Q-Value Weighted Regression

Reinforcement Learning with Limited Data

Piotr Kozakowski[1,3], Łukasz Kaiser[2], Henryk Michalewski[1,3], Afroz Mohiuddin[2], Katarzyna Kańska [1]

[1] Google Cloud [2] Google Brain [3] University of Warsaw

Motivation

AWR: Advantage Weighted Regression

Very simple: just 2 regressions.

Policy evaluation:

$$\arg\max_V \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}||\mathcal{R}_{\mathcal{D}}^{\mathbf{s},\mathbf{a}} - V(\mathbf{s})||^2$$

Weighted regression towards the previously-performed actions.

Policy improvement:

$$\arg\max_\pi \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\left[\log\pi(\mathbf{a}|\mathbf{s})\exp\left(\frac{1}{\beta}(\mathcal{R}_{\mathcal{D}}^{\mathbf{s},\mathbf{a}} - V(\mathbf{s}))\right)\right]$$

Good and stable! But not very sample efficient.

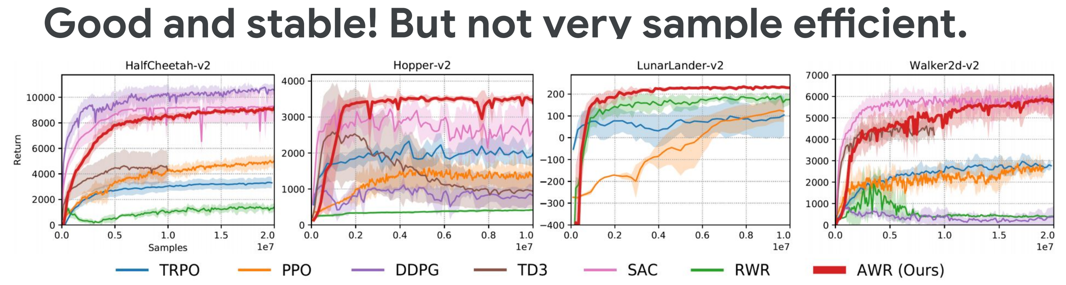


Figure borrowed from Peng et al, 2019.

Struggles with large state spaces.

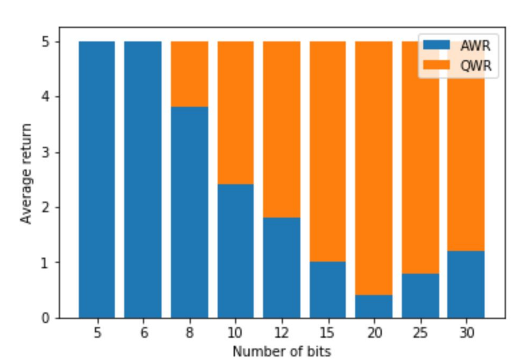
BitFlip: simple env with many states.

State: N bits - 2^N states in total.

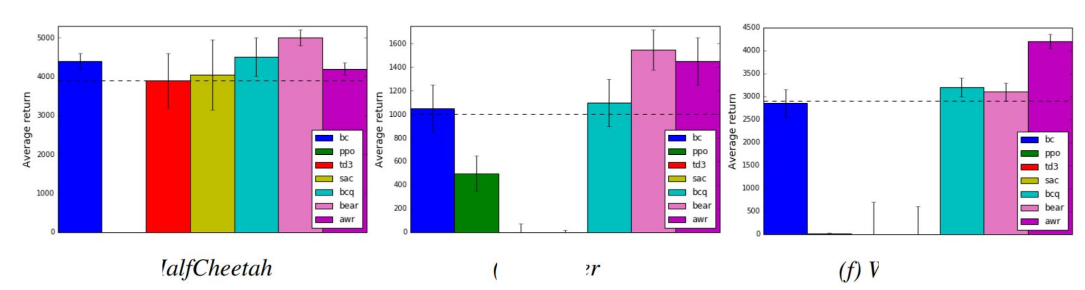
Action: flip one of the bits.

Reward: 1 if we flipped 0 -> 1, -1 otherwise.

Episode terminated after 5 steps.



However, it works offline.



Our method

QWR: Q-Value Weighted Regression

Why is AWR not sample-efficient?

