Distributional Multivariate Policy Evaluation and Exploration with the Bellman GAN

Dror Freirich [1] Tzahi Shimkin [1] Ron Meir [1] Aviv Tamar [2]

Abstract

The recently proposed distributional approach to reinforcement learning (DiRL) is centered on learning the distribution of the reward-to-go, often referred to as the value distribution. In this work, we show that the distributional Bellman equation, which drives DiRL methods, is equivalent to a generative adversarial network (GAN) model. In this formulation, DiRL can be seen as learning a deep generative model of the value distribution, driven by the discrepancy between the distribution of the current value, and the distribution of the sum of current reward and next value. We use this insight to propose a GAN-based approach to DiRL, which leverages the strengths of GANs in learning distributions of high-dimensional data. In particular, we show that our GAN approach can be used for DiRL with multivariate rewards, an important setting which cannot be tackled with prior methods. The multivariate setting also allows us to unify learning the distribution of values and state transitions, allowing us to devise a novel exploration method that is driven by the discrepancy in estimating both values and states.

1. Introduction

Deep Reinforcement Learning (DRL) has been applied to a wide range of problems in robotics and control, where policies can be learned according to sensory inputs without assuming a model of the environment (Mnih et al., 2015; Schulman et al., 2015). Until recent years, most RL methods have relied on estimating the expected future return, a.k.a. the 'value function', for carrying out the next action. Recent studies (Bellemare et al., 2017; Dabney et al., 2017) have suggested that a Distributional Reinforcement Learning (DiRL) approach, where the value distribution, rather than the expectation are learned, leads to improved learning performance[1].

DiRL algorithms such as C51 (Bellemare et al., 2017) and Quantile Regression DQN (Dabney et al., 2017) learn a mapping from states to a parametric distribution over the return, based on a distributional Bellman equation. Empirically, the distributional perspective to RL has been shown to significantly improve performance on challenging benchmarks (Bellemare et al., 2017; Hessel et al., 2017).

In this work, we provide a new approach to DiRL, building on an equivalence between the distributional Bellman equation and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014; Arjovsky et al., 2017). From this perspective, DiRL can be seen as learning a deep generative model of the value distribution, driven by the discrepancy between the distribution of the current value, and the distribution of the sum of current reward and next value. This view allows us to leverage GAN techniques for improving DiRL algorithms. In particular, GANs are known to be effective models for high-dimensional and correlated data such as images. We exploit this fact to develop a DiRL method for multivariate rewards, a setting for which previous DiRL methods are not suitable.

Multivariate rewards are important for domains where the natural performance evaluation of a policy depends on several different factors that cannot be easily combined into a single reward scalar (Vamplew et al., 2011). Here, we also show that the multivariate reward approach allows us to unify learning the distribution of future rewards and states under a common learning framework. We use this combined framework to develop a novel exploration strategy for RL, where the signal driving the exploration is the discrepancy in the learned distribution of states and rewards. We demonstrate the efficiency on high-dimensional test-benches.

In summary, our specific contributions in this work are the following. (1) Demonstrate an equivalence between the distributional Bellman equation and GANs, where we con-

[1]The Viterbi Faculty of Electrical Engineering, Technion - Israel Institute of Technology [2]Berkeley AI Research Lab, UC Berkeley. Correspondence to: Dror Freirich <drorfrc@gmail.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

[1]While the value is customarily defined as an expectation over a random variable, namely the return given in (1), we follow (Bellemare et al., 2017) and use the term value distribution for the distribution of the return.

sider the Wasserstein metric (Arjovsky et al., 2017; Villani, 2008) as a distributional distance measure. (2) Introduce the multivariate distributional Bellman equation and demonstrate policy evaluation on vector-valued reward functions. (3) Establish a novel reward-systematic exploration scheme, combining exploration with transition model learning.

2. Preliminaries and Formulation

Let $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ be a Markov Decision Process (MDP; Bertsekas 2005), where $\mathcal{S}$ is the state-space, $\mathcal{A}$ is the action space and $\mathcal{R}$ is the space of possible rewards. The states transition according to the *unknown* rule, $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$ and $\gamma \in [0, 1)$ is the discount factor. The action a_t is drawn w.r.t. some policy $\pi(\cdot|s_t)$. We discuss the case of full state observations for simplicity. A reward function $r_t = r(s_t, a_t) \in \mathcal{R}$ is supplied by the environment and may be unknown. By $\mathcal{Z}$ we denote the set of transformations from state-action space to probability distributions over rewards, $Z : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{R})$. Given a state s and an action a, we consider the (random) *return*, when starting from s and taking action a

$$R(s, a) \triangleq \sum_{t=0}^{\infty} \gamma^t r_t, \quad \text{s.t. } s_0 = s, a_0 = a. \tag{1}$$

Following (Bellemare et al., 2017), we use the notation $Z^\pi(s, a)$ for the return, and $Z^\pi(s, a) \in \mathcal{Z}$ to imply that $Z^\pi(s, a)$ is distributed according to a distribution belonging to the set of distributions $\mathcal{Z}$. Since we will be assuming a fixed policy for much of this work, we will simply use Z, rather than Z^π, to denote the random return.

Notation Throughout this paper we use the following acronyms. MDP - Markov Decision Process, RL - Reinforcement Learning, NN - Neural Network, GAN - Generative Adversarial Network, DQN - Deep Q-Network. W_p is the Wasserstein-p metric . For any two random variables X and Y, $X = Y$ denotes equality in distribution. By $\|\cdot\|$ we denote the l_2-norm, and by I_m the identity matrix of dimension m. δ_A is the indicator function of the set A.

