Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eidgenössische Technische Hochschule Zü
Swiss Federal Institute of Technology Zuric

Prof. Name

Master Thesis

Thesis title

Spring Term 2020

Supervised by:
Sebastian Curi

Author:
Nuria Armengol Urpi

Contents

Preface iii

Abstract v

Symbols vii

1 Introduction 1

2 Problem description and Related work 2

2.1 Problem description 2

2.1.1 Reinforcement Learning 2

2.2 Risk 3

2.2.1 Conditional Value-at-Risk (CVaR) 4

3 The algorithm 7

3.1 Off-policy Deterministic Actor-Critic 7

3.1.1 Preliminaries 7

3.1.2 Deterministic policy gradients 8

3.1.3 Off-policy Deterministic actor-critic 8

3.2 Distributional off-policy deterministic actor critic 9

3.2.1 Distributional Critic 9

3.2.2 Actor 10

3.3 Technical Details of the algorithm 11

4 Distributional RL 13

4.1 Distributional RL 13

4.1.1 Example showing interest in learning the distribution 13

4.1.2 Distributional Bellman Operator 14

4.1.3 Quantile approximation 16

4.1.4 Quantile projection: 17

4.1.5 Quantile Regression 17

4.1.6 Quantile Regression Temporal Difference Learning 18

5 Batch RL **20**

5.1 Details of VAE 21

6 Results **23**

6.1 Current results Car 23

6.1.1 Case 1: No velocity penalization 23

6.1.2 Case 2: Velocity penalization with probability 1 24

6.1.3 Case 3: Velocity penalization with probability P 25

6.2 Current results Batch RL HalfCheetah 26

Bibliography **33**

A Anything **35**

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

Decision-making is the process of taking choices by identifying an optimal strategy from current system states. A crucial element in the process is the performance measure used to assess how good the choice is for the decision-maker or agent. In sequential decision-making problems, the most common optimization criterion is the expectation of the cumulative reward collected by the agent, which leads to a *risk-neutral* behavior.

However, this approach neither takes in to account the variability of the rewards (i.e fluctuations around the mean) nor its sensitivity to modeling errors. In some scenarios affected by measurement or modelling errors and in which the safety of the agent is particularly important, such as autonomous navigation or or finances, it is crucial to ensure that only *safe* action strategies will take place.
In many works, the concept of safety, or its opposite risk, is related to the inherent stochasticity of the environment. Under those environments, even an optimal policy with respect to the expected cumulative reward, may perform very poorly in some cases. Since maximizing the expected cumulative-reward does not necessarily imply the avoidance of rare occurrences of large negative outcomes, we need other criteria to evaluate risk. (García and Fernández, 2015).

Risk-sensitive decision-making provides a promising approach to compute robust and safe policies, but finding computationally tractable and conceptually meaningful methodologies for such a goal is non-trivial and still a challenge.

In this thesis, we focus on the reinforcement learning (RL) framework, a branch of machine learning that focuses on dynamic decision making in unknown environments, and propose a risk-sensitive approach to act *safely* in a non-deterministic environment.

Chapter 2

Problem description and Related work

2.1 Problem description

Standard reinforcement learning approaches aim to find policies which maximize the expected cumulative reward. However, this approach neither takes into account variability of the reward around its mean neither sensitivity of the policy to modelling errors.
In this thesis, we change the objective function of the standard RL approach to one that optimizes another metric of the reward distribution which takes into account the risk of the actions taken by the agent. While several metrics have been designed to assess risk, we will focus on a particular risk metric called Conditional Vale at Risk (CVaR).
We, thereby, present a RL algorithm that aims to find policies with optimal CVaR.

2.1.1 Reinforcement Learning

Reinforcement learning is an approach to learn a mapping from situations or states to actions so as to maximize a numerical reward signal. The learner or *agent* is not told which actions to take but instead must discover which actions yield the most reward by trying them. In most of the cases, actions may affect not only the immediate reward but also the next state and consequently, all subsequent rewards. These two characteristics—trial-and-error search and delayed reward—are the two most important distinguishing features of reinforcement learning Sutton and Barto (1998).

Markovian Decision Processes (MDPs)

We formalize the problem of RL by using the framework of Markov decision processes (MDPs) to define the interaction between a learning agent and its environment in terms of states, actions and rewards.

MDPs are discrete-time stochastic control processes which provide a mathematical framework for modeling sequential decision making in situations where outcomes are partly random and partly under the control of a decision maker and where actions influence not only immediate rewards, but also future ones.
An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$, where $\mathcal{S}, \mathcal{A}$ are the state and action spaces respectively, $R(x, a)$ is the reward distribution, $P(\cdot|x, a)$ is the transition probability distribution and $\gamma \in [0, 1]$ is the discount factor.

State transitions of an MDP satisfy the Markov property, in which the set of transition probabilities to next states depend only on the current state and action of the system, but are conditionally independent of all previous states and actions. Hence, the state must provide information about all aspects of the past agent-environment interaction that make a difference for the future.

Solving an MDP involves determining a sequence of policies π (mapping states to actions) that maximize an objective function. A commonly considered class of policies in literature are the class of Markovian policies Π_M where at each time-step t the policy π_t is a function that maps state x_t to the probability distribution over the action space $\mathcal{A}$. In the special case when the policies are time-homogeneous, i.e. $\pi_j = \pi \; \forall j \geq 0$ then the class of policies is known as stationary Markovian $\Pi_{M,S}$.
This set of policies, under which actions only depend on current state information and its state-action mapping is time-independent, makes the problem of solving for an optimal policy more tractable and common solution techniques involve dynamic programming algorithms (Bertsekas, 1995) such as Bellman iteration.
When the objective function is given by a risk-neutral expectation of cumulative reward, i.e.:

$$\min_{\pi \in \Pi_H} \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t R(x_t, a_t)|x_0, a_t \sim \pi_t(\cdot|h_t)\big] \tag{2.1}$$

where Π_H represents the set of all history-dependant policies, the Bellman's principle of optimality (Bertsekas, 1995) shows that the optimal policy lies in the class of stationary Markovian policies $\Pi_{M,S}$

When dealing with other types of objective functions that aim towards more risk-sensitive policies, these nice properties doesn't normally hold and require extra mathematical formulations, such as MDP state augmentation (Chow et al., 2015).

2.2 Risk

Standard reinforcement learning agents aim to maximize the expected cumulative reward and hence do not take risk into account. In some scenarios the shape of the reward distribution might be unimportant, since highly different distributions still can have same expectation. However, in real world scenarios, in when catastrophic losses can occur, risk must be taken into account.
We can find 3 types of strategies with respect to risk, namely *risk neutral, risk averse* and *risk seeking*

As an example, suppose a participant in a game is told to choose between two doors. One door hides 1000CHF and the other 0CHF. The host also allows the contestant to take 500CHF instead of choosing a door. The two options (choosing between door 1 and door 2, or taking 500CHF) have the same expected value of 500CHF. But it can clearly be seen that the risk among two options is different. Since the expected value is the same, risk neutral contestant is indifferent between these choices. A risk-seeking contestant will maximize its utility from the uncertainty and hence choose a door, whereas the risk-averse contestant will accept the guaranteed 500CHF.

