Ablation: Implicit Quantile Networks for Distributional Reinforcement Learning

Yu-Lun Hsu, Cheng-You Zhong, Bing-Zhi Ke
Department of Computer Science
National Yang Ming Chiao Tung University
{alan890104.cs07, yojoh1225.cs07, jklzxcvbnm1225.cs07}@nctu.edu.tw

1 Problem Overview

- **The main research problem tackled by the paper:**
  The main research problem tackled by the paper is the problem encountering on QR-DQN. In QR-DQN, the quantiles are fixed lacking of flexibility. The hyperparameters we can tune are limited. To increase the precision of our distribution, we need sufficiently many quantiles, which seems inefficeint in computing. Due to the problems mentioned above, IQN appeared and tried to solve thes problems with some tricks.
- **High-level description of the proposed method:**
  Comapred with QR-DQN, IQN's structure of the neural network has changed to a different appearence, including inputs, outputs, the connection between layers, and so on. Instead of approximating quantile functions at fixed points, IQN approximates Z_τ with $f(\psi(x), \phi(\tau))_a$. In addition, IQN reparameterizes sampling distribution via the learned quantile function to approximate an error, which yields an implicitly defined return distribution and gives rise to a large class of risk-sensitive policies.

2 Background and The Algorithm

IQN is a deterministic parametric fuction trained to reparameterize samples from a base distribution Z to the respective quantile values of a target distribution. The optimization problem is to optimized the Q function by learning an implicit representaion of the return distribution.

In IQN, we have $Z_\tau \approx f(\psi(x) \odot \phi(\tau))$. τ is the quatile of the distribution. ψ is a function used to encode our input states $\psi : \mathcal{X} \to \mathbb{R}^d$. $\odot$ denotes the element-wise (Hadamard) product. ϕ, $\psi : [0, 1] \to \mathbb{R}^d$, is a function computing an embedding for the sample point τ:

$$\phi_j(\tau) := RELU(\textstyle\sum_{i=0}^{n-1} cos(\pi i \tau) w_{ij} + b_j)$$

where n is the number of the neurons in ϕ, w represent the weights and b is denoted as the biases.

$\beta : [0, 1] \to [0, 1]$ is a distortion risk measure, with identity corresponding to risk-neutrality. Then, the distorted expectation of $Z(x, a)$ under β is given by:

$$Q_\beta(x, a) := \mathop{\mathbb{E}}_{\tau \sim U([0,1])} \left[Z_{\beta(\tau)}(x, a) \right]$$

The corresponding policy to $Q_\beta(x, a)$ is :

$$\pi_\beta = \mathop{argmax}_{a \in \mathcal{A}} Q_\beta(x, a)$$

Preprint. Under review.

During the implementation on IQN, we get K samples of $\tau \sim U([0,1])$ to approximate our Q_β. Therefore, we changed the form of the equation above to :

$$\widetilde{\pi}_\beta(x) = \underset{a \in \mathcal{A}}{argmax} \frac{1}{K} \sum_{k=1}^{K} Z_{\beta(\widetilde{\tau}_k)}(x, a)$$

For two samples $\tau, U([0,1])$, and policy π_β, the sampled TD error at step t is:

$$\delta_t^{\tau,\tau'} = r_t + \gamma Z_{\tau'}(x_{t+1}, \pi_\beta(x_{t+1})) - Z_\tau(x_t, a_t)$$

After getting the TD error, we can compute the Huber quantile regression loss $\mathcal{L}$:

$$\mathcal{L}(x_t, a_t, r_t, x_{t+1}) = \frac{1}{N'} \sum_{i=1}^{N} \sum_{j=1}^{N'} \rho_{\tau_i}^K(\delta_t^{\tau_i, \tau_j'})$$

where $\rho_{\tau_i}^K(u) = L_k(u)|\tau - \delta_{u<0}|$. Finally, we can update our neural network by the loss function above.

3 Detailed Implementation

According to the above, we design some experiments to test the robustness and stability IQN itself with removing some features from it, seeing how that affect performance. We select **Open AI Gym**, a toolkit for developing and comparing reinforcement learning algorithms, as our testing infrastructure.

We implement our ablation on three control theory problems with continues state space:

- **CartPole-v1:**
  CartPole will terminate when it meet the clearance condition in total 2000 episodes. The clearance condition is triggered as the EWMA reward achieves 190.
- **Acrobot-v1**
  The acrobot system has discrete action set as CartPole. The goal is to swing the end of the lower link up to a given height. However, since the clearance condition is not defined, we will observe acrobot for 4000 episodes and give an elaboration of the result.
- **Pendulum-v0**
  The pendulum starts in a random position, and the goal is to swing it up so it stays upright. It contains a continues action space with range $[2, 8]$, therefore we modify the environment, dividing the action set into 20 pieces. The setting of termination is same as Acrobot.