2.1. Distributional RL

Given an initial state s and an action a, the Q-function is defined as the expectation

$$Q^\pi(s, a) \triangleq \mathbb{E}_\pi R(s, a) \tag{2}$$

A useful property of Q-functions is that they obey the Bellman equation (Bertsekas, 2005),

$$Q^\pi(s, a) = \mathbb{E}_\pi[r(s, a) + \gamma Q^\pi(s', a')], \tag{3}$$

where $s' \sim \mathcal{P}(\cdot|s, a)$, and $a' \sim \pi(\cdot|s')$.

An *optimal policy* is a policy satisfying $\pi^* \in \operatorname{argmax}_\pi \mathbb{E}_\pi Q(s, \pi(\cdot|s))$ for all $s \in \mathcal{S}$. The Q-function of an optimal policy, denoted Q^*, satisfies the optimality equation

$$Q^*(s, a) = \mathbb{E}[r(s, a) + \gamma \max_{a'} Q^*(s', a')]. \tag{4}$$

In RL we typically try to maximize the expected cumulative reward, namely the Q-function. Therefore, many RL approaches involve approximating solutions of (4) (Bertsekas, 1995; Mnih et al., 2015).

While the goal in RL is to maximize the expected return, it has recently been observed that learning the full distribution of the return $Z(s, a)$, rather than the expectation $\mathbb{E}[Z(s, a)]$, leads in practice to better performing algorithms (Bellemare et al., 2017; Hessel et al., 2017).

For learning the distribution of $Z(s, a)$ under a fixed policy π, Bellemare et al. (2017) showed that the *Bellman operator*,

$$\mathcal{T}^\pi Z(s, a) \triangleq r(s, a) + \gamma Z(s', a'), \tag{5}$$

is a $\gamma-$contraction over $\mathcal{Z}$ under the metric

$$\bar{d}_p = \sup_{s,a} W_p(Z_1(s, a), Z_2(s, a)), \qquad Z_1, Z_2 \in \mathcal{Z}. \tag{6}$$

Here again, s' is a random variable drawn from $\mathcal{P}(\cdot|s, a)$, $a' \sim \pi(\cdot|s')$, and W_p is the Wasserstein-p metric.

For practical reasons, DiRL algorithms use a parametric family of distributions $Z_\theta(s, a)$ to approximate $Z(s, a)$. Previous work explored using deep neural networks to map from s, a to either a distribution defined by a fixed set of particles (Bellemare et al., 2017), or a mixture of uniform distributions (Dabney et al., 2017). An approximation of the Bellman operator was used to update the distribution parameters.

2.2. Generative Adversarial Networks

GANs train two competing models (typically NNs) (Goodfellow et al., 2014). The *generator* takes noise $z \sim P_z$ as input and generates samples according to some transformation, $G_\theta(z)$. The *discriminator* takes samples from both the generator output and the training set as input, and aims to distinguish between the input sources. Goodfellow et al. (2014) measured discrepancy between the generated and the real distribution using the Kullback-Leibler divergence. However, this approach was improved by Arjovsky et al. (2017), by using the Wasserstein-1 distance. Wasserstein-GANs exploit the Kantorovich-Rubinstein duality (Villani, 2008),

$$W_1(\mathbb{P}_r, \mathbb{P}_g) = \sup_{f \in 1-\text{Lip}} \left\{ \mathbb{E}_{x\sim\mathbb{P}_r} f(x) - \mathbb{E}_{x\sim\mathbb{P}_g} f(x) \right\}, \tag{7}$$

where $1 - \text{Lip}$ is the class of Lipschitz functions with Lipschitz constant 1, in order to approximate the distance between the real distribution, $\mathbb{P}_r$, and the generated one, $\mathbb{P}_g$. The GAN objective is then to train a generator model $G_\theta(z)$ with noise distribution P_z at its input, and a critic $f \in 1-\mathrm{Lip}$, achieving

$$\min_{G_\theta(\cdot)} \max_{f \in 1-\mathrm{Lip}} \left\{ \mathbb{E}_{z\sim\mathbb{P}_r} f(x) - \mathbb{E}_{z\sim P_z} f(G_\theta(z)) \right\}. \quad (8)$$

The WGAN-GP (Gulrajani et al., 2017) is a stable training algorithm for (8) that employs stochastic gradient descent and a penalty on the norm of the gradient of the critic with respect to its input. The gradient norm is sampled at random points along lines between a real sample, x, and a generated sample, $G_\theta(z)$,

$$\tilde{x} = \varepsilon x + (1-\varepsilon) G_\theta(z), \;\; x \sim \mathbb{P}_r, z \sim P_z, \varepsilon \sim U[0,1]. \quad (9)$$

The penalty term $\lambda \mathbb{E}_{\tilde{x}} \left(\|\nabla_{\tilde{x}} f(\tilde{x})\| - 1 \right)^2$ is then added to the discrimintaor optimization objective, where λ is the gradient penalty coefficient. By employing this penalty, the norm of the critic gradient is penalized for deviating from 1, thus encouraging f to belong to $1-\mathrm{Lip}$.

