Nonlinear Distributional Gradient Temporal-Difference Learning

Chao Qu [1] **Shie Mannor** [2] **Huan Xu** [3 4]

Abstract

We devise a distributional variant of gradient temporal-difference (TD) learning. Distributional reinforcement learning has been demonstrated to outperform the regular one in the recent study (Bellemare et al., 2017a). In the policy evaluation setting, we design two new algorithms called distributional GTD2 and distributional TDC using the Cramér distance on the distributional version of the Bellman error objective function, which inherits advantages of both the nonlinear gradient TD algorithms and the distributional RL approach. In the control setting, we propose the distributional Greedy-GQ using the similar derivation. We prove the asymptotic almost-sure convergence of distributional GTD2 and TDC to a local optimal solution for general smooth function approximators, which includes neural networks that have been widely used in recent study to solve the real-life RL problems. In each step, the computational complexities of above three algorithms are linear w.r.t. the number of the parameters of the function approximator, thus can be implemented efficiently for neural networks.

1. Introduction

Reinforcement learning (RL) considers a problem where an agent interacts with the environment to maximize the cumulative reward trough time. A standard approach to solve the RL problem is called value function based reinforcement learning, which finds a policy that maximizes the value function $V(s)$ (Sutton & Barto, 1998). Thus, the estimation of the value function of a given stationary policy of a Markov Decision Process (MDP) is an important subroutine of *generalized policy iteration* (Sutton & Barto, 1998) and a key intermediate step to generate good control policy (Gelly & Silver, 2008; Tesauro, 1992). The value function is known to solve the Bellman equation, which succinctly describes the recursive relation on state-action value function $Q(s,a)$.

$$Q^{\pi}(s,a) = \mathbb{E}R(s,a) + \gamma \mathbb{E}_{s',a'} Q^{\pi}(s',a'),$$

where the expectation is taken over the next state $s' \sim P(\cdot|s,a)$, the reward $R(s,a)$ and the action a' from policy π, γ is the discount factor. Hence, many RL algorithms are based on the idea of solving the above Bellman equation in a sample driven way, and one popular technique is the temporal-difference (TD) learning (Sutton & Barto, 1998).

The last several years have witnessed the success of the TD learning with the value function approximation (Mnih et al., 2015; Van Hasselt et al., 2016), especially when using a deep neural network. In their seminal work, Tsitsiklis & Van Roy (1996) proved that the TD(λ) algorithm converges when a *linear function approximator* is implemented and states are sampled according to the policy evaluated (sometimes referred as *on-policy setting* in RL literature). However, if either the function approximator is non-linear, or the on-policy setting does not hold, there are counterexamples that demonstrate that TD(λ) may diverge. To mitigate this problem, a family of TD-style algorithms called Gradient Temporal Difference (GTD) are proposed by (Sutton et al., 2009b;a) that address the instability of the TD algorithm with the linear function approximator in the off-policy setting. These works rely on the objective function called mean-squared projected Bellman error (MSPBE) whose unique optimum are the fixed points of the TD(0) algorithm. Bhatnagar et al. (2009) extend this idea to the non-linear smooth function approximator (e.g., neural networks) and prove the convergence of the algorithm under mild conditions. In the control setting, Maei et al. (2010) propose Greedy-GQ which has similar objective function as MSPBE but w.r.t. the Bellman optimality operator.

Recently, the distributional perspective on reinforcement learning has gained much attention. Rather than study on the expectation of the long term return (i.e., $Q(s,a)$), it explicitly takes into consideration the stochastic nature of the long term return $Z(s,a)$ (whose expectation is $Q(s,a)$). The recursion of $Z(s,a)$ is described by the distributional

[1]Ant Financial Services Group, Hang Zhou, China [2]Faculty of Electrial Engineering, Technion, Haifa, Israel [3]Alibaba Group, Seattle, USA [4]H. Milton Stewart School of Industrial and Systems Engineering, Georgia Tech, Atlanta, USA. Correspondence to: Chao Qu <luoji.qc@antfin.com>.

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

Bellman equation as follows,

$$Z(s,a) \overset{D}{=} R(s,a) + \gamma Z(s',a'), \tag{1}$$

where $\overset{D}{=}$ stands for "equal in distribution" (see Section 2 for more detailed explanations). The distributional Bellman equation essentially asserts that the distribution of Z is characterized by the reward R, the next random state-action (s',a') following policy π and its random return $Z(s',a')$. Following the notion in (Bellemare et al., 2017a) we call Z the *value distribution*. Bellemare et al. (2017a) showed that for a fixed policy the Bellman operator over value distributions is a γ-contraction in *a maximal form of the Wasserstein metric*, thus making it possible to learn the value distribution in a sample driven way. There are several advantages to study the value distributions: First, real-life decision makers sometimes are interested in seeking big wins on rare occasions or avoiding a small chance of suffering a large loss. For example, in financial engineering, this risk-sensitive scenario is one of the central topics. Because of this, risk-sensitive RL has been an active research field in RL (Heger, 1994; Defourny et al., 2008; Bagnell & Schneider, 2003; Tamar et al., 2016), and the value distribution obviously provides a very useful tool in designing risk-sensitive RL algorithms. Second, it can model the uncertainty. Engel et al. (2005) leveraged the distributional Bellman equation to define a Gaussian process over the unknown value function. Third, from the algorithmic view, as discussed in (Bellemare et al., 2017a), the distributional Bellman operator preserves multimodality in value distribution, which leads to more stable learning. From the exploration-exploitation tradeoff perspective, if the value distribution is known, the agent can explore the region with high uncertainty, which is often called "optimism in the face of uncertainty" (Kearns & Singh, 2002; O'Donoghue et al., 2017).

Contributions: Although distributional approaches on RL (e.g., C51 in (Bellemare et al., 2017a)) have shown promising results, theoretical properties of them are not well understood yet, especially when the function approximator is nonlinear. As nonlinear function approximation is inevitable if we hope to combine RL with deep neural networks – a paradigm with tremendous recent success that enables automatic feature engineering and end-to-end learning to solve the real problem. Therefore, to extend the applicability of the distributional approach to the real problem and close the gap between the theory and practical algorithms, we propose the nonlinear distributional gradient temporal-difference learning. It inherits the merits of non-linear gradient TD and distributional approaches mentioned above. Using the similar heuristic, we also propose a distributional control algorithm called distributional Greedy-GQ.

