Risk-averse Distributional Reinforcement Learning

A CVaR optimization approach

Silvestr Stanko[1]

[1]Department of Computer Science
Czech Technical University

Saturday 26th May, 2018

Outline

1. Introduction
   - 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

Outline

1. Introduction
   - 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

Motivation

Figure: Robotics

Figure: Finance

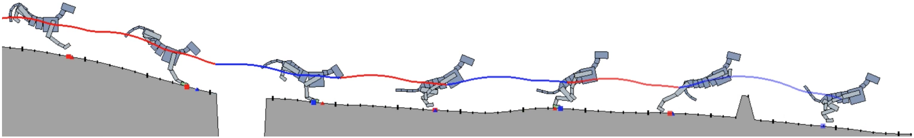


Figure: AI safety

Ultimate goals of AI

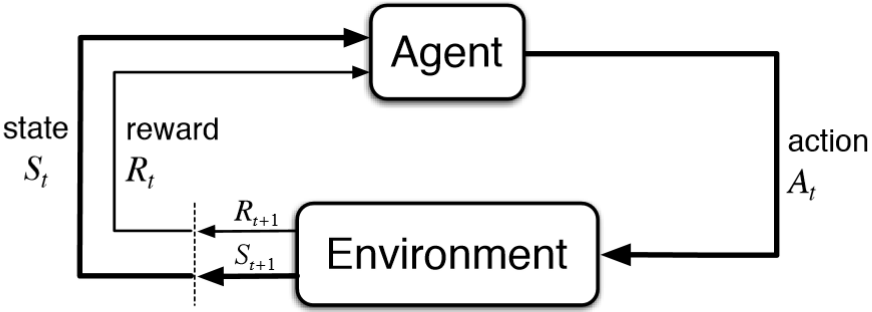
General AI

- Learning from experience
- Learning *tabula rasa*
- Beyond purpose-specific AI
- Beyond human-level performance

Safe AI

- Avoiding catastrophic events
- Robust to environment changes or adversaries

Reinforcement Learning



Recent successes

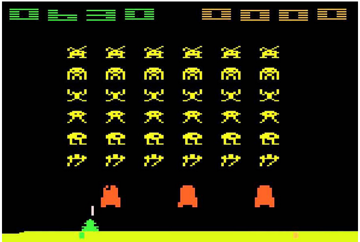


Figure: Atari games

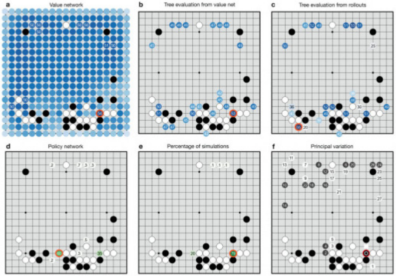


Figure: Go

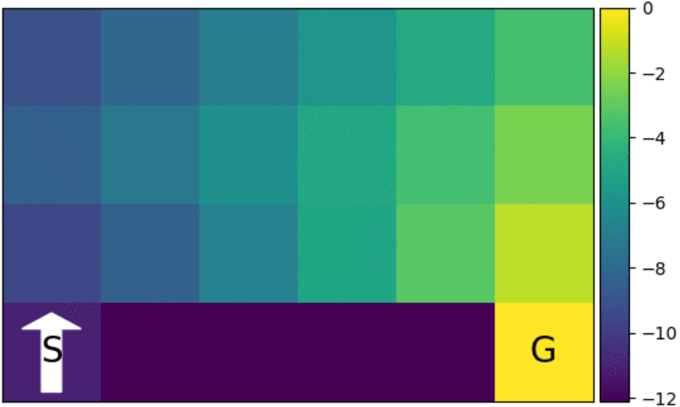
Markov Decision Processes

Definition

An MDP is a 5-tuple $\mathcal{M} = (\mathcal{X}, \mathcal{A}, R, P, \gamma)$, where

- $\mathcal{X}$ is the state space
- $\mathcal{A}$ is the action space
- $R(x, a)$ is a random variable representing the reward generated by being in state x and selecting action a
- $P(\cdot|x, a)$ is the transition probability distribution
- $\gamma \in [0, 1)$ is a discount factor

Markov Decision Process - example



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^*

$$\pi^* = \arg\max_\pi \exp Z^\pi(x_0)$$

Potential problems

Problems

- Solutions must avoid catastrophic events and be **safe**
- RL is sample inefficient $\rightarrow$ expensive training
- Solutions must be **robust** to small model changes

Solution

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

Risk

Definition

Risk is the potential of gaining or losing something of value.