2.3. Intrinsic-Reward Based Exploration

Exploration is a key challenge in RL. While efficient algorithms and performance guarantees are available for the model-based setting with finite state and action spaces (e.g., Kearns & Singh 2002; Osband et al. 2013; Tewari & Bartlett 2008), the situation is very different in the model free setting, and, in particular, for large state and action spaces. Within a model-free setting, the unknown environment and rewards are not directly modeled, and actions are learned by the agent through trial-and-error experience. At each stage of the process an agent must balance exploration and exploitation. Exploitation uses the knowledge gained so-far about the environment, and attempts to use this to maximize the return. Since the knowledge gained is always partial and approximate, exploration is required in order to improve the agent's knowledge about environment, thereby improving future exploitation. Exploration is particularly difficult in a model free setting, where the environment itself is not modeled. Simple exploration techniques in model-free RL draw randomized actions (e.g. ϵ-greedy, Boltzmann exploration (Sutton & Barto, 1998)), or mildly perturb policy improvements (Lillicrap et al., 2015).

A promising recent approach to exploration in model free settings uses the notion of curiosity and internal reward (Oudeyer et al., 2007; Schmidhuber, 2010) in order to direct the learner to regions in state-action space where system uncertainty is large. Such methods aim to explore regions in state-action space through actions that lead to poorly predictable outcomes, namely to 'surprise'.

These methods often set a trade-off between exploitation and exploration using a tuning parameter η and a combined reward function,

$$r'(s_t, a_t, s_{t+1}) = r(s_t, a_t) + \eta r^i(s_t, a_t, s_{t+1}). \quad (10)$$

In this formulation, intrinsic rewards $r^i(s_t, a_t, s_{t+1})$ measure information gains about the agent's internal belief of a dynamical model it holds. For example, Houthooft et al. (2016) capture information gain through the notion of mutual information which is approximated and estimated by their algorithm.

3. Related Work

The C51 algorithm of (Bellemare et al., 2017) represented the value distribution using a fixed set of 51 particles, and learned the probability of each particle. Dabney et al. (2017) extended this approach to particles with adjustable locations, where each particle corresponds to a fixed uniform distribution. Both approaches, which rely on 'discretization' of the value distribution, do not scale to high-dimensional multivariate reward distributions.

Concurrently, and independently from us, Doan et al. (2018) showed a similar equivalence between the distributional Bellman equation and GANs, and used it to develop a GAN Q-learning algorithm. Compared to that work, which did not show any significant improvement of GAN Q-learning over conventional DRL methods, we show that the GAN approach can be used to tackle multivariate rewards, and use it to develop a novel exploration strategy.

In the context of model based RL, Asadi et al. (2018) showed an equivalence between Wasserstein and value-aware methods for learning state transition models. This is different from our framework, that is able to learn both the state transitions and the value distribution using a single deep generative model. Tang & Agrawal (2018) proposed a distributional RL based exploration strategy with Bayesian parameter updates. This approach introduces a minimization objective combining the expected distributional discrepancy from observed data together with an exploration term, encouraging a high-entropy distribution. In our work, we suggest that the learning of a distributional discrepancy by itself naturally yields a motivation for exploration.

In the context of intrinsic-reward based exploration, the VIME method of Houthooft et al. (2016) described above, does not take information about the distribution of incoming rewards into consideration. This may become an advantage where rewards are sparse, but misses crucial information about the task elsewhere.

4. Equivalence Between DiRL and GANs

In this section, we show that the distributional Bellman equation can be interpreted as a GAN, leading to a novel approach for learning the return distribution in DiRL. Given a fixed stochastic policy π, where $a_t \sim \pi(\cdot|s_t)$, we construct a state-action conditioned generative model with an output distribution $Z(s,a) \in \mathcal{Z}$ such that

$$Z(s,a) \overset{\bar{d}_1}{=} T^\pi Z(s,a), \tag{11}$$

$$T^\pi Z(s,a) \triangleq r(s,a,s') + \gamma Z(s',a'), \tag{12}$$

where $s' \sim \mathcal{P}(\cdot|s,a), a' \sim \pi(\cdot|s')$. The notation (11) implies that $\overline{d_1} \triangleq \sup_{s,a} W_1(Z(s,a), T^\pi Z(s,a))$ vanishes. We use a more general notation where r may depend on s'.

We can solve (11) via a game formulation: due to the Kantorovich-Rubinstein duality (7) we have a 1-Lipschitz function (in its first argument), $f(r|s,a)$, s.t.

$$W_1(Z(s,a), T^\pi Z(s,a)) = \tag{13}$$
$$\mathbb{E}_{r\sim T^\pi Z(s,a)} f(r|s,a) - \mathbb{E}_{r\sim Z(s,a)} f(r|s,a),\ \forall s,a.$$

Similarly to (Arjovsky et al., 2017), we make use of this duality for training a *conditional GAN* in a novel configuration we call a *Bellman-GAN* (Figure 1(a)). The generator $G_\theta(\cdot|s,a)$ is a generative NN model with parameters θ, whose input consists of random noise with distribution P_z, and whose output distribution imitates that of $Z^\pi(s,a)$. The discriminator learns a critic $f_\omega(\cdot|s,a)$ (we will sometimes omit the state-action condition for simplicity) . The optimization objective is

$$\min_{G_\theta(\cdot|s,a)} \max_{f_\omega(\cdot|s,a)\in 1-\mathrm{Lip}} \mathcal{L}_\pi(G,f), \tag{14a}$$

$$\mathcal{L}_\pi(G,f) = \tag{14b}$$
$$\mathbb{E}_{z\sim P_z, a_{t+1}\sim\pi(\cdot|s_{t+1})} \Lambda(G_\theta, f_\omega))|_{(z,s_t,a_t,r_t,s_{t+1},a_{t+1})}.$$

Here, $z \sim P_z$ is the input noise and Λ is defined by

$$\Lambda(G_\theta, f_\omega)|_{(z,s,a,r,s',a')} \triangleq$$
$$f_\omega(r + \gamma G_\theta(z|s',a')) - f_\omega(G_\theta(z|s,a)). \tag{15}$$

We emphasize that unlike the conventional WGAN (Arjovsky et al., 2017), in the Bellman GAN (Figure 1(a)), both distributions are generated and there is no 'real' distribution. We train our critic using the WGAN-GP scheme (Gulrajani et al., 2017), with a penalty factor of λ (see Section 2.2). We term our algorithm Value Distribution GAN Learning (VDGL); pseudo-code is provided in Algorithm 1.