The contributions of this paper are the following.

- We propose a *distributional MSPBE (D-MSPBE)* as the objective function to optimize, which is an extension of MSPBE when the stochastic nature of the random return is considered.
- We derive two stochastic gradient algorithms to optimize the D-MSPBE using the weight-duplication trick in (Sutton et al., 2009b; Bhatnagar et al., 2009). In each step, the computational complexity is linear w.r.t. the number of parameters of the function approximator, thus can be efficiently implemented for neural networks.
- We propose a distributional RL algorithm in the control setting called distributional Greedy-GQ, which is an distributional counterpart of (Maei et al., 2010).
- We prove distributional GTD2 and TDC converge to a local optimal solution in the policy evaluation setting under mild conditions using the two-time-scale stochastic approximation argument. If the linear function approximator is applied we have the finite sample bound.

Remarks: More precisely, we have m addition operations in each step of the algorithm (see our algorithm) but the costs of them are negligible compared to computations in neural networks in general. Thus the computational complexity in each step is still linear w.r.t. the number of parameters of the function approximator (neural networks).

2. Problem setting and preliminaries

We consider a standard setting in reinforcement learning, where an agent acts in a stochastic environment by sequentially choosing actions over a sequence of time steps, in order to maximize the cumulative reward (Sutton & Barto, 1998). This problem is formulated as a Markov Decision Process (MDP) which is a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$: $\mathcal{S}$ is the finite state space, $\mathcal{A}$ is the finite action space, $\mathcal{P} = (P(s'|s,a))_{s,s' \in \mathcal{S}, a \in \mathcal{A}}$ are the transition probabilities, $R = (R(s,a))_{s,s' \in \mathcal{S}, a \in \mathcal{A}}$ are the real-valued immediate rewards and $\gamma \in (0,1)$ is the discount factor. A policy is used to select actions in the MDP. In general, the policy is stochastic and denoted by π, where $\pi(s_t, a_t)$ is the conditional probability density at a_t associated with the policy. We also define $R^\pi(s) = \sum_{a \in \mathcal{A}} \pi(s,a) R(s,a)$, $P^\pi(s,s') = \sum_{a \in \mathcal{A}} \pi(s,a) P(s'|s,a)$.

Suppose the policy π to be evaluated is followed and it generates a trajectory $(s_0, a_0, r_1, s_1, a_1, r_2, s_2, ...)$. We are given an infinite sequence of 3-tuples (s_t, r_t, s'_t) that satisfies the following assumption.

Assumption 1. $(s_t)_{t \geq 0}$ is an $\mathcal{S}$-valued stationary Markov process, $s_t \sim d(\cdot)$, $r_t = R^\pi(s_t)$ and $s'_t \sim P^\pi(s_t, \cdot)$.

Here $d(\cdot)$ denotes the probability distribution over initial states for a transition. Since stationarity is assumed, we can drop the index t in the t^{th} transition (s_t, r_t, s'_t) and use (s, r, s') to denote a random transition. The (state-action) value function Q^π of a policy π describes the expected return from taking action $a \in \mathcal{A}$ from state $s \in \mathcal{S}$.

$$Q^\pi(s,a) := \mathbb{E}\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t),$$

$s_t \sim P(\cdot|s_{t-1}, a_{t-1}), a_t \sim \pi(\cdot|s_t), s_0 = s, a_0 = a$. It is well known that the value function $Q^\pi(s,a)$ satisfies the Bellman equation. $Q^\pi(s,a) = \mathbb{E}R(s,a) + \gamma\mathbb{E}Q^\pi(s',a')$. Define the Bellman operator as $(\mathcal{T}Q^\pi)(s,a) := \mathbb{E}R(s,a) + \gamma\mathbb{E}Q^\pi(s',a')$, then the Bellman equation becomes $Q^\pi = \mathcal{T}Q^\pi$. To lighten the notation, from now on we may drop the superscript π when the policy to be evaluated is kept fixed.

2.1. Distributional Bellman equation and Cramér distance

Recall that the return Z is the sum of discounted reward along the agent's trajectory of interactions with the environment, and hence $Q(s,a) = \mathbb{E}Z(s,a)$. When the stochastic nature of the return Z is considered, we need a distributional variant of the Bellman equation which the *distribution of* Z satisfies. Following the notion in (Bellemare et al., 2017a), we define the transition operator P^π:

$$P^\pi Z(s,a) \overset{D}{:=} Z(s',a'), \quad s' \sim P(\cdot|s,a), a' \sim \pi(\cdot|s'),$$

where $A \overset{D}{:=} B$ indicates that the random variable A is distributed according to the same law of B. The distributional Bellman operator $\mathcal{T}^\pi$ is $\mathcal{T}^\pi Z(s,a) \overset{D}{:=} R(s,a) + \gamma P^\pi Z(s,a)$. For more rigorous definition and discussions on this operator, we refer reader to (Bellemare et al., 2017a).

Bellemare et al. (2017a) prove that the distributional Bellman operator is a γ-contraction in a maximal form of the Wasserstein metric. However as pointed by them (see proposition 5 in their paper), in practice, it is hard to estimate the Wasserstein distance using samples and furthermore the gradient estimation w.r.t. the parameter of the function approximator is biased in general. Thus KL divergence is implemented instead in the algorithm C51 rather than the Wasserstein metric. However the KL divergence may not be robust to the discrepancies in support of distribution (Arjovsky et al., 2017). In this paper, we adapt the Cramér distance (Székely, 2003; Bellemare et al., 2017b) instead, since the unbiased sample gradient estimaton of Cramér distance can be easily obtained in the setting of reinforcement learning (Bellemare et al., 2017b). The square root of Cramér distance is defined as follows: Suppose there are two distributions P and Q and their cumulative distribution functions are F_P and F_Q respectively, then the square root of Cramér distance between P and Q is

$$\ell_2(P,Q) := \Big(\int_{-\infty}^{\infty} (F_P(x) - F_Q(x))^2 dx\Big)^{1/2}.$$

Intuitively, it can be thought of as the two norm on the distribution function. Indeed, the distributional Bellman operator is a $\sqrt{\gamma}$-contraction in a maximal form of the square root of Cramér distance. Here, for two random return Z_1, Z_2 with distribution P_{Z_1} and P_{Z_2}, a maximal form of the square root of Cramér distance is defined as $\bar{\ell}_2(P_{Z_1}, P_{Z_2}) = \sup_{s,a} \ell_2(P_{Z_1}(s,a), P_{Z_2}(s,a))$.