We can segment RL algorithms accounting for risk in two main categories: the first transform the optimization criterion to introduce the concept of risk, whereas the second, modifies the exploration process to avoid exploratory actions that can lead to undesirable situations.
We will focus on the first category, which can be divided into 3 subcategories: worst-case criterion, constrained criterion and risk-sensitive criterion.
Worst-case or minimax criterion has been studied by Heger (1994), Coraluppi and Marcus (2000) and Coraluppi and Marcus (1999), in which the objective is to compute a control policy that maximizes the expectation of the return with respect to the worst case scenario. In general, minimax criterion has been found to be too restrictive as it takes into account severe but extremely rare events which may never occur.
Constrained criterion aims to maximize the expectation of the return while keeping other types of expected utilities higher than some given bounds (Altman, 1993). It may be seen as finding the best policy π in the space of considered safe policies. This space may be restricted by different constraints: ensuring that the expectation of return exceeds some specific minimum threshold (Geibel, 2006) or that te variance of return doesn't exceed a given threshold (Tamar et al., 2012). This constraint problems can be converted to equivalent unconstrained ones by using penalty methods or a Lagrangian approach.
Finally, risk-sensitive criterion use other utility metrics to be maximized instead of expectation of cumulative rewards. Lot of research has been done using exponential utility functions (Howard and Matheson, 1972; Chung and Sobel, 1987).
A risk metric that has recently gained a lot of popularity is the Conditional Value at Risk, due to its favorable computation properties and superior ability to safeguard a decision maker from the "outcomes that hurt the most" Serraino and Uryasev (2013).
In this thesis we present an optimization approach for this metric.

2.2.1 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 the reward distribution. 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):

$$\mathrm{CVaR}_\alpha(Z) = \mathbb{E}[Z|Z \leq \mathrm{VaR}_\alpha] = \frac{1}{\alpha}\int_0^\alpha F_Z^{-1}(\tau)d\tau = \frac{1}{\alpha}\int_0^\alpha \mathrm{VaR}_\tau(Z)d\tau \qquad (2.2)$$

While both VaR and CVaR are risk measures, only CVaR is coherent in the sense of Artzner et al. (1999). In addition, CVaR takes into account the possibility of tail events where losses exceeds VaR whereas VaR is incapable of distinguishing situations beyond it.

Given its properties, Rockafellar and Uryasev (2000) also showed that CVaR is equivalent to the solution of the following optimization problem:

$$\mathrm{CVaR}_\alpha(Z) = \max_\nu \bigl\{\nu + \frac{1}{\alpha}\mathbb{E}_Z[[Z-\nu]^-]\bigr\} \tag{2.3}$$

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 Artzner et al. (1999):

$$\mathrm{CVaR}_\alpha(Z) = \min_{\xi \in U_{\mathrm{CVaR}}(\alpha,\mathbb{P})} \mathbb{E}_\xi[Z] \tag{2.4}$$

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

$$U_{\mathrm{CVaR}}(\alpha,\mathbb{P}) = \Bigl\{\xi | \xi(w) \in \bigl[0, \frac{1}{\alpha}\int_{w\in\Omega} \xi(w)\mathbb{P}(w)dw = 1\bigr]\Bigr\} \tag{2.5}$$

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

Due to its superior mathematical properties and practical implications, CVaR optimization has gained a lot interest in the risk-sensitive literature.
CVaR optimization aims to find the parameters θ that maximizes $\mathrm{CVaR}_\alpha(Z)$, where the reward distribution Z is parameterized by a controllable parameter θ, such that: $Z = f(X;\theta)$.
In the simplest scenarios, where X is not dependant on θ CVaR optimization may be solved using various approaches such in Rockafellar and Uryasev (2000).
However, in many domains, such as reinforcement learning, the tunable parameter θ also affects the distribution of X, and hence the standard existing approaches for CVaR optimization are not suitable.
Additionally, most of the work in these domains, has been done in the context of MDPs when the model is known (Chow et al., 2015; Petrik and Subramanian, 2012), by using dynamic programming methods and not much research has been done for the RL framework.

Morimura et al. (2010a) and Morimura et al. (2010b) focused on estimating the density of the returns to handle CVaR risk criteria. However the resulting distributional-SARSA-with-CVaR algorithm they propose has proved effectiveness only in very simple and discrete MDP.
Tamar et al. (2015a) proposed a policy gradient (PG) algorithm for CVaR optimization, by deriving a sampling based estimator for the gradient of CVaR and used it to optimize the CVaR by stochastic gradient descent. However they only considered continuous loss distributions and they used empirical α-quantile to estimate VaR_α which is known to be a biased estimator for small samples.
Chow and Ghavamzadeh (2014) also proposed a PG algorithm for mean-CVaR optimization, which has several disadvantages also shared with Tamar et al. (2015b). First, by definition of PG algorithms, they suffer from high variance on the gradient estimates, especially when the trajectories are long. This high variance is even more exacerbated when using very low quantiles α for the CVaR since the averaging on the rewards is effectively only on αN samples. Second, they are both very sample

inefficient since only the rewards below the quantile are used to compute the gradient.
Third, they are both trajectory-based (not incremental), i.e they only update after observing one or more full trajectories.
Chow and Ghavamzadeh (2014) also proposed both a trajectory-based and incremental actor-critic approaches which help in reducing the variance of PG algorithms. However, they require state augmentation of the original MDP formulation to a state-space $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{Y} = (0, 1]$ is an additional continuous state that represents the different confidence levels. This allows to include as a critic the CVaR value function and it is necessary due to the time-inconsistency of CVaR metric.
It is also important to be noticed, that all the aforementioned algorithms are on-policy approaches. This is first another source of sample inefficiency because they cannot exploit data from experts or other sources. Additionally they constraint the learning process to happen online while interacting with the environment, which in real-case scenarios, for example robotics, can lead to very detrimental situations for the robot and hence it is, paradoxically, risky.

Most recently, Dabney et al. (2018a) presented a distributional RL approach to train risk-averse and risk-seeking agents, using, among others, CVaR as risk objective. The approach uses Q-learning and hence, it is only suitable for discrete action spaces.

In this thesis, we present an off-policy, actor-critic, model-free algorithm for CVaR optimization, based on the deterministic policy gradient algorithm that can operate over continuous action spaces.
Instead of empirically estimating the VaR from the observed rewards or using the CVaR Bellman equation, we rely on the recent advances in distributional RL (Bellemare et al., 2017; Dabney et al., 2018b,a) to estimate the full *value distribution* (i.e. the distribution of the random return received by a RL agent). We then compute the CVaR of the current policy via sampling from the estimated value distribution and maximize it via stochastic gradient ascent.

