Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eidgenössische Technische Hochschule Zü
Swiss Federal Institute of Technology Zuric

Prof. Name

Master Thesis

LaTeX -Latex Template

Spring Term 2020

Supervised by:
Supervisor 1

Author:
Nuria Armengol Urpi

Contents

Preface iii

Abstract v

Symbols vii

1 Introduction 1

2 Distributional RL 3

2.1 Distributional RL . . . 3

2.1.1 Example showing interest in learning the distribution . . . 3

2.1.2 Distributional Bellman Operator . . . 4

2.1.3 Quantile approximation . . . 6

2.1.4 Quantile projection: . . . 6

2.1.5 Quantile Regression . . . 7

2.1.6 Quantile Regression Temporal Difference Learning . . . 7

3 Conditional Value at Risk (CVaR) 9

3.1 CVaR . . . 9

3.1.1 Conditional Value-at-Risk (CVaR) . . . 9

4 Algorithm: Off-policy deterministic AC 11

4.1 Distributional Deterministic Policy Gradients . . . 11

4.1.1 Off-policy Deterministic policy gradient algorithm . . . 11

4.1.2 Distributional approach . . . 12

5 Results 15

5.1 Current results Car . . . 15

5.1.1 Case 1: No velocity penalization . . . 15

5.1.2 Case 2: Velocity penalization with probability 1 . . . 15

5.1.3 Case 3: Velocity penalization with probability P . . . 17

5.2 Current results Batch RL HalfCheetah . . . 18

5.3 Current results Batch RL HalfCheetah . . . 21

5.4 CVaR minimization using DDPG and IQN . . . 23

Bibliography 25

A Anything 27

Preface

Bla bla . . .

Abstract

Bla bla ...

Symbols

Symbols

ϕ, θ, ψ	roll, pitch and yaw angle

Indices

x	x axis
y	y axis

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule

Chapter 1

Introduction

Introduction

Chapter 2

Distributional RL

2.1 Distributional RL

Recent research has been done demonstrating the importance of learning the value distribution, i.e, the distribution of the random return received by a RL agent. This differs from the common RL approach which is focused on learning the expected value of this return.
One of the major goals of RL is to teach an agent so that it learns how to act so that it maximizes its expected utility, Q [1] Bellman's equation describes this value Q in terms of the expected reward and expected outcome of the random transition $(x, a) \to (X', A')$, showing the particular recursive relationship between the value of a state and the values of its successor states:

$$Q(x, a) = \mathbb{E}[R(x, a)] + \gamma \mathbb{E}[Q(X', A')] \tag{2.1}$$

Distributional RL aims to go beyond the notion of *value* and training to study instead the random return Z.

2.1.1 Example showing interest in learning the distribution

Imagine the example in which we are playing a board game and we roll 2 dices. If we get a 3, we fall in prison and need to pay 2000CHF (ie reward of -2000CHF), whereas otherwise we we collect a salary of 200CHF (ie reward of +200CHF). If we consider the common reinforcement learning approach and we compute the expected immediate ($\gamma = 1$) reward:

$$\mathbb{E}[R(x)] = \frac{1}{36}(-2000\,\text{CHF}) + \frac{35}{36}(200\,\text{CHF}) = 138.88\,\text{CHF} \tag{2.2}$$

Hence, the expected immediate return is +138.88CHF. However, in any case we will get a return of +138.88CHF. Instead:

$$R(x) = \begin{cases} -2000\,\text{CHF}, & \text{w.p } \frac{1}{36} \\ 200\,\text{CHF}, & \text{w.p } \frac{35}{36} \end{cases}$$

We define the random return $Z^\pi(x, a)$ as the random variable that represents the sum of discounted rewards obtained by starting from position x taking action a and thereupon following policy π.
This variable captures intrinsic randomness from:

1. Immediate rewards
2. Stochastic dynamics
3. Possibly an stochastic policy

Having defined $Z^\pi(x, a)$, we can clearly see that:

$$Q^\pi(x, a) = \mathbb{E}[Z^\pi(x, a)] \tag{2.3}$$

Z is also described by a recursive equation, but of a distributional nature:

$$Z^\pi(x, a) \overset{D}{=} R(x, a) + \gamma Z(x', a') \tag{2.4}$$

where $x' \backsim p(\cdot|x, a)$ and $a' \backsim \pi(\cdot|x')$
where $\overset{D}{=}$ denotes that the RV on both sides of the equation share the same probability distribution. The *distributional Bellman equation* defined in (2.4), states that the distribution of Z is characterized by the interaction of 3 RV's: the random variable reward R, the next state-action (X',A') and its random return Z(X',A'). From here on, we will view Z^π as a mapping from state-action pairs to distributions over returns, and we call this distribution the *value distribution.*

2.1.2 Distributional Bellman Operator

In the policy evaluation setting [1], one aims to find the value function $V\pi$ associated with a given fixed policy π. In the distributional case, we aim to find $Z\pi$. [2] defined the Distributional Bellman operator T^π. We view the reward function as a random vector R $\in \mathbb{Z}$ and define the transition operator $P^\pi : \mathbb{Z} \to \mathbb{Z}$

$$P^\pi Z(x, a) \overset{D}{=} Z(X', A') \tag{2.5}$$

$$X' \backsim P(\cdot|x, a) \text{ and } A' \backsim \pi(\cdot|X') \tag{2.6}$$

where we use capital letters to emphasize the random nature of the next state-action pair (X',A') Then, the Distributional Bellman operator T^π is defined as:

$$T^\pi Z(x, a) \overset{D}{=} R(x, a) + \gamma P^\pi Z(x, a) \tag{2.7}$$

[2] showed that (2.7) is a contraction mapping in Wasserstein metric whose unique fixed point is the random return Z^π.

Wasserstein metric:

The p-Wasserstein metric W_p, for $p \in [1, \infty]$, also known as the Earth Mover's Distance when $p = 1$ is an integral probability metric between distributions. The p-Wasserstein distance is characterized as the L^p metric on inverse cumulative distribution functions (CDF). Tht is, the p-Wasserstein metric between distributions U and Y is given by:

$$W_p(U, Y) = \Big(\int_0^1 |F_Y^{-1}(w) - F_U^{-1}(w)|^p dw \Big)^{\frac{1}{p}} \tag{2.8}$$

where for a random variable Y, the inverse CDF F_Y^{-1} of Y is defined by:

$$F_Y^{-1}(w) := \inf\{y \in \mathbb{R} \,| w \;\leq F_Y(w)\} \tag{2.9}$$

where $F_Y(w) = Pr(y \leq Y)$.

