Mini-project weeks 7 & 8
Understanding PCL

GUERIN Cyril, POT Eline

1 Presentation of PCL, Soft Q-learning, and PGQL

What does it have in common with soft Q-learning and PGQL, and what are the main differences?

1.1 PCL

PCL is based on the maximization of expected reward with a discounted entropy regularization. This regularization term brings an interesting connection between softmax temporal value consistency and policy optimality. This led to the Path Consistency Learning algorithm(PCL).

$$\begin{aligned} O_{ENT}(s,\pi) &= O_{ER} + \tau \mathbf{H}(s,\pi) \\ &= \sum_a \pi(a|s)[r(s,a) - \tau \log \pi(a|s) + \gamma O_{ENT}(s',\pi)] \end{aligned}$$

where O_{ER} is the expected discounted reward objective and $\mathbf{H}(s,\pi)$ is the discounted entropy.
PCL combines aspects of Actor-Critic and Q-Learning. It is similar to actor-critic because it maintains and learns models for state values and policy simultaneously. It is also similar to Q-Learning because it minimizes a measure of temporal consistency error.
The authors also introduced Unified PCL, a variant that uses a single model for both policy and values, challenging the traditional actor-critic approach.
Unlike standard policy-based algorithms, PCL and Unified PCL work with both on-policy and off-policy trajectory samples. Moreover, unlike value-based algorithms, they can leverage multi-step consistencies.

1.2 Comparaison to Soft Q-Learning

Soft Q-Learning introduces a method for learning energy-based policies for continuous states and actions. It also uses maximum entropy reinforcement learning. The agent collects new experience from the environment, and store them in a replay buffer as Q-Learning. It samples a minibatch from the replay memory and then updates the soft Q-function and the policy.

1.3 Comparaison to PGQL

The authors of PGQL (Policy Gradient and Q-Learning) combine policy gradient with off-policy Q-Learning and use experiences from the replay buffer. They established a connection between the fixed points of the regularised policy gradient algorithm and Q-values, that enables them to estimate Q-values from the actions preferences of the policy, on which they apply Q-learning updates. They also use regularized policy gradient algorithms but do not use an entropy one.

2 Hyper-parameters of PCL

What are the hyper-parameters? Do you believe PCL is easier or harder to tune than competitors?

2.1 PCL

The hyperparameters that characterize PCL are :

- η_ν : policy learning rate
- η_π : value learning rate
- τ : entropy regularization factor (temperature)
- d: rollout factor

It seems quite easy to tune because there are only a few parameters. To give an idea of the order of magnitude of these hyperparameters, the researchers provided the values they used and tested during their experiments (cf Table 1, and Table 2). They noticed that it was simple to parameterize the critic learning rate in terms of the actor learning rate: $\eta_\nu = C\eta_\pi$ where C is called the critic weight.

parameter	**implementation value/grid search**
batch size	10
rollout d	3
discount factor	1
replay buffer capacity	10,000
parameter for PCL's replay buffer α	1
action learning rate η_π	{0.01,0.05,0.1}
critic weight C	{0.1, 0.5, 1}
entropy regularizer τ	{0.005, 0.01, 0.025, 0.05, 0.1, 0.5, 1}
optimizer	standard gradient descent

Table 1: Values of the hyperparameters tested during the experiments for the Synthetic Tree environment

2.2 Comparaison to Soft Q-Learning

The main hyperparameters used in Soft Q-Learning algorithm are :

- ϵ : energy function (neural network for instance)
- α : entropy regularization factor
- κ : kernel function

parameter	**implementation value/grid search**
batch size	400
rollout d	10
discount factor	0.9, 1
replay buffer capacity	100,000
parameter for PCL's replay buffer α	{0.1, 0.5}
action learning rate η_π	0.005
critic weight C	{0.1, 1}
entropy regularizer τ	{0.005, 0.01, 0.025, 0.05, 0.1, 0.15}
optimizer	Adam

Table 2: Values of the hyperparameters tested during the experiments for the algorithmic tasks

2.3 Comparaison to PGQL

The main hyperparameters used in PGQL algorithm are :

- η : weighting parameter (controls how much of each update is applied)
- α : regularization parameters

3 Would it work in stochastics domains ?

A reviewer doubts that PCL would work in stochastic domains. Do you believe that is the case?

3.1 PCL

All the theorems of the article were claimed under the assumption of a deterministic domain. However, in the Supplemental Material, the researchers gave proof of their results in the more general scenario of a stochastic environment.

3.2 Comparaison to Soft Q-Learning

When compared to Soft Q-Learning, it becomes apparent that this algorithm is a technique for acquiring stochastic policies through the utilization of an energy-based model (EBM). Here, the energy function is the 'soft' Q-function, obtained by optimizing the maximum entropy objective. The authors work in expectation, for instance, $\mathbf{E}_{(s_t,a_t)\sim p_\pi}$ where $p_\pi(s_t, a_t)$ represents the state-action marginals of the trajectory distribution induced by a policy $\pi(a_t|s_t)$.