Chapter 3

The algorithm

We introduce an off-policy, model-free algorithm for CVaR optimization using deep function approximators that can learn policies in high-dimensional, continuous action spaces. Our work is based on the deterministic policy gradient algorithm.
Specifically, we use an actor-critic approach: the critic uses a distributional variant of RL and it is trained to estimate the whole value distribution, whereas the actor is trained via gradient ascent to maximize the CVaR of this distribution.
In the following, we briefly describe the standard DPG algorithm 3.1, then we introduce our distributional approach and the way we move towards risk-sensitive policies. Extensive and more theoretical information on the distributional RL approach is addressed in chapter 4

3.1 Off-policy Deterministic Actor-Critic

change title not to repeat with subsection?

3.1.1 Preliminaries

Goal in standard RL is to learn a policy π^* which maximizes the expected return or (discounted) cumulative reward R collected by the agent when acting in an environment E starting from any initial state x. Action-value function is used in many RL algorithms and describes the expected return after taking an action a in state x, and thereafter following policy π:

$$Q^\pi(x_t, a_t) = \mathbb{E}_{r_{i\geq t}, x_{i>t}\sim E, a_{i>t}\sim\pi}\Big[\sum_{i=t}^{T}\gamma^{(i-t)} r(x_i, a_i)\Big] \tag{3.1}$$

Bellman's equation describes this value Q using the recursive relationship between the action-value of a state and the action-values of its successor states:

$$Q^\pi(x_t, a_t) = \mathbb{E}_{r_t, x_{t+1}\sim E}\Big[r(x_t, a_t)] + \gamma\mathbb{E}_{a_{t+1}\sim\pi}\big[Q^\pi(x_{t+1}, a_{t+1})\big]\Big] \tag{3.2}$$

DPG algorithm (Silver et al., 2014) is characterized for using deterministic policies $a_t = \mu_\theta(x_t)$.

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, hence we can remove the inner expectation in 3.2 to:

$$Q^{\pi}(x_t, a_t) = \mathbb{E}_{r_t, x_{t+1} \sim E}\Big[r(x_t, a_t) + \gamma Q^{\pi}(x_{t+1}, \pi(x_{t+1}))\Big] \tag{3.3}$$

This means that it is possible to learn the value function Q^{π} off-policy, i.e. using environment interactions which are generated by acting under a different stochastic behavior policy β (where $\beta \neq \pi$) which ensures enough exploration. An advantage of off-policy algorithms is that we can treat the problem of exploration independently from the learning algorithm.

To learn the optimal policy, Q-learning Watkins and Dayan (1992), a commonly used off-policy algorithm, first learns the optimal value function Q^* by iteratively applying the Bellman optimality operator to the current Q estimate:

$$Q(x_t, a_t) \leftarrow \mathbb{E}_{r_t, x_{t+1} \sim E}\Big[r(x_t, a_t) + \gamma \max_{a_{t+1}} Q(x_{t+1}, a_{t+1})\Big] \tag{3.4}$$

which is a contraction mapping proved to converge exponentially to Q^*, and then derives the optimal policy $\pi^*$from it via the greedy policy $a^* = \operatorname*{argmax}_a Q^*(x, a)$.

When dealing with continuous actions, it is not possible to apply Q-learning straightforward 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, policy gradient methods are used in which a *parameterized policy* is learnt to be able to select actions without consulting the value function.

3.1.2 Deterministic policy gradients

The deterministic policy gradient described by Silver et al. (2014) updates the parameters of the deterministic policy π_θ via gradient ascent to maximize an objective function $J(\pi_\theta)$:

$$J(\pi_\theta) = \mathbb{E}_{x \sim \rho^{\pi}}\left[Z(x, \pi_\theta(x))\right] \tag{3.5}$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{x \sim \rho^{\pi}}\left[\nabla_\theta \pi_\theta(x) \nabla_a Q^{\pi}(x, a)|_{a=\pi_\theta(x)}\right] \tag{3.6}$$

where ρ^{π} is the discounted state distribution when acting under policy π.

3.1.3 Off-policy Deterministic actor-critic

When we both learn approximations of Q and policy, the method is called deterministic actor-critic. The actor is the learned policy which is updated with respect to the current value estimate, or critic. Sutton and Barto (1998).

In the off-policy setting, the critic parameterized by θ^Q estimates the action-value function $Q^{\pi}(x, a)$ off-policy from trajectories generated by a behavior policy β (argumented in 3.1.1) using the Bellman equation. It learns by minimizing the loss:

$$\mathcal{L}(\theta^Q) = \mathbb{E}_{x_t \sim \rho^\beta, a_t \sim \beta, r_t \sim E}\left[(Q(x_t, a_t|\theta^Q) - y_t)^2\right] \tag{3.7}$$

$$y_t = r(x_t, a_t) + \gamma Q(x_{t+1}, \pi(x_{t+1})|\theta^Q) \tag{3.8}$$

The actor, parameterized by θ^π, learns via gradient ascent by using the off-policy deterministic policy gradient, which was proved by Silver et al. (2014) to be the policy gradient, i.e. the gradient of the policy's performance:

$$J_\beta(\pi|\theta^\pi) = \int_{\mathcal{X}} \rho^\beta(s) Q^\pi(x, \pi(x|\theta^\pi))dx$$

$$\nabla_{\theta^\pi} J_\beta(\pi|\theta^\pi) \approx \mathbb{E}_{x \sim \rho^\beta}\left[\nabla_{\theta^\pi}\pi(x, |\theta^\pi)\nabla_a Q^\pi(x, a)|_{a=\pi(x|\theta^\pi)}\right] \tag{3.9}$$

where ρ^β is the discounted state distribution when acting under behavior policy β. By propagating the gradient through both policy and Q, the actor learns an approximation to the maximum of the value function under target policy π averaged over the state distribution of the behavior policy β.

A term that depends on $\nabla_\theta Q^{\mu_\theta}(x, a)$ has been dropped in following a justification given by Degris et al. (2012) that argues that this is a good approximation since it can preserve the set of local optima to which gradient ascent converges.

Similarly to Lillicrap et al. (2016), we will use neural networks as non-linear function approximators for learning both action-value functions and the deterministic target policy.

3.2 Distributional off-policy deterministic actor critic

We present the CVaR optimization algorithm which is the main contribution of this thesis. The algorithm is based on the original off-policy deterministic actor critic explained in previous section 3.1, but introduces a distributional critic, which estimates the whole value distribution instead of only its expected value. With this extra information, the actor can learn to maximize other metrics than the expected value, specifically the CVaR. We proceed to present the components of the algorithm:

3.2.1 Distributional Critic

We use a distributional variant of the standard critic function, which maps from state-action pairs to distributions, similar to the implicit quantile network (IQN) introduced in Dabney et al. (2018a).
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 $Z(x, a)$ as the random variable representing the return, with cumulative distribution function $F(z) := P(Z \leq z)$ and we define $F_Z^{-1}(\tau) := Z(x, a; \tau)$ as its quantile function (or inverse cumulative distribution function) at $\tau \in [0, 1]$. Thus, for $\tau \in U([0, 1])$, the resulting state-action return distribution sample is $Z(x, a; \tau) \sim Z(x, a)$.