Add Figure 2.1 in [3]

Unlike the Kullback-Leibler divergence , the Wasserstein metric is a true probability metric and considers both the probability of and the distance between various outcome events, which makes it well-suited to domains where an underlying similarity in outcome is more important than exactly matching likelihoods.

Contraction in $\hat{d}_p$:

Let $\mathcal{Z}$ be the space of action-value distributions:

$$\mathcal{Z} = \{Z \mid \mathcal{X} \times \mathcal{A} \to \wp(\mathbb{R}) \tag{2.10}$$
$$\mathbb{E}[|Z(x,a)|^p < \infty, \forall(x,a), p \geq 1]\} \tag{2.11}$$

check first line the $\wp$

Then, for two action-value distribution $Z_1, Z_2 \in \mathcal{Z}$, the maximal form of the Wasserstein metric is defined by:

$$\hat{d}_p(Z_1, Z_2) := \sup_{x,a} W_p(Z_1(x,a), Z_2(x,a)) \tag{2.12}$$

[2] showed that $\hat{d}_p$ is a metric over value distributions and furthermore, the distributional Bellman operator T^π is a contraction in $\hat{d}_p$. Consider the process $Z_{k+1} := T^\pi Z_k$, starting with some $Z_0 \in \mathcal{Z}$.
$T^\pi Z : \mathcal{Z} \to \mathcal{Z}$ is a γ-contraction in the Wasserstein metric $\hat{d}_p$, which implies that not only the first moment (expectation) converges exponentially to Q^π, but also in all moments.
Lemma 1: (Lemma 3 in [2])
T^π is a γ-contraction: for any two $Z_1, Z_2 \in \mathcal{Z}$,

$$\hat{d}_p(T^\pi Z_1, T^\pi Z_2) \leq \gamma \hat{d}_p(Z_1, Z_2) \tag{2.13}$$

Using Banach's fixed point theorem, it is proven that T^π has a unique fixed point, which by inspection must be Z^π.
Hence the $\hat{d}_p$ metric is shown to be useful metric for studying behavior of distributional RL algorithms, and to showed their convergence to a fixed point. Moreover, shows than en effective way to learn a value distribution is to attempt minimize the Wasserstein distance between a distribution Z and its distributional Bellman update $T^\pi Z$, analogously to the way that TD-learning attempts to iteratively minimize the L^2 distance between Q and TQ.
We have so fare considered a policy evaluation setting, ie trying to learn a value distribution for a fixed policy π, and we studied the behavior of its associated distributional operator T^π. In the control setting, ie, when we try to find a policy π^* that maximizes a value, or its distributional analogous, ie that induces an optimal value distribution. However, while all optimal policies attain the same value Q^*, in general there are many optimal value distributions.
The distributional analogue of the Bellman optimality operator converges, in a weak sense, to the set of optimal value distributions, but this operator is *not a contraction in any metric between distributions.*.
Let Π^* be the set of optimal policies.
Definition 1: An optimal value distribution is the value distribution of an optimal policy. The set of optimal value distributions is

$$\mathcal{Z}^* := \{Z^{\pi^*} | \pi^* \in \Pi^*\}$$

Not all value distributions with expectation Q^* are optimal, but they must match the full distribution of the return under some optimal policy. **Definition 2:** A greedy policy π for Z $in \mathcal{Z}$ maximizes the expectation of Z. The set of greedy policies for Z is:

$$\mathcal{G}_{\mathcal{Z}} := \{\pi \mid \sum_a \pi(a|x)\mathbb{E}(Z(x,a) = \max_{a' \in \mathcal{A}} Q(x', a')\}$$

We will call a *distributional Bellman optimality operator* any operator $\mathcal{T}$ which implements a greedy selection rule, ie:

$$\mathcal{T}Z = \{\mathcal{T}^{\pi} Z \text{ for some } \pi \in \mathcal{G}_{\mathcal{Z}}\}$$

As in the policy evaluation setting, we are interested in the behavior of the iterates $Z_{k+1} := \mathcal{T}Z_k, Z_0 \in \mathcal{Z}$. Lemma 4 in [2] shows that $\mathbb{E}Z_k$ behaves as expected: **Lemma 4:** Let $Z_1, Z_2 \in \mathcal{Z}$. Then:

$$\|\mathbb{E}\mathcal{T}Z_1 - \mathbb{E}\mathcal{T}Z_2\|_\infty \leq \gamma \|\mathbb{E}Z_1 - \mathbb{E}Z_2\|_\infty$$

and in particular $\mathbb{E}Z_1 \to Q^*$ exponentially quickly. However, Z_k is not assured to converge to a fixed point. Specifically, they provide a number of negative results concerning $\mathcal{T}$:
Proposition 1: The operator $\mathcal{T}$ is not a contraction.
Proposition 2: Not all optimality operators have a fixed point $Z^* = \mathcal{T}Z^*$ **Proposition 3**: That $\mathcal{T}$ has a fixed point $Z^* = \mathcal{T}Z^*$ is insufficient to guarantee the convergence of $\{Z_k\}$ to Z^*
Another result, shows that we cannot in general minimize the Wasserstein metric, viewed as a loss, using stochastic gradient descent methods. This limitation, is crucial in a practical context, when the value distribution needs to be approximated.

2.1.3 Quantile approximation

[3] used the theory of quantile regression [4], to design an algorithm applicable in a stochastic approximation setting. Quantile regression is used to estimate the quantile function at precisely chosen points. Then the Bellman update is applied onto this parameterized quantile distribution. This combined operator is proven to be a contraction and the estimated quantile function is shown to converge to the true value distribution when minimized using stochastic approximation.