3.3 Comparaison to PGQL

At the beginning of the article, it is mentioned that it is considered that the agent uses a stochastic policy. In the mathematical expressions, the authors indeed work in expectation, for instance, $\mathbf{E}_{(s,a)\sim p_\pi}$.

4 Could it be applied to continuous actions?

Could PCL be applied to continuous actions? If not, why, and could we change something to turn the answer into a yes? If yes, do you believe it would work well?

4.1 PCL

PCL could not be applied to continuous actions because its methods are based on a discretization of the action space. As outlined in Section 2, the algorithm's methodology involves modeling the agent's behavior through a parametric distribution, denoted as $\pi_\theta(a|s)$ and implemented by a neural network, operating over a finite set of actions.
To extend PCL to continuous actions, one approach is to replace the summation with integration across the action space, and operate in expectation with respect to a distribution over the action space.

4.2 Comparaison to Soft Q-Learning

On the other hand, Soft Q-Learning can be applied to continuous actions as specified in the abstract and the preliminaries of the paper. The authors propose a method for learning expressive energy-based policies where the state space $\mathcal{S}$ and actions space $\mathcal{A}$ are assumed to be continuous. In particular, they defined the state transition probability that represents the probability density of the next state $s_{t+1} \in \mathcal{S}$ given the current state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$: $p_s : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$.

4.3 Comparaison to PGQL

In the PGQL article, the authors considered the infinite horizon, discounted, finite state and action space Markov decision process. So it seems like PGQL could only be applied to finite actions space. However, PGQL is described as a combination of policy gradient and off-policy Q-learning. As both of these algorithms can be applied to continuous space, it does not seem unreasonable to think that PGQL could be applied or adapted to continuous action spaces.

5 Additional proofs and detailed calculations

5.1 Equation (7)

$$\boxed{\textbf{(7) :} \quad \pi(a|s) \propto \exp\left(\frac{r(s,a) - \tau \log \pi(a|s) + \gamma O_{ENT}(s', \pi)}{\tau}\right)}$$

Proof. The objective function that we have to maximize with respect to π is :

$$\begin{aligned} O_{ENT}(s,\pi) &= O_{ER} + \tau \mathbf{H}(s,\pi) \\ &= \sum_a \pi(a|s)[r(s,a) - \tau \log \pi(a|s) + \gamma O_{ENT}(s',\pi)] \end{aligned}$$

and :

$$\begin{aligned} \nabla_{\pi(a|s)} O_{ENT}(s,\pi) &= [r(s,a) - \tau \log \pi(a|s) + \gamma O_{ENT}(s',\pi)] - \pi(a|s)\tau \frac{1}{\pi(a|s)} \\ &= r(s,a) - \tau \log \pi(a|s) + \gamma O_{ENT}(s',\pi) - \tau \end{aligned}$$

Since we let $V^\star(s) = \max_\pi(O_{ENT}(s,\pi))$, we have in the optimum $\pi^\star(a|s)$:

$$\begin{cases} O_{ENT}(s,\pi^\star) = V^\star(s') \\ \nabla_{\pi(a|s)} O_{ENT}(s,\pi) = 0 \end{cases}$$

i.e. :

$$\begin{aligned} r(s,a) - \tau \log \pi(a|s) + \gamma O_{ENT}(s',\pi) - \tau &= 0 \\ \text{i.e. } \tau \log \pi(a|s) &= r(s,a) - \tau \log \pi(a|s) + \gamma O_{ENT}(s',\pi) - \tau \\ \text{i.e. } \pi(a|s) &= \exp\left(\frac{r(s,a) - \tau \log \pi(a|s) + \gamma O_{ENT}(s',\pi) - \tau}{\tau}\right) \\ \text{i.e. } \pi(a|s) &= \exp\left(\frac{r(s,a) - \tau \log \pi(a|s) + \gamma O_{ENT}(s',\pi)}{\tau} - 1\right) \\ \text{Finally : } \pi(a|s) &\propto \exp\left(\frac{r(s,a) - \tau \log \pi(a|s) + \gamma O_{ENT}(s',\pi)}{\tau}\right) \end{aligned}$$

□

5.2 Theorem 1

Theorem 1 : For $\tau > 0$, the policy $\pi^\star$ that maximizes O_{ENT} and state values $V^star(s) = \max_\pi O_{ENT}(s,\pi)$ satisfy the following temporal consistency property for any state s and action a (where $s' = f(s,a)$),

$$\boxed{V^\star(s) - \gamma V^\star(s') = r(s,a) - \tau \log \pi^\star(a|s)}$$

Proof. Theorem 1 is demonstrated in the additional material under the stochastic assumption. Let us detail the calculation in the deterministic case.
The (10) states that :

$$\begin{aligned}
\pi^\star(a|s) &= \frac{\exp((r(s,a)+\gamma V^\star(s'))/\tau)}{\exp(V^\star(s)/\tau)} \\
\text{i.e. } \log(\pi^\star(a|s)) &= \frac{(r(s,a)+\gamma V^\star(s'))}{\tau)} - \frac{V^\star(s)}{\tau} \\
\text{i.e. } \tau\log(\pi^\star(a|s)) &= r(s,a)+\gamma V^\star(s') - V^\star(s) \\
\text{Finally : } V^\star(s) - \gamma V^\star(s') &= r(s,a) - \tau\log\pi^\star(a|s)
\end{aligned}$$