Risk-averse: disinclined or reluctant to take risks

Risk-neutral: indifferent to or balanced with respect to risk.

Risk-seeking: inclined or eager to take risks

Example

Choose between recieving:

1. \$100 in 100% cases
2. \$200 in 50% cases and \$0 in 50% cases
3. \$10,000 in 1% cases and \$0 in 99% cases

Measuring Risk

Value-at-Risk (VaR)

- Easy to understand
- Historically the most used risk-measure
- Undesirable computational properties
- Does not differentiate between large and catastrophic losses

Definition

Let Z be a random variable representing reward, with cumulative distribution function $F(z) = \mathbb{P}(Z \leq z)$. The Value-at-Risk at confidence level $\alpha \in (0, 1)$ is the α-quantile of Z, i.e.

$$\text{VaR}_\alpha(Z) = F^{-1}(\alpha) = \inf \{z | \alpha \leq F(z)\}$$

Measuring Risk

Conditional Value-at-Risk (CVaR)

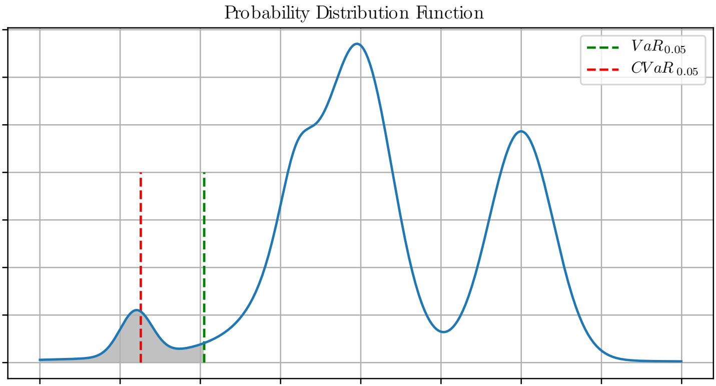
- Good computational properties
- Basel Committee on Banking Supervision: VaR $\rightarrow$ CVaR
- Equivalent to robustness

Definition

The Conditional Value-at-Risk (CVaR) at confidence level $\alpha \in (0, 1)$ is defined as the expected reward of of outcomes worse than the α-quantile (VaR_α):

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

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



Conditional Value-at-Risk as an optimal point

Definition

$$\mathrm{CVaR}_\alpha(Z) = \max_s \left\{ \frac{1}{\alpha} \mathbb{E}\left[(Z-s)^-\right] + s \right\}$$

where $(x)^- = \min(x, 0)$ and in the optimal point it holds that $s^* = VaR_\alpha(Z)$

$$\mathrm{CVaR}_\alpha(Z) = \frac{1}{\alpha} \mathbb{E}\left[(Z - VaR_\alpha(Z))^-\right] + VaR_\alpha(Z)$$

it's dual is

$$\mathrm{CVaR}_\alpha(Z) = \min_{\xi \in \mathcal{U}_{\mathrm{CVaR}}(\alpha, p(\cdot))} \mathbb{E}_\xi[Z]$$

$$\mathcal{U}_{\mathrm{CVaR}}(\alpha, p(\cdot)) = \left\{ \xi : \xi(z) \in \left[0, \frac{1}{\alpha}\right], \int \xi(z) p(z) \mathrm{d}z = 1 \right\}$$

