Risk-averse Distributional Reinforcement Learning

A CVaR optimization approach

Silvestr Stanko[1]

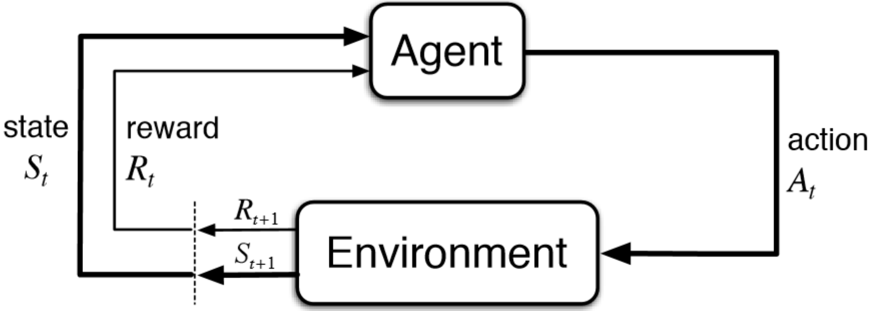
[1]Department of Computer Science
Czech Technical University

Sunday 3rd June, 2018

Outline

1. Introduction
   - Motivation
   - Reinforcement Learning
   - Risk
   - Risk-averse Reinforcement Learning
2. CVaR Value Iteration
   - Previous results
   - Linear-time improvement
3. CVaR Q-learning
   - Var-based policy improvement
4. Deep CVaR Q-learning

Reinforcement Learning



Reinforcement Learning - Goal

Definition

$Z^\pi(x_t)$ Is a random variable representing the discounted reward along a trajectory generated by the MDP by following the policy π, starting at state x_t.

$$Z^\pi(x_t) = \sum_{t=0}^{\infty} \gamma^t R(x_t, \pi(x_t))$$

Reinforcement Learning goals

Our goal is to find a globally optimal policy π^*

$$\pi^* = \arg\max_\pi \mathbb{E}\, Z^\pi(x_0)$$

Potential problems

Problems

1. Solutions should be **robust** to small environment changes
2. Solutions should avoid catastrophic events and be **safe**

Solution

Instead of maximizing the expected return, focus on other criteria that take into account the **risk** of the potential reward.