The critic IQN network $Z(x, a; \tau|\theta^Z)$ parameterized by θ^Z, is hence a parametric function used to represent the quantile function at specific quantile levels.

As in Dabney et al. (2018a),we train the IQN network using the sampled quantile regression loss (Koenker, 2005) on the pairwise temporal-difference (TD)-errors. For two samples $\tau, \tau' \sim U([0,1])$, and current policy π_{θ^π}, the sampled TD error is:

$$\delta^{\tau,\tau'} = r + \gamma Z(x_{t+1}, \pi(x_{t+1}|\theta^\pi); \tau'|\theta^Z) - Z(x_t, a_t; \tau|\theta^Z) \tag{3.10}$$

Then, we compute the quantile regression loss:

$$\mathcal{L}_{QR}(x_t, a_t, r_t, x_{t+1}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N'} \rho_{\tau_i}(\delta^{\tau_i,\tau'_j}) \tag{3.11}$$

where ρ is as defined as:

$$\rho_\tau(u) = u(\tau - \delta_{u<0}), \forall u \in \mathbb{R}$$

where N and N' are the number of iid samples $\tau_i, \tau'_j \sim U([0,1]$ used to estimate the loss.
By minimizing 3.11 via stochastic gradient descent with respect to θ^Z we move towards the true quantile function (Koenker, 2005).

3.2.2 Actor

The policy is updated via deterministic policy gradient ascent. We modify equation (3.9), to include the action-value distribution.

$$\nabla_{\theta^\pi} J_\beta(\pi|\theta^\pi) \approx \mathbb{E}_{x\sim\rho^\beta}\left[\nabla_{\theta^\pi}\pi(x, |\theta^\pi)\nabla_a Q^\pi(x,a)|_{a=\pi(x|\theta^\pi)}\right] \tag{3.12}$$
$$= \mathbb{E}_{x\sim\rho^\beta}\left[\nabla_{\theta^\pi}\pi(x, |\theta^\pi)\mathbb{E}[\nabla_a Z^\pi(x, a|\theta^Z)]|_{a=\pi(x|\theta^\pi)}\right] \tag{3.13}$$

Step from (3.12) to (3.13) comes by the fact that

$$Q^\pi(x,a) = \mathbb{E}[Z^\pi(x,a)] \tag{3.14}$$

With our goal of CVaR optimization in mind:

$$\arg\max_\pi \mathrm{CVaR}_\alpha[Z(x, \pi(x))] \quad \forall x \in \mathcal{X} \tag{3.15}$$

we can make use of the information provided by the Z distribution to optimize other objective functions rather than the expected value.

To approach this, we use as a performance objective the distorted expectation of Z(x,a) under the distortion risk measure $\phi : [0,1] \to [0,1]$, with identity corresponding to risk-neutrality, i.e.:

$$Q_\phi(x,a) = \mathbb{E}_{\tau\sim U([0,1])}[Z_{\phi(\tau)}(x,a)] \tag{3.16}$$

which is actually equivalent to the expected value of $F^{-1}_{Z(x,a)}$ weighted by ϕ, i.e.:

$$Q_\phi(x,a) = \int_0^1 F_Z^{-1}(\tau) d\phi(\tau) \tag{3.17}$$

When we use as mapping $\phi(\tau) = \alpha\tau$, 3.17 is actually the CVaR of Z(x,a) as already presented in 2.2, and as a reminder:

$$\text{CVaR}_\alpha(Z) = \frac{1}{\alpha}\int_0^\alpha F_Z^{-1}(\tau) d\tau \tag{3.18}$$

We can hence approximate (3.18) via sampling, by taking K samples of $\tau \sim U[0,\alpha]$:

$$\text{CVaR}_\alpha(Z) \approx \frac{1}{\alpha}\frac{1}{K}\sum_{i=1}^{K} Z(x,a;\tau_i) \qquad \tau_i \sim U[0,\alpha] \quad \forall i \in [1,K] \tag{3.19}$$

Finally, then, we use the following *distorted policy gradient* for the actor network:

call it like that?

$$\begin{aligned} J_\beta^{CVAR}(\pi|\theta^\pi) &= \int_{\mathcal{X}} \rho^\beta(s)\text{CVaR}_\alpha(Z^\pi(x,\pi(x|\theta^\pi))dx \\ \nabla_{\theta^\pi} J_\beta(\pi|\theta^\pi) &\approx \mathbb{E}_{x\sim\rho^\beta}\left[\nabla_{\theta^\pi}\pi(x,|\theta^\pi)\nabla_a[\frac{1}{\alpha}\frac{1}{k}\sum_{i=1}^{k} Z(x,a;\tau_i)|\theta^Z)]|_{a=\pi(x|\theta^\pi)}\right] \end{aligned} \tag{3.20}$$

where $\tau_i \in U([0,\alpha])\ \forall i$

We again remark the capability of the algorithm to be implemented off-policy. We use a more-exploratory behavior policy β to learn a target policy π (where $\beta \neq \pi$) and, in contrast to stochastic off-policy actor-critic algorithms (Degris et al., 2012), we can avoid importance sampling both in the actor and the critic. This is due to the deterministic essence of the policies, which removes the integral over actions in the policy gradient and the expected value over actions in the Bellman equation for the critic.

3.3 Technical Details of the algorithm

s

- Explain replay buffer characteristics
- Target networks and update frequencies
- Adam optimizers (p4 lillicrap)
- Ornstein Noises and exponential decays
- Add pseudo-code of the algorithm

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.

use replay buffer, target networks as DQN to deal with problems of Qlearning with functions approximators.

Since distorted expectations can be expressed as weighted average over the quantiles Dhaene et al. (2012), we can use a specific sampling base distribution $\beta : [0, 1] \rightarrow [0, 1]$ to sample the quantile levels $\tau \in [0, 1])$ from our critic network $Z(x, a; \tau)$.