Algorithm 1 Value Distribution GAN Learning (VDGL)

```
Input: discriminator parameters $\omega$, generator parameters $\theta$, fixed policy $\pi(\cdot|s)$, discount factor $\gamma$.
Parameters: number of steps $T$, learning rate $\alpha$, penalty factor $\lambda$, minibatch size $m$, $n_{critic}$ = 5.
for $t = 1$ to T do
    Act according to $a_t \sim \pi(\cdot|s_t)$.
    Observe $(s_t, a_t, s_{t+1})$ and draw reward $r_t$.
    Draw the next action according to $a_{t+1} \sim \pi(\cdot|s_{t+1})$.
    Store $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ in replay pool.
end for
for $n = 1$ to $n_{critic}$ (Train critic) do
    1. Sample $\{(s_t^{(i)}, a_t^{(i)}, r_t^{(i)}, s_{t+1}^{(i)}, a_{t+1}^{(i)})\}_{i=1}^m$ from replay pool.
    2. Sample both $\{z^{(i)}\}$ and $\{z'^{(i)}\}$, $i = 1,\dots,m$, from $P_z$, and $\{\varepsilon^{(i)}\}$, $i = 1,\dots,m$, from $U[0,1]$.
    3. $x_\theta^{(i)} = G_\theta(z^{(i)}|s_t^{(i)}, a_t^{(i)})$, $x'^{(i)}_\theta = r_t^{(i)} + \gamma G_\theta(z'^{(i)}|s_{t+1}^{(i)}, a_{t+1}^{(i)})$.
    4. $\tilde{x}^{(i)} \leftarrow \varepsilon^{(i)} x_\theta^{(i)} + (1-\varepsilon^{(i)}) x'^{(i)}_\theta$.
    5. $g_\omega \leftarrow \frac{1}{m}\nabla_\omega \sum_{i=1}^m \Big[ f_\omega\big(x_\theta^{(i)}\big) - f_\omega\big(x'^{(i)}_\theta\big) + \lambda\big(\|\nabla_{\tilde{x}} f_\omega(\tilde{x}^{(i)})\| - 1\big)^2 \Big]$.
    6. $\omega \leftarrow \mathrm{Adam}(\omega, g_\omega, \alpha)$.
end for
(Train generator)
    1. Sample $\{(s_t^{(i)}, a_t^{(i)}, r_t^{(i)}, s_{t+1}^{(i)}, a_{t+1}^{(i)})\}_{i=1}^m$ from replay pool.
    2. Sample both $\{z^{(i)}\}$ and $\{z'^{(i)}\}$, $i = 1,\dots,m$, from $P_z$.
    3. $x_\theta^{(i)} = G_\theta(z^{(i)}|s_t^{(i)}, a_t^{(i)})$, $x'^{(i)}_\theta = r_t^{(i)} + \gamma G_\theta(z'^{(i)}|s_{t+1}^{(i)}, a_{t+1}^{(i)})$.
    4. $g_\theta \leftarrow -\frac{1}{m}\nabla_\theta \sum_{i=1}^m \Big[ f_\omega\big(x_\theta^{(i)}\big) - f_\omega\big(x'^{(i)}_\theta\big) \Big]$.
    5. $\theta \leftarrow \mathrm{Adam}(\theta, g_\theta, \alpha)$.
```

4.1. Q-function and Z-distribution Estimation

Learning the discounted reward distribution of $Z(s,a)$ at every point of state-action space may lead to slow convergence for large spaces (recall that we are concerned here with a fixed policy). However, learning the Q-function is a simpler task, and may be utilized at an earlier stage of training, forming a baseline for learning Z. By setting the generator architecture given in Figure 1(b), we manage to estimate the Q and Z functions concurrently, without significant additional computational cost. Here $Q(s,a)$ is trained using DQN to satisfy the optimality equation (4), and $G_\theta(\cdot|s,a) = \hat{Z}_\theta(\cdot|s,a) + Q(s,a)$ is trained by VDGL to satisfy (11). We believe other distributional methods may benefit from this setting.

5. Multivariate Rewards

The equivalence between DiRL and GANs proposed in the previous section allows us to train deep generative models for approximating the value distribution. One advantage of this approach is when the reward is a *vector*, requiring learning a multivariate distribution for the value function.

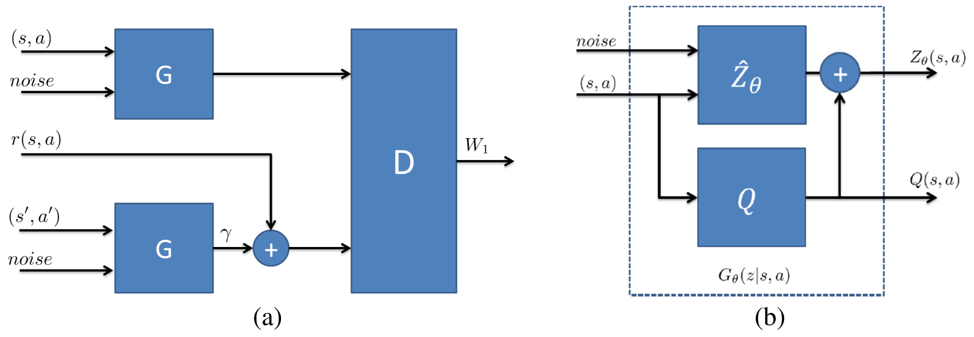


Figure 1. GAN configurations. (a) Bellman-GAN; (b) Simultaneous estimation of Q-function and Z-distribution.