Proposition 1. *$\bar{\ell}_2$ is a metric over value distributions and $\mathcal{T}^\pi$ is a $\sqrt{\gamma}-$ contraction in $\bar{\ell}_2$.*

The proof is similar to Theorem 2 in (Bellemare et al., 2017b) and and proposition 2 in (Rowland et al., 2018).

2.2. Gradient TD and Greedy-GQ

We now review linear and nonlinear gradient TD and Greedy-GQ proposed by (Sutton et al., 2009b; Bhatnagar et al., 2009; Maei et al., 2010), which helps to better understand the nonlinear distributional gradient TD and distributional Greedy-GQ. One approach in reinforcement learning for large scale problems is to use a linear function approximation for the value function V. Particularly, the value function is $\hat{V}(s) = \theta^T\phi(s)$, where the feature map is $\phi : \mathcal{S} \to \mathbb{R}^d$, and $\theta \in \mathbb{R}^d$ is the parameter of the linear model. The objective function of the gradient TD family is the mean squared projected Bellman error (MSPBE).

$$MSPBE(\theta) = \frac{1}{2}\|\hat{V} - \Pi T\hat{V}\|_D^2, \tag{2}$$

where $\hat{V}$ is the vector of value function over $\mathcal{S}$, D is a diagonal matrix with diagonal elements being the stationary distribution $d(s)$ over $\mathcal{S}$ induced by the policy π, and Π is the weighted projection matrix onto the linear space spanned by $\phi(1), ..., \phi(|\mathcal{S}|)$, which is $\Pi = \Phi(\Phi^T D\Phi)^{-1}\Phi^T D$. Substitute Π into (2), the MSPBE can be written as

$$\begin{aligned} MSPBE(\theta) =& \frac{1}{2}\|\Phi^T D(\hat{V} - T\hat{V})\|^2_{(\Phi^T D\Phi)^{-1}} \\ =& \frac{1}{2}\mathbb{E}[\delta\phi]^T \mathbb{E}[\phi\phi^T]^{-1}\mathbb{E}[\delta\phi], \end{aligned} \tag{3}$$

where δ is the TD error for a given transition (s,r,s'), i.e., $\delta = r + \gamma\theta^T\phi' - \theta^T\phi$. Its negative gradient is $-\frac{1}{2}\nabla MSPBE(\theta) = \mathbb{E}[(\phi - \gamma\phi')\phi^T]w$, where $w = \mathbb{E}[\phi\phi^T]^{-1}\mathbb{E}[\delta\phi]$. Sutton et al. (2009b) use the weight-duplication trick to update w on a "faster" time scale as follows $w_{t+1} = w_t + \beta_t(\delta_t - \phi_t^T w_t)\phi_t$. Two different ways to update θ leads to GTD2 and TDC.

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha_t(\phi_t - \gamma\phi'_t)(\phi_t^T w_t) \quad (GTD2), \\ \theta_{t+1} &= \theta_t + \alpha_t\delta_t\phi_t - \alpha_t\gamma\phi'_t(\phi_t^T w_t) \quad (TDC). \end{aligned} \tag{4}$$

Once the nonlinear approximation is used, we can optimize a slightly different version of MSPBE. There is an additional term h_t in the update rule

$$\theta_{t+1} = \theta_t + \alpha_t\{(\phi_t - \gamma\phi_t')(\phi_t^T w_t) - h_t\} \ \ GTD2$$

See more discussion in section 3.

Similarly, Greedy-GQ optimizes the following objective function,

$$J(\theta) = \|\Pi T^{\pi(\theta)} Q_\theta - Q_\theta\|_D^2,$$

where π_θ is a greedy policy w.r.t. Q_θ. Reusing the weight-duplication trick, Maei et al. (2010) give the update rule.

$$\theta_{t+1} = \theta_t + \alpha_t[\delta_t\phi_t - \gamma(w_t^T\phi)\hat{\phi}],$$
$$w_{t+1} = w_t + \beta_t[\delta_{t+1} - \phi^T w_t]\phi_t,$$

where $\hat{\phi}$ is an unbiased estimate of expected value of the next state under π_θ.

3. Nonlinear Distributional Gradient TD

In this section, we propose distributional Gradient TD algorithms by considering the Cramér distance between value distribution of $Z(s,a)$ and $\mathcal{T}Z(s,a)$ which is a distributional counterpart of (Bhatnagar et al., 2009). To ease the exposition, in the following we consider the value distribution on state s rather than the state-action pair (s,a) since the extension to (s,a) is straightforward.

3.1. Distributional MSPBE (D-MSPBE)

Suppose there are $|\mathcal{S}| = n$ states. One simple choice of the objective function is as follows

$$\sum_{i=1}^{n} d(s_i)\ell_2^2(Z(s_i), \mathcal{T}Z(s_i)). \tag{5}$$

However, a major challenge to optimize (5) is the double sampling problem (Dann et al., 2014), i.e., two independent samples are required from each state . To see that, notice that if we only consider the *expectation* of the return, (5) reduces to the mean squared Bellman error (MSBE), and the corresponding algorithms to minimize MSBE are the well-known residual gradient algorithms (Baird, 1995), which is known to require two independent samples for each state (Dann et al., 2014). To get around the double sampling problem, we instead adapt MSPBE into its distributional version. To simplify the problem, we follow (Bellemare et al., 2017a) and assume that the value distribution is discrete with range $[V_{\min}, V_{\max}]$ and whose support is the set of atoms $\{z_j = V_{\min} + (j-1)\Delta z : 1 \le j \le m\}$, $\Delta z := \frac{V_{\max} - V_{\min}}{m-1}$. In practice $V_{\min}$ and $V_{\max}$ are not hard to get. For instance, suppose we know the bound on the reward $|r| < b$, then we can take $V_{\min}, V_{\max}$ as $\pm\frac{b}{1-\gamma}$. We further assume that the atom probability can be given by a parametric model θ such as a softmax function:

$$p_\theta(s_i, z_j) = \frac{\exp(l_\theta(s_i, z_j))}{\sum_{j=1}^m \exp(l_\theta(s_i, z_j))},$$

where $\ell_\theta(s_i, z_j)$ can be a non-linear function, e.g., a neural network. From an algorithmic perceptive, such assumption or approximation is necessary, since it is hard to represent the full space of probability distributions.