Chapter 4

Distributional RL

4.1 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 Sutton and Barto (1998) 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:

$$Q(x, a) = \mathbb{E}[R(x, a)] + \gamma \mathbb{E}[Q(X', A')] \tag{4.1}$$

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

4.1.1 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 (i.e. reward of -2000CHF), whereas otherwise we we collect a salary of 200CHF (i.e. reward of +200CHF). If we consider the common reinforcement learning approach and we compute the expected immediate ($\gamma = 1$) reward:

$$\mathbb{E}[R(x)] = \frac{1}{36}(-2000\,\text{CHF}) + \frac{35}{36}(200\,\text{CHF}) = 138.88\,\text{CHF} \tag{4.2}$$

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:

$$Q^{\pi}(x,a) = \mathbb{E}[Z^{\pi}(x,a)] \tag{4.3}$$

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

$$Z^{\pi}(x,a) \overset{D}{=} R(x,a) + \gamma Z(x',a') \tag{4.4}$$

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 (4.4), 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.*

4.1.2 Distributional Bellman Operator

In the policy evaluation setting Sutton and Barto (1998), one aims to find the value function $V\pi$ associated with a given fixed policy π. In the distributional case, we aim to find $Z\pi$. Bellemare et al. (2017) 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 $P^{\pi} : \mathbb{Z} \rightarrow \mathbb{Z}$

$$P^{\pi}Z(x,a) \overset{D}{=} Z(X',A') \tag{4.5}$$

$$X' \backsim \ P(\cdot|x,a) \text{ and } A' \backsim \ \pi(\cdot|X') \tag{4.6}$$

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:

$$T^{\pi}Z(x,a) \overset{D}{=} R(x,a) + \gamma P^{\pi}Z(x,a) \tag{4.7}$$

Bellemare et al. (2017) showed that (4.7) 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:

$$W_p(U, Y) = \Big(\int_0^1 |F_Y^{-1}(w) - F_U^{-1}(w)|^p dw \Big)^{\frac{1}{p}} \tag{4.8}$$

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

$$F_Y^{-1}(w) := \inf\big\{ y \in \mathbb{R} \,|w \ \leq F_Y(w) \big\} \tag{4.9}$$

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

Add Figure 2.1 in Dabney et al. (2018b)

Contraction in $\hat{d}_p$:

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

$$\mathcal{Z} = \big\{ Z \mid \mathcal{X} \times \mathcal{A} \rightarrow \wp(\mathbb{R}) \tag{4.10}$$

$$\mathbb{E}[|Z(x, a)|^p < \infty, \forall (x, a), p \geq 1] \big\} \tag{4.11}$$

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:

$$\hat{d}_p(Z_1, Z_2) := \sup_{x,a} W_p(Z_1(x, a), Z_2(x, a)) \tag{4.12}$$

Bellemare et al. (2017) 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 $Z_0 \in \mathcal{Z}$.
$T^\pi Z : \mathcal{Z} \rightarrow \mathcal{Z}$ 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 Bellemare et al. (2017))
T^π is a γ-contraction: for any two $Z_1, Z_2 \in \mathcal{Z}$,

$$\hat{d}_p(T^\pi Z_1, T^\pi Z_2) \leq \gamma \hat{d}_p(Z_1, Z_2) \tag{4.13}$$

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, i.e. 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, i.e., when we try to find a policy π^* that maximizes a value, or its distributional analogous, i.e. 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 $\mathit{\Pi}^*$ 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

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

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:

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

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

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

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 Bellemare et al. (2017) 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.

4.1.3 Quantile approximation

Dabney et al. (2018b) used the theory of quantile regression Koenker (2005), 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.

4.1.4 Quantile projection:

Our current aim is to estimate quantiles of the target distribution, i.e. 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:

$$Z_\theta(x, a) \coloneqq \frac{1}{N} \sum_{i=1}^{N} \delta_{\theta_i(x,a)} \tag{4.14}$$

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:

$$\prod_{W_1} Z \coloneqq \underset{Z_\theta \in \mathcal{Z}_Q}{\arg\min} W_1(Z, Z_\theta) \tag{4.15}$$

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 (4.14) with support $\{\theta_1, ... \theta_N\}$ by:

$$W_1(Y, U) = \sum_{i=1}^{N} \int_{\tau_{i-1}}^{\tau_i} |F_Y^{-1}(w) - \theta_i| dw \tag{4.16}$$

Lemma 2 in Dabney et al. (2018b) 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 $\hat{\tau}_i = \frac{\tau_{i-1}+\tau_i}{2}$.

4.1.5 Quantile Regression

Quantile regression is a method for approximating quantile functions of a distribution at specific points, i.e. 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:

$$\mathcal{L}^{\tau}_{QR}(\theta) = \mathbb{E}_{\hat{Z}\backsim Z}[\rho_{\tau}(\hat{Z} - \theta)] \tag{4.17}$$
$$\rho_{\tau}(u) = u(\tau - \delta_{u<0}), \forall u \in \mathbb{R}$$

Given that the minimizer of the quantile regression loss for τ is $F_Z^{-1}(\tau)$, and using Lemma 2 in Dabney et al. (2018b), 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:

$$\sum_{i=1}^{N} \mathbb{E}_{\hat{Z}\sim Z}[\rho_{\hat{\tau}_i}(\hat{Z} - \theta_i)] \tag{4.18}$$

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

add huberloss

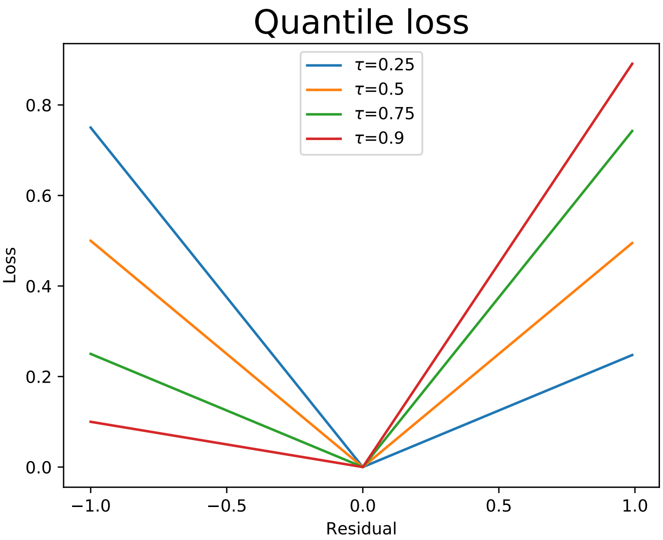

Figure 4.1: Quantile loss for different quantile values

Proposition 2 in Dabney et al. (2018b) 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 $\hat{Z}^{\pi}$.

4.1.6 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 (4.17). 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:

$$
\begin{aligned}
u &= r + \gamma z' - \theta_i(x) && (4.19)\\
\theta_i(x) &\leftarrow \theta_i(x) + \alpha(\hat{\tau}_i - \delta_{u<0}) && (4.20)\\
a \backsim \pi(\cdot|x), r \backsim R(x,a), x' &\backsim P(\cdot|x,a), z' \backsim Z_\theta(x') && (4.21)
\end{aligned}
$$

where Z_θ is a quantile distribution as in (4.14) and $\theta_i(x)$ is the estimated value of $F^{-1}_{Z^\pi(x)}(\hat{\tau}_i)$ in state x.

Chapter 5

Batch RL

- Watch videos Levine
- Papers bear and bcq

We decide to test the capabilities of our algorithm in a *fully* off-policy setting, also called *batch RL setting* or *offline* RL. In this setting, the agent can only learn from a fixed dataset without further interaction with the environment.