2.1.4 Quantile projection:

Our current aim is to estimate quantiles of the target distribution, ie the values of the return that divide the value distribution in equally sized parts. We will call it a quantile distribution, and we will let $\mathcal{Z}_Q$ be the space of quantile distributions. We denote the cumulative probabilities associated with such a distribution by $\tau_1, \tau_2 ... \tau_N$, so that $\tau_i = \frac{i}{N}$ for $i = 1, ...N$.
Formally, let $\theta : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^N$ be some parametric model. A quantile distribution $Z_\theta \in \mathcal{Z}_Q$ maps each state-action pair (x,a) to a uniform probability distribution supported on $\{\theta_i(x,a)\}$. Hence we can approximate it by a uniform mixture of N Diracs:

$$Z_\theta(x,a) := \frac{1}{N}\sum_{i=1}^{N} \delta_{\theta_i(x,a)} \tag{2.14}$$

with each θ_i assigned a fixed quantile. We aim to learn the support of these Diracs, i.e learn $\theta_i \forall i, a, x$. We will do it by quantifying the projection of an arbitrary value distribution $Z \in \mathcal{Z}$ onto $\mathcal{Z}_Q$, that is:

$$\prod_{W_1} Z := \underset{Z_\theta \in \mathcal{Z}_Q}{\arg\min} W_1(Z, Z_\theta) \tag{2.15}$$

This projection $\prod_{W_1}$ is the quantile projection.
We can quantify the projection between a distribution with bounded first moment Y and U, a uniform distribution over N Diracs as in (2.14) with support $\{\theta_1, ...\theta_N\}$ by:

$$W_1(Y, U) = \sum_{i=1}^{N} \int_{\tau_{i-1}}^{\tau_i} |F_Y^{-1}(w) - \theta_i| dw \tag{2.16}$$

Lemma 2 in [3] establishes that the values $\{\theta_1, ..., \theta_N\}$ for the returns that minimize $W_1(Y, U)$ are given by $\theta_i = F_Y^{-1}(\hat{\tau}_i)$, where $\hat{\tau}_i = \frac{\tau_{i-1}+\tau_i}{2}$.

2.1.5 Quantile Regression

Quantile regression is a method for approximating quantile functions of a distribution at specific points, ie its inverse cumulative distribution function. The quantile regression loss, for quantile $\tau \in [0, 1]$, is an asymmetric convex lox function that penalizes underestimation errors with weight τ and overestimation errors with weight $1 - \tau$.
For a distribution Z, and given quantile τ, the value of the quantile function $F_Z^{-1}(\tau)$ may be characterized as the minimizer of the quantile regression loss:

$$\mathcal{L}_{QR}^{\tau}(\theta) = \mathbb{E}_{\hat{Z} \backsim Z}[\rho_\tau(\hat{Z} - \theta)] \tag{2.17}$$
$$\rho_\tau(u) = u(\tau - \delta_{u<0}), \forall u \in \mathbb{R}$$

Given that the minimizer of the quantile regression loss for τ is $F_Z^{-1}(\tau)$, and using Lemma 2 in [3], which claims that the values of $\{\theta_1, ...\theta_N\}$ that minimize $W_1(Z, Z_\theta)$ are given by $\theta_i = F_Y^{-}1(\hat{\tau}_i)$; we can claim that the values of $\{\theta_1, ...\theta_N\}$ are the minimizers of the following objective:

$$\sum_{i=1}^{N} \mathbb{E}_{\hat{Z} \sim Z}[\rho_{\hat{\tau}_i}(\hat{Z} - \theta_i)] \tag{2.18}$$

This loss gives unbiased sample gradients and hence, we can find the minimizing $\{\theta_1, ...\theta_N\}$ by stochastic gradient descent.

add huberloss

Proposition 2 in [3] states that the combined quantile projection $\prod_{W_1}$ with the Bellman update $\mathcal{T}^\pi$ has a unique fixed point $\hat{Z}^\pi$, and the repeated application of this operator, or its stochastic approximation, converges to $\hat{Z}^\pi$.

2.1.6 Quantile Regression Temporal Difference Learning

Temporal difference learning updates the estimated value function with a single unbiased sample following policy π. Quantile regression allows to improve the estimate of the quantile function for some target distribution Y(x), by observing samples $y \sim Y(x)$ and minimizing equation (2.17). Using the quantile regression loss, we can obtain an approximation with minimal 1-Wasserstein distance from the original. We can combine this with the distributional Bellman operator to give a target distribution for quantile regression, creating the quantile regression temporal difference learning algorithm:

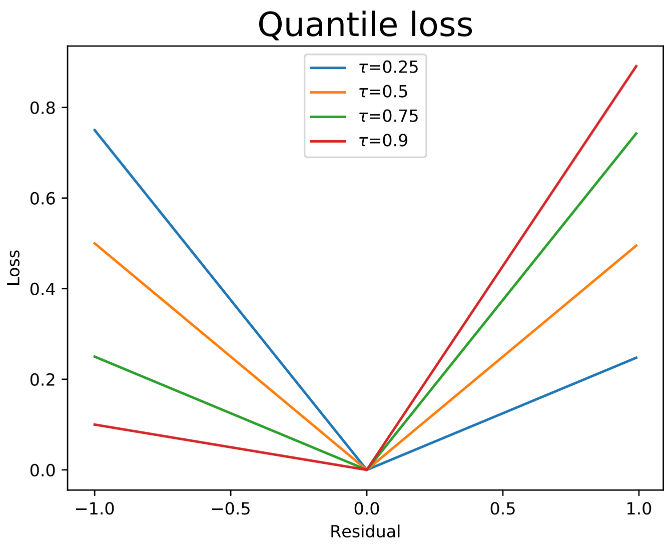

Figure 2.1: Quantile loss for different quantile values

$$u = r + \gamma z' - \theta_i(x) \tag{2.19}$$

$$\theta_i(x) \leftarrow \theta_i(x) + \alpha(\hat{\tau}_i - \delta_{u<0}) \tag{2.20}$$

$$a \backsim \pi(\cdot|x), r \backsim R(x,a), x' \backsim P(\cdot|x,a), z' \backsim Z_\theta(x') \tag{2.21}$$

where Z_θ is a quantile distribution as in (2.14) and $\theta_i(x)$ is the estimated value of $F^{-1}_{Z^\pi(x)}(\hat{\tau}_i)$ in state x.