Motivation

Figure: Simulation vs Real world

Figure: Critical Applications

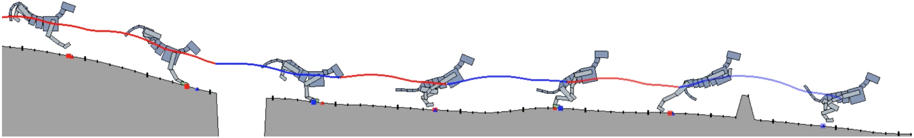


Figure: AI safety

Risk-averse Reinforcement Learning - example

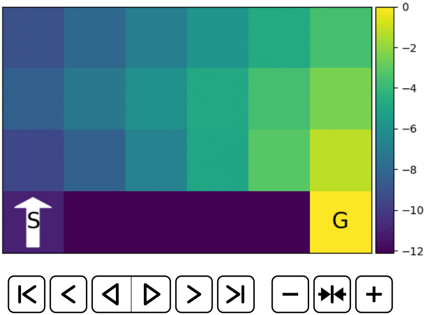


Figure: Greedy agent

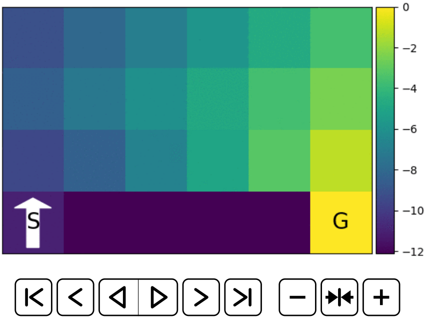
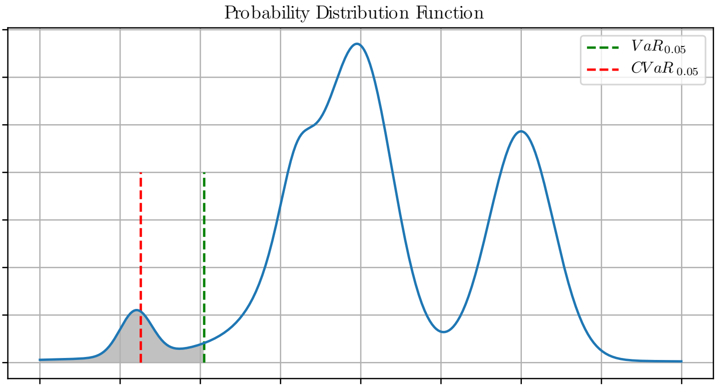


Figure: Risk-averse agent

Value-at-Risk, Conditional Value-at-Risk



Risk-averse Reinforcement Learning - goals

Definition

$Z^\pi(x_t)$ Is a random variable representing the discounted reward along a trajectory generated by the MDP by following the policy π, starting at state x_t.

$$Z^\pi(x_t) = \sum_{t=0}^{\infty} \gamma^t R(x_t, \pi(x_t))$$

Reinforcement Learning with CVaR

For a given α, our goal is to find a globally optimal policy π^*

$$\pi^* = \arg\max_{\pi} \mathrm{CVaR}_\alpha(Z^\pi(x_0))$$

Original Contributions

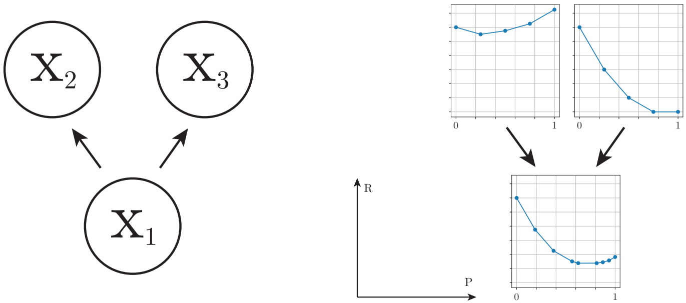
1. **Faster CVaR Value Iteration**
    - Polynomial $\rightarrow$ linear time.
    - Formally proved for increasing, unbounded distributions.
    - Experimentally verified for general distributions.
2. **CVaR Q-learning**
    - Sampling version of CVaR Value Iteration.
    - Based on the distributional approach.
    - Experimentally verified.
3. **Distributional Policy improvement**
    - Proved monotonic improvement for distributional RL.
    - Used as a heuristic for extracting π^* from CVaR Q-learning.
4. **Deep CVaR Q-learning**
    - TD update $\rightarrow$ loss function.
    - Experimentally verified in a deep learning context.

CVaR Value Iteration

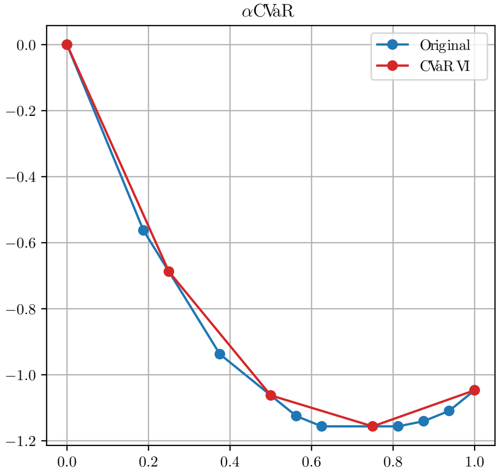
Theorem (CVaR Value Iteration)

The following Bellman operator is a contraction:

$$\mathbf{T}C(x,y) = \max_a \left[R(x,a) + \gamma \min_\xi \sum_{x'} p(x'|x,a)\xi(x')C\left(x', y\xi(x')\right) \right]$$



Linear interpolation



CVaR VI computational complexity

Problem

- CVaR Value Iteration requires computing a Linear Program for each state and atom.
- LP computation is slow

Solution

CVaR Value Iteration with quantile representation.