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^*

$$\pi^* = \arg\max_\pi CVaR_\alpha^\pi(Z^\pi(x_0))$$

Risk-averse Reinforcement Learning - example

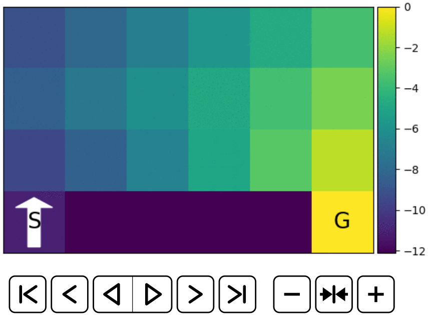


Figure: Greedy agent

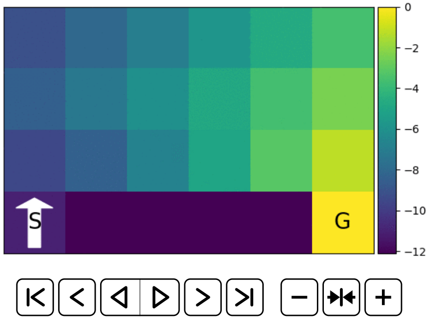


Figure: Risk-averse agent

Outline

Value Iteration

Definition

Value function $V(x)$ represents the expected return when starting in state x and following the optimal policy π^* thereafter.

Value Iteration

Initialize $V_0(x)$ for each state (arbitrary value, e.g. 0).
Update each state:

$$V_{k+1}(x) = \max_a \left[R(x,a) + \gamma \sum_{x'} p(x'|x,a) V_k(x') \right]$$

Repeat.

The algorithm converges to the optimal policy π^*: $\lim_{k\to\infty} V_k(x) = V(x)$

Value Iteration with CVaR

Theorem (CVaR decomposition)

The conditional CVaR under policy π obeys the following decomposition:

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

TODO: pic of mdp with cvars

CVaR Value Iteration

Theorem (CVaR Value Iteration)

The following Bellman operator is a contraction:

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

The operator $\mathbf{T}$ describes the following relationship:

$$\mathbf{T}CVaR_y(Z(x)) = \max_a \left[R(x,a) + \gamma CVaR_y(Z(x')) \right]$$
$$x' \sim p(\cdot|x,a)$$

Linear interpolation

Computing operator $\mathbf{T}$ s intractable, as the state-space is continuous. A solution would be to approximate the operator with linear interpolation.

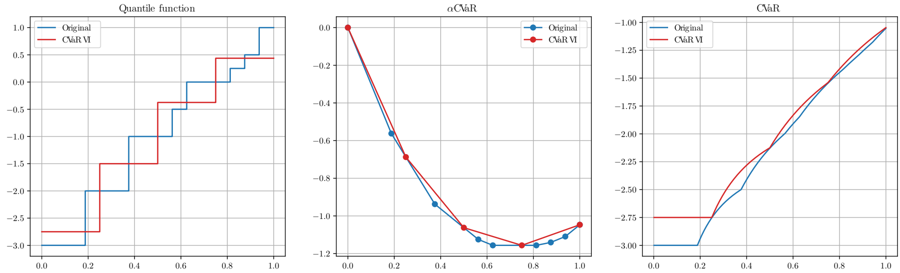
Theorem

The function $\alpha CVaR_\alpha$ is convex. The operator $\mathbf{T}_\mathcal{I}$ is a contraction.

$$\mathcal{I}_x[C](y) = y_i C(x, y_i) + \frac{y_{i+1} C(x, y_{i+1}) - y_i C(x, y_i)}{y_{i+1} - y_i}(y - y_i)$$

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

This iteration can be formulated and solved as a linear program.


Quantile function
Original
CVaR VI
αCVaR
Original
CVaR VI
CVaR
Original
CVaR VI

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.

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.