Multivariate rewards are important when the natural objective in the decision problem cannot be easily composed into a single scalar quantity (Mannor & Shimkin, 2004; Vamplew et al., 2011), and where different objectives may need to be balanced. Examples include RL for clinical trials (Zhao et al., 2009), but also learning auxiliary predictions within a standard RL problem (Sutton et al., 2011; Dosovitskiy & Koltun, 2016).

Since GANs have been shown to generate remarkably convincing high dimensional multivariate signals such as images (Goodfellow et al., 2014; Radford et al., 2015), we propose that our GAN approach to DiRL would scale well to the multivariate reward case.

Consider the case where the reward is a vector $r_t \in \mathbb{R}^m$. Given a fixed policy π, we define the return vector

$$Z(s,a) = R(s,a) = \sum_{t=0}^{\infty} \Gamma^t r_t \in \mathbb{R}^m,$$

where $s_0 = s, a_0 = a$. $\Gamma = \gamma I_m$ is the discount matrix, and the multivariate Bellman equation is

$$Z(s,a) \overset{\bar{d}_1}{=} r(s,a,s') + \Gamma Z(s',a'), \tag{16}$$

where s', a' are drawn from $\mathcal{P}(\cdot|s,a)$ and policy π.

We note that the discussion of Section 4 applies to the multivariate reward case. The only modification is that in this case, the generator output and discriminator input are m-dimensional vectors.

6. Wasserstein-Distance Motivated Exploration

In this section, we propose to use our GAN-based DiRL method for exploration in MDPs. In exploration, the agent attempts to reach state-space regions where knowledge about the environment or reward function is insufficient, thereby gaining better experience for future action learning. The inability to predict the expected rewards or transitions is generally a result of one of three factors. (1) Inherent uncertainty of the environment, (2) Inadequate learning model, or (3) Poor experience. The first factor is directly related to the structure of the environment, and we assume that it is beyond our control. The second factor can be mitigated by using a richer class of models, e.g., expressive models such as deep NNs. Here, we focus on the third factor, and propose a method for directing the agent to areas in state space where its past experience is not sufficient to accurately predict the reward and transitions of the state.

The key feature of our approach to exploration is the following. Assume we have a model of $Z(s,a)$ (namely a model for the distribution of $Z(s,a)$), trained to minimize $D(Z(s,a), T^\pi Z(s,a))$ for some distributional distance D. In regions where we have rich enough experience, the approximated distance $D(Z(s,a), T^\pi Z(s,a))$ will be small, even though $Z(s,a)$ itself may have very high moments. Regions where the drawn rewards are statistically different from past experience will be characterized by a higher discrepancy. As a result, distributional learning may help us guide exploration to state-action regions where more samples are required.

We now propose an exploration method based on DiRL. Recall that every generator training step involves an update of the NN parameters $\theta \to \theta'$ according to a sampled batch of data. We measure the effect on the distributional model by inspecting the Wasserstein-1 distance, $W_1(G_\theta(\cdot|s_t,a_t), G_{\theta'}(\cdot|s_t,a_t))$.[2] Following Arjovsky et al. (2017), we assume $G_\theta(z|s,a)$ to be locally Lipschitz in (z,θ) with constants $L(z,\theta|s,a)$, such that for all (s,a),

$$\mathbb{E}_{z\sim P_z} L(z,\theta|s,a) = L(\theta|s,a) < \infty. \tag{17}$$

Arjovsky et al. (2017) showed that for any given (θ, s, a), there exists a neighborhood of θ, denoted $U_{s,a}(\theta)$, such that

[2] Although many distributional metrics may be considered, corresponding to the methods suggested in Section 4, here we use the Wasserstein-1 distance.

for all $\theta' \in U_{s,a}(\theta)$ we have

$$W_1(G_\theta(\cdot|s,a), G_{\theta'}(\cdot|s,a)) \leq L(\theta|s,a)\|\theta - \theta'\|. \quad (18)$$

That is, for small enough updates, Equation (18) bounds the distributional change of the generator output for each (s,a) by terms of the parameter difference. Using a gradient-descent method implies an update of the parameters that is proportional to the gradient,

$$\theta - \theta' \propto \nabla_\theta \mathcal{L}_\pi(G_\theta, f), \quad (19)$$

where $\nabla_\theta \mathcal{L}_\pi(G_\theta, f)$ equals

$$\nabla_\theta \mathbb{E}_{z\sim P_z, s'\sim\mathcal{P}(\cdot|s,a), a'\sim\pi(\cdot|s')} \Lambda(G_\theta, D_\omega)|_{(z,s,a,r,s',a')}.$$

