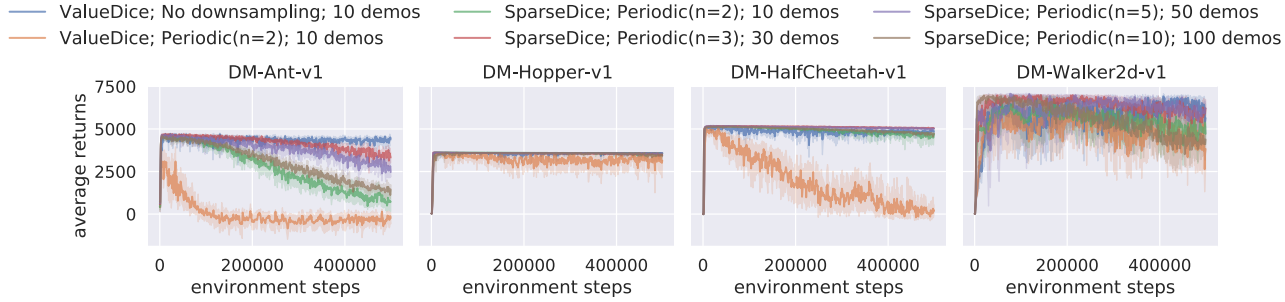
1. Can we learn to imitate expert behavior with just a few data points? If so, how does the quality of the policies learned degrade when we reduce the number of training examples?

2. How shall we generate expert data? If we had limited resources, we want to understand whether it is better to generate a few dense demonstrations, or a more larger set of sparse demonstrations.

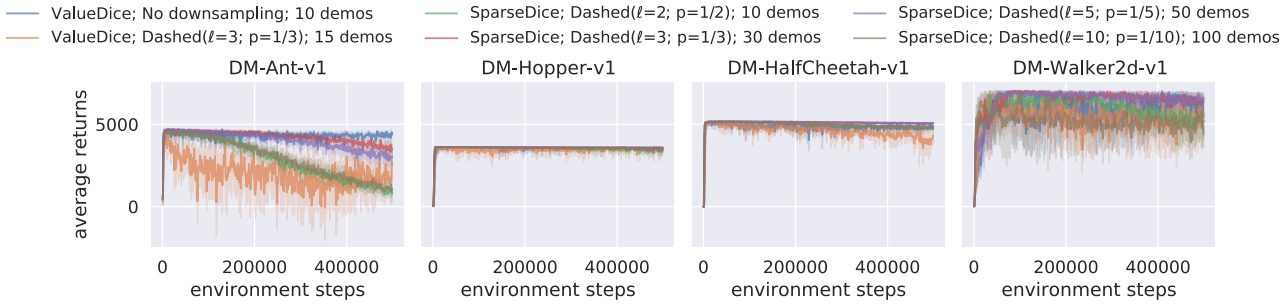
Experimental setup. We evaluated the performance of SparseDice on four popular continuous control tasks (Ant, Hopper, Half Cheetah, and Walker 2D) extracted from the Mujoco suite (Todorov et al., 2012). Expert demonstrations were generated by running executions of a TRPO agent (that was trained with rewards). Each (dense) expert demonstration comprises several thousands of consecutive examples (s, a, s'), and showcases several cycles of a robot moving



(a) Demonstrations were sampled uniformly.



(b) Demonstrations were sampled periodically.



(c) Snippets of demonstrations were sampled uniformly.

Figure 4: It is often better to sample from a more diverse dataset.

straightforward. The actions have certain stochasticity that make the robotic behavior be not exactly cyclic. Our results are averaged over 10 runs.

Implementation details. We implemented SparseDice on top of ValueDice (Kostrikov et al., 2020a), by adding the regularization losses listed in the preceding section. The objective function, $J(\pi, \nu)$, is parameterized on two functions ν and π. Each of these two functions is modeled with a feed-forward neural network with three layers. In SparseDice, we augmented the output layer of the ν network with a sigmoid activation. Otherwise, we observed that the outputs diverge as a side effect of some of the regularizers. We perform mini-batch training steps after each interaction with the environment, with a batch size of 256. When we perform training steps we freeze the weights of the target networks to compute the gradients.

5.1. Limitations of Learning with Sparse Data

We wanted to know whether it is possible to learn from a few sparse data points. To this end, we conducted a series of experiments to evaluate the performance of SparseDice when it is trained with sparse demonstrations. We generated a variety of sparse datasets by subsampling a fixed number $N = 10$ of dense expert demonstrations. We experimented with different sampling distributions, and different sampling rates.

The results of our experiments are summarized in Figure 3. As expected, the quality of the policies learned by SparseDice degrades when downsampling is more aggressive. However, it is remarkable that SparseDice is more stable than ValueDice, even when SparseDice is given less amount of expert data. Our empirical evaluation showcased that SparseDice is more robust to sparse demonstrations

than ValueDice. The policies learned with SparseDice are more stable, and in many cases they have better-quality than those learned by ValueDice.

5.2. Adding More Diversity on Training Data

Generating training data is costly, and the amount of training data that we can make available to the learning system may be limited. Under these conditions, we wanted to learn how we shall generate our training data. More precisely, it better to generate a few dense trajectories, or many sparse trajectories? Furthermore, what are the best sampling strategies at the time of generating the dataset of expert examples?

We conducted a series of experiments where we fixed (on average) the number of examples in the datasets of demonstrations. We constructed different datasets of experience by sampling from 10, 30, 50, and 100 demonstrations. Sampling was done in a way that the expected number of data samples is fixed. However, by sampling from the set of 100 demonstrations we get more diverse data than sampling from the set of 10 demonstrations.