Add motivation in doing so: envs where collection of data is expensive, unsafe for robotics or autonomous vehicles

The ’off-policy’ algorithm we presented in previous sections falls in the category of off-policy *”growing batch learning”* in which data is collected by using near-on-policy policies such as ϵ-greedy and stored in a replay buffer. After used for training, the data is replaced with *fresher* data obtained from interaction of the agent with the environment using an updated policy. As a result, the dataset used tends to be heavily correlated to the current policy.

Issues with Batch RL

Most of off-policy algorithms fail to learn in the off-line setting. This is due to a fundamental problem of off-policy RL, called extrapolation error (Fujimoto et al., 2019) or bootstrapping error (Kumar et al., 2019). This error is introduced due to a mismatch between the dataset distribution and the state-action visitation distribution induced by the current target policy. At every train step the Q estimate is updated in the direction to reduce the Bellman error, i.e. the mean squared error between the current value estimate and the expected Q value under the current target policy at the next state. The Q function estimator, however, is valid only when evaluated on actions sampled from the behavior policy, which in the batch-RL case is the distribution of the dataset. Using unfamiliar (unlikely or not contained in the dataset) action (also called out of distribution (OOD) actions in (Kumar et al., 2019)) for the next-state, results on a new Q value estimate which is affected by this extrapolation error, resulting in pathological values that incur large absolute error from the optimal desired Q-value.

It is good to notice, that for an on-policy settings, extrapolation error is generally something positive, since it leads to a beneficial exploration. In this case, if the value function is overestimating the value at a (state-action) pair, the current policy will lead the agent to that pair, collect the data at that point and hence, the value estimate will be corrected afterwards. In the off-policy setting, the correction step is not possible due to the inability of collecting new data.

Our approach To overcome this issue, we inspire ourselves on the approach presented in Fujimoto et al. (2019), where a generative model G_w is trained to generate actions with high similarity to the dataset. For the generative model we use a conditional variational auto-encoder (VAE) Kingma and Welling (2014) which generates action samples as a reasonable approximation to $\underset{a}{\operatorname{argmax}} P^G_{\mathcal{B}}(a|s)$, where $P^G_{\mathcal{B}}(a|s)$ is the conditioned marginal likelihood.

rewrite

5.1 Details of VAE

A variational autoencoder aims to maximize the marginal log-likelihood $\log p(X) = \sum_{i=1}^{N} \log p(x_i)$, where X is the dataset with iid samples $\{x_1, x_2, ...x_N\}$. It is assumed that data is generated by some random process, involving an unobserved continuous random variable **z**. The process consists of two steps: (1) a value z_i is generated from some prior distribution p(z) and (2) a value x_i is generated from some conditional distribution $p(x|z)$. Given that the true probabilities are unknown, a recognition model $q(z|x;\phi)$ is introduced as an approximation to the intractable true posterior $p(z|x;\theta)$.

The recognition model $q(z|x;\phi)$ is called an *encoder*, since given a datapoint x it produces a *random latent vector* z. $p(x|z;\theta)$ is called a *decoder*, since given the random latent vector z it reconstructs the original sample x.

Since computing the desired marginal $p(X;\theta)$ is intractable, VAE algorithm optimizes a lower bound instead:

$$\log p(X;\theta) \geq \mathcal{L}(\theta,\phi;X) = \mathbb{E}_{q(z|X;\phi)}[\log p(X|z;\theta)] - D_{KL}(q(z|X;\phi) \mid\mid p(z;\theta)) \quad (5.1)$$

For our implementation, the prior $p(z;\theta)$ is chosen to be a multivariate normal distribution $\mathcal{N}(0, Id)$, hence it lacks parameters.

For the probabilistic encoders and decoders we use neural networks. For the encoder $q(z|x_i;\phi)$ we used a neural network with Gaussian output, specifically a multivariate Gaussian with a diagonal covariance structure $\mathcal{N}(z|\mu(X), \sigma^2(X)Id)$, where μ and σ are the outputs of the neural network, i.e. nonlinear functions of datapoint $x_i := (state_i, action_i)$ and ϕ. To sample from the posterior $z_i \sim q(z|x_i;\phi)$ we use the reparameterization trick: $z_i = g(x_i, \epsilon; \phi) = \mu_i + \sigma_i \odot \epsilon$ where $\epsilon \sim \mathcal{N}(0, Id)$ and $\odot$ is the element-wise product. For the decoder $p(x|z;\theta)$ we used another neural network with deterministic output, i.e. nonlinear function of datapoint $\hat{x}_i := (state_i, z_i)$ and θ.

The VAE is trained to maximize reconstruction loss and a KL-divergence term according to the distribution of the latent vector:

When it comes to training the VAE, both recognition model parameters ϕ and the generative model parameters θ are learnt jointly to maximize the variational lower bound $\mathcal{L}(\theta, \phi; X)$ via gradient ascent which includes the expected reconstruction error loss and the KL-divergence term according to the distribution of the latent vectors. When both prior and posterior are Gaussian, KL-divergence can be computed analytically:

$$- D_{KL}(q(z|X;\phi) \mid\mid p(z;\theta)) = \frac{1}{2}\sum_{j=1}^{J}(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2) \tag{5.2}$$

where J is the dimensionality of z, and μ_j, σ_j represent the jth element of these vectors. The expected reconstruction error $\mathbb{E}_{q(z|X;\phi)}[\log p(X|z;\theta)]$ requires estimation by sampling, and we will use the mean-squared error between the $action_i$ from the dataset and the reconstructed action.

Finally, when acting during evaluation or deployment, random values of z will be sampled from the multivariate normal and passed through the decoder to produce actions.

add:

- New actor network
- VAE(state) + perturbation_level × DeterministicActorNN(vae_action, state)
- Train DeterministicActorNN as in the algorithm presented previously for off-policy RL.
- Remove randn sampling of z in deployment? It works :)

Chapter 6

Results

6.1 Current results Car

Problem: A car with fully-observable 2D-state: [position, velocity] needs to move from initial position $x_0 = 0\text{m}$ 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 $T_F = 400\text{ts}$ the episode ends (with no extra penalization).

6.1.1 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 $x_F = 2\text{m}$ with 20 time-steps. Hence the final cumulative reward $G_T = (20 + 1)R_{\text{ts}} + R_F = 60$.

Starting from $x_0 = 0\text{m}$, the car reaches a velocity of $1\text{m}\,\text{ts}^{-1}$ after 10 time-steps, at $x_{\tau=10} = 0.5$. 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 $G_T = (10 + 14 + 1)R_\tau + R_F = 74$. 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.