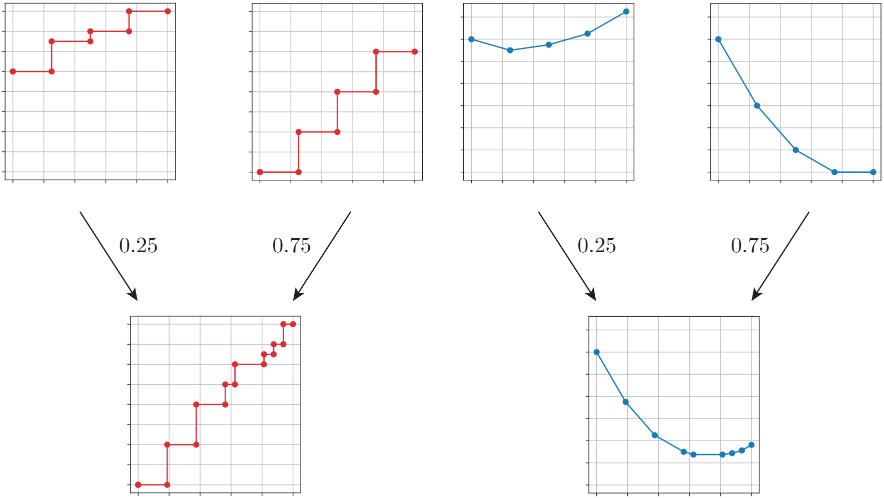
αCVaR$_\alpha$ $\Rightarrow$ VaR

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

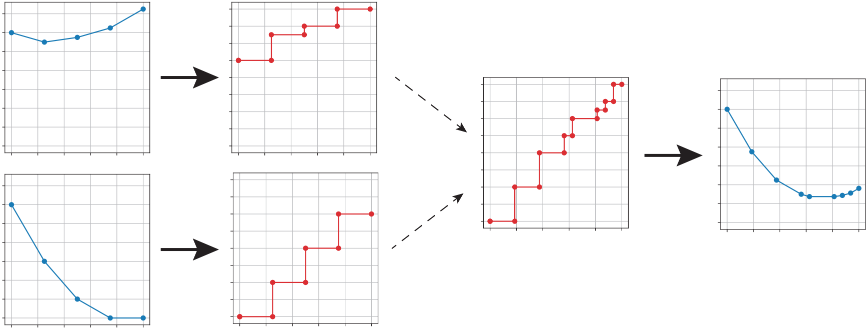
αCVaR$_\alpha$ $\Leftarrow$ VaR

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

Next state CVaR computation



Next state CVaR computation



Linear-time Computation

Theorem

Solution to minimization problem

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

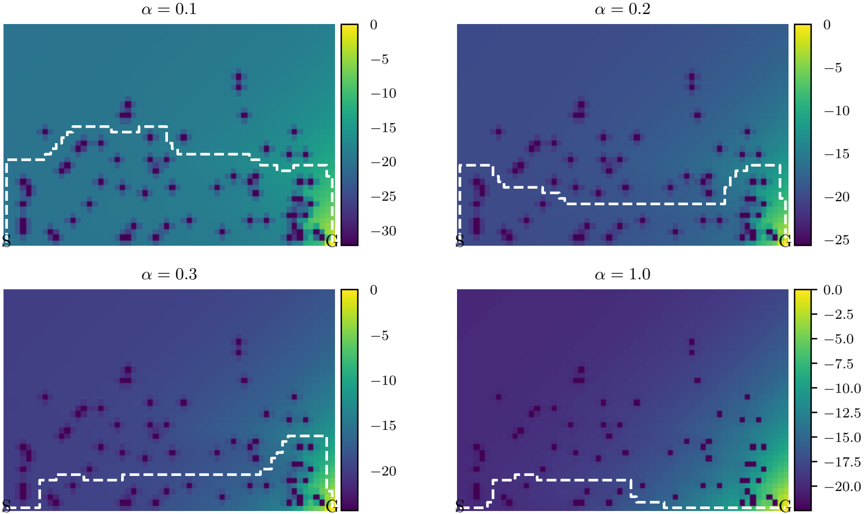
can be computed by setting

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

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



Outline

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.

