Replication: Combining Policy Gradient and Q-Learning

0751231 Yang Tzeng, 0610767 Cheng Huang, 0612230 Chia-An Hou
Department of Computer Science
National Chiao Tung University
Yang850409@gmail.com
vin30731.ee06@nycu.edu.tw
doubt1103.am06@nctu.edu.tw

1 Problem Overview

This paper describe a new technique that combines policy gradient with off-policy Q-learning, drawing experience from a replay buffer. This is motivated by making a connection between the fixed points of the regularized policy gradient algorithm and Q-value. This connection allows us to estimate the Q-values from the action preferences of the policy to which we apply Q-learning updates. Refer to the new technique as 'PGQL', for policy gradient and Q-learning.

2 Background and The Algorithm

The **advantage function** is the difference between the action-value and the value function:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$$

Since $V^\pi(s) = \sum_a \pi(s,a) Q^\pi(s,a)$, so the identity $\sum_a \pi(s,a) A^\pi(s,a) = 0$, which simply states that the policy π has no advantage over itself.

The **Bellman operator** $\mathcal{T}^\pi$ for policy π is defined as

$$\mathcal{T}^\pi Q(s,a) = \mathop{\mathbf{E}}_{s',r,b}(r(s,a) + \gamma Q(s',b))$$

where the expectation is over next state $s' \sim P(\cdot, s, a)$, the reward $r(s,a)$, and the action b from policy $\pi_{s'}$. The Q-value function for policy π is the fixed point of the Bellman operator for π, then the optimal Bellman operator $\mathcal{T}^*$ is defined as

$$\mathcal{T}^* Q(s,a) = \mathop{\mathbf{E}}_{s',r}(r(s,a) + \gamma \max_b Q(s',b))$$

In the online setting the Bellman operator is approximated by sampling and bootstrapping, whereby the Q-values at any state are updated using the Q-values from the next visited state. Exploration is achieved by not always taking the action with the highest Q-value at each time step.
In this paper, the author mainly uses the method called 'Boltzmann exploration', it is a popular technique where the policy is given by the softmax over the Q-values with a temperature T

$$\pi(s,a) = \frac{\exp \frac{Q(s,a)}{T}}{\sum_b \exp \frac{Q(s,b)}{T}}$$

Preprint. Under review.

where it is common to decrease the temperature over time

The **policy gradient theorem** states that the gradient of J with respect to the parameters of the policy is given by

$$\bigtriangledown_\theta J(\pi) = \mathop{\mathbf{E}}_{s,a} Q^\pi(s,a) \bigtriangledown_\theta \log \pi(s,a) \tag{1}$$

where the expectation is over (s,a) with probability $d^\pi(s)\pi(s,a)$. Throughout this paper will use the non-discounted distribution of states.
In the online case it is common to add an entropy regularizer to the gradient in order to prevent the policy becoming deterministic. This ensures that the agent will explore continually. The updates becomes

$$\triangle\theta \propto \mathop{\mathbf{E}}_{s,a} Q^\pi(s,a) \bigtriangledown_\theta \log \pi(s,a) + \alpha \mathop{\mathbf{E}}_{s} \bigtriangledown_\theta H^\pi(s) \tag{2}$$

where $H^\pi(s) = -\sum_a \pi(s,a) \log \pi(s,a)$ denotes the entropy of policy π, and $\alpha > 0$ is the regularization penalty parameter. Throughout this paper will make use of entropy regularization, however many of the results are true for other choices of regularizers with only minor modification.

Then derive a relationship between the policy and the Q-values when using a regularized policy gradient algorithm. There are two cases, tabular case and general case.

Tabular case
Define gradient $f(\theta) = \mathbf{E}_{s,a} Q^\pi(s,a) \bigtriangledown_\theta \log \pi(s,a) + \alpha \mathbf{E}_s \bigtriangledown_\theta H(\pi_s)$, and $g_s(\pi) = \sum_a \pi(s,a) \forall s$.
A fixed point is one where can no longer update θ in the direction of $f(\theta)$ without violating one the constraints $g_s(\pi) = 1$.
So any fixed point must satisfy $f(\theta) = \sum_s \lambda_s \bigtriangledown_\theta g_s(\pi)$, for each s the Lagrange multiplier $\lambda_s \in \mathbb{R}$ ensure that $g_s(\pi) = 1$. Then we obtain

$$\mathop{\mathbf{E}}_{s,a} (Q^\pi(s,a) - \alpha \log \pi(s,a) - c_s) \bigtriangledown_\theta \log \pi(s,a) = 0 \tag{3}$$