αCVaR$_\alpha$ describes a quantile function

Lemma

Any discrete distribution has a piece-wise linear $\alpha CVaR_\alpha$ function. Similarly, any a piece-wise linear $\alpha CVaR_\alpha$ function can be seen as representing a certain discrete distribution.

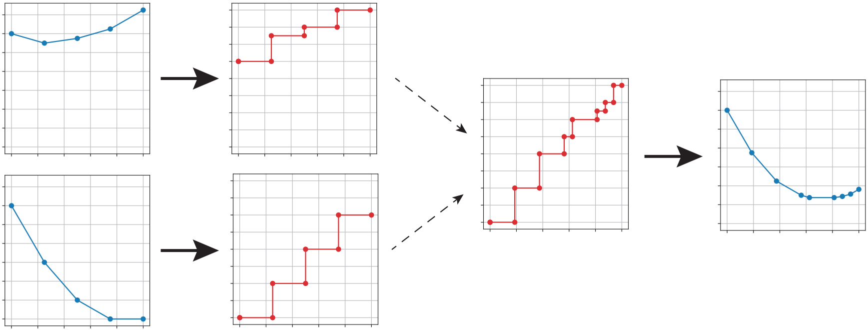
αCVaR$_\alpha$ $\Rightarrow$ VaR

$$\frac{\partial}{\partial\alpha}\alpha\mathrm{CVaR}_\alpha(Z) = \frac{\partial}{\partial\alpha}\int_0^\alpha VaR_\beta(Z)d\beta = VaR_\alpha(Z)$$

αCVaR$_\alpha$ $\Leftarrow$ VaR

$$\alpha\mathrm{CVaR}_\alpha(Z) = \int_0^\alpha VaR_\beta(Z)\mathrm{d}\beta$$

Next state CVaR computation



Linear-time Computation

Theorem

Solution to minimization problem

$$\min_{\xi \in \mathcal{U}_{CVaR}(\alpha, p(\cdot|x,a))} \sum_{x'} p(x'|x,a)\xi(x') CVaR_{\xi(x')\alpha}\left(Z^{\pi}(x')\right)$$

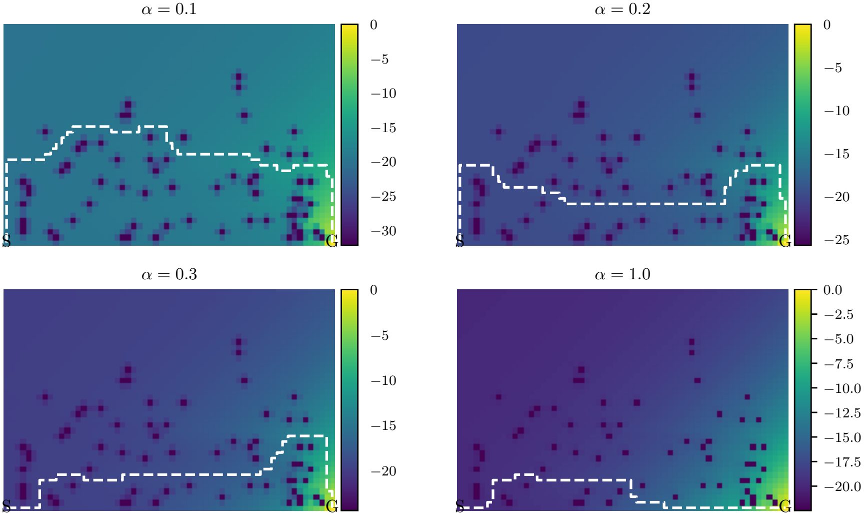
can be computed by setting

$$\xi(x') = \frac{F_{Z(x')}(F^{-1}_{Z(x,a)}(\alpha))}{\alpha}$$

The computational complexity is $O(n \cdot m)$ where n is the number of transition states and m is the number of atoms.