Chapter 3

Conditional Value at Risk (CVaR)

3.1 CVaR

We focus on the importance of *value distribution*, the distribution of the random return received by a RL agent, in contrast to the common approach in RL of modelling the expectation of this return. The latter neither takes int account the variability of the cost (i.e fluctuations around the mean), nor its sensitivity to modeling errors. [5]
We aim to learn this distribution and try to minimize other metrics rather than its mean, which can be crucial for some environments in which ensuring that the cost is always above a certain value with certain probability is crucial.
A metric that has recently gained a lot of popularity is the Conditional Value at Risk, eg in finance, due to its favorable computation properties and superior ability to safeguard a decision maker from the "outcomes that hurt the most" [6]

3.1.1 Conditional Value-at-Risk (CVaR)

Let Z be a bounded-mean random variable, i.e $\mathbb{E}[|Z|] < \infty$, on a probability space $(\Omega, \mathbb{F}, \mathbb{P})$, with cumulative distribution function $F(z) = \mathbb{P}(Z \leq z)$. We interpret Z as a reward. The value-at-risk (VaR) at confidence level $\alpha \in (0, 1)$ is the α quantile of Z, i.e, $\mathrm{VaR}_\alpha(Z) = \inf\{z \mid F(z) \geq \alpha\}$. The conditional value-at-risk (CVaR) at confidence level $\alpha \in (0, 1)$ is defined as the expected reward of outcomes worse than the α-quantile (VaR_α):

$$\mathrm{CVaR}_\alpha(Z) = \frac{1}{\alpha}\int_0^\alpha F_Z^{-1}(\beta)d\beta = \frac{1}{\alpha}\int_0^\alpha \mathrm{VaR}_\beta(Z)d\beta \tag{3.1}$$

Rockafellar and Uryasev [7] also showed that CVaR is equivalent to the solution of the following optimization problem:

$$\mathrm{CVaR}_\alpha(Z) = \max_\nu\big\{\nu + \frac{1}{\alpha}\mathbb{E}_Z[[Z-\nu]^-]\big\} \tag{3.2}$$

where $(x)^- = \min(x, 0)$. In the optimal point it holds that $\nu^* = \mathrm{VaR}_\alpha(Z)$.
A useful property of CVaR, is its alternative dual representation [8]:

$$\mathrm{CVaR}_\alpha(Z) = \min_{\xi \in U_{\mathrm{CVaR}}(\alpha, \mathbb{P})} \mathbb{E}_\xi[Z] \tag{3.3}$$

where $\mathbb{E}_\xi[Z]$ denotes the ξ-weighted expectation of Z, and the risk envelope U_{CVaR} is given by:

$$U_{\text{CVaR}}(\alpha, \mathbb{P}) = \left\{ \xi | \xi(w) \in \left[0, \frac{1}{\alpha} \int_{w \in \Omega} \xi(w)\mathbb{P}(w)dw = 1\right] \right\} \tag{3.4}$$

Thus, the CVaR of a random variable may be interpreted as the worst case expectation of Z, under a perturbed distribution $\xi\mathbb{P}$.

Chapter 4

Algorithm: Off-policy deterministic AC

4.1 Distributional Deterministic Policy Gradients

We introduce an off-policy actor-critic distributional algorithm. The actor uses a distributional variant of the deterministic policy gradient algorithm.

4.1.1 Off-policy Deterministic policy gradient algorithm

We will use an actor-critic approach based on the DPG algorithm [9] which uses deterministic policies $a = \mu_\theta(s)$. From a practical viewpoint, using stochastic policies requires integrating over both state and action spaces to compute the policy gradient, whereas the deterministic case only needs to integrate over the state space. Hence, stochastic policy gradients may require much more samples, especially if the action space has many dimensions.

In general, behaving according to a deterministic policy does not ensure adequate exploration, and may lead to suboptimal solutions. However, if the policy is deterministic, the expected cumulative reward in the next-state depends only on the environment. This means that it is possible to learn the value function Q under policy μ off-policy, ie using transitions which are generated from a different stochastic behavior policy β which act on the environment and ensures enough exploration. that ensures enough exploration. will use an off-policy actor-critic. An advantage of offpolicy algorithms is that we can treat the problem of exploration independently from the learning algorithm.

Q-learning [10], a commonly used off-policy algorithm, uses the greedy policy $\mu(s) = \underset{a}{\text{argmax}} Q(x, a)$. In a continuous action space, it is not possible to apply Q-learning straight-forward because finding the greedy policy requires an optimization of a at every timestep, which is too slow to be practical with large action spaces. In this case, actor-critic methods are commonly used, where action selection is performed through a separate policy network, known as the actor,and updated with respect to a value estimate, known as the critic [1]. The policy can be updated following the deterministic policy gradient theorem [9], which corresponds to learning an approximation to the maximum of Q, by propagating the gradient through both policy and Q. Specifically maximizing, the performance objective which is the value function of the target policy μ, averaged over the state distribution of the behavior policy β:

$$
\begin{aligned}
J_\beta(\mu|\theta^\mu) &= \int_{\mathcal{S}} \rho^\beta(s) Q^\mu(x, \mu(x|\theta^\mu))dx \\
\nabla_{\theta^\mu} J_\beta(\mu|\theta^\mu) &\approx \mathbb{E}_{x\sim\rho^\beta}\left[\nabla_{\theta^\mu}\mu(x, |\theta^\mu)\nabla_a Q^\mu(x,a)|_{a=\mu(x|\theta^\mu)}\right] \qquad (4.1)
\end{aligned}
$$

(4.1) gives the off-policy deterministic policy gradient, which was proved by [9] to be the policy gradient, ie the gradient of the policy's performance. A term that depends on $\nabla_\theta Q^{\mu_\theta}(x,a)$ has been dropped in following a justification given by [11] that argues that this is a good approximation since it can preserve the set of local optima to which gradient ascent converges.
Similarly to [12], we will use neural networks as non-linear function approximators for learning both action-value functions and the deterministic target policy.