□

5.3 Corollary 2

Corollary 2 : For $\tau > 0$, the optimal policy $\pi^\star$ and optimal state values V^star satisfy the following extended temporal consistency property for any state s_1 and any action sequence $a_1, ..., a_{t-1}$ (where $s_{i+1} = f(s_i, a_i)$),

$$\boxed{V^\star(s_1) - \gamma^{t-1}V^\star(s_t) = \sum_{i=1}^{t-1}\gamma^{i-1}[r(s_i,a_i) - \tau\log\pi^\star(a_i|s_i)]}$$

Proof. We can work recursively and use Theorem 1. For a fixed trajectory $(s_1, a_1, s_2, \dots, s_t)$, we have :

$$\begin{aligned}
&V^\star(s_1) - \gamma^{t-1}V^\star(s_t) \\
&= V^\star(s_1) - \gamma V^\star(s_2) + \gamma V^\star(s_2) - \gamma^2 V^\star(s_3) + \gamma^2 V^\star(s_3) + \cdots - \gamma^{t-1}V^\star(s_t) \\
&= V^\star(s_1) - \gamma V^\star(s_2) + \gamma[V^\star(s_2) - \gamma V^\star(s_3)] + \gamma^2[V^\star(s_3) - \gamma V^\star(s_4)] + \cdots + \gamma^{t-2}[V^\star(s_{t-1}) - \gamma V^\star(s_t)] \\
&= r(s_1,a_1) - \tau\log\pi^\star(a_1|s_1) + \gamma[r(s_2,a_2) - \tau\log\pi^\star(a_2|s_2)] + \cdots + \gamma^{t-2}[r(s_{t-1},a_{t-1}) - \tau\log\pi^\star(a_{t-1}|s_{t-1})] \\
&= \sum_{i=1}^{t-1}\gamma^{i-1}[r(s_i,a_i) - \tau log\pi^\star(a_i|s_i)]
\end{aligned}$$

□

5.4 Updates (16) and (17)

The PCL update rules for θ and $\Phi$are derived by calculating the gradient of O_{PCL}. For a given trajectory $s_{i:i+d}$ these take the form,

$$\boxed{\textbf{(16) :}\ \Delta\theta = \eta_\pi C(s_{i:i+d},\theta,\Phi)\sum_{j=0}^{d-1}\gamma^j\nabla_\theta\log\pi_\theta(a_{i+j}|s_{i+j})}$$

$$\text{and } \boxed{\textbf{(17) :}\ \Delta\Phi = \eta_\nu C(s_{i:i+d},\theta,\Phi)(\nabla_\Phi V_\Phi(s_i) - \gamma^d\nabla_\Phi V_\Phi(s_{i+d}))}$$

Proof. We have :

$$O_{PCL}(\theta, \Phi = \sum_{s_{i:i+d} \in E} \frac{1}{2} C(s_{i:i+d}, \theta, \Phi)^2$$
$$= \sum_{s_{i:i+d} \in E} \frac{1}{2} \left(-V_\Phi(s_i) + \gamma^d V_\Phi(s_{i+d}) + \sum_{j=0}^{d-1} \gamma^j [r(s_{i+j}, a_{i+j}) - \tau \log \pi_\theta(a_{i+j}|s_{i+j})] \right)^2$$

Considering a fixed trajectory $s_{i:i+d}$:

$$\nabla_\theta O_{PCL}(\theta, \Phi) = \frac{1}{2} 2C(s_{i:i+d}, \theta, \Phi) \left(\sum_{j=0}^{d-1} -\gamma^j \tau \nabla_\theta \log \pi_\theta(a_{i+j}|s_{i+j})] \right)$$
$$= -\tau C(s_{i:i+d}, \theta, \Phi) \sum_{j=0}^{d-1} \gamma^j \tau \nabla_\theta \log \pi_\theta(a_{i+j}|s_{i+j})]$$

If we use a value learning rate η_π for the update of θ, that will multiplied to the gradient $\nabla_\theta O_{PCL}(\theta, \Phi)$ we can consider that η_π can be written as a proportional factor of τ, and that τ is 'absorbed' by η_π since the multiplication of two constants is a constant.
Also,

$$\nabla_\Phi O_{PCL}(\theta, \Phi) = \frac{1}{2} 2C(s_{i:i+d}, \theta, \Phi) \left(-\nabla_\Phi V_\Phi(s_i) + \gamma^d \nabla_\Phi V_P hi(s_{i+d}) \right.$$
$$= C(s_{i:i+d}, \theta, \Phi) \left(-\nabla_\Phi V_\Phi(s_i) + \gamma^d \nabla_\Phi V_\Phi(s_{i+d}) \right)$$

□