Risk-Averse Distributional Reinforcement Learning: Bonus Materials

May 2019

1 Proofs

Proof of Lemma 1

Proof. The fact that discrete distributions have a piecewise linear $y\mathrm{CVaR}_y$ function has already been shown by [Rockafellar and Uryasev, 2000].
According to definition (2) we have

$$y\mathrm{CVaR}_y(Z) = y\frac{1}{y}\int_0^y \mathrm{VaR}_\beta(Z)\mathrm{d}\beta = \int_0^y \mathrm{VaR}_\beta(Z)\mathrm{d}\beta$$

by taking the y derivative, we have

$$\frac{\partial}{\partial y} y\mathrm{CVaR}_y(Z) = \frac{\partial}{\partial y}\int_0^y \mathrm{VaR}_\beta(Z)d\beta = \mathrm{VaR}_y(Z)$$

□

Proof of Theorem 1

Proof. Since we are interested in the minimal argument, we can ease the computation by focusing on the $\alpha\mathrm{CVaR}_\alpha$ function instead of CVaR_α. When working with two states, the equation of interest simplifies to

$$\begin{aligned}
\alpha\mathrm{CVaR}_\alpha(Z(x,a)) = \min_\xi\ & p_1\xi_1\alpha\mathrm{CVaR}_{\xi_1\alpha}\left(Z(x_1')\right) + p_2\xi_2\alpha\mathrm{CVaR}_{\xi_2\alpha}\left(Z(x_2')\right) \\
\text{s.t.}\quad & p_1\xi_1 + p_2\xi_2 = 1 \\
& 0 \le \xi_1 \le \frac{1}{\alpha} \\
& 0 \le \xi_2 \le \frac{1}{\alpha}
\end{aligned}$$

therefore

$$\begin{aligned}
\alpha\mathrm{CVaR}_\alpha(Z(x,a)) &= \min_{\xi} p_1\xi_1\alpha\mathrm{CVaR}_{\xi_1\alpha}(Z(x_1')) + (1-p_1)\frac{1-p_1\xi_1}{1-p_1}\alpha\mathrm{CVaR}_{\frac{1-p_1\xi_1}{1-p_1}\alpha}(Z(x_2')) \\
&= \min_{\xi} p_1\int_0^{\xi_1\alpha}\mathrm{VaR}_\beta(Z(x_1'))\,\mathrm{d}\beta + (1-p_1)\int_0^{\frac{1-p_1\xi_1}{1-p_1}\alpha}\mathrm{VaR}_\beta(Z(x_2')\,\mathrm{d}\beta
\end{aligned}$$

To find the minimal argument, we find the first derivative w.r.t. ξ_1

$$\begin{aligned}
\frac{\partial\alpha\mathrm{CVaR}_\alpha}{\partial\xi_1} &= p_1\alpha\mathrm{VaR}_{\xi_1\alpha}(Z(x_1')) + (1-p_1)\alpha\frac{-p_1}{1-p_1}\mathrm{VaR}_{\frac{1-p_1\xi_1}{1-p_1}\alpha}(Z(x_2') \\
&= p_1\mathrm{VaR}_{\xi\alpha}(Z(x_1')) - p_1\mathrm{VaR}_{\frac{1-p\xi}{1-p}\alpha}(Z(x_2'))
\end{aligned}$$

By setting the derivative to 0 , we get

$$\mathrm{VaR}_{\xi_1\alpha}(Z(x_1')) \stackrel{!}{=} \mathrm{VaR}_{\frac{1-p\xi}{1-p}\alpha}(Z(x_2')) = \mathrm{VaR}_{\xi_2\alpha}(Z(x_2'))$$

[Bernard and Vanduffel, 2015] have shown that in the case of strictly increasing c.d.f. with unbounded support, it holds that

$$\begin{aligned}
\mathrm{VaR}_{\xi_1\alpha}(Z(x_1')) &= \mathrm{VaR}_{\xi_2\alpha}(Z(x_2')) \\
&= \mathrm{VaR}_\alpha(Z(x,a)) \\
F^{-1}_{Z(x_1')}(\xi_1\alpha) &= F^{-1}_{Z(x_2')}(\xi_2\alpha) \\
&= F^{-1}_{Z(x,a)}(\alpha)
\end{aligned}$$

and we can extract the values of $\xi_1\alpha, \xi_2\alpha$ using the

$$\begin{aligned}
F^{-1}_{Z(x_1')}(\xi_1\alpha) &= F^{-1}_{Z(x,a)}(\alpha) && /F_{Z(x_1')} \\
F_{Z(x_1')}\left(F^{-1}_{Z(x_1')}(\xi_1\alpha)\right) &= F_{Z(x_1')}\left(F^{-1}_{Z(x,a)}(\alpha)\right) \\
\xi_1\alpha &= F_{Z(x_1')}\left(F^{-1}_{Z(x,a)}(\alpha)\right)
\end{aligned}$$

And similarly for ξ_2.