We denote the (cumulative) distribution function of $Z(s)$ as $F_\theta(s,z)$. Notice $F_\theta(s,z)$ is non-linear w.r.t. θ in general, thus it is not restricted to a hyperplane as that in the linear function approximation. Following the non-linear gradient TD in (Bhatnagar et al., 2009), we need to define the tangent space at θ. Particularly, we denote $F_\theta \in \mathbb{R}^{nm\times 1}$ as a vector of $F_\theta(s_i, z_j), i = 1, ..., n, j = 1, ...m$ and assume F_θ is a differentiable function of θ. $\mathcal{M} = \{F_\theta \in \mathbb{R}^{nm\times 1} | \theta \in \mathbb{R}^d\}$ becomes a differentiable submanifold of $\mathbb{R}^{mn}$. Define $\phi_\theta(s,z) = \frac{\partial F_\theta(s,z)}{\partial \theta}$, then the tangent space at θ is $T\mathcal{M}_\theta = \{\Phi_\theta a | a \in \mathbb{R}^d\}$, where $\Phi_\theta \in \mathbb{R}^{mn\times d}$ is defined as $(\Phi_\theta)_{((i,j),l)} = \frac{\partial}{\partial \theta_l} F_\theta(s_i, z_j)$, i.e., each row of it is $\phi_\theta^T(s_i, z_j)$. Let Π_θ be the projection that projects vectors to $T\mathcal{M}$. Particularly, to project the distribution function $F(s,z)$ onto the $\mathcal{T}M$ w.r.t. the Cramér distance, we need to solve the following problem

$$\hat{\alpha} = \arg\min_\alpha \sum_{i=1}^{n}\sum_{j=1}^{m} d(s_i)\big(F(s_i, z_j) - \phi_\theta(s_i, z_j)^T\alpha\big)^2, \tag{6}$$

where $F(s_i, z_j)$ is the value of distribution function of $Z(s_i)$ at z_j. Since this is a least squares problem, we have that the projection operator has a closed form

$$\Pi_\theta = \Phi_\theta(\Phi_\theta^T D \Phi_\theta)^{-1}\Phi_\theta^T D,$$

where D is a $nm \times nm$ diagonal matrix with diagonal elements being $d(s_1)I^{m\times m}, d(s_2)I^{m\times m}, ..., d(s_n)I^{m\times m}$. Given this projection operator, we propose the distributional MSPBE (D-MSPBE). Particularly, the objective function to optimize is as follows

$$\underset{\theta}{\text{minimize}}: \quad \|F_\theta - \Pi G_\theta\|_D^2,$$

where $G_\theta \in \mathbb{R}^{nm\times 1}$ is the vector form of $G_\theta(s_i, z_j)$, and $G_\theta(s_i, z_j)$ is the value of distribution function of $\mathcal{T}Z(s_i)$ at atom z_j. Assume $(\Phi_\theta^T D \Phi_\theta)^{-1}$ is non-singular, similar to the MSPBE, we rewrite the above formulation into another form.

D-MSPBE:

$$\underset{\theta}{\text{minimize}}: \quad J(\theta) := \|\Phi_\theta^T D(F_\theta - G_\theta)\|^2_{(\Phi_\theta^T D \Phi_\theta)^{-1}}. \tag{7}$$

To better understand D-MSPBE, we compare it with MSPBE. First, in equation (3), we have the term $\hat{V} - T\hat{V}$, which is the difference between the value function $\hat{V}$ and $T\hat{V}$, while we have the difference between the distribution of Z and TZ in equation (7). Second, the D matrix is slightly different, since in each state we need m atoms to describe the value distribution. Thus we have the diagonal element as $d(s_i)I^{m\times m}$. Third, Φ_θ is a gradient for F_θ and thus depends on the parameter θ rather than a constant feature matrix, which is similar to (Bhatnagar et al., 2009).

3.2. Distributional GTD2 and Distributional TDC

In this section, we use the stochastic gradient to optimize the D-MSPBE (equation 7) and derive the update rule of distributional GTD2 and distributional TDC.

The first step is to estimate $\Phi_\theta^T D(F_\theta - G_\theta)$ from samples. We denote $\hat{G}_\theta$ as the empirical distribution of G_θ and $\mathbb{E}\hat{G}_\theta = G_\theta$. Notice one unbiased empirical distribution $\hat{G}_\theta(s,\cdot)$ at state s is the distribution of $r+\gamma Z(s')$, whose distribution function is $F_\theta(s', \frac{z-r}{\gamma})$ by simply shifting and shrinking the distribution of $Z(s')$. Then we have

$$\Phi_\theta^T D(F_\theta - G_\theta) = \mathbb{E}\sum_{j=1}^m \phi_\theta(s,z_j)\big(F_\theta(s,z_j) - \hat{G}_\theta(s,z_j)\big)$$

Then we can write D-MSPBE in the following way

$$\begin{aligned}J(\theta) = {}&\mathbb{E}\big(\sum_{j=1}^m \phi_\theta(s,z_j)(F_\theta(s,z_j) - \hat{G}_\theta(s,z_j))\big)^T \times\\ &\big(\mathbb{E}\sum_{j=1}^m \phi_\theta(s,z_j)\phi_\theta^T(s,z_j)\big)^{-1}\big(\mathbb{E}\sum_{j=1}^m \phi_\theta(s,z_j)\times\\ &(F_\theta(s,z_j) - \hat{G}_\theta(s,z_j))\big).\end{aligned}$$

We define $\boldsymbol{\delta_\theta(s,z_j)} = \hat{G}_\theta(s,z_j) - F_\theta(s,z_j)$, analogous to the temporal difference, and call it *temporal distribution difference*. To ease the exposition, we denote $A = \mathbb{E}\sum_{j=1}^m \phi_\theta(s,z_j)\phi_\theta^T(s,z_j)$. Then we have

$$\begin{aligned}J(\theta) = {}&\mathbb{E}\big(\sum_{j=1}^m \phi_\theta(s,z_j)\delta_\theta(s,z_j)\big)^T A^{-1}\times\\ &\mathbb{E}\sum_{j=1}^m \big(\phi_\theta(s,z_j)\delta_\theta(s,z_j)\big).\end{aligned} \tag{8}$$

In the following theorem, we choose $\hat{G}_\theta(s,z) = F_\theta(s', \frac{z-r}{\gamma})$, an unbiased empirical distribution we mentioned above and give the gradient of D-MSPBE w.r.t. θ. We defer the proof to the appendix.