- Proved for increasing unbounded distributions

CVaR Value Iteration - Experiments



Q-learning

Problem

- In practice, we often don't have access to the transition probabilities $p(x'|x, a)$
- We need to learn through direct interaction with the environment.

Solution

Q-learning: Sampling version of Value Iteration.

CVaR estimation

Requirements:

- Store a single value
- Expected value of updates is CVaR

Recursive CVaR Estimation

$$V_{t+1} = V_t + \beta_t \left[1 - \frac{1}{\alpha} \mathbb{1}_{(V_t \geq r)} \right]$$

$$C_{t+1} = (1 - \beta_t) C_t + \beta_t \left[V_t + \frac{1}{\alpha} (r - V_t)^{-} \right]$$

CVaR Q-learning

Pseudocode

1. Sample a transition x, a, x', r
2. Create a target distribution $\mathbf{d}$
3. Sample from the target distribution
4. Update current estimates of VaR and CVaR towards the sample

Improvement

- ~~Sample from target distribution~~
- Update proportionally to the target distribution

VaR-based Policy Improvement

Theorem

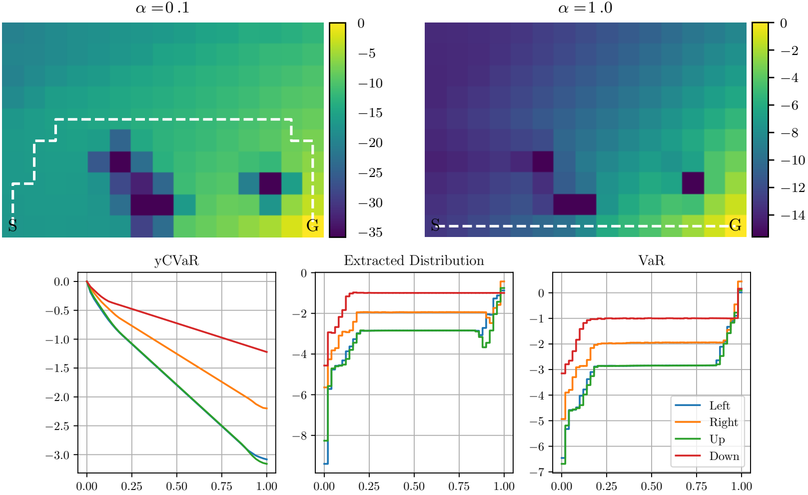
Let π be a fixed policy, $\alpha \in (0, 1]$. By following policy π' from the following algorithm, we will improve $CVaR_\alpha(Z)$ in expectation:

$$CVaR_\alpha(Z^\pi) \leq CVaR_\alpha(Z^{\pi'})$$

VaR-based Policy Improvement

$a = \arg\max_a \mathrm{CVaR}_\alpha(Z(x_0, a))$
$s = \mathrm{VaR}_\alpha(Z(x_0, a))$
Take action a, observe x, r
while x is not terminal **do**
  $s = \dfrac{s - r}{\gamma}$
  $a = \arg\max_a \mathbb{E}\left[(Z(x, a) - s)^-\right]$
  Take action a, observe x, r
end while

CVaR Q-learning - Experiments



Approximate Q-learning

Problem

Q-learning is intractable for large state spaces.

Solution

Use approximate Q-learning.