The table below is our hyperparameter settings of the original IQN algorithm. Moreover, considering reproducibility of our experiments, we fixed all the random seeds to 20.

Hyperparameter	value	description
γ	0.97	discount factor
α	0.0001	learning rate
ε	0.05	epsilon
$capacity$	10000	replay buf size
$exploration$	1000	start train size
$updatefreq.$	100	load freq.

Hyperparameter	value	description
N	3	for current net
N'	3	for target net
K	8	sample for Q
$\|\tau\|$	64	size of τ
$updatefreq.$	100	load freq.
$batchsize$	32	batch size

Figure 1 shows the depictions of our testing algorithms. The original algorithm is Figure 1(a). τ samples from uniform distribution and then transforms by a risk distortion measure β. No sooner had the quantile sample encoding ϕ generated than we performed a element-wise product as a mixture of them. After passing the last layer f, the IQN network outputs $Z_{\beta(\tau)}(x, a)$ corresponding to the sample τ. The process is the main spirit of IQN, also called "Inverse Transfrom Sampling".

The topic can be seperated into several categories:

1. **Degenerate:**

   As we mention, IQN tackles the problems encountering on QR-DQN: The quantiles are fixed,lacking of flexibility. The more the precision we expect, the more quantiles we need in QR-DQN. However, IQN samples quantiles from uniform distribution, which is a continues space, thus the problem is solved.

   Now, considered a degenerate form, QR-DQN-like IQN with 64 fixed $\tau \in [0, 1]$. We wonder whether the performance is still good or not. We anticipate a worse EWMA reward would gain. Notice that **quantile location is the median of each 64 interval** in order to simulate QR-DQN, whereas the original IQN range $[0, 1]$ will lead to a terrible performance, but the reason is not yet known.

2. **Remove cosine transform:**

   The authors of the paper do not give the mathematical reason of cosine transformation. So, we peel off the cosine function, sending τ into ϕ directly, and observe the effect.

3. **Change mixture way:**

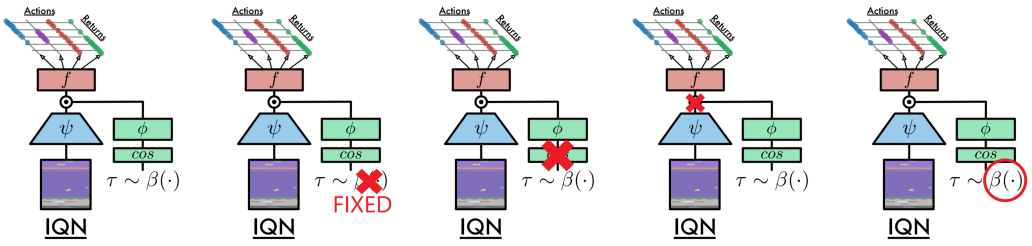
   Since the network is not deep, the paper chooses element-wise product as a mixture of two encodings ϕ and ψ. We substitute other methods for the element-wise product, removing it, changing to residual $\psi(1 + \phi)$ and concatenation being considered. It's a pitty that concatenation never converge at all, so we would not show the result in our empirical evaluation. It's noteworthy that **the dimension may change according to the applied method**.

4. **Modify risk distortion:**

   β function results in a risk-averse or risk-seeking agent. CVaR(.25) and Wang(.75) are the candidates of our ablation. Furthermore, another task is that we would try normal sampling rather than original uniform sampling.

5. **Modify hyperparameter:**

   We have a test on $N = N' = 1$, the number of sampling while calculating loss. It is conceivable that a slower convergence speed or worse early performance will hold.



(a) Original (b) QR-DQN like (c) No cos transform (d) Other Mixture (e) Change β

Figure 1: Comparison between algorithms

4 Empirical Evaluation

In the three environment, We all find some differences of our ablation, especially, the operatoring of removing the cosine function before quantile encoding. We will show our empirical results in the order of the three environment we introduce above respectively.

- Carpole

In first environmevt, carpole-v0, our success requirment is the EWNA reward over a hundred and ninety-five points. Then, the traing will be automatically terminated. There are finally six models converging in the environment. And five of them get well trained when terminated. Here is the picture overlap of the models:

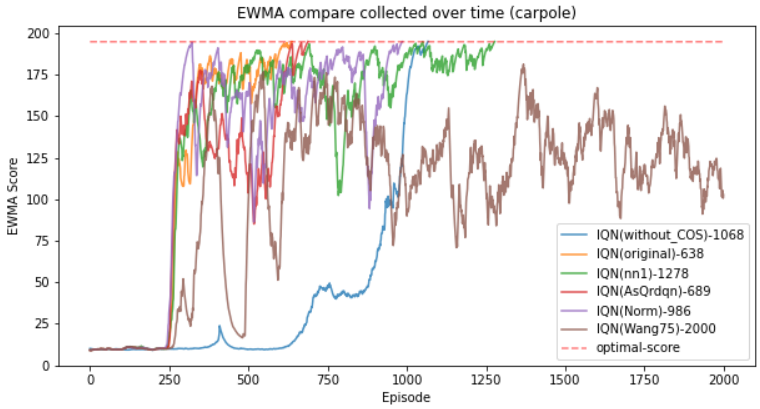
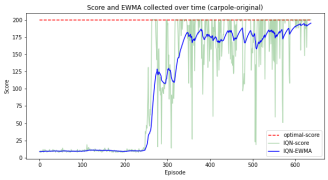
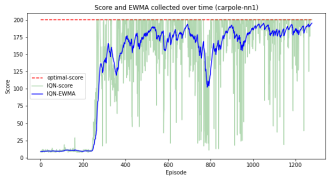


Figure 2: Carpole compare

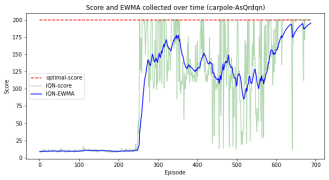
In figure 1, we see that when we set 'N' and 'N" to one, the time it need to reach the success requirment is the longest. The second one to converge is the model without 'cosine' function, and it also has the most different converge curve comparing to others. Following after that, the third one is the model with normal distribution quantile initialization. In addition, what surprise us the most in our result is that most of the β function mentioned in the paper cannot successfully converge to the success requirment. The only one that shows a little convergence trend is β function of 'Wang(0.75)'. However, it eventually does not reach the requirment.



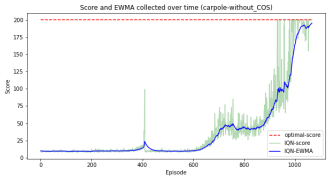
(a) Native



(b) N = N' = 1



(c) QR-DQN like



(d) Remove cosine transform

Figure 3: Carepole EWMA reward with scores per episode

To observe the empirical results carefully, let's take a look at the EWMA reward curve with its score per episode in figure 2. As the comparing result, we know that setting 'N' and 'N" to one need the longest time to converge. In figure 2b, it presents that the change of EWMA

reward curve is huger than the curve in native setting(figure 2a). And, there are no scores reaching to the optimal score around 800 episode. In our opinion, it due to the numbers of sample 'N' and 'N" is too small to correctly calculate the loss of the policy. Therefore, the convergence of the policy need longer period and the EWMA reward is more unstable. In figure 2c, it seems that the curve does not converge as stable as native setting(figure 2a) after fixing the quantiles. Its EWMA reward drops several times and the scores are apparently different between initial stage and middle stage. In initial stage, it has even no scores less than fifty after curve ascending. In middle stage, it, however, gets floating performances strongly influencing the convergence of the EWMA reward. In figure 2d, when we remove the 'cosine' function before the quantile encoding, as our expectation, the converge speed is much slower. Nevertheless, the convergence curve is more stable than all the other operatorings, and the score also ascends in gradually instead of changing between two hundreds and zero in short period.

- Acrobot

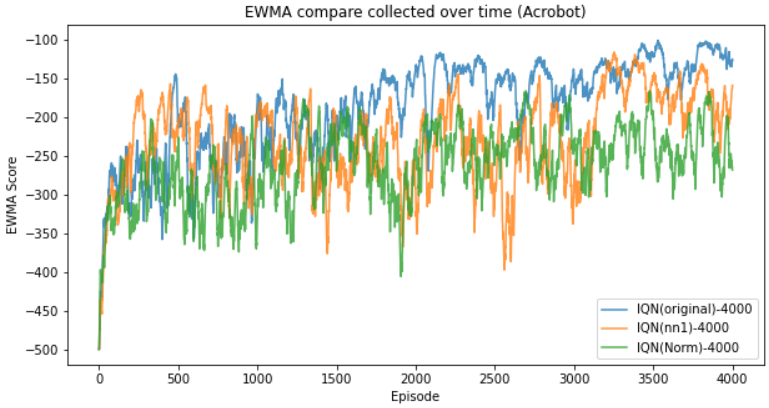
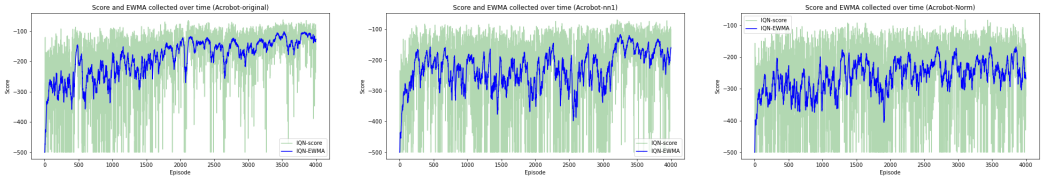