Theorem 1. *Assume that $F_\theta(s,z_j)$ is twice continuously differentiable in θ for any $s\in\mathcal{S}$ and $j\in\{1,...,m\}$, $\mathbb{E}\sum_{j=1}^m \phi_\theta(s,z_j)\phi_\theta^T(s,z_j)$ is non-singular in a small neighborhood of θ. Denote $h = \mathbb{E}\sum_{j=1}^m(\delta_\theta(s,z_j) - w^T\phi_\theta(s,z_j))\nabla_\theta^2 F_\theta(s,z_j)w$, then we have*

$$\begin{aligned}-\frac{1}{2}\nabla_\theta J(\theta) = {}&\mathbb{E}\sum_{j=1}^m \big(\phi_\theta(s,z_j) - \phi_\theta(s', \frac{z_j - r}{\gamma})\big)\times\\ &\phi_\theta^T(s,z_j)w - h,\end{aligned} \tag{9}$$

which has another form

$$\begin{aligned}-\frac{1}{2}\nabla_\theta J(\theta) = {}&-\mathbb{E}\sum_{j=1}^m \phi_\theta(s', \frac{z_j - r}{\gamma})\phi_\theta^T(s,z_j)w\\ &+\mathbb{E}\sum_{j=1}^m(\phi_\theta(s,z_j)\delta_\theta(s,z_j)) - h,\end{aligned} \tag{10}$$

where $w = A^{-1}\mathbb{E}\sum_{j=1}^m \big(\phi_\theta(s,z_j)\delta_\theta(s,z_j)\big)$.

Based on Theorem 1, we obtain the algorithm of distributional GTD2 and distributional TDC. Particularly (9) leads to Algorithm 1, and (10) leads to Algorithm 2. The difference between distributional gradient TD methods and regular ones are highlighted in boldface.

Algorithm 1 Distributional GTD2 for policy evaluation

```
**Input:** step size $\alpha_t$, step size $\beta_t$, policy $\pi$.
**for** $t = 0, 1, ...$ **do**
    $w_{t+1} = w_t + \beta_t \sum_{\boldsymbol{j=1}}^{\boldsymbol{m}} \big(-\phi_{\theta_t}^T(s_t,z_j)w_t + \boldsymbol{\delta_{\theta_t}}\big)\phi_{\theta_t}(s_t,z_j)$
    $\theta_{t+1} = \Gamma[\theta_t + \alpha_t\{\sum_{\boldsymbol{j=1}}^{\boldsymbol{m}} \big(\phi_{\theta_t}(s_t,z_j) - \boldsymbol{\phi_{\theta_t}(s_{t+1}, \frac{z_j - r_t}{\gamma})}\big)\phi_{\theta_t}^T(s_t,z_j)w_t - h_t\}]$
    $\Gamma : \mathbb{R}^d \to \mathbb{R}^d$ is a projection onto an compact set $C$ with a smooth boundary.
    $h_t = \sum_{\boldsymbol{j=1}}^{\boldsymbol{m}}(\boldsymbol{\delta_{\theta_t}} - w_t^T\phi_{\theta_t}(s_t,z_j))\nabla^2 F_{\theta_t}(s_t,z_j)w_t$,
    where $\boldsymbol{\delta_{\theta_t}} = F_{\theta_t}(s_{t+1}, \frac{z_j - r_t}{\gamma}) - F_{\theta_t}(s_t,z_j)$.
**end for**
```

Some remarks are in order. We use distributional GTD2 as an example, but all remarks hold for the distributional TDC as well.

(1). We stress the difference between the the update rule of GTD2 in (4) and that of the distributional GTD2 (highlighted in boldface): In the distributional GTD2, we use the temporal distribution difference $\boldsymbol{\delta_{\theta_t}}$ instead of the temporal difference in GTD2. Also notice there is a summation over z_j, which corresponds to the integral in the Cramér distance, since we need the difference over the whole distribution rather than a certain point. The term $\frac{z_j - r_t}{\gamma}$ comes from

Algorithm 2 Distributional TDC for policy evaluation

```
Input: step size $\alpha_t$, step size $\beta_t$, policy $\pi$.
for $t = 0, 1, ...$ do
    $$w_{t+1} = w_t + \beta_t \sum_{j=1}^{m} \left(-\phi_{\theta_t}^T(s_t, z_j) w_t + \boldsymbol{\delta_{\theta_t}}\right) \phi_{\theta_t}(s_t, z_j).$$
    $$\theta_{t+1} = \Gamma[\theta_t + \alpha_t\{\sum_{j=1}^{m} \left(\boldsymbol{\delta_{\theta_t}} \phi_{\theta_t}(s_t, z_j) - \boldsymbol{\phi_{\theta_t}(s_{t+1}, \frac{z_j - r_t}{\gamma}}) (\phi_{\theta_t}^T(s_t, z_j) w_t)\right) - h_t\}].$$
    $\Gamma : \mathbb{R}^d \to \mathbb{R}^d$ is a projection onto an compact set $C$ with a smooth boundary.
    $h_t = \sum_{j=1}^{m} (\boldsymbol{\delta_{\theta_t}} - w_t^T \phi_{\theta_t}(s_t, z_j)) \nabla^2 F_{\theta_t}(s_t, z_j) w_t$,
    where $\boldsymbol{\delta_{\theta_t}} = F_{\theta_t}(s_{t+1}, \frac{z_j - r_t}{\gamma}) - F_{\theta_t}(s_t, z_j)$.
end for
```

the shifting and shrinkage on the distribution function of $Z(s_{t+1})$.