Practically, we approximate over a batch of samples. Based on (13)-(15), we have that

$$\nabla_\theta \mathcal{L}_\pi(G_\theta, f) \approx \quad (20)$$
$$\hat{\mathbb{E}}_{z\sim P_z, s'\sim\mathcal{P}(\cdot|s,a), a'\sim\pi(\cdot|s')} \nabla_\theta \Lambda(G_\theta, D_\omega)|_{(z,s,a,r,s',a')},$$

where by $\hat{\mathbb{E}}$ we denote the empirical mean over a batch. Our idea is that the gradient $\nabla_\theta \Lambda(G_\theta, D_\omega)|_{(z,s,a,r,s',a')}$ is, in effect, a measure of the error in predicting the return distribution at the state-action tuple s, a. Thus, we propose to use the magnitude of the gradient as an intrinsic reward function (cf. Section 2.3). We introduce the combined reward function, where $\eta > 0$ is the trade-off parameter:

$$\hat{r}(s_t, a_t, s_{t+1}) = r(s_t, a_t, s_{t+1}) + \eta r^i(s_t, a_t, s_{t+1}), \quad (21a)$$

$$r^i(s_t, a_t, s_{t+1}) = \quad (21b)$$
$$\|\mathbb{E}_{z\sim P_z, a_{t+1}\sim\pi(\cdot|s_{t+1})} \nabla_\theta \Lambda(G_\theta, D_\omega)|_{(z,s_t,a_t,r_t,s_{t+1},a_{t+1})}\|.$$

Based on this definition, we introduce the Distributional Discrepancy Motivated Exploration (W-1ME) method, described in Algorithm 2.

6.1. Reward-Systematic Exploration

In domains where the reward is sparse, uncertainty in the value distribution may not be an informative signal for exploration. To overcome sparsity, several exploration methods enrich the reward signal using some 'intrinsic' reward, based on informative measures of transitions (see e.g. Houthooft et al. (2016)). Following the ideas of model-based exploration (Houthooft et al., 2016), we learn the transition probability $\mathcal{P}(\cdot|s,a)$, together with the value distribution.

Our main insight is that, by adopting the multivariate reward framework of Section 5, we can unify learning the distribution of return and transitions under a single GAN framework. Assume that $\mathcal{S} \subseteq \mathbb{R}^n$ has a L_2 (or equivalent) norm. Then, learning a dynamic model is a special case of (16), where the reward vector is

$$\tilde{r}(s,a,s') = \begin{pmatrix} r(s,a,s') \\ s' \end{pmatrix} \quad (22)$$

and

$$\Gamma = \begin{pmatrix} \gamma I_m & 0 \\ 0 & 0_{n\times n} \end{pmatrix}. \quad (23)$$

Thus, by simply adding the state to the NNs in the GAN generator and discriminator, and setting the discount factor appropriately, we obtain a GAN that predicts the joint distribution of the return together with the state transition. Note that, since we are learning *joint distributions*, any dependency between the return and state transition should be learned by such a GAN. In addition, the intrinsic reward in Equation (21a) can be immediately applied to this GAN, and will in this case include a reward bonus for errors both in the return distribution and in the state transition.

Algorithm 2 Distributional Discrepancy Motivated Exploration (W-1ME)

```
Input: initial policy $\pi(\cdot|s)$, trained model $(G_\theta, D_\omega)$.
Parameters: number of steps $T$, number of noise samples $N_{explore}$, trade-off parameter $\eta$.
for $t = 1$ to $T$ do
  Act according to $a_t \sim \pi(\cdot|s_t)$.
  Observe $(s_t, a_t, s_{t+1})$ and draw reward $r_t$.
  Draw $N_{explore}$ noise samples $z^{(i)} \sim \mathcal{N}(0, I)$ and actions $a_{t+1}^{(i)} \sim \pi(\cdot|s_{t+1})$ .
  Approximate (21b) by the empirical mean over sample $(s_t, a_t, r_t, s_{t+1})$:
      $r^i(s_t, a_t, s_{t+1}) = \left\| \frac{1}{N_{explore}} \sum_{i=1}^{N_{explore}} \nabla_\theta \Lambda(G_\theta, D_\omega)|_{p^{(i)}} \right\|,$
      $p^{(i)} = (z^{(i)}, s_t, a_t, r_t, s_{t+1}, a_{t+1}^{(i)}).$
  Construct combined rewards $\hat{r}(s_t, a_t, s_{t+1})$, applying (21a) using parameter $\eta$.
end for
Update policy $\pi$ using rewards $\hat{r}(s_t, a_t, s_{t+1})$ and any standard RL method.
Train $(G_\theta, D_\omega)$ using stored $(s_t, a_t, r_t, s_{t+1})$, $\pi$ and Algorithm 1.
```

7. Experimental Results

In this section we demonstrate our methods empirically. Our experiments are designed to address the following questions: (1) Can the VDGL algorithm learn accurate distributions of multivariate returns? (2) Does the W-1ME algorithm result in effective exploration?

7.1. Multivariate Policy Evaluation

Here, we aim to demonstrate that our GAN-based method can learn return distributions of multivariate returns, with non-trivial correlation structure between the return variables. For this task, we designed a toy domain where the correlation structure is easy to visualize.

We consider a multi-reward maze with a tabular state space – the Four-room maze in Fig. 2(a). The four allowed actions are moving one unit UP, DOWN, LEFT or RIGHT. We have 8 reward types corresponding to the subsets $A, \dots, H \subseteq \mathcal{S}$, where in each subset of the state space only one type of reward is active, $r(s_t, a_t, s_{t+1}) = (1-\gamma)(\delta_A(s_{t+1}), \dots, \delta_H(s_{t+1}))^T \in \mathbb{R}^8$.

Each room contains a pair of reward types, while the aisle is neutral. Thus, when exploring each room, only two reward types can be collected. We consider an agent that moves randomly in the maze. Because of the maze structure, which contains narrow passages between the rooms, the agent is likely to spend most of its time in a single room, and we therefore expect the return distribution to contain different *modes*, where in each mode only two rewards are dominant.