where we have absorbed all constants into $c \in \mathbb{R}^{|\mathcal{S}|}$.
Indicator function: $\mathbf{1}_A(x) \begin{cases} 1 & if x \in A \\ 0 & if x \notin A \end{cases}$
In the tabular case π is represented by a single number for each state and action pair, and the gradient of the policy with respect to the parameters is the indicator function, that is $\bigtriangledown_\theta \pi(s,a) = \mathbf{1}_{(t,b)=(s,a)}$.
Then from (3) can get $Q^\pi(s,a) - \alpha \log \pi(s,a) - c_s = 0$ for each s.
Multiplying by $\pi(a,s)$, and summing over $a \in \mathcal{A}$:

$$\begin{aligned} &\sum_a \pi(a,s) \cdot (Q^\pi(s,a) - \alpha \log \pi(s,a) - c_s = 0 \\ \Rightarrow &V^\pi(s) - \alpha \pi(a,s) \log \pi(s,a) - c_s = 0 \\ \Rightarrow &c_s = \alpha H^\pi(s) + V^\pi(s) \end{aligned}$$

Substituting c into equation (3),

$$\pi(s,a) = \exp(A^\pi(s,a)/\alpha - H^\pi(s)), \quad \forall s \in \mathcal{S} \quad \forall a \in \mathcal{A} \tag{4}$$

Use the policy to derive an estimate of the Q-values,

$$\tilde{Q}^\pi(s,a) = \tilde{A}^\pi(s,a) + V^\pi(s) = \alpha(\log \pi(s,a) + H^\pi(s)) + V^\pi(s) \tag{5}$$

Rewrite the gradient update (2) as

$$\triangle\theta \propto \mathop{\mathbf{E}}_{s,a} (Q^\pi(s,a) - \tilde{Q}^\pi(s,a)) \bigtriangledown_\theta \log \pi(s,a) \tag{6}$$

Equation (4) states that the action preferences are equal to the Q-values scale by $1/\alpha$, up to an additive per-state constant.

General case

$$\begin{aligned} \text{minimize} \quad & \mathop{\mathbf{E}}_{s,a} (q(s,a) - \alpha \log \pi(s,a))^2 \\ \text{subject to} \quad & \sum_a \pi(s,a) = 1, \quad s \in \mathcal{S} \end{aligned} \tag{7}$$

Consider optimization problem (7), over variable θ with parameterizes π, where we consider both the measure in the expectation and the values $q(s)$ to be independent of θ. The optimality condition for this problem is

$$\mathbf{E}\, s, a(q(s,a) - \alpha \log \pi(s,a) + c_s) \bigtriangledown_\theta \log \pi(s,a) = 0$$

Comparing to equation (3), if $q = Q^\pi$ and the measure in the expectation is the same then they describe the same set of fixed point. In the general case of using an approximation architecture we can interpret equation (3) as indicating that the error between Q^π and $\tilde{Q}^\pi$ is orthogonal to $\bigtriangledown_{\theta_i} \log \pi$ for each i.

PGQL update

The main contribution of the paper is a technique to combine policy gradient with Q-learning, call this techinuque 'PGQL'.

Define the estimate of Q using the policy as

$$\tilde{Q}^\pi(s,a) = \alpha(\log \pi(s,a) + H^\pi(s,)) + V(s) \tag{8}$$

At the fixed point the Bellman residual($\| \mathcal{T} Q^{\pi_\alpha} - Q^{\pi_\alpha} \| \to 0$ with decreasing regularization penalty α) will be small for small α, then consider updating the parameters to reduce the Bellman residual in a fasion similar to Q-learning

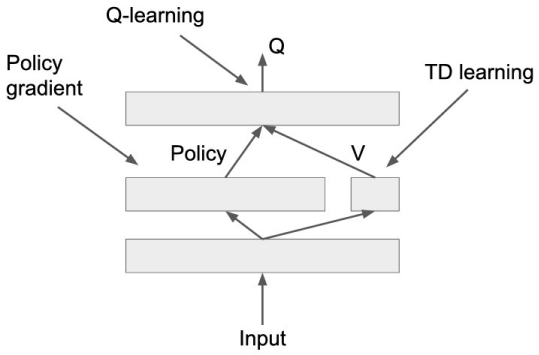
$$\triangle\theta \propto \mathop{\mathbf{E}}_{s,a}(\mathcal{T}^* \tilde{Q}^\pi(s,a) - \tilde{Q}^\pi(s,a)) \bigtriangledown_\theta \log \pi(s,a), \quad \triangle w \propto \mathop{\mathbf{E}}_{s,a}(\mathcal{T}^* \tilde{Q}^\pi(s,a) - \tilde{Q}^\pi(s,a)) \bigtriangledown_w V(s) \tag{9}$$