4.1.2 Distributional approach

Critic

Instead of using the standard critic network that approximates the value function, we will use the distribution variant which maps from state-action pairs to distributions, similar to the implicit quantile network (IQN) introduced in [13].
IQN is a deterministic parametric function trained to reparameterize samples from a base distribution, e.g $\tau \in U([0,1])$, to the respective quantile values of a target distribution. We define $Z(x,a;\tau)$ as the quantile function at $\tau \in [0,1]$ for the random variable $Z(x,a)$. Thus, for $\tau \in U([0,1])$, the resulting state-action return distribution sample is $Z(x,a;\tau) \sim Z(x,a)$
The parameters θ^Z of the IQN network are updated using backpropagation of the sampled quantile regression loss. The quantile regression loss is computed on the sampled temporal-difference error using the distributional Bellman operator: For two samples $\tau, \tau' \sim U([0,1])$, and current policy μ_θ, the sampled TD error is:

$$
\delta^{\tau,\tau'} = r + \gamma Z(x', \mu(x'); \tau'|\theta^\mu) - Z(x,a;\tau) \qquad (4.2)
$$

Then, we compute the quantile regression loss:

$$
\mathcal{L}(x,a,r,x') = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N'} \rho_{\tau_i}(\delta^{\tau_i,\tau'_j}) \qquad (4.3)
$$

where ρ is as defined in (2.17) and where N and N' are the number of iid samples $\tau_i, \tau'_j \sim U([0,1]$ used to estimate the loss.

Actor

The policy is updated via deterministic policy gradient ascent. We modify equation (4.1), to include the action-value distribution.

$$
\begin{aligned}
\nabla_{\theta^\mu} J_\beta(\mu|\theta^\mu) &\approx \mathbb{E}_{x\sim\rho^\beta}\left[\nabla_{\theta^\mu}\mu(x, |\theta^\mu)\nabla_a Q^\mu(x,a)|_{a=\mu(x|\theta^\mu)}\right] \qquad (4.4) \\
&= \mathbb{E}_{s\sim\rho^\beta}\left[\nabla_{\theta^\mu}\mu(x, |\theta^\mu)\mathbb{E}[\nabla_a Z(x,a|\theta^Z)]|_{a=\mu(x|\theta^\mu)}\right] \qquad (4.5)
\end{aligned}
$$

The term $\mathbb{E}[\nabla_a Z(x,a|\theta^Z)]$ comes by the fact that

$$
Q^\mu(x,a) = \mathbb{E}[Z^\mu(x,a)] \qquad (4.6)
$$

However, we could modify the objective of DPG algorithm

$$\max_{\mu} Q(x, \mu(x)) := \max_{\mu} \mathbb{E}[Z(x, \mu(x))] \tag{4.7}$$

for a more general one:

$$\max_{\mu} \mathrm{CVAR}_{\alpha}[Z(x, \mu(x))] \tag{4.8}$$

since we can see the cvar of the distribution as a distorted expectation of the value distribution. and which allows us for risk-sensitive, risk-averse or risk-neutral policies depending on the value of α. add pag 2 proof sent to sebastian last day
add replay batch, target networks.. p4 lillicrap

Chapter 5

Results

5.1 Current results Car

Problem: A car with fully-observable 2D-state: [position, velocity] needs to move from initial position $x_0 = 0$m and initial velocity $v_0 = 0\text{m}\,\text{ts}^{-1}$ to goal position $x_F = 2.0\,\text{m}$. The action taken at every time-step ts, with a discretization of $t_d = 0.1$, determines the car acceleration. The control input a is constrained to range between [-1.0,1.0]$\text{m}\,\text{ts}^{-2}$. Per every time-step passed before it reaches the goal, the car receives a penalization reward $R_{\text{ts}} = -10$ If the car reaches the goal position, it receives a reward $R_F = +270$ and the episode ends. Otherwise, after $T_F = 400$ts the episode ends (with no extra penalization).

5.1.1 Case 1: No velocity penalization

Using both DDPG and CVAR-DDPG algorithms, the car arrives at the goal position. Both with a maximum acceleration kept throughout the whole episode.
For this setup we have:

$$x = x_0 + v_0 \frac{\text{ts}}{10} + 0.5a\Big(\frac{\text{ts}}{10}\Big)^2$$

In the optimal case, the car keeps an acceleration of $1\,\text{m}\,\text{ts}^{-2}$ for the whole episode, and hence reaches $x_F = 2$m with 20 time-steps. Hence the final cumulative reward $G_T = (20 + 1)R_{\text{ts}} + R_F = 60$.

5.1.2 Case 2: Velocity penalization with probability 1

The experiment is carried out to ensure the two algorithms manage to learn the new reward function when there is no uncertainty. In this setup, when the car velocity exceeds $1\text{m}\,\text{ts}^{-1}$, it receives a penalization of $R_v = -20$. We expect both algorithms to perform similarly since there is no reward uncertainty. As expected, both DDPG and CVAR-DDPG algorithms learn to accelerate with maximum value till a velocity of $1\text{m}\,\text{ts}^{-1}$ is reached, and then they keep the velocity constant until the goal is reached.
Starting from $x_0 = 0$m, the car reaches a velocity of $1\text{m}\,\text{ts}^{-1}$ after 10 time-steps, at $x_{\text{ts}=10} = 0.5$. Keeping velocity $1\text{m}\,\text{ts}^{-1}$ through the rest of the episode, it reaches the goal position after 14 time-steps. Hence the final cumulative reward $G_T = (10 + 14 + 1)R_{\text{ts}} + R_F = 20$. The reward values were chosen in order to make sure that, for this Case 2 setting, driving with a velocity higher than $1\text{m}\,\text{ts}^{-1}$ never induces higher cumulative rewards.

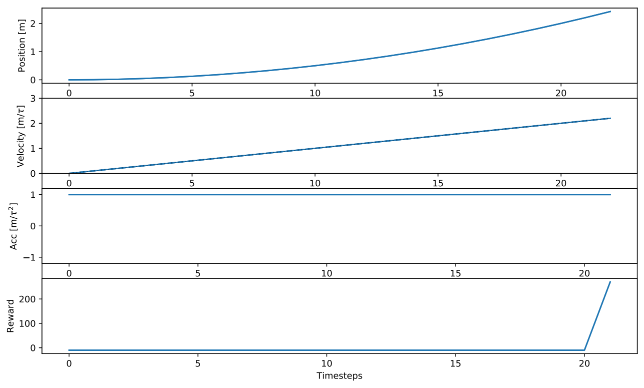

Figure 5.1: Car trajectory using DDPG and algorithm without velocity penalization. (Same behavior for CVAR-DDPG algorithm).