We trained the VDGL algorithm for 1500 episodes of 350 steps, and sampled $Z(s, a)$ at three different states (s_0, s_1, s_2). Action a was randomly chosen at each sample, where we denote the obtained value distribution $Z(s) \triangleq Z(s, a),\ a \sim U(\mathcal{A})$. Figure 2(b) shows that at each location, $Z(s)$ generated higher probability of values for near-by reward types; while samples from $Z(s_1)$ tends to present higher A and B rewards, $Z(s_2)$ predicts higher G's and H's. When starting at the center of the maze, the return distribution $Z(s_0)$ is indeed multi-modal, with each mode showing high returns for two different reward types, corresponding to one of the four rooms.

We emphasize that previous DiRL algorithms, such as C-51 (Bellemare et al., 2017), or quantile regression DQN (Dabney et al., 2017), which rely on a discretization of the value distribution, cannot be used to practically estimate an 8-dimensional return vector. As our results show, our GAN approach can effectively handle multivariate, correlated, return distributions.

7.2. DiRL-Based Exploration

To evaluate the W-1ME exploration algorithm, we propose the 2-Face Climber testbench. In the 2-Face Climber (see full description in the Appendix), a climber is about to conquer the summit of a mountain, and there are two possible ways to reach the top. The South Face is mild and easy to climb. The North face is slippery and much harder to climb, but the route is shorter, and reaching the top bears a greater reward. The climb starts at camp ($s_0 = 0$), where the climber chooses the face to climb. Then, at each step, there are two possible actions. One action progresses the climber towards the summit, while the effect of the other action depends on the mountain face. On the South Side, it causes the climber to slip, and move back to the previous state, while on the North Side, with some probability it can also cause her to fall several steps, or even restart at base-camp. The idea here is that once the model is known, it is easy to always take actions that progress towards the summit, and in this case the north side should be preferred. During exploration, however, the south side is more forgiving when taking a wrong action.

Figure 3 shows the state-space visit counts of W-1ME exploration using different values of η in (21a), where we used reward-systematic exploration (Sec. 6.1). For policy improvement in Algorithm 2 we used DQN, where we refer to the full algorithm as W-1ME+DQN. $\eta = 0$ is for DQN, where we applied an ϵ-greedy exploration ($\epsilon = 0.05$). We run over 100 independent seeds, with 1000 episodes at each experiment. Here we can see that indeed, higher exploration η's increment the visit rate to the North face states, resulting in higher average returns (for detailed results we refer the reader to the Appendix).

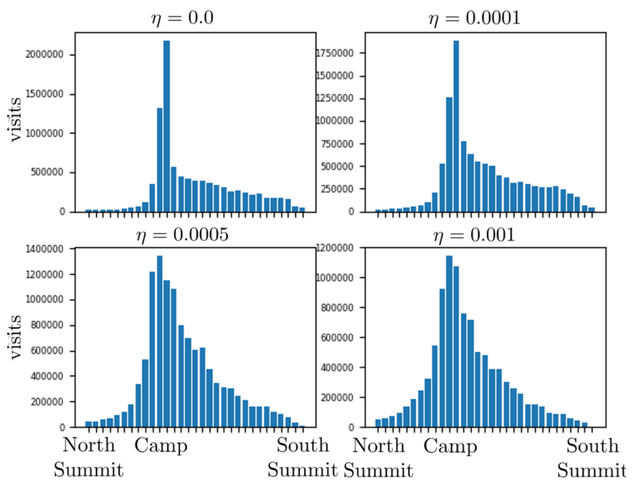
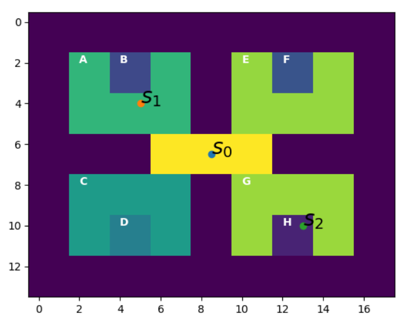


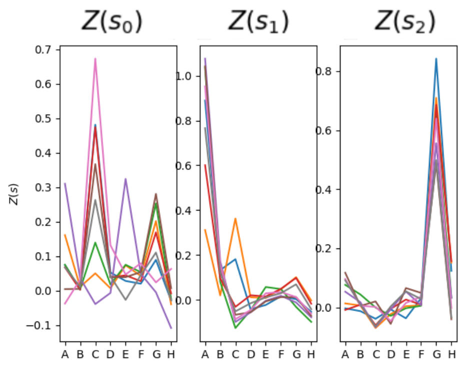
Figure 3. W-1ME exploration for the 2Face climber. Histograms present the number of visits to each state. Observe that higher η's incremented the visit rate to the North face states, resulting in higher average returns. This shows the utility of our exploration method.

7.3. Exploration in Continuous Domains

In this section, we evaluate W-1ME on continuous control tasks: a LQR setup with a Gaussian cost, and sparse reward tasks CartPoleSwingup and SwimmerGather (Houthooft et al., 2016). Here we used TRPO (Schulman et al., 2015) as a basic RL method, and the reward-systematic exploration (Sec. 6.1). For exact setup we refer the reader to the Appendix.

