1 Thought progression - 2nd semester

1.1 22.2.

Solve a simplified problem first: just estimate CVaR of some distribution - imagine a simple MDP with one action where u get random rewards. The estimation must be a memoryless process, like the exponential running average. The CVaR estimates must be an expectation of *something* over transition probabilities.
Solution idea: The CVaR estimation is similar to estimating std:

$$\mu^* = \arg\min \mathbb{E}(Z-\mu)^2]$$
$$\sigma^2 = \min \mathbb{E}[(Z-\mu)^2]$$
$$VaR = \arg\min \frac{1}{\alpha}\mathbb{E}((Z-s)^-)] + s$$
$$CVaR = \min \frac{1}{\alpha}\mathbb{E}((Z-s)^-)] + s$$

=> it will be necessary to remember both VaR and CVaR estimates. does it converge? is the estimation unbiased?
If this estimation procedure works: we can use the techniques used in distributional quantile regression: Take the next-state distribution, sum with reward and 'generate' samples (importance sampling will be necessary if not uniform). Use samples to shift the VaR estimates, use Var estimates and samples to shift CVaR estimates (more like yCVaR). Important: the VaR estimates are internal parameters, the samples are being generated from the yCVaR estimates by conversion.
Notes: can we be sure there doesn't exist a single parameter exp procedure that yields cvar? **investigate the cvar decomposition**

1.2 28.2.

There already exists a theory for recursively estimating CVaR based on stochastic approximation:

$$\mathrm{VaR}_\alpha = \mathrm{VaR}_\alpha - \beta\left[1 - \frac{1}{\alpha}\mathbb{1}_{(\mathrm{VaR}\le x)}\right]$$
$$\mathrm{CVaR}_\alpha = (1-\beta)\mathrm{CVaR}_\alpha + \beta\left[\mathrm{VaR}_\alpha + \frac{1}{\alpha}(x-\mathrm{VaR}_\alpha)^-\right]$$

rewrite var, examine convergence

Algorithm 1 CVaR Q-learning: big picture

Initialize $VaR, CVaR$, e.g. $\mathrm{VaR}_{y_i}(x,a) = 0, \mathrm{CVaR}_{y_i}(x,a) = 0$ for all x, a, y_i
while not converged **do**
  $x = x_0$
  $s = \mathrm{VaR}_\alpha(x, \arg\max_a \mathrm{CVaR}_\alpha(x,a))$
  while x is not terminal **do**
    $a = \varepsilon - \arg\max_a CVaRFromS(x,a,s)$
    $x', r = transition(x,a)$
    $UPDATE(x,a,x',r)$
    $s = \frac{r-s}{\gamma}$
    $x = x'$
  end while
end while

reasoning: when converged, the distribution's CVaR_y is the best approximation.

but is it the best cvar-s aproximation? better than var-alpha-cvar? Note: cvar<=var but the same doesn't hold for the underlying distribution

VaR representation? ($\tau/2$?)

Algorithm 2 UPDATE

$UPDATE(x, a, x', r)$
for each y_i **do**
  $a' = \arg\max_{a''} \text{CVaR}_{y_i}(x', a'')$
  generate 'samples' from distribution underlying $\text{CVaR}_{\cdot}(x', a')$
  for each sample v **do**
    $\text{VaR}_{y_i}(x, a) = \text{VaR}_{y_i}(x, a) - \beta \left[1 - \frac{1}{y_i} \mathbb{1}_{(\text{VaR}_{y_i}(x,a) \leq r + \gamma v)}\right]$
    $\text{CVaR}_{y_i}(x, a) = (1 - \beta)\text{CVaR}_{y_i}(x, a) - \beta \left[\text{VaR}_{y_i}(x, a) + \frac{1}{y_i}(r + \gamma v - \text{VaR}_{y_i}(x, a))\right]$
  end for
end for

Algorithm 3 CVaRFromS

$CVaRFromS(x, a, s)$
convert $\text{CVaR}_{\bullet}(x, a) \to \text{VaR}$
find y where $\text{VaR}_y = s$
return $\text{CVaR}_y(x, a)$

Algorithm 4 CVaRFromS, alternate

$CVaRFromS(x, a, s)$
find y where $\text{VaR}_y(x, a) = s$
return $\text{CVaR}_y(x, a)$