Since the problem is convex, we have found the optimal point. □

Proof of Theorem 2

Proof. Let s^* be a solution to $\max_s \frac{1}{\alpha}\mathbb{E}\left[(Z^\pi(x_0)-s)^-\right]+s$. Then by optimizing $\frac{1}{\alpha}\mathbb{E}\left[(Z^\pi - s^*)^-\right]$ over π, we monotonously improve the optimization criterion $CVaR_\alpha(Z(x_0))$.

$$\begin{aligned}
\mathrm{CVaR}_\alpha(Z^\pi) &= \max_s \frac{1}{\alpha}\mathbb{E}\left[(Z^\pi - s)^-\right] + s && = \frac{1}{\alpha}\mathbb{E}\left[(Z^\pi - s^*)^-\right] + s^* \\
&\le \max_{\pi'} \frac{1}{\alpha}\mathbb{E}\left[(Z^{\pi'} - s^*)^-\right] + s^* && = \frac{1}{\alpha}\mathbb{E}\left[(Z^{\pi^*} - s^*)^-\right] + s^* \\
&\le \max_{s'} \frac{1}{\alpha}\mathbb{E}\left[(Z^{\pi^*} - s')^-\right] + s' && = CVaR_\alpha(Z^{\pi^*})
\end{aligned}$$

When optimizing w.r.t. π we can ignore the scaling term $\frac{1}{\alpha}$ and a constant term s^* without affecting the optimal argument. We can therefore focus on optimization of $\mathbb{E}\left[(Z^\pi(x_0) - s^*)^-\right]$.

$$
\begin{aligned}
\mathbb{E}\left[(Z_t - s)^-\right] &= \mathbb{E}\left[(Z_t - s)\mathbb{1}(Z_t \leq s)\right] = \mathbb{E}\left[(r_t + \gamma Z_{t+1} - s)\mathbb{1}(Z_{t+1} \leq \frac{s - r_t}{\gamma})\right] \\
&= \sum_{x_{t+1}, r_t} P(x_{t+1}, r_t \,|\, x_t, a)\mathbb{E}\left[(r_t + \gamma Z(x_{t+1}) - s)\mathbb{1}(Z(x_{t+1}) \leq \frac{s - r_t}{\gamma})\right] \\
&= \sum_{x_{t+1}, r_t} P(x_{t+1}, r_t \,|\, x_t, a)\mathbb{E}\left[\gamma\left(Z(x_{t+1}) - \frac{s - r_t}{\gamma}\right)\mathbb{1}(Z(x_{t+1}) \leq \frac{s - r_t}{\gamma})\right] \\
&= \gamma \sum_{x_{t+1}, r_t} P(x_{t+1}, r_t \,|\, x_t, a)\mathbb{E}\left[\left(Z(x_{t+1}) - \frac{s - r_t}{\gamma}\right)\mathbb{1}(Z(x_{t+1}) \leq \frac{s - r_t}{\gamma})\right] \\
&= \gamma \sum_{x_{t+1}, r_t} P(x_{t+1}, r_t \,|\, x_t, a)\mathbb{E}\left[\left(Z(x_{t+1}) - \frac{s - r_t}{\gamma}\right)^-\right]
\end{aligned}
\tag{1}
$$

where we used the definition of return $Z_t = R_t + \gamma Z_{t+1}$ and the fact that probability mixture expectations can be computed as $\mathbb{E}[f(Z)] = \sum_i p_i \mathbb{E}[f(Z_i)]$ for any function f.

Now let's say we sampled reward r_t and state x_{t+1}, we are still trying to find a policy π^* that maximizes

$$
\begin{aligned}
\pi^* &= \arg\max_\pi \mathbb{E}\left[(Z(x_t) - s)^- \,\middle|\, x_{t+1}, r_t\right] \\
&= \arg\max_\pi \mathbb{E}\left[\left(Z(x_{t+1}) - \frac{s - r_t}{\gamma}\right)^-\right]
\end{aligned}
\tag{2}
$$

Where we ignored the unsampled states, since these are not a function of x_{t+1}, and the multiplicative constant γ that will not affect the maximum argument.

At the starting state, we set $s = s^*$. At each following state we select an action according to equation (2). By induction we maximize the criterion (**??**) in each step. □

2 Algorithms

Algorithm 1 CVaR Computation via Quantile Representation

function extractDistribution
  input: vectors $\mathbf{C}, \mathbf{y}$
  # Note: $y_0 = C(x', y_0) = 0$
  for $i \in \{1, ..., |\mathbf{y}|\}$ **do**
    $d_i = \dfrac{C(x', y_i) - C(x', y_{i-1})}{y_i - y_{i-1}}$
  end for
  output vector $\mathbf{d}$

function extractC
  input: vectors $\mathbf{d}, \mathbf{p}$
  $C_0 = 0$
  for $i \in \{1, ..., |\mathbf{p}|\}$ **do**
    $C_i = C_{i-1} + d_i \cdot p_i$
  end for
  output vector $\mathbf{C}$