Figure 4: Acrobot compare

In acrobot-v1, there are only three models that show the convergence trend. To our surprised, the fixed quantiles model does not get well trained. It seems that directly fixes the quantile cannot replace the ability of randomly sampling quantile. The model without cosine and all of the β function do not get good performance in our training, either. All of them are on the line of minus five-hundreds points. Therefore, we do not show it on the comparing(figure 3). In our comparing(figure 3), the native model does the best performance. The other two models seem to be similiar. However, the curve of setting 'N' and 'N" to one seems to be more stable in the end of training. The scores it has after 3000 episode(figure 4b) show less bad performance than the scores in normal distribution(figure 4c). Further, as the result in carpole, the model of setting 'N' and 'N" to one need more time to get a stable convergence.



(a) Native (b) N = N' = 1 (c) Normal distribution

Figure 5: Acrobot EWMA reward with scores per episode

- pendulum

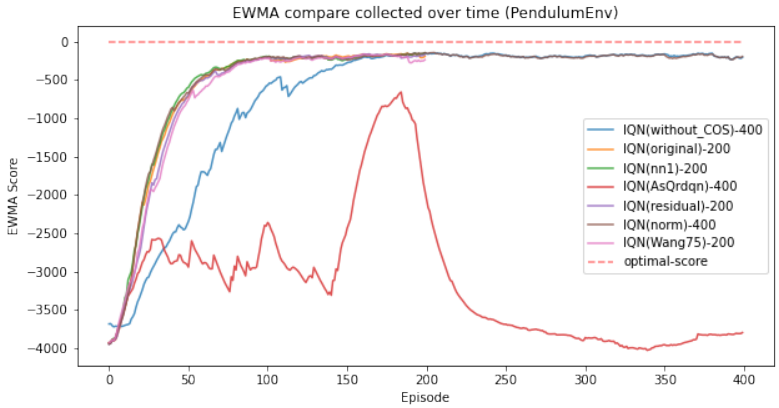
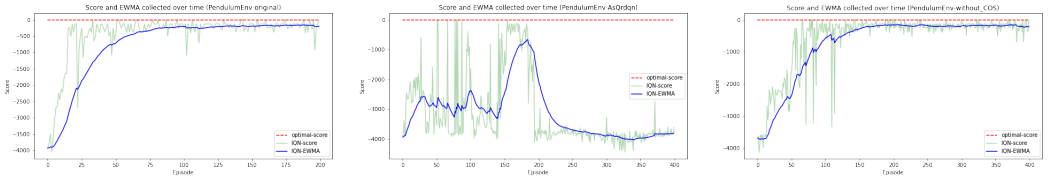


Figure 6: Pendulum compare

There are apparently three kinds of convergence curve in this environment, showed in figure 5. Similiar to the result in carpole, 'Wang(0.75') is the only convergence model among all β function under our hyperparameters, and shows the same result as native model. In addition, the model removing cosine transform shows a steady convergence like what it do in carpole environment. To look carefully into the socre in each episode(figure 6c), its scores also have the same change. It ascends in gradually rather than suddenly improve. As to the QR-DQN like model, it does not converge as smooth as the other two models. The curve ascends suddenly around 150 episode, but soon turns terrible. In fact, the scores around 150 episode are all stay on the area close to the optimal value. It means that it once gets a well trained model, but fails with the following training.



(a) Native (b) QR-DQN like (c) Remove cosine transform

Figure 7: Pendulum EWMA reward with scores per episode

5 Conclusion

- **The potential future research directions:**
  The result of our ablation research shows that the β function might be not suitable for all of the environment. In the environment we choose, only 'Wang(0.75)' can get a better performance than other β functions. In the paper, it means a risk seeking function. Perhaps it is due to the environments we choose are all upward convergence which makes the risk seeking function can show well, or it is due to the environments we choose are too uncomplicated. We need more experiments on different environment to research on it.

  Another result that is worth to talk about is the ablation of fixing the quantiles. In our expectation, the result should be as same as the one in QR-DQN. However, it seems that we have a wrong guess. The result shows not that good as the native IQN. In some case, it

even can not get a well trained model. We guess it might be the quantile encoding network that have a effect on it.

- **Any technical limitations:**

  Due to the restriction of the design of neural networks, it is hard to use IQN in environment with multi-action space. In pendulum, we split the continuous action in to lot of fractions to deal with the unlimited action. But, if there are multi-action space, the number of the output layer will increase in exponential form. It really cause the difficulty of training the model.