For this Case 2 setting, the trained models were saved using Early stopping with *maximal reward in episode evaluation* as a metric and with a patience of 100 episodes. The quantiles used for learning the actor for the CVAR-DDPG algorithm were sampled uniformly $\backsim U[0,1]$

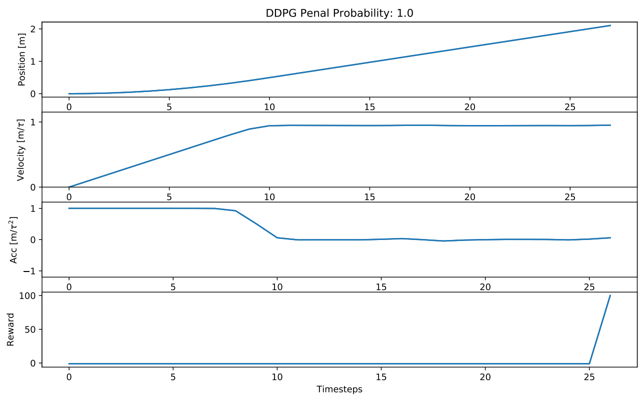

Figure 5.2: Car trajectory using DDPG algorithm and velocity penalization with probability 1

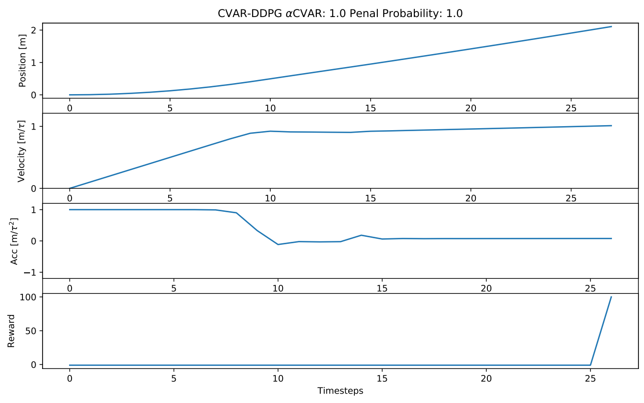

Figure 5.3: Car trajectory using CVAR-DDPG algorithm and velocity penalization with probability 1. (α-CVAR = 1)

5.1.3 Case 3: Velocity penalization with probability P

The experiment is carried out to show the risk-sensitiveness property of the CVAR-DDPG algorithm.

The models saved were the ones that obtained a maximum CVAR (with a window of 10 episodes) of the cumulative rewards during evaluation The quantiles used for learning the actor for the CVAR-DDPG algorithm were sampled uniformly $\backsim$ $U[0, \alpha]$ where $\alpha = 0.1$

For $P = 0.2$ the CVAR-DDPG algorithm learns to saturate the velocity, even though the probability of a penalization is low, whereas the DDPG algorithm doesn't, and keeps a linear increase of the velocity during the whole episode. The CVAR algorithm reaches its maximum CVAR of 64.0 at episode 220, whereas the DDPG reaches its maximum CVAR value of 49.0 at episode 1435.

Important issue: Although CVAR-DDPG finds a risk-sensitive trajectory at episode 220, it doesn't converge there and keeps oscillating and even moves towards a risk-neutral behavior later on. The graph in figure 5.7 , shows the evolution of the sampled mean of the tail of the sampled cumulative value distribution (CDF). (ie we compute via IQN the quantile values from the tail value distribution (VD) and take the mean). The value it converges to coincides with the maximum value of the CVAR we achieved ,but then the actor doesn't seem to behave accordingly.

$$\mathrm{CVaR}_\alpha(Z) = \frac{1}{\alpha}\int_0^\alpha F_Z^{-1}(\tau)d\tau = \frac{1}{\alpha}\int_0^\alpha IQN(\tau)d\tau \approx \frac{1}{\alpha}\frac{1}{K}\sum_{i=0}^{K} IQN(\tau_i) \tag{5.1}$$

where $\tau_i \sim U[0, \alpha]$, and IQN is the output of the IQN network for given τ, representing the value of the return for the given quantile.
(Values of the sampled CVAR showed in 5.7 are not divided by α neither K)

A binomial distribution can be observed.

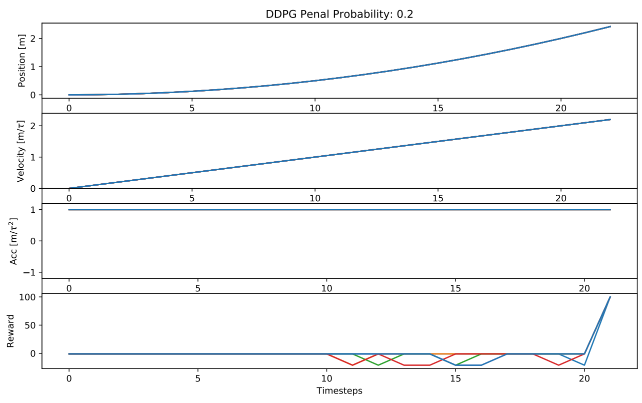

Figure 5.4: Car trajectory using DDPG algorithm and velocity penalization with probability P=0.2

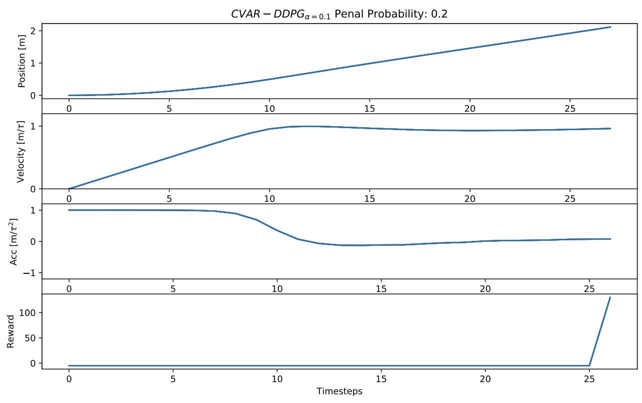

Figure 5.5: Car trajectory using CVAR-DDPG algorithm and velocity penalization with probability 0.2 and (α-CVAR = 0.2)

5.2 Current results Batch RL HalfCheetah

We use one of the D4RL datasets. Specifically *halfcheetah-medium-v0*, which uses 1M samples from a policy trained to approximately 1/3 the performance of the expert.
We introduce stochasticity in the original cost function in a way that makes the environment stochastic enough to have a meaningful assessment of risk in terms of tail performance. A reward of -100 is given wp 0.05, if the velocity of the cheetah is greater than 4. We train using the distributional critic and a policy that consists of a variational autoencoder to sample from the dataset distribution and then a second perturbing network that shifts the action towards maximizing the sampled CVaR. The perturbation is up to 0.5 (paper originally 0.05).