The full scheme simply combines two updates to the policy, the regularized policy gradient update (2) and the Q-learning update (13). The parameter updates can be written as

$$\begin{aligned} \triangle\theta &\propto (1-\eta) \mathop{\mathbf{E}}_{s,a}(Q^\pi - \tilde{Q}^\pi) \bigtriangledown_\theta \log \pi + \eta \mathop{\mathbf{E}}_{s,a}(\mathcal{T}^* \tilde{Q}^\pi - \tilde{Q}^\pi) \bigtriangledown_\theta \log \pi, \\ \triangle w &\propto (1-\eta) \mathop{\mathbf{E}}_{s,a}(Q^\pi - \tilde{Q}^\pi \bigtriangledown_w V - \eta \mathop{\mathbf{E}}_{s,a}(\mathcal{T}^* \tilde{Q}^\pi - \tilde{Q}^\pi) \bigtriangledown_w V, \end{aligned} \tag{10}$$

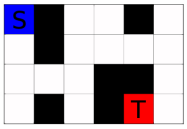
here $\eta \in [0,1]$ is a weighting parameter that controls how much of each update we apply.

3 Detailed Implementation

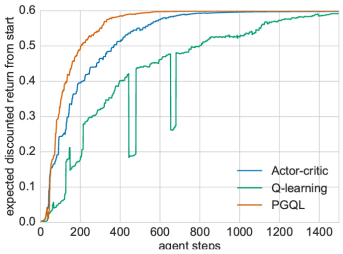


The part of the implementation uses the architecture network in the paper and the equation(8) to estimate the Q value, and use these two to estimate Q-learning.

The difficulty we encountered in the implementation is because this paper does not explain many aspects of the hyperparameters and the architecture of the model. For example, the entire paper of the model architecture only uses the above figure, but it does not explain how it run in the architecture. So in implementation, we can only try to find a result.



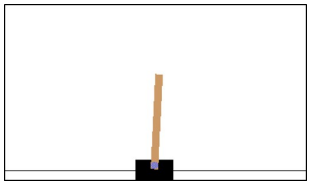
(a) Grid world.

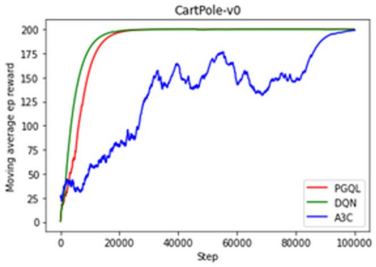


(b) Performance versus agent steps in grid world.

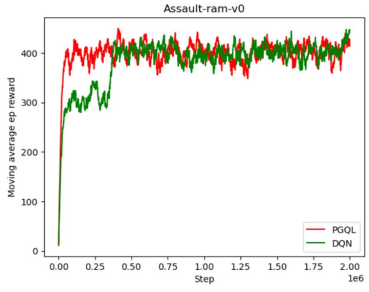
Figure (a) is the first experiment done in this paper, using the grid world environment. But the author has said that this grid world is his own version, so we can't find this version on the Internet. Figure (b) is the experimental result about compare PGQL, Q-learning and Actor-critic. PGQL here can use fewer steps to achieve better results.

4 Empirical Evaluation





Because we can't use the author's grid-world environment, for the convenience of implementation, we chose to use the more common environment, which is "CartPole". Consider that there are many steps in CartPole, we think that if we simply use Q -learning is not suitable, so we change Q-learning to DQN.
Then we compare three different algorithms, PGQL, DQN and A3C. We try to set the same parameters as possible (learning rate is 0.001 and batch size is 32), and replay buffer is also useful. For implementation reasons, our exploration method adopts ε-greedy, this ε will gradually decrease with the number of steps.



The author's main experiment was done on the atari game. The author experimented with 47 games, and PGQL performed best in 34 games. The results of PGQL performance are also shown in the paper.
The problem of implementation is that the author has done 5e7 steps in each game, but because the hardware and time, in order to run more parameters and results, we can't run for a long time every experiment. Another problem is that the author ran a lot of games, but he did not list the hyperparameters used in each game, so we can only keep trying different hyperparameters to see if

we can get better results.
We have actually tried three games, but none of the other games have produced better results, so we only have the assault game first. The comparison result is due to hardware reasons, we can only list the first 2e6 steps. The figure shows that the effect of PGQL is better than that of DQN, but our parameters are not adjusted very well, this result may not be compared with the experimental results on the paper.

5 Conclusion

This paper made a connection between the fixed point of regularized policy gradient techniques and the Q-value of the resulting policy. This is an innovative method and we have benefited a lot from it. It also helps us understand the content of reinforcement learning when learning this method. But the number of experiments we implemented this time is not enough, and the performance of this method remains to be seen.

References