(2). The term h_t results from the nonlinear function approximation, which is zero in the linear case. This term is similar to the one in nonlinear GTD2 (Bhatnagar et al., 2009). Notice we do not need to explicitly calculate the Hessian in the term $\nabla^2 F_{\theta_t}(s, z) w$. This term can be evaluated using forward and backward propagation in neural networks with the complexity scaling linearly w.r.t. the number of parameters in neural networks, see the work (Pearlmutter, 1994) or chapter 5.4.6 in (Christopher, 2016). We give an example in the appendix to illustrate how to calculate this term.

(3). It is possible that $\frac{z_j - r_t}{\gamma}$ is not on the support of the distribution in practice. Thus we need to approximate it by projecting it on the support of the distribution, e.g., round to the nearest atoms. This projection step would lead to further errors, which is out of scope of this paper. We refer readers to related discussion in (Dabney et al., 2017; Rowland et al., 2018), and leave its analysis as a future work.

(4). The aim of w_t is to estimate w for a fixed value of θ. Thus w is updated on a "faster" timescale and parameter θ is updated on a "slower" timescale.

4. Distributional Greedy-GQ

In practice, we care more about the control problem. Thus in this section, we propose the distributional Greedy-GQ for the control setting. Now we denote $F_\theta((s, a), z)$ as the distribution function of $Z(s, a)$. Policy π_θ is a greedy policy w.r.t. $Q(s, a)$. i.e., the mean of $Z(s, a)$. $G_\theta((s, a), z)$ is the distribution function of $\mathcal{T}^{\pi_\theta} Z((s_i, a_i))$. The aim of the distributional Greedy-GQ is to optimize following objective function.

$$\min_\theta \|F_\theta - \Pi G_\theta\|_D^2$$

Using almost similar derivation as distributional GTD2 (With only difference in notation and we omit the term h_t here), we give the following algorithm 3, analogous to the Greedy-GQ (Maei et al., 2010).

Algorithm 3 Distributional Greedy-GQ

```
Input: step size $\alpha_t$, step size $\beta_t$, $0 \le \eta \le 1$
for $t = 0, 1, ...$ do
    $Q(s_{t+1}, a) = \sum_{j=1}^{m} z_j p_j(s_t, a)$, where $p_j(s_t, a)$ is the density function w.r.t. $F_\theta((s_t, a))$. $a^* = \arg\max_a Q(s_{t+1}, a)$.
    $$w_{t+1} = w_t + \beta_t \sum_{j=1}^{m} \left(-\phi_{\theta_t}^T((s_t, a_t), z_j) w_t + \delta_{\theta_t}\right) \times \phi_{\theta_t}((s_t, a_t), z_j).$$
    $$\theta_{t+1} = \theta_t + \alpha_t\{\sum_{j=1}^{m} \left(\delta_{\theta_t} \phi_{\theta_t}((s_t, a_t), z_j) - \eta \phi_{\theta_t}((s_{t+1}, a^*), \frac{z_j - r_t}{\gamma}) (\phi_{\theta_t}^T((s_t, a_t), z_j) w_t)\right)\}.$$
    where $\delta_{\theta_t} = F_{\theta_t}((s_{t+1}, a^*), \frac{z_j - r_t}{\gamma}) - F_{\theta_t}((s_t, a_t), z_j)$.
end for
```

Some remarks are in order.

- $0 \le \eta \le 1$ interpolates the distributional Q-learning and distributional Greedy-GQ. When $\eta = 0$, it reduces to the distributional Q-learning with Cramér distance while C51 uses KL-divergence. When $\eta = 1$ it is the distributional Greedy-GQ where we mainly use the temporal distribution difference δ_{θ_t} to replace the TD-error in (Maei et al., 2010).
- Unfortunately for the nonlinear function approximator and control setting, so far we do not have convergence guarantee. If the linear function approximation is used, we may obtain a asymptotic convergence result following the similar argument in (Maei et al., 2010). We leave both of them as the future work.

5. Convergence analysis

In this section, we analyze the convergence of distributional GTD2 and distributional TDC and leave the convergence of distributional Greedy-GQ as a future work. To the best of our knowledge, the convergence of control algorithm even in the non-distributional setting is still a tricky problem. The argument essentially follows the two time scale analysis (e.g., theorem 2 in (Sutton et al., 2009a) and (Bhatnagar et al., 2009)). We first define some notions used in our theorem. Given a compact set $C \subset \mathbb{R}^d$, let $\mathcal{C}(C)$ be the space of continuous mappings $C \mapsto \mathbb{R}^d$. Given projection Γ onto C, let operator $\hat{\Gamma} : \mathcal{C}(C) \mapsto \mathcal{C}(\mathbb{R}^d)$ be

$$\hat{\Gamma} v(\theta) = \lim_{0<\epsilon\to 0} \frac{\Gamma(\theta + \epsilon v(\theta)) - \theta}{\epsilon}.$$

When $\theta \in C^\circ$ (interior of C), $\hat{\Gamma}v(\theta) = v(\theta)$. Otherwise, if $\theta \in \partial C$, $\hat{\Gamma}v(\theta)$ is the projection of $v(\theta)$ to the tangent space of ∂C at θ. Consider the following ODE:

$$\dot{\theta} = \hat{\Gamma}(-\frac{1}{2}\nabla_\theta J)(\theta), \theta(0) \in C,$$

where $J(\theta)$ is the D-MSPBE in (7). Let K be the set of all asymptotically stable equilibria of the above ODE. By the definitions $K \subset C$. We then have the following convergence theorem, which proof is deferred to the appendix.