---

function mixDistributions
  input: tuples $(\mathbf{d^{(1)}}, p^{(1)}), ..., (\mathbf{d^{(K)}}, p^{(K)})$ and vector $\mathbf{y}$
  # $\sum_{k=1}^{K} p_k = 1$
  for $i, k \in \{1, ..., K\} \times \{1, ..., |\mathbf{y}|\}$ **do**
    # Weigh atom probabilities by transitions
    $p_i^{(k)} = p^{(k)} \cdot (y_i - y_{i-1})$
  end for
  # Join all tuples together:
  $atoms = \left\{(d_1^{(1)}, p_1^{(1)}), ..., (d_N^{(1)}, p_N^{(1)}), (d_1^{(2)}, p_1^{(2)}), ..., (d_N^{(K)}, p_N^{(K)})\right\}$
  Sort *atoms* by d
  Unwrap vectors $\mathbf{d}, \mathbf{p}$ from sorted tuples
  output $\mathbf{d}, \mathbf{p}$

Main
input: tuples $(C(x_i', \bullet), p^{(1)}), ..., (C(x_i', \bullet), p^{(K)})$ and vector $\mathbf{y}$
for $i \in \{1, ..., K\}$ **do**
  $\mathbf{d^{(i)}}$ = extractDistribution($C(x_i', \bullet), \mathbf{y}$)
end for
$\mathbf{d_{mix}}, \mathbf{y_{mix}} =$ mixDistributions($(\mathbf{d^{(1)}}, p^{(1)}), ..., (\mathbf{d^{(K)}}, p^{(K)}), \mathbf{y}$)
$\mathbf{C_{out}} =$ extractC($\mathbf{d_{mix}}, \mathbf{y_{mix}}$)
output: $\mathbf{C_{out}}$

Algorithm 2 CVaR Q-learning policy

input: α, converged V, C

$x = x_0$

$a = \arg\max_a C(x, a, \alpha)$

$s = V(x, a, y)$

while x is not terminal **do**

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

  $a = \arg\max_a \text{expMinInterp}(s, \mathbf{d}, V(x', a, \bullet), \mathbf{y})$

  Take action a, observe r, x'

  $s = \dfrac{s - r}{\gamma}$

  $x = x'$

end while

\# Compute $\mathbb{E}\left[(\mathbf{d}_a - s)^-\right]$ with linear interpolation

function expMinInterp

  input: s, vectors $\mathbf{d}, \mathbf{V}, \mathbf{y}$

  $z = 0$

  for $i \in \{1, \ldots, |\mathbf{y}|\}$ **do**

    if $S < V_i$ **then**

      break

    end if

    $z = z + d_i \cdot (y_i - y_{i-1})$

  end for

  $p_{\text{last}} = \dfrac{s - V_{i-1}}{V_i - V_{i-1}}(y_i - y_{i-1})$

  $z = z + d_i \cdot p_{\text{last}}$

  output z

Algorithm 3 Deep CVaR Q-learning with experience replay

Initialize replay memory M
Initialize the VaR function V with random weights θ_v
Initialize the CVaR function C with random weights θ_c
Initialize target CVaR function C' with weights $\theta_c' = \theta_c$
for each episode **do**
  $x = x_0$
  while x is not terminal **do**
    Choose a using a policy derived from C (ε-greedy)
    Take action a, observe r, x'
    Store transition (x, a, r, x') in M
    $x = x'$
    Sample random transitions (x_j, a_j, r_j, x_j') from M
    Build the loss function $\mathcal{L}_{\text{VaR}} + \mathcal{L}_{\text{CVaR}}$ (Algorithm **??**)
    Perform a gradient step on $\mathcal{L}_{\text{VaR}} + \mathcal{L}_{\text{CVaR}}$ w.r.t. θ_v, θ_c
    Every N_{target} steps set $\theta_c' = \theta_c$
  end while
end for

3 Figures

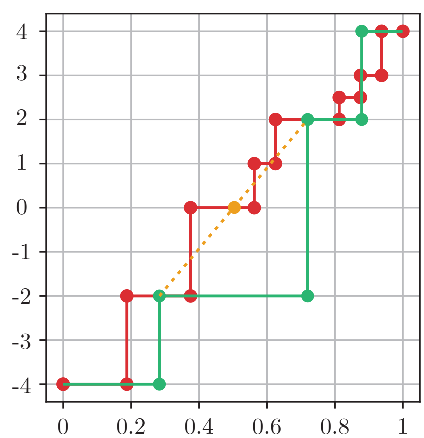

Figure 1: Visualization of the VaR-based heuristic. Quantile function of the exact distribution (unknown to the model) is shown in red and the VaR estimates at selected α-levels are shown in green. Let's say we now want to know y where $\text{VaR}_y = 0$. We use linear interpolation between the nearest known VaRs, shown in orange. In this case the interpolation estimate is $y = 0.5$.

References

Bernard, C. and Vanduffel, S. (2015). Quantile of a mixture with application to model risk assessment. *Dependence Modeling*, 3(1).

Rockafellar, R. T. and Uryasev, S. (2000). Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42.