- Formulate CVaR Q-learning update as a minimizing argument
- Use methods of convex optimization to find the optimal point

TD update → loss function

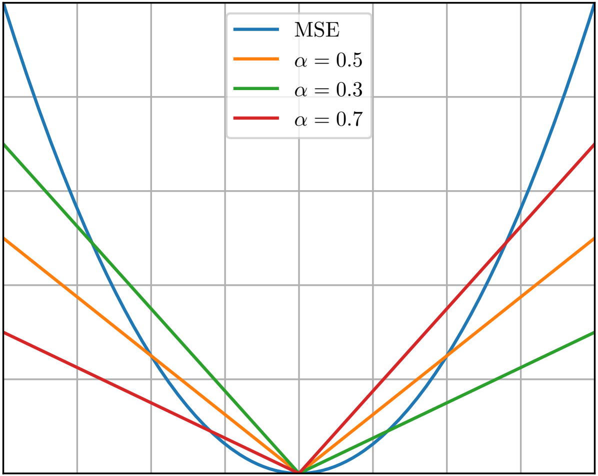
Standard RL

$$Q(x,a) = (1-\beta)Q(x,a) + \beta \mathcal{T}Q(x,a)$$
$$\downarrow$$
$$\min_{\theta} \mathbb{E}\left[\left(Q_\theta(x,a) - \mathcal{T}Q(x,a)\right)^2\right]$$

CVaR RL

$$V(x,a,y_i) = V(x,a,y_i) + \beta \mathop{\mathbb{E}}_{j}\left[1 - \frac{1}{y_i}\mathbb{1}_{(V(x,a,y_i)\geq \mathcal{T}d_j)}\right]$$
$$C(x,a,y_i) = (1-\beta)C(x,a,y_i) + \beta \mathop{\mathbb{E}}_{j}\left[V(x,a,y_i) + \frac{1}{y_i}\left(r + \gamma d_j - V(x,a,y_i)\right)^{-}\right]$$
$$\downarrow$$
???

Quantile loss



TD update → loss function

VaR loss

$$\mathcal{L}_{\text{VaR}} = \sum_{i=1}^{N} \mathbb{E}_j \left[(r + \gamma d_j - V_i(x, a))(y_j - \mathbb{1}_{(V_i(x,a) \geq r + \gamma d_j)}) \right]$$

CVaR loss

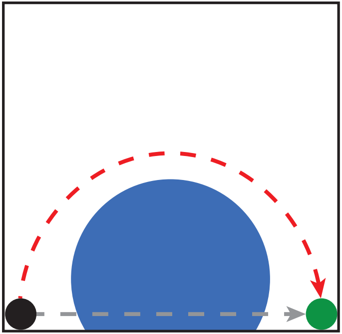
$$\mathcal{L}_{\text{CVaR}} = \sum_{i=1}^{N} \mathbb{E}_j \left[\left(V_i(x, a) + \frac{1}{y_i} (r + \gamma d_j - V_i(x, a))^{-} - C_i(x, a) \right)^2 \right]$$

$$\mathcal{L} = \mathbb{E} \left[\mathcal{L}_{\text{VaR}} + \mathcal{L}_{\text{CVaR}} \right]$$

Deep CVaR Q-learning

- Model: Convolutional Neural Network
  1. Input: $84 \times 84 \times 4$
  2. Convolution: $8 \times 8 \times 32$ (stride 4)
  3. Convolution: $4 \times 4 \times 64$ (stride 2)
  4. Convolution: $3 \times 3 \times 32$ (stride 1)
  5. Fully connected: 256 hidden units
  6. Output: $|\mathcal{A}| \times 100$
- Replay Memory
- Target network C'
- Optimizer: Adam (Stochastic Gradient Descent)

Deep CVaR Q-learning - Experiments



1. Video: $\alpha = 1$
2. Video: $\alpha = 0.3$

Summary

1. **Faster CVaR Value Iteration**
    - Polynomial $\rightarrow$ linear time.
    - Formally proved for increasing, unbounded distributions.
    - Experimentally verified for general distributions.
2. **CVaR Q-learning**
    - Sampling version of CVaR Value Iteration.
    - Based on the distributional approach.
    - Experimentally verified.
3. **Distributional Policy improvement**
    - Proved monotonic improvement for distributional RL.
    - Used as a heuristic for extracting π^* from CVaR Q-learning.
4. **Deep CVaR Q-learning**
    - TD update $\rightarrow$ loss function.
    - Experimentally verified in a deep learning context.