Theorem 2. *Let $(s_t, r_t, s'_t)_{t\geq 0}$ be a sequence of transitions satisfying Assumption 1. The positive step-sizes in Algorithm 1 and 2 satisfy $\sum_{t=0}^{\infty} a_t = \infty$, $\sum_{t=0}^{\infty} \beta_t = \infty$, $\sum_{t=0}^{\infty} \alpha_t^2, \sum_{t=1}^{\infty} \beta_t^2 < \infty$ and $\frac{\alpha_t}{\beta_t} \to 0$, as $t \to \infty$. Assume that for any $\theta \in C$ and $s \in \mathcal{S}$ s.t. $d(s) > 0$, F_θ is three times continuously differentiable. Further assume that for each $\theta \in C$, $\left(\mathbb{E} \sum_{j=1}^{m} \phi_\theta(s, z_j)\phi_\theta^T(s, z_j)\right)$ is nonsingular. Then $\theta_t \to K$ in Algorithm 1 and 2, with probability one, as $t \to \infty$.*

If we assume the distribution function can be approximated by the *linear function*. We can obtain a *finite sample* bound. Due to the limit of space we defer it to appendix.

6. Experimental result

6.1. Distributional GTD2 and distributional TDC

In this section, firstly we use a simple grid world experiment to test the convergence of our distributional GTD2 and distributional TDC in the off-policy setting. Then we assess the empirical performance of them and compare the performance with their non-distributional counterparts, namely, $GTD2$ and TDC. Particularly, we use a simple cartpole problem to test the algorithm, where we do several policy evaluation steps to get a accurate estimation of value function and then do a policy improvement step. To apply distributional GTD2 or distributional TDC, we use a neural network to approximate the distribution function $F_\theta((s, a), z)$. Particularly, in both experiments, we use a neural network with one hidden layer. The inputs are state-action pairs, and the output is a softmax function. There are 50 hidden units and we choose the number of atoms as 30 in the distribution, i.e., the number of outputs in softmax function is 30. In the policy evaluation step, the update rule of w and θ is simple, since we just need to calculate the gradient of F_θ, which can be obtained by the forward and backward propagation. The update rule of h_t is slightly more involved, where we have the term $\nabla^2 F_\theta(s, z_j)w_t$. Roughly speaking, it requires four times as many computations as the regular back propagation and we present the update rule in the appendix. In the grid world problem, the target policy is set to be the optimal policy. The data-generating policy is a small perturbation on the optimal policy. In the control problem (cartpole), we use the ϵ-greedy policy over the expected action values, where ϵ starts at 0.1 and decreases gradually to 0.02. To implement regular nonlinear GTD2 and TDC (Bhatnagar et al., 2009), we still use one hidden layer neural network with 30 hidden units. The output is $Q_\theta(s, a)$. The control policy is ϵ greedy with $\epsilon = 0.1$ at the beginning and decreases to $\epsilon = 0.02$ gradually. In the experiment, we choose discount factor $\gamma = 0.9$. Since reward is bounded in $[0, 1]$ in cartpole problem, in distributional GTD2 and distributional TDC, we choose $V_{\min} = 0$ and $V_{\max} = 10$. In the experiment, we use 20 episodes to evaluate the policy, and then choose the policy by the ϵ-greedy strategy. We report all experimental results in Figure 1. In the left panel of Figure 1, we test the convergence of the distributional GTD2 and distributional TDC in D-MSPBE. It is clearly to see that both algorithms converge. In the middle panel of Figure 1, we compare distributional GTD2 and TDC with their vanilla counterparts. We observe that the distributional GTD2 has the best result, followed by the distributional TDC. The distributional TDC seems to improve the policy faster at the early stage of the training and then slows down. The performance of regular GTD2 and TDC are inferior than their distributional counterparts. We also observe that standard deviations of the distributional version are smaller than those of regular one. In addition, the performance of the distributional algorithms increases steadily with few oscillations. Thus the simulation results show that the distributional RL is more stable than the regular one which matches the argument and observations in (Bellemare et al., 2017a). In the right panel, we draw a distribution function of $Z(s, a)$ estimate by the algorithm.

6.2. Distributional Greedy-GQ

In practice, we are more interested in the control problem. Therefore, the aim of this section is to test the performance of distributional Greedy-GQ and compare the result with DQN and distributional DQN (C51). Particularly, we test the algorithm in the environment Cartpole v0, lunarlander v2 in the openai gym (Brockman et al., 2016) and vizdoom (Kempka et al., 2016). All above algorithms are implemented in the off-policy manner with experience replay and target networks. In the platform of vizdoom, we choose defend the center as the environment, where the agent occupies the center of a circular arena. Enemies continuously

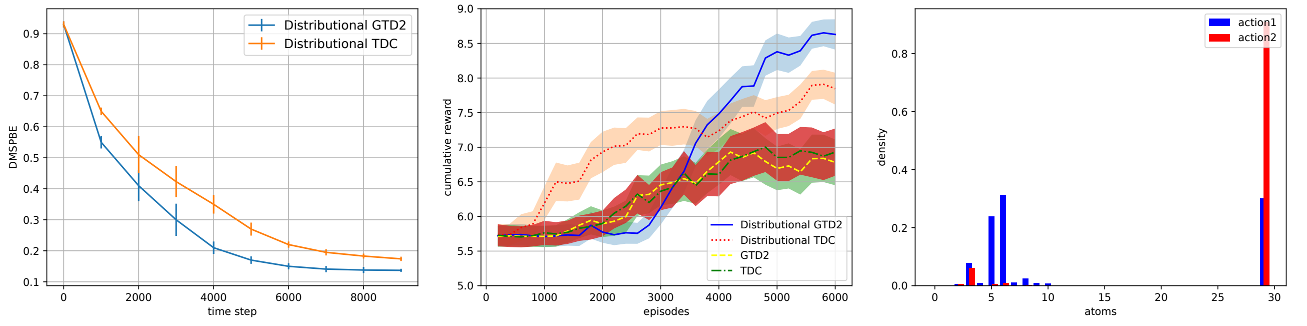

Figure 1. Left: Convergence of distributional GTD2 and distributional TDC. Middle: Performance of algorithms in Cartpole. Right: distribution of $Z(s,a)$ at some state s.