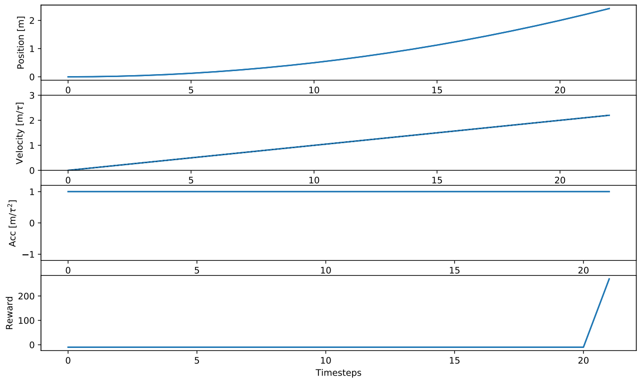

Figure 6.1: Car trajectory using DDPG and algorithm without velocity penalization. (Same behavior for CVAR-DDPG algorithm).

6.1.2 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 $1\mathrm{m\,ts}^{-1}$, 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 $1\mathrm{m\,ts}^{-1}$ is reached, and then they keep the velocity constant until the goal is reached.

Starting from $x_0 = 0\mathrm{m}$, the car reaches a velocity of $1\mathrm{m\,ts}^{-1}$ after 10 time-steps, at $x_{\mathrm{ts}=10} = 0.5$. Keeping velocity $1\mathrm{m\,ts}^{-1}$ through the rest of the episode, it reaches the goal position after 14 time-steps. Hence the final cumulative reward $G_T = (10 + 14 + 1)R_{\mathrm{ts}} + R_F = 20$. The reward values were chosen in order to make sure that, for this Case 2 setting, driving with a velocity higher than $1\mathrm{m\,ts}^{-1}$ never induces higher cumulative rewards.

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 $\backsim U[0,1]$

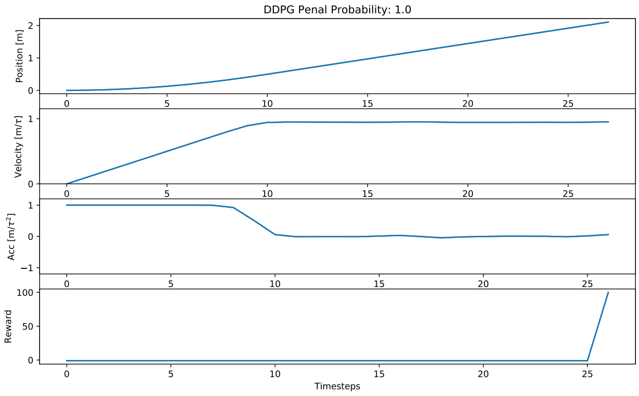

Figure 6.2: Car trajectory using DDPG algorithm and velocity penalization with probability 1

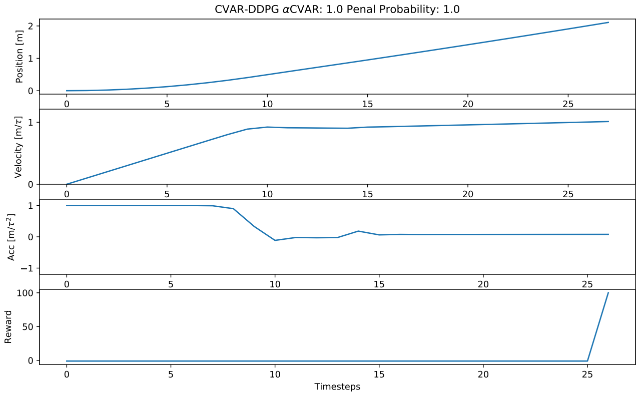

Figure 6.3: Car trajectory using CVAR-DDPG algorithm and velocity penalization with probability 1. (α-CVAR = 1)

6.1.3 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 $\alpha = 0.1$

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 $\alpha = 0.2$.

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 6.7 , shows the evolution of the sampled mean of the tail of the sampled cumulative value distribution (CDF). (i.e. 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.

$$\mathrm{CVaR}_\alpha(Z) = \frac{1}{\alpha}\int_0^\alpha F_Z^{-1}(\tau)d\tau = \frac{1}{\alpha}\int_0^\alpha IQN(\tau)d\tau \approx \frac{1}{\alpha}\frac{1}{K}\sum_{i=0}^{K} IQN(\tau_i) \qquad (6.1)$$

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 6.7 are not divided by α neither K)

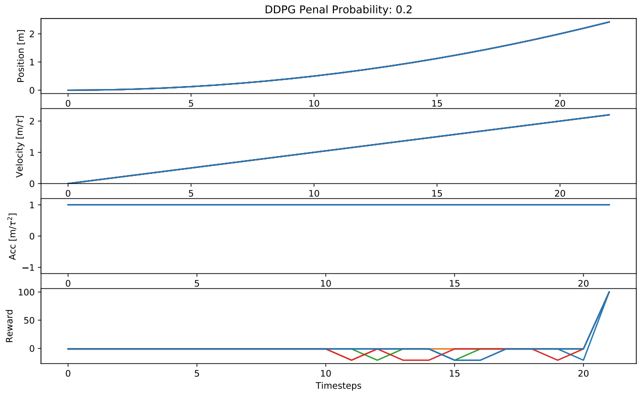

Figure 6.4: Car trajectory using DDPG algorithm and velocity penalization with probability P=0.2

A binomial distribution can be observed.

6.2 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

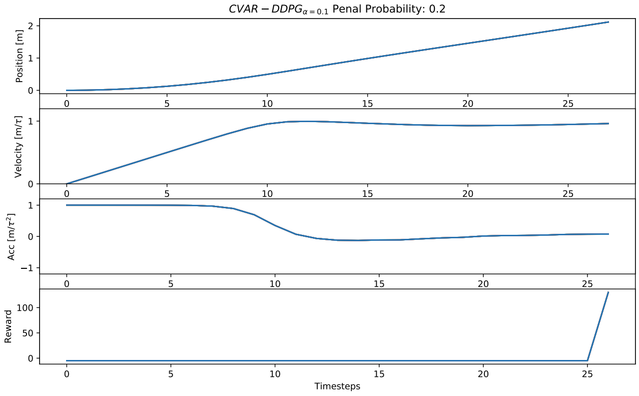

Figure 6.5: Car trajectory using CVAR-DDPG algorithm and velocity penalization with probability 0.2 and (α-CVAR $= 0.2$)

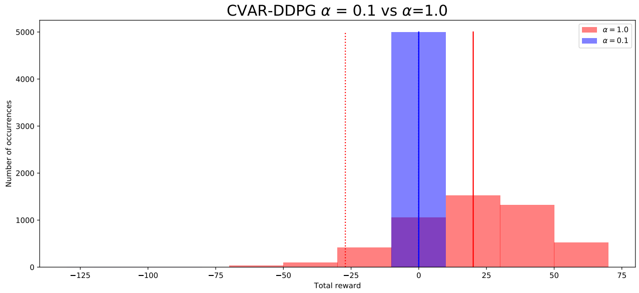

Figure 6.6: Comparison of cumulative rewards achieved with CVAR-DDPG algorithms with α= 0.2 and α= 1 when the probability of velocity penalization = 0.2. Algorithm with α= 1 achieves a higher expected value ($\mu = 20.11$) but has a lower CVAR (CVaR$_{\alpha=0.1}$=-27.12 compared to the algorithm with α= 0.1, which has $\mu = 0$ and CVaR$_{\alpha=0.1}$=0.0 5000 episodes were ran after training each algorithm.