Q-learning

Value Iteration

$$Q(x,a) \leftarrow \mathcal{T}Q(x,a)$$

$$Q(x,a) \leftarrow R(x,a) + \gamma \sum_{x'} p(x'|x,a) \max_{a'} Q(x',a')$$

Q-learning

$$Q(x,a) \leftarrow (1-\beta)Q(x,a) + \beta\mathcal{T}Q(x,a)$$

$$Q(x,a) \leftarrow (1-\beta)Q(x,a) + \beta \left[R(x,a) + \gamma \max_{a'} Q(x',a') \right]$$

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

CVaR Q-learning

CVaR Q-learning: Uniform case

input: x, a, x', r
for each i **do**
  $C(x', y_i) = \max_{a'} C(x', a', y_i)$
end for
$\mathbf{d} = \text{extractDistribution}\left(C(x', \bullet), \mathbf{y}\right)$
for each i, j **do**
  $V(x, a, y_i) = V(x, a, y_i) + \beta \left[1 - \frac{1}{y_i} \mathbb{1}_{(V(x,a,y_i) \geq r + \gamma d_j)}\right]$
  $C(x, a, y_i) =$
  $(1-\beta) C(x, a, y_i) + \beta \left[V(x, a, y_i) + \frac{1}{y_i} \left(r + \gamma d_j - V(x, a, y_i)\right)^{-}\right]$
end for

CVaR Q-learning

CVaR Q-learning: General case

input: x, a, x', r

for each i **do**

$\quad C(x', y_i) = \max_{a'} C(x', a', y_i)$

end for

$\mathbf{d} = \text{extractDistribution}\left(C(x', \bullet), \mathbf{y}\right)$

for each i **do**

$\quad V(x, a, y_i) = V(x, a, y_i) + \beta\, \mathbb{E}_j \left[1 - \frac{1}{y_i} \mathbb{1}_{(V(x,a,y_i) \geq r + \gamma d_j)}\right]$

$\quad C(x, a, y_i) =$

$\quad (1 - \beta) C(x, a, y_i) + \beta\, \mathbb{E}_j \left[V(x, a, y_i) + \frac{1}{y_i}\left(r + \gamma d_j - V(x, a, y_i)\right)^{-}\right]$

end for

Optimal policy extraction

Standard RL: Optimal policy

$$\pi^*(x) = \arg\max_a Q(x, a)$$

CVaR VI: Optimal policy

$$\begin{aligned}
\pi^*(x_0) &= \arg\max_a C(x, a, \alpha) \\
\pi^*(x_1) &= \arg\max_a C(x_1, a, \alpha\xi^*(x_0)) \\
&\vdots \\
\pi^*(x_t) &= \arg\max_a C(x_t, a, y_{t-1}\xi^*(x_{t-1}))
\end{aligned}$$

Optimal policy extraction

Problem

- To compute y_t, we would need to know ξ^*.
- To compute ξ^*, we would need to know the probability of transitions $p(x'|x, a)$

Solution

- Use transition reward to compute the next-state y

Distributional Policy Improvement

CVaR primal definition:

$$\mathrm{CVaR}_\alpha(Z) = \max_s \left\{ \frac{1}{\alpha} \mathbb{E}\left[(Z - s)^- \right] + s \right\}$$

Our goal can then be rewritten as

$$\max_\pi \mathrm{CVaR}_\alpha(Z^\pi) = \max_\pi \max_s \frac{1}{\alpha} \mathbb{E}\left[(Z^\pi - s)^- \right] + s$$

Recall: The primal solution is equivalent to $\mathrm{VaR}_\alpha(Z)$
Idea: If we knew the value $s^* = \mathrm{VaR}_\alpha(Z)$ in advance, we could simplify the problem to maximize only

$$\max_\pi \mathrm{CVaR}_\alpha(Z^\pi) = \max_\pi \frac{1}{\alpha} \mathbb{E}\left[(Z^\pi - s^*)^- \right] + s^* \tag{1}$$

Distributional Policy Improvement

Given that we have access to the return distributions, we can improve the policy by simply choosing an action that maximizes CVaR_α in the first state $a_0 = \arg\max_\pi \mathrm{CVaR}_\alpha(Z^\pi(x_0))$, setting $s^* = \mathrm{VaR}_\alpha(Z(x_0, a_0))$ and focus on maximization of the simpler criterion.