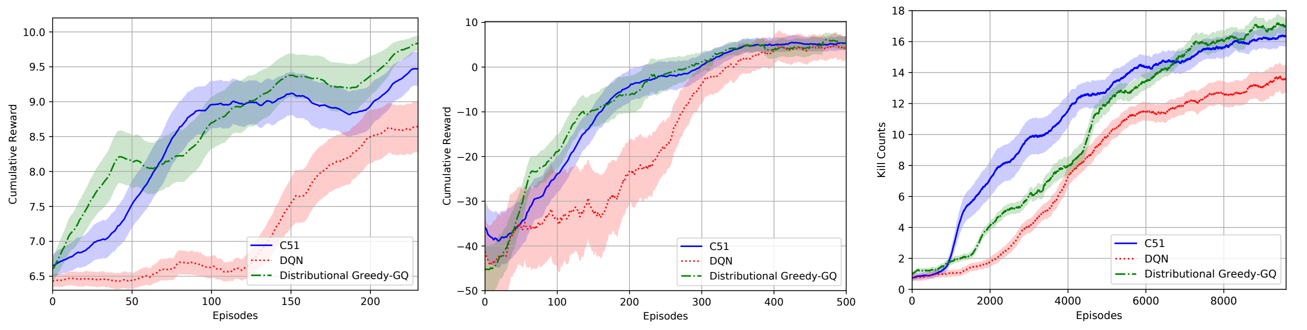

Figure 2. Left: Result in Cartpole v0. Middle: Result in lunarlander V2. Right: Result in vizdoom. In the left and middle panels, the x-axis is the training episode; the y-axis is the cumulative reward.

get spawned from far away and move closer to the agent until they are close enough to attack. The death penalty -1 is given by the environment. We give the penalty -0.1 if the agent loses ammo and health. Reward $+1$ is received if one enemy is killed. Totally, there are 26 bullets. The aim of the agent is to kill enemy and avoid being attacked and killed. For the environment Cartpole and lunarlander, to implement distributional Greedy-GQ and C51, we use two hidden-layer neural network with 64 hidden units to approximate the value distribution where activation functions are relu. The outputs are softmax functions with 40 units to approximate the probability atoms. We apply Adam with learning rate 5e-4 to train the agent. In vizdoom experiment, the first three layers are CNN and then it follows a dense layer where all activation functions are relu. The outputs are softmax functions with 50 units. We set $V_{\min} = -10$ and $V_{\max} = 20$ in the experiment.

We demonstrate all experiments in Figure 2. In the experiments of Cartpole and vizdoom, the performance of distributional Greedy-GQ are comparable with C51 and both of them are better than the DQN. Particularly, in the left panel, distributional Greedy-GQ is slightly better than C51. The variance of them are both smaller than that of DQN possibly because the distributional algorithms are more stable. In the experiment of vizdoom, C51 learns faster than distributional Greedy-GQ at the beginning but after 7000 episodes training distributional Greedy-GQ has the same performance with C51 and starts to outperform C51 later. In middle panel, C51 and distributional Greedy-GQ are slightly betten than the non-distributional counterpart DQN.

7. Conclusion and Future work

In this paper, we propose two non-linear distributional gradient TD algorithms and prove their convergences to a local optimum, while in the control setting, we propose the distributional Greed-GQ. We compare the performance of our algorithm with their non-distributional counterparts, and show their superiority. Distributional RL has several advantages over the regular approach, e.g., it provides richer set of prediction, and the learning is more stable. Based on this work on distributional RL, we foresee many interesting future research directions about performing RL beyond point of the estimation of the value function. An immediate interesting one is to develop efficient exploration to devise the control policy using more distribution information rather than using the expectation.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan. *stat*, 1050:26, 2017.

Bagnell, J. A. and Schneider, J. Covariant policy search. IJCAI, 2003.

Baird, L. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pp. 30–37. Elsevier, 1995.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458, 2017a.

Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., and Munos, R. The cramer distance as a solution to biased wasserstein gradients. *arXiv preprint arXiv:1705.10743*, 2017b.

Bhatnagar, S., Precup, D., Silver, D., Sutton, R. S., Maei, H. R., and Szepesvári, C. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*, pp. 1204–1212, 2009.

Borkar, V. S. and Meyn, S. P. The ode method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization*, 38 (2):447–469, 2000.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Christopher, M. B. *PATTERN RECOGNITION AND MACHINE LEARNING.* Springer-Verlag New York, 2016.

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

Dann, C., Neumann, G., and Peters, J. Policy evaluation with temporal differences: A survey and comparison. *The Journal of Machine Learning Research*, 15(1):809–883, 2014.

Defourny, B., Ernst, D., and Wehenkel, L. Risk-aware decision making and dynamic programming. 2008.

Engel, Y., Mannor, S., and Meir, R. Reinforcement learning with gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pp. 201–208. ACM, 2005.

Gelly, S. and Silver, D. Achieving master level play in 9 x 9 computer go. In *AAAI*, volume 8, pp. 1537–1540, 2008.

Heger, M. Consideration of risk in reinforcement learning. In *Machine Learning Proceedings 1994*, pp. 105–111. Elsevier, 1994.

Juditsky, A., Nemirovski, A., and Tauvel, C. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 1(1):17–58, 2011.

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

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pp. 1–8. IEEE, 2016.

Maei, H. R., Szepesvári, C., Bhatnagar, S., and Sutton, R. S. Toward off-policy learning control with function approximation. In *ICML*, pp. 719–726, 2010.

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, 2015.

O'Donoghue, B., Osband, I., Munos, R., and Mnih, V. The uncertainty bellman equation and exploration. *arXiv preprint arXiv:1709.05380*, 2017.

Pearlmutter, B. A. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

Rowland, M., Bellemare, M., Dabney, W., Munos, R., and Teh, Y. W. An analysis of categorical distributional reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 29–37, 2018.

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

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 993–1000. ACM, 2009a.

Sutton, R. S., Maei, H. R., and Szepesvári, C. A convergent $o(n)$ temporal-difference algorithm for off-policy learning with linear function approximation. In *Advances in neural information processing systems*, pp. 1609–1616, 2009b.

Székely, G. E-statistics: The energy of statistical samples. *Bowling Green State University, Department of Mathematics and Statistics Technical Report*, 3(05):1–18, 2003.

Tamar, A., Di Castro, D., and Mannor, S. Learning the variance of the reward-to-go. *Journal of Machine Learning Research*, 17(13):1–36, 2016.

Tesauro, G. Practical issues in temporal difference learning. In *Advances in neural information processing systems*, pp. 259–266, 1992.

Tsitsiklis, J. and Van Roy, B. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42:674–690, 1996.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *AAAI*, volume 16, pp. 2094–2100, 2016.