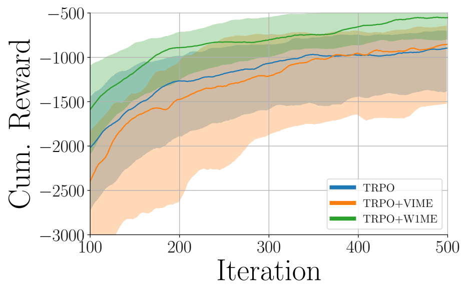


(a)

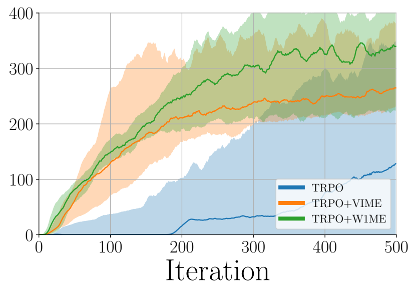


(b)

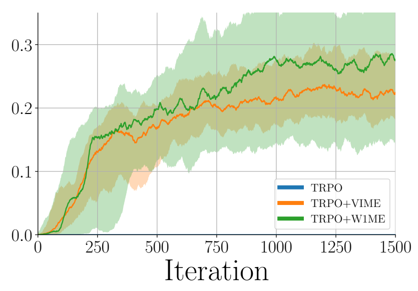
Figure 2. Random policy evaluation on multi-reward maze. (a) maze configuration; (b) generated $Z(s)$ samples at different locations, where each line represents a sample in $\mathbb{R}^8$. Note that when starting at the center of the maze (s_0), the distribution contains several modes, where in each mode only two reward types have a high return. Conversely, when starting at s_1, only returns of type A and B are observed.



(a) LQR



(b) CartpoleSwingup



(c) SwimmerGather

Figure 4. W-1ME vs. VIME exploration on continuous control tasks. W-1ME exploration outperforms TRPO+VIME and TRPO on both (a) and (b), and is comparable to VIME on (c). The cumulative reward is displayed.

Figure 4 shows the median of the averaged cumulative reward (at each iteration) for each algorithm over a fixed set of random seeds (20 for LQR, 10 for CartpoleSwingup, 8 for SwimmerGather). In 4(c) the median is averaged over a window of 15 iterations. Shaded areas represent the interquartile range. For the first two tasks, we set $\eta = 10^{-7}$ which gave the best results for both exploration methods. For Swimmergather, we set $\eta = 10^{-4}$, as used by Houthooft et al. (2016) in their code.

We can see that W-1ME exploration outperforms both VIME and plain TRPO on LQR and CartpoleSwingup tasks, and is comparable to VIME on SwimmerGather environment. This shows that in domains where reward uncertainty dominates, such as in random cost LQR, our method that exploits this uncertainty for exploration is preferred, while in domains where state uncertainty is most important, such as in SwimmerGather, we are not worse than the state of the art.

8. Conclusion and Outlook

In this work we showed an interesting equivalence between the distributional Bellman equation and GANs. Based on this equivalence, we proposed a GAN based algorithm for DiRL, which can handle high-dimensional, multivariate rewards. We also showed that the multivariate reward formulation allows to unify learning of return and next state distributions, and we proposed a novel exploration method based on this idea, where the prediction error in both return and next state distribution is used as an intrinsic reward. We empirically validated our methods in several RL domains.

Our work paves the way for investigations of a distributional approach to multi-objective RL (Vamplew et al., 2011). Such would require a distributional *policy optimization* algorithm that can exploit the multi-variate reward distribution from the Bellman GAN. Our unified method for learning state and value distributions also suggests a new direction for model-based RL in high-dimensional state spaces.

Acknowledgements

The work is partially supported by grant 451/17 from the Israel Science Foundation, by the Ollendorff Center of the Viterbi Faculty of Electrical Engineering at the Technion, and by the Skillman chair in biomedical sciences. Support from NVIDIA Corporation with the donation of the Titan Xp GPU is also acknowledged. The authors wish to thank Josh Achiam for helpful discussions.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Asadi, K., Cater, E., Misra, D., and Littman, M. L. Equivalence between wasserstein and value-aware model-based reinforcement learning. *arXiv preprint arXiv:1806.01265*, 2018.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*, 2017.

Bertsekas, D. *Dynamic programming and optimal control*, volume 1. Athena Scientific, 1995.

Bertsekas, D. Dynamic programming and optimal control: Volume II. 2005.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. Distributional reinforcement learning with quantile regression. *arXiv preprint arXiv:1710.10044*, 2017.

Doan, T., Mazoure, B., and Lyle, C. Gan q-learning. *arXiv preprint arXiv:1805.04874*, 2018.

Dosovitskiy, A. and Koltun, V. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117, 2016.

Kearns, M. and Singh, S. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Mannor, S. and Shimkin, N. A geometric approach to multi-criterion reinforcement learning. *Journal of machine learning research*, 5(Apr):325–360, 2004.

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

Osband, I., Russo, D., and Van Roy, B. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pp. 3003–3011, 2013.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2): 265–286, 2007.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Schmidhuber, J. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

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

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

Tang, Y. and Agrawal, S. Exploration by distributional reinforcement learning. *arXiv preprint arXiv:1805.01907*, 2018.

Tewari, A. and Bartlett, P. L. Optimistic linear programming gives logarithmic regret for irreducible mdps. In *Advances in Neural Information Processing Systems*, pp. 1505–1512, 2008.

Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., and Dekker, E. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine learning*, 84(1-2):51–80, 2011.

Villani, C. *Optimal transport: old and new*. Springer Science & Business Media, 2008.

Zhao, Y., Kosorok, M. R., and Zeng, D. Reinforcement learning design for cancer clinical trials. *Statistics in medicine*, 28(26):3294–3315, 2009.