Note: Ideas based on working with cvar alone led to nowhere, we have no way of recovering both cvar and var (which we need for the expectation)
Note: should include importance sampling for log-spaced atoms
Idea: the improved VI procedure computes dual of the dual cvar decomposition (proof required), so the above procedure is doing exactly what the VI does
Thesis Idea: separate var-based stationary improvement to a separate chapter after VI, show that AI gridworld is improved

2 Thesis Plan

2.1 Goals

1. Sample-based Q-learning algorithm that finds global CVaR minimum
   (a) Proved convergence (hard)
   (b) verified on an interesting large problem (time-consuming)
2. Approximate Q-learning
   (a) verified on an interesting large problem (time-consuming)

2.2 Todos

1. CVaR Value Iteration
   (a) Prove that the linear procedure does what it's supposed to
   (b) Extend it to Q function (to help with q-learning)
   (c) Idealy prove that var-based policy does the same (to help with q-learning proof)
2. CVaR Q-learning
   (a) ? quantile-regression for cvar
   (b) ? wasserstein convergence

   (c) ? tamar is a measure

3. Sensible testing environments
   (a) table-based (VI + Q-learning)
   (b) large for aproximation
   (c) interesting (e.g. AI grid)
   (d) sensitive to different CVaRs
4. Motivation
   (a) gradient-based methods are worse than exp?
5. Learn
   (a) measure theory basics
   (b) CVaR+, CVaR-: VaR-based connection

2.3 Thesis Layout

1. Introduction
   (a) Motivation
   (b) Contributions
   (c) Outline
2. Preliminaries
   (a) Literature survey
   (b) Definitions
3. Value Iteration
   (a) Tamar explained
   (b) Linear-time explained
   (c) Proof
   (d) Experiments
4. Q-learning
   (a) quantile regression for cvar
   (b) Linear-time extended
   (c) Q-learning
   (d) VaR-based
   (e) Experiments
5. Approximate Q-learning
   (a) quantile regression for cvar
   (b) approximators
   (c) Experiments
6. Conclusion

3 Thought progression - distributional cvar policy iteration

Policy iteration with argmax CVaR does NOT work (see counterexample).

BUT. It should work with the CVaR bellman operator (see RSRDM, cvar optimization approach). And we have what we need, because we have the CVaR values for all alpha (computable from the distribution). This should hold for policy iteration, but maybe even for value iteration?

Idea - **simplified action choice**: I know my current (s_t) distribution. If I knew the action I would take in the next state (s_{t+1}), I can get the next cvar-alpha value by checking the quantile of the current VaR in the next state(because the cvar cared about only these values. It can also happen that we were lucky and all values lower than the previous VaR have 0 probability, or if they have probability 1 - we maximize expected value. These were just examples.). I would need to keep the probabilities of actions in previous iterations. Then I can sample the new policy.

Not really, optimal policy is history-dependent. Important is to choose actions according to different CVaRs, following the procedure [5] in RSRDM.
Because the policies are history-dependent, we don't have access to the actual distributions (these differ depending with which cvar value we came). This is a BIG problem, can't see the solution now. RSRDM overcomes this issue by pure math - the operator is a contraction -> problem solved.

(Not sure what happens if we ignore the nonstationarities and just approximate the values by sampling, just assuming there is a prior on the nonstationary behaviour. Seems unlikely that this is a valid approach.)

The distributional perspective paper hints that nonstationary policies could have better guarantees than stationary - but how? (maybe its refering to the periodical nonstationarities that are used to strenghten aproximation bounds?).

Also, what does it mean to have a value function for nonstationary policies?

Note: For the finite horizon case, only nonstationary policies are optimal - maybe more info could be found there. (Could it be possible to translate nonstationarity-demanding goals to finite horizons? Seems like wishful thinking.)

Note: the policy iteration optimizing utility based shortfall is trivial (use a greedy policy w.r.t. the criterion -> done)(see risk-sensitive reinforcement learning (shen et al)).