The results are summarized in Figure 4. We observed that SparseDice can benefit from being trained with more diverse data. However, there exists tradeoffs. Given a fixed amount of resources to generate examples, the best number of individual demonstrations to sample from is uncertain, and we do not have a rule to determine such number beforehand.

5.3. Ablations

We conducted ablations to assess the contribution of each of the regularization losses to the quality of the policies learned by SparseDice. We found this to be the case. In our experiments, the "policy expert imitation loss" resulted to be the most beneficial regularizer.

6. Related Work

Regularization is a core paradigm in machine learning. While regularization has a long history in supervised learning, it has recently come into focus for RL and imitation learning due to interest in transfer learning (Cobbe et al., 2018) and offline RL (Levine et al., 2020). In transfer RL, the focus is on learning feature representations which generalize to other tasks given online access to a set of training tasks, whereas our setting is closer to offline RL, for which the main challenge is learning from a static dataset that exhibits sparsity in the state-action space. This may partially explain why we found poor performance of regularizers (e.g., batch normalization) which are known to perform well in transfer RL (Cobbe et al., 2018).

In the offline RL setting, much of the regularization is focused on keeping the learned policy close to the offline dataset, via various behavior regularization techniques (Fujimoto et al., 2019; Wu et al., 2019). In the imitation learning scenario, this is not as much of an issue, as the imitation learning already compels the learned policy to stay close to the offline dataset. Rather, many of our techniques can be interpreted as regularization on the *value function*, which also appears in the offline RL literature. Namely, our various regularization objectives on ν can be related to critic regularizers in the offline RL literature (Nachum et al., 2019; Kostrikov et al., 2021; Kumar et al., 2020). Many of these existing works also contrastive-like regularizers which encourage larger critic values on actions in the offline dataset and smaller values on actions sampled from the learned policy.

As the ValueDICE objective (Kostrikov et al., 2020a) is derived from a GAN-like loss (Goodfellow et al., 2014), it is important to mention that regularizers are popular in the GAN literature as well, where mode-collapse is a well-known issue (Srivastava et al., 2017). One of the simplest and most popular regularizers is the gradient penalty (Arjovsky et al., 2017; Mroueh & Sercu, 2017). However, we note that ValueDICE already includes a gradient penalty based on these existing works; although such a regularizer helps, we found that it is still not enough to maintain good performance in extremely sparse settings.

Other than applying regularizers, other works have suggested mitigating effects of sparse data by *augmenting* the data. Namely, one applies perturbations (e.g., random noise) to the given dataset to generate more synthetic data. This technique is popular in supervised learning (Chen et al., 2020) and has also been demonstrated in online RL (Kostrikov et al., 2020b) and offline RL (Sinha & Garg, 2021) settings. However, the use of such techniques relies on prior knowledge of the task to understand what perturbations are valid, and in fact, choosing the right perturbations can be a challenging problem on its own (Raileanu et al., 2020).

7. Discussion

Imitation learning is a convenient approach to sequential decision making. We argued that many real-world applications of imitation leaning may need to handle sparse demonstration data—e.g., because of limited budget. Existing algorithms for imitation learning that were designed to handle dense demonstrations, such as ValueDice, may not be as effective with sparse data. We contributed with regularizers for ValueDice, that make learning more stable. We saw that it is still possible to learn from a few sparse data points, algthough there exist tradeoffs between data sparsity and diversity that need be further studied. Our studies open the door to future research on imitation learning with sparse demonstrations.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In Brodley, C. E. (ed.), *International Conference on Machine Learning (ICML)*, volume 69 of *ACM*. ACM, 2004.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International conference on machine learning (ICML)*, pp. 214–223. PMLR, 2017.

Camacho, A., Varley, J., Zeng, A., Jain, D., Iscen, A., and Kalashnikov, D. Disentangled planning and control in vision based robotics via reward machines. *arXiv preprint arXiv:2012.14464*, 2020a.

Camacho, A., Varley, J., Zeng, A., Jain, D., Iscen, A., and Kalashnikov, D. Reward machines for vision-based robotic manipulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020b.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning (ICML)*, pp. 1597–1607. PMLR, 2020.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

Donsker, M. and Varadhan, S. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36 (2):183–212, March 1983. ISSN 0010-3640. doi: 10.1002/cpa.3160360204.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, pp. 2052–2062. PMLR, 2019.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., and Tompson, J. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *7th International Conference on Learning Representations (ICLR)*, 2019.

Kostrikov, I., Nachum, O., and Tompson, J. Imitation learning via off-policy distribution matching. In *8th International Conference on Learning Representations (ICLR)*, 2020a.

Kostrikov, I., Yarats, D., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020b.

Kostrikov, I., Tompson, J., Fergus, R., and Nachum, O. Offline reinforcement learning with fisher divergence critic regularization. *arXiv preprint arXiv:2103.08050*, 2021.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Mroueh, Y. and Sercu, T. Fisher gan. *arXiv preprint arXiv:1705.09675*, 2017.

Nachum, O., Dai, B., Kostrikov, I., Chow, Y., Li, L., and Schuurmans, D. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

Raileanu, R., Goldstein, M., Yarats, D., Kostrikov, I., and Fergus, R. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*, 2020.

Russell, S. J. and Norvig, P. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, volume 37, pp. 1889–1897, 2015.

Sinha, S. and Garg, A. S4rl: Surprisingly simple self-supervision for offline reinforcement learning. *arXiv preprint arXiv:2103.06326*, 2021.

Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U., and Sutton, C. Veegan: Reducing mode collapse in gans using implicit variational learning. *arXiv preprint arXiv:1705.07761*, 2017.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033. IEEE, 2012.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.