Recall the AWR policy update rule:

$$\arg\max_\pi \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\left[\log\pi(\mathbf{a}|\mathbf{s})\exp\left(\frac{1}{\beta}(\mathcal{R}_{\mathcal{D}}^{\mathbf{s},\mathbf{a}} - V(\mathbf{s}))\right)\right]$$

If we have only one action for each state:

$$\text{for all } (s,a),(s',a') \in \mathcal{D} : s = s' \implies a = a'$$

Then the policy network can learn a small replay buffer **exactly** - ignoring advantage weights!

$$\pi(\mathbf{a}|\mathbf{s}) = \mathbf{1}\left[(s,a) \in \mathcal{D}\right]$$

Which gives **no policy improvement**.

Solution: many actions for the same state.

Policy improvement:

$$\arg\max_\pi \mathbb{E}_{(\mathbf{s},\mu)\sim\mathcal{D}}\ \mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\left[\log\pi(\mathbf{a}|\mathbf{s})\exp\left(\frac{1}{\beta}(Q(\mathbf{s},\mathbf{a}) - \hat{V}(\mathbf{s}))\right)\right]$$

where $\hat{V}(\mathbf{s}) = \mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\ Q(\mathbf{s},\mathbf{a})$ Similar to AWR, but using a Q-network *Q(s, a)*.

We estimate the loss across multiple actions sampled from μ.

This **prevents overfitting** on actions - like using an infinite dataset.

Q-networks can learn improved baselines.

Policy "evaluation":

$$\arg\min_Q \mathbb{E}_{(\mathbf{s},\mu)\sim\mathcal{D}}\ \mathbb{E}_{\mathbf{a}\sim\mu(\cdot|\mathbf{s})}\ ||Q^\star(\mathbf{s},\mathbf{a}) - Q(\mathbf{s},\mathbf{a})||^2$$

$$Q^\star(\mathbf{s},\mathbf{a}) = \mathbf{r} + \gamma\mathbb{E}_{\mathbf{a}'_1,\ldots,\mathbf{a}'_n\sim\mu(\cdot|\mathbf{s}')}F\{Q(\mathbf{s}',\mathbf{a}'_i) \mid i \in \{1..n\}\}$$

$$F(X) = \tau\log\left[\frac{1}{|X|}\sum_{x\in X}\exp(x/\tau)\right]$$

With Q-networks, we can estimate the advantage of an **improved** policy μ^* using a **backup operator** *F*.

Similar to **Q-learning** - but with action sampling to handle continuous action spaces.

In practice, we set *F = log-sum-exp*.

Results

Much **more sample-efficient than AWR. On-par with SAC.**

Algorithm	Half-Cheetah	Walker	Hopper	Humanoid
QWR-LSE	2323 ± 332	$\mathbf{1301 \pm 445}$	$\mathbf{1758 \pm 735}$	511 ± 57
QWR-MAX	2250 ± 254	1019 ± 1185	1187 ± 345	503 ± 49
QWR-AVG	1691 ± 682	1052 ± 231	420 ± 65	455 ± 41
AWR	-0.4 ± 0	67 ± 11	110 ± 81	500 ± 4
SAC	$\mathbf{5492 \pm 8}$	493 ± 6	1197 ± 175	$\mathbf{645 \pm 27}$
PPO	51 ± 41	-14 ± 98	15 ± 75	72 ± 18

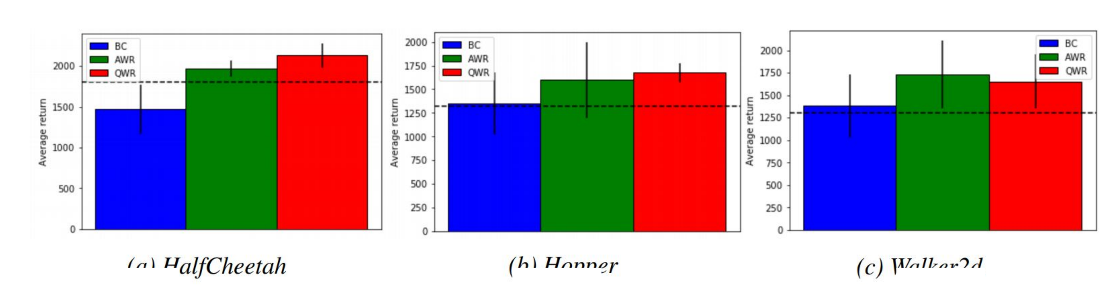
Scores @ 100K interactions.

Works on Atari with the same hyperparameters!