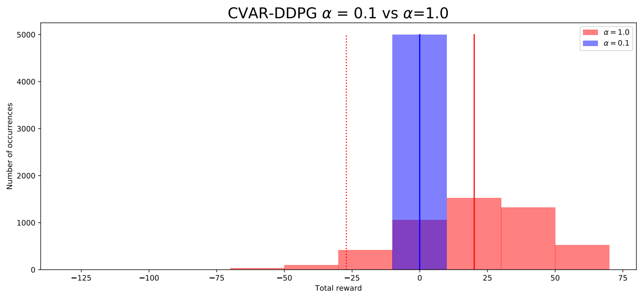

Figure 5.6: Comparison of cumulative rewards achieved with CVAR-DDPG algorithms with α= 0.2 and α= 1 when the probability of velocity penalization = 0.2. Algorithm with α= 1 achieves a higher expected value ($\mu = 20.11$) but has a lower CVAR (CVaR$_{\alpha=0.1}$=-27.12 compared to the algorithm with α= 0.1, which has $\mu = 0$ and CVaR$_{\alpha=0.1}$=0.0 5000 episodes were ran after training each algorithm.

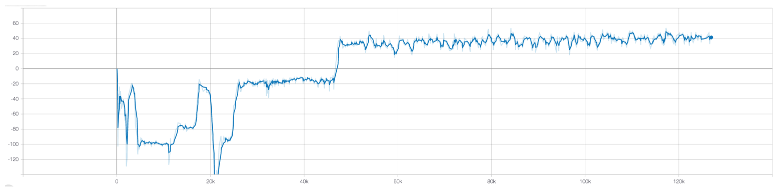

Figure 5.7: Evolution of the sampled mean of the tail of the Cumulative Value Distribution during training epochs

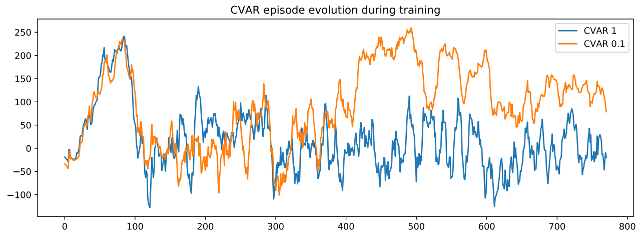

Figure 5.8: CVaR of the cumulative episode rewards over 10 episodes while training

5.3 Current results Batch RL HalfCheetah

We use one of the D4RL datasets. Specifically *halfcheetah-medium-v0*, which uses 1M samples from a policy trained to approximately 1/3 the performance of the expert.
We introduce stochasticity in the original cost function in a way that makes the environment stochastic enough to have a meaningful assessment of risk in terms of tail performance. A reward of -100 is given wp 0.05, if the velocity of the cheetah is greater than 4. We train using the distributional critic and a policy that consists of a variational autoencoder to sample from the dataset distribution and then a second perturbing network that shifts the action towards maximizing the sampled CVaR. The perturbation is up to 0.5 (paper originally 0.05).

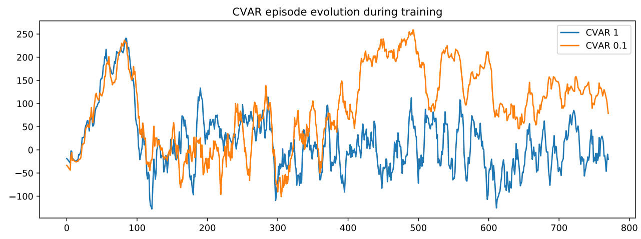

Figure 5.9: CVaR of the cumulative episode rewards over 10 episodes while training

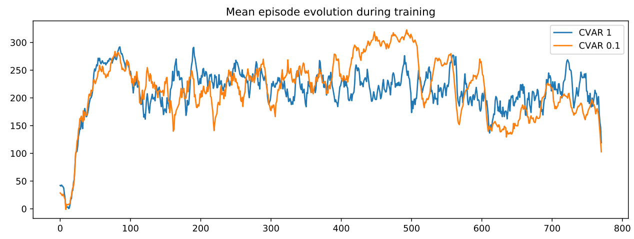

Figure 5.10: Mean of the cumulative episode rewards over 10 episodes while training

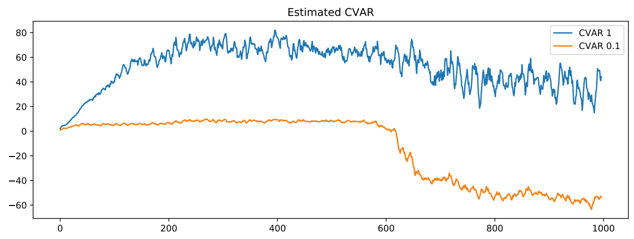

Figure 5.11: Sampled CVaR while training

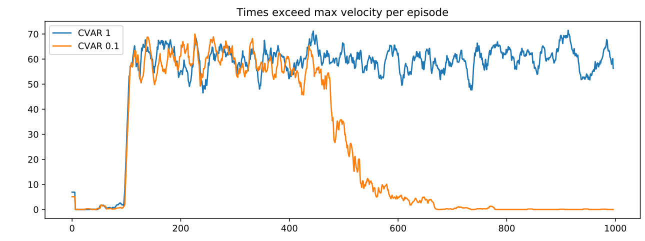

Figure 5.12: Times of velocity exceed while training

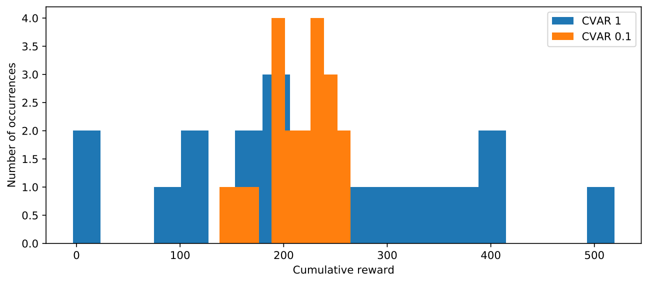

Figure 5.13: Histogram of cumulative reward during 200 timesteps using the trained final policies