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).

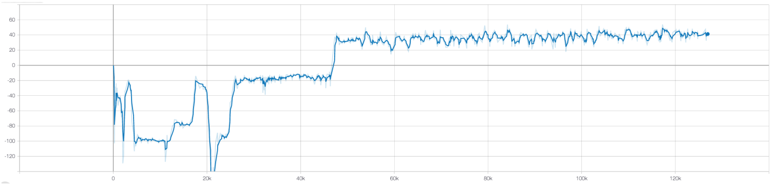

Figure 6.7: Evolution of the sampled mean of the tail of the Cumulative Value Distribution during training epochs

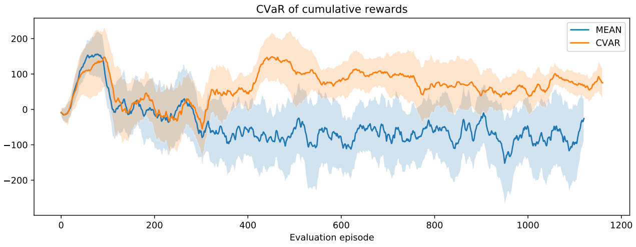

Figure 6.8: Evolution during training of CVaR ($\alpha = 0.1$) of the cumulative rewards over 5 evaluation episodes

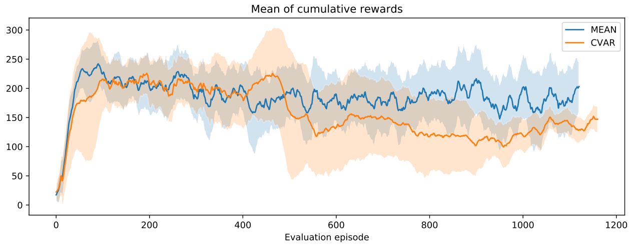

Figure 6.9: Evolution during training of mean of the cumulative rewards over 5 evaluation episodes

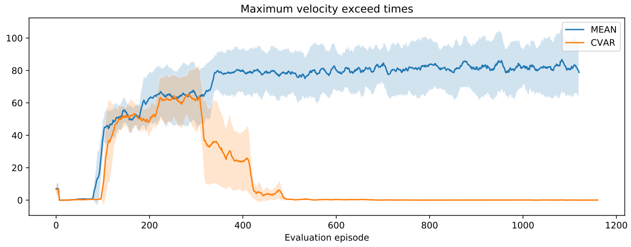

Figure 6.10: Evolution during training of mean of times of maximum velocity exceed over 5 evaluation episodes

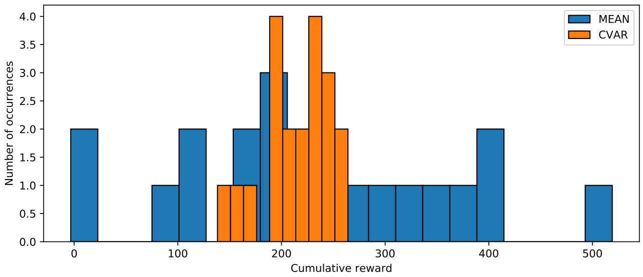

Figure 6.11: Histogram of cumulative rewards during 200 time steps using the trained final policies

Bibliography

J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," 2015.

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

D. Bertsekas, *Dynamic Programming and Optimal Control*, 1995, vol. 1.

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.

M. Heger, "Consideration of Risk in Reinforcement Learning," in *Machine Learning Proceedings 1994*, 1994.

S. P. Coraluppi and S. I. Marcus, "Mixed risk-neutral/minimax control of discrete-time, finite-state Markov decision processes," *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 528–532, 2000.

——, "Risk-sensitive and minimax control of discrete-time, finite-state Markov decision processes," *Automatica*, vol. 35, no. 2, pp. 301–309, 1999.

E. Altman, "Asymptotic properties of constrained Markov Decision Processes," *ZOR - Methods and Models of Operations Research*, 1993.

P. Geibel, "Reinforcement Learning for MDPs with Constraints," in *Machine Learning: ECML 2006*, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 646–653.

A. Tamar, D. Di Castro, and S. Mannor, "Policy gradients with variance related risk criteria," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 2012.

R. A. Howard and J. E. Matheson, "Risk-Sensitive Markov Decision Processes," *Management Science*, vol. 18, no. 7, pp. 356–369, 1972.

K. J. Chung and M. J. Sobel, "DISCOUNTED MDP'S: DISTRIBUTION FUNCTIONS AND EXPONENTIAL UTILITY MAXIMIZATION." *SIAM Journal on Control and Optimization*, 1987.

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

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

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

M. Petrik and D. Subramanian, “An approximate solution method for large risk-averse markov decision processes,” in *Uncertainty in Artificial Intelligence - Proceedings of the 28th Conference, UAI 2012*, 2012.

T. Morimura, M. Sugiyama, H. Kashima, H. Hachiya, and T. Tanaka, “Parametric return density estimation for Reinforcement Learning,” in *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, UAI 2010*, 2010.

——, “Nonparametric return distribution approximation for reinforcement learning,” in *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, 2010.

A. Tamar, Y. Glassner, and S. Mannor, “Optimizing the CVaR via sampling,” *Proceedings of the National Conference on Artificial Intelligence*, vol. 4, pp. 2993–2999, 2015.

Y. Chow and M. Ghavamzadeh, “Algorithms for CVaR optimization in MDPs,” *Advances in Neural Information Processing Systems*, vol. 4, no. January, pp. 3509–3517, 2014.

A. Tamar, Y. Chow, M. Ghavamzadeh, and S. Mannor, “Policy gradient for coherent risk measures,” *Advances in Neural Information Processing Systems*, vol. 2015-Janua, pp. 1468–1476, 2015.

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.

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

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.

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.

C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, 1992.

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.

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.

R. Koenker, *Quantile Regression*, ser. Econometric Society Monographs. Cambridge University Press, 2005.

J. Dhaene, A. Kukush, D. Linders, and Q. Tang, “Remarks on quantiles and distortion risk measures,” *European Actuarial Journal*, 2012.

S. Fujimoto, D. Meger, and D. Precup, “Off-policy deep reinforcement learning without exploration,” in *36th International Conference on Machine Learning, ICML 2019*, 2019.

A. Kumar, J. Fu, G. Tucker, and S. Levine, “Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction,” no. NeurIPS, 2019.

D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

Appendix A

Anything

Bla bla …