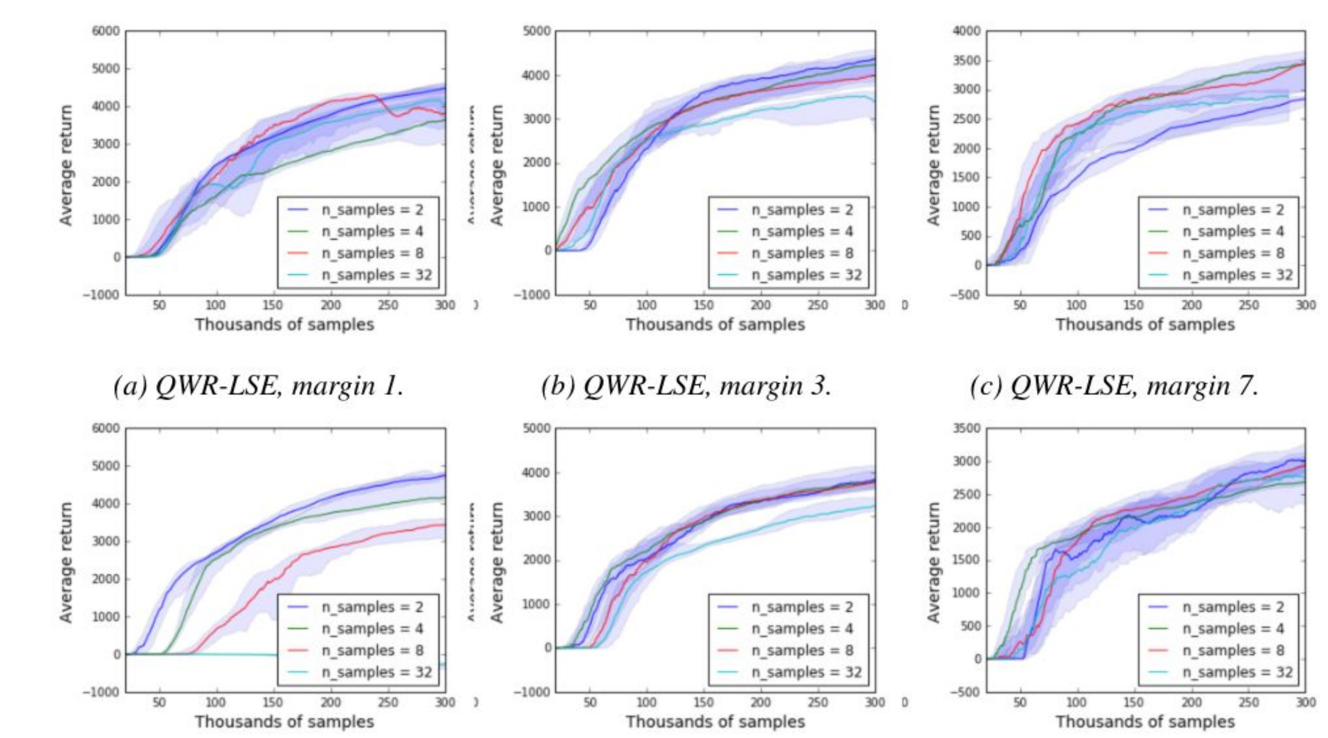
Algorithm	Boxing	Breakout	Freeway	Gopher	Pong	Seaquest
QWR-LSE	**4.6**	**8**	21.2	**776**	−7.6	308
QWR-MAX	−1.8	0.8	16.8	580	**−2**	252
QWR-AVG	−0.8	1.4	19.2	548	−9	296
PPO	−3.9	5.9	8	246	−20.5	**370**
Rainbow	2.5	1.9	**27.9**	349.5	−19.3	354.1
MPR	16.1	14.2	23.1	341.5	−10.5	361.8
MPR-aug	30.5	15.6	24.6	593.4	−3.8	603.8
SimPLe	9.1	16.4	20.3	845.6	12.8	683.3
Random	0.1	1.7	0	257.6	−20.7	68.4

Scores @ 100K interactions.

Works offline out-of-the-box, just like AWR.



Robust to hyperparameter choices.



(a) QWR-LSE, margin 1. *(b) QWR-LSE, margin 3.* *(c) QWR-LSE, margin 7.*

(d) QWR-MAX, margin 1. *(e) QWR-MAX, margin 3.* *(f) QWR-MAX, margin 7.*

In QWR-LSE, *F = log-sum-exp*. In QWR-MAX, *F = max*.

"margin" is the number of steps in multi-step targets for Q training.

Conclusion

QWR is a simple, yet effective RL algorithm.

- Based on AWR update rule, but with action sampling.
- Q-network critic with extra improvement similar to Q-learning.
- Much more sample-efficient than AWR, on-par with SAC on MuJoCo @ 100k.
- Better than Rainbow on Atari @ 100k.
- On-par with AWR in offline RL on MuJoCo.