5.4 CVaR minimization using DDPG and IQN

With DDPG algorithm we consider a parameterized deterministic policy $\mu_{\theta_{policy}}$ and we want to maximize the expected value of this policy by optimizing:

$$J(\theta) = \mathbb{E}[Q(x, \mu_{\theta_{policy}})] \tag{5.2}$$

Our goal:
Find policy $\mu_{\theta_{policy}}$ that minimizes the following:

$$\min_{\theta_{policy}} CVaR_\alpha[Z^\theta(x_0, \mu_{\theta_{policy}})] \tag{5.3}$$

where

$$Z^\theta(x_0, \mu_{\theta_{policy}}) = \sum_k^N \gamma^k c(x_k, a_k | \mu_{\theta_{policy}}, x_0) \tag{5.4}$$

is the distribution over returns.
We first red ourselves of the deterministic policy gradient theorem [**?**]:
Consider a determinstic policy $\mu_\theta(s) : \mathcal{S} \to \mathcal{A}$ with parameter vector $\theta \in \mathcal{R}^n$. We define: $G_T^\gamma = \sum_{k=0}^\infty \gamma^k r(x_k, a_k)$
We define a performance objective $J(\mu_\theta) = \mathbb{E}[G_T | \mu_{\theta_{policy}}, x_0)]$ We denote density

$$\min_{\theta_{policy}} \min_\nu \{\nu + \frac{1}{1-\alpha}\mathbb{E}_Z[[Z^\theta(x_0, \mu_{\theta_{policy}}) - \nu]^+]\} \tag{5.5}$$

SKETCH:
1) **Act:** using deterministic policy + exploration noise. Save in memory replay (s,a,r,s')
2) Learn critic: ie $Z^\theta(x_0, \mu_{\theta_{policy}})$ OFFLINE via IQN:
Batch replay (s,a,r,s').

$$Z(s, a|\tau, \theta_{currentcritic}) Z(s', \mu_{\theta_{targetpolicy}}(s')|\tau, \theta_{targetcritic}) + r \tag{5.6}$$

where $\tau \sim U[0,1]$ (same or different for Z and Z'?)
Update via gradient descent $\theta_{currentcritic}$ using quantile regression loss. From time to time update parameters of $\theta_{targetcritic}$
Issue: having continuous actions requires action to be passed as an input to the IQN network. (In the IQN paper they deal with discrete actions only so IQN outputs the value of the quantile per every action.) In our case: embed action input with state and quantile inputs with some operation (Hadamard product?)
3) LEARN ACTOR, ie $\mu_{\theta_{criticpolicy}}$:

$$\min_{\theta_{policy}} \min_\nu \{\nu + \frac{1}{1-\alpha}\mathbb{E}_Z[[Z^\theta(x_0, \mu_{\theta_{policy}}) - \nu]^+]\} \tag{5.7}$$

To be discussed: See next for updates My idea: Fix ν, and learn it separately.

$$\nabla_{\theta_{policy}}\{\nu + \frac{1}{1-\alpha}\mathbb{E}_Z[[Z^\theta(x_0, \mu_{\theta_{policy}}) - \nu]^+]\} == \frac{1}{1-\alpha}\nabla_{\theta_{policy}}\mathbb{E}_Z[[Z^\theta(x_0, \mu_{\theta_{policy}}) - \nu]^+] \tag{5.8}$$

From same batch replay samples (s,a,s',r): Compute sampled distribution $Z(s, \mu_{\theta_{targetpolicy}}|\tau, \theta_{currentcritic})$ where $\tau \sim U[0,1]$ Take samples from distribution via IQN:
$Z^\theta(s, \mu_{\theta_{policy}}(s)|\tau_j, \theta_{currentcritic}) - \nu]^+]$.mean() for every $\tau_j, j \in [0, N]$
and sum over all batch size samples.

Update $\theta_{currentpolicy}$ with gradient descent (min cvar for risk sensitive policies)
From time to time update parameters of $\theta_{targetpolicy}$
4) Learn ν using CVaR Sebastian code.
Problem: We need a ν fore every initial state x_0 I think? How to learn it?
3.1) LEARN ACTOR, ie $\mu_{\theta_{criticpolicy}}$:

$$\min_{\theta_{policy}} \min_{\nu} \{\nu + \frac{1}{1-\alpha} \mathbb{E}_Z[[Z^\theta(x_0, \mu_{\theta_{policy}}) - \nu]^+]\} \tag{5.9}$$

Sebastian says it is a jointly convex function so we can learn them together. Mainly, we don't even need to learn *nu*, just sample $\tau \sim U[0, \alpha]$, and minimize the mean of the samples. An advantage is that we wont' "lose" samples from the good-side of the distribution, since we have the RELU operation,which would discard those.

Bibliography

[1] R. Sutton and A. Barto, "Reinforcement Learning: An Introduction," *IEEE Transactions on Neural Networks*, 1998.

[2] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *34th International Conference on Machine Learning, ICML 2017*, 2017.

[3] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018.

[4] R. Koenker and K. F. Hallock, "Quantile regression," *Journal of Economic Perspectives*, 2001.

[5] Y. Chow, A. Tamar, S. Mannor, and M. Pavone, "Risk-sensitive and robust decision-making: A CVaR optimization approach," *Advances in Neural Information Processing Systems*, vol. 2015-Janua, pp. 1522–1530, 2015.

[6] G. Serraino and S. Uryasev, "Conditional Value-at-Risk (CVaR)," in *Encyclopedia of Operations Research and Management Science*, 2013.

[7] R. T. Rockafellar and S. Uryasev, "Optimization of conditional value-at-risk," *The Journal of Risk*, vol. 2, no. 3, pp. 21–41, 2000.

[8] P. Artzner, F. Delbaen, J. M. Eber, and D. Heath, "Coherent measures of risk," *Mathematical Finance*, 1999.

[9] D. Silver, G. Lever, D. Technologies, G. U. Y. Lever, and U. C. L. Ac, "Deterministic Policy Gradient (DPG)," *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[10] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, 1992.

[11] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 2012.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[13] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," in *35th International Conference on Machine Learning, ICML 2018*, 2018.

Appendix A

Anything

Bla bla . . .