Idea: **VaR-based policy iteration**. We could iteratively adjust the acceptance level in accordance to the VaR of the distribution from the initial state. Will this converge to π^*_{cvar}? Counterexample: last node - $0.01 \cdot 0, 0.99 \cdot 100$ - agent won't pick a_1 since a_2 has better expectation before 1. But still may have some empirical merit.
–> This would work if we adjust the var value as we go. This has the same disadvantages as the alpha option, but we would need the transition prob.
The problem with both of these is that they essentially increase the statespace (new input: the current alpha/var). We could save the previous distributions to select actions from (compute it from the old dists), but there is a recurrent relationship, meaning we would have to save the full distribution for each iteration.
But maybe even this wouldn't work since the policy is nonstationary :(.
Middle ground: use maxExp as the baseline in each iteration? Would

Is there really no way to optimize cvar without the extended state-space? Can we prove it?

Simple Idea: we can learn by maximizing expected value, then run with the alpha-altering procedure. Also, we can use the maxexp policy as a starting point for policy iterations. (this is kinda dumb)

Does the inability to converge to a specific distribution has something to do with the fact that we can't distinguish between states if some values have nonzero probabilities in multiple state-action pairs? (Remember the cvar computation, we can choose which part contributes, it doesn't make a difference.)

Use CVaR^+ as an empirical device for exploration?

What about prioritized replay? It would make sense to apply it separately for each atom. (we get a sample with low probability->surprising)
Is distributional PI sure to converge? seems like it on the first glance

---

if cvar is good for robust behavior it makes sense to learn greedy in simulation and apply cvar minimal policy in real runs? or is it the other way around?

3.1 20.11.

Instead of just CVaR, consider 1-step improvement for coherent risk measures. Is the dual CVaR counterexample a counterexample to this? **Double check the example**.
What about markov risk measures? PI could work on them. But is there a point?
Policy gradients for CVaR - are they naive? or nonstationary? does this mean they can be worse than exp? if so, we have excellent motivation for our policy improvement.

3.2 25.11.

alpha-based could be used for a q-learning approach, since we dont need to know the probs (!?) -> this would open a path towards completely solving cvar

3.3 28.11.

Notice that the linearized yV approach equals approximating a discrete distribution
The alpha-based equation cannot be used for xi computation since we do not have F_x.
But we could use the VaR equalities.
Problem is that these hold for continuous variables only - could be solvable after some thinking (?)
this is stupid, we can just combine the distributions by sorting and weighting by p

---

tamars VI is in principle the same as distributional value iteration (representing only discrete distributions - is this minimizing wasserstein?). So if we chose to update each atom according to argmax over that particular alpha - is this the same as tamar but extended to Q approximators?
note on the var-based tamar implementation: construct the distribution by combining cdfs (like on paper), compute the cvar exactly (nlogn because of sort), choose action, update the values
It may be problematic to retrieve xis from the sampled distributions (without knowing transition probs) -> we can use var-based policy here, since when knowing the real s** it (?) equals the optimal policy! (but it is not a real distribution so exp doesnt make sense? investigate)
on first try it seems the d/dalpha solution is the wasserstein LP1 var-based solution. Maybe comparing this with the tamar approach will reveal similarities (is it the same? that would be cool)

3.4 2.12.

observation: empirically, simple sort IS the same as tamar
observation: tamar doesn't optimize wasserstein
observation: tamar is a biased estimate (allways >=) could wasserstein be unbiased? (seems unlikely, it has an integration error)
what if we used wasserstein minimizer in the cvar VI?

3.5 5.12.

xi can be also extracted from sort - its the used portion in computing cvar (check)

3.6 7.12.

The tamar-like procedure for wasserstein may be overoptimistic. BUT the wasserstein VI converges to max exp (check) eventho it doesn't fit it in one step
Empirical observation: tamar-wasserstein behaves like tamar (same policy)
check: var-based actualy optimizes yV (as a function of var), there is a connection

3.7 10.12.

cannot see the q-learning procedure with quantiles alone working (not sure)
is it possible to use the var trick? for a single transition, for each atom: get atoms var, transform, use to select action, bring closer

3.8 20.12.

investigate var-based quantile regression: $V = (1 - \alpha)V + \alpha(r + \gamma V')$ if we know e.g. that all atoms of V' are higher than all atoms of V, we should shift $V \to V'$
about the distance measure: something like $C_{atoms}(p, q) = \sum_{atoms} |CVaR(p) - CVaR(q)|$?