This can be seen as coordinate ascent with the following phases:

1. Maximize $\frac{1}{\alpha}\mathbb{E}\left[(Z^\pi(x_0) - s)^-\right] + s$ w.r.t. s while keeping π fixed. This is equivalent to computing CVaR according to the primal.
2. Maximize $\frac{1}{\alpha}\mathbb{E}\left[(Z^\pi(x_0) - s)^-\right] + s$ w.r.t. π while keeping s fixed. This is the policy improvement step.
3. Recompute $\mathrm{CVaR}_\alpha(Z^{\pi^*})$ where π^* is the new policy.

TODO: simpler slide

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

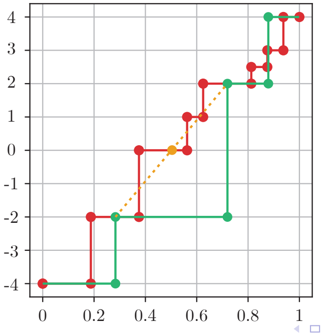
VaR-based heuristic

Problem

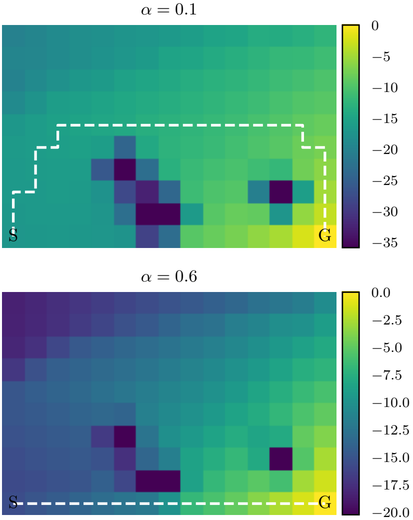
When using quantile discretization, we don't have access to the exact VaR.

Solution

Use linear interpolation as a heuristic.



CVaR Q-learning - Experiments



Outline

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

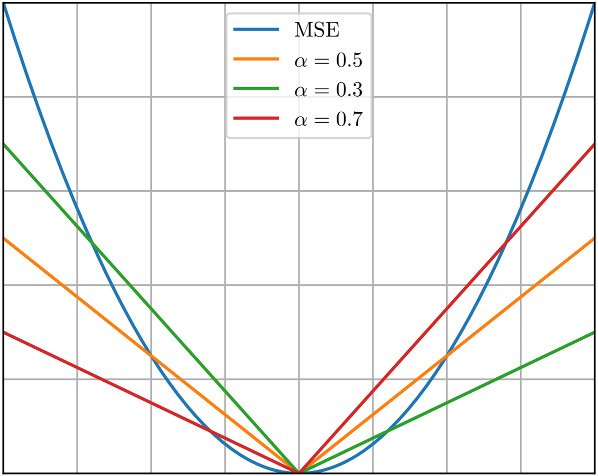
$$Q(x,a) = (1-\beta)Q(x,a) + \beta \mathcal{T} Q(x,a)$$
$$\downarrow$$
$$\min_\theta \mathbb{E}\left[\left(Q_\theta(x,a) - \mathcal{T} Q(x,a)\right)^2\right]$$

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

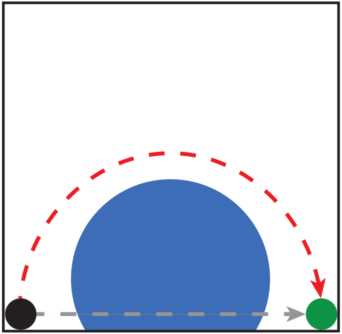
$$\mathcal{L}_{\text{CVaR}} = \sum_{i=1}^{N} \mathbb{E}_{j}\left[\left(V_i(x,a) + \frac{1}{y_i}(r + \gamma d_j - V_i(x,a))^{-} - C_i(x,a)\right)^2\right]$$

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

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.

Future work

Theory

- CVaR Value Iteration - General equivalence proof
- Q-learning - convergence proof

Practice

- Larger state spaces
- Practical problems (e.g. finance)