Rapport DDPG

Simon Groc

March 9, 2025

Contents

1 Introduction

Le bute de se documment et de clarifier le foncionement de ddpg au traver d'un parcoure du code BBRL que nous avont fais avec des explications du coure et la retrencription de celle-ci en code pyhton.

2 Marche aléatoire

Avent de commenser a enténer le modelle il nous faux des donnés sur l'environement pour cela nous alons devoir nous deplaser de manier aléatiore dans celui-ci et recurperet des sample de la forme suivente ("env/terminated", "env/reward" "critic/q_value", "target-critic/q_value"). C'est informations sont recupere a chaque step de la marche aléatiore dans l'environement dans notre cas de figure l'on en fais 10000 avec un agent qui utilise un Gaussian pour se deplacer.

```
class AddGaussianNoise(Agent):
    def __init__(self, sigma):
        super().__init__()
        self.sigma = sigma

    def forward(self, t, **kwargs):
        act = self.get(("action", t))
        dist = Normal(act, self.sigma)
        action = dist.sample()
        self.set(("action", t), action)
```

3 Calcule de la loss du critique

le calcule de la loss du critique et une fonctions qui prent les parametre suivent:

- cfg: Le dictioner des parametre du modelle
- reward: Un tablaux des reconpense en fonctions des transition (taille 2xB)
- must_bootstrap: Un tablaux de booléen qui indique la fin d'un episode (taille 2xB)
- q_values: Le calcule de la Q-values pare le critique (taille 2xB)
- target_q_values: La Q-values calculer par le target (taille 2xB)

ensuite on dois calculer cette formule $y_i = r_{i+1} + \gamma \hat{Q}_{\phi}^{\pi\theta}\big(s_{i+1}, \pi(s_{i+1})\big)$
pour pouvoir la comparer a $\hat{Q}_{\phi}^{\pi\theta}(s_i, a_i \mid \phi)$
et caculer la loss $L = \frac{1}{N}\sum_i \Big(y_i - \hat{Q}_{\phi}^{\pi\theta}(s_i, a_i \mid \phi)\Big)^2$

Pour se faire on commense par le cacule de $\hat{Q}_{\phi}^{\pi\theta}(s_i, a_i \mid \phi)$. Il nous faut donc indique a lagent de caculer les nouvelle q_values pour sa il va utilise les état et les actions du workspace trouver pendands la marche aléatiore puis recuper les q_values du workspace.

```
q_agent(rb_workspace, t=0, n_steps=1)
q_values = rb_workspace["critic/q_values"]
```

une fois que on a cette valeur on cacule notre q_valeus courente il nous faux $\hat{Q}_{\phi}^{\pi\theta}\big(s_{i+1}, \pi(s_{i+1})\big)$ et pour sa on comment pare executer le actore sur s_{i+1} se qui nous donne les action maximale $\pi(s_{i+1})$ une fois optenus on peux enfin metre a jour les valeur du target critique avec s_{i+1} comme indique dans la formule $\hat{Q}_{\phi}^{\pi\theta}\big(s_{i+1}, \pi(s_{i+1})\big)$.

```
with torch.no_grad():

ag_actor(rb_workspace, t=1, n_steps=1)

target_q_agent(rb_workspace, t=1, n_steps=1, detach_actions=True)

post_q_values = rb_workspace["target-critic/q_values"]
```

Dans le bloc ci-dessu aucun gradient nais pris en conte.
Ensuite on appelle la loss foncition :

```
target = reward[:-1].squeeze() + cfg.algorithm.discount_factor * target_q_values[1].s

mse = nn.MSELoss()
critic_loss = mse(q_values[0].squeeze(-1), target)
```

On commense par calculer $y_i = r_{i+1} + \gamma \hat{Q}_{\phi}^{\pi\theta}\big(s_{i+1}, \pi(s_{i+1})\big)$ dans cette forume r_{i+1} et reward[:-1], γ represent le degrés dentisipations du modelle, et $\hat{Q}_{\phi}^{\pi\theta}\big(s_{i+1}, \pi(s_{i+1})\big)$ c'est simplement q_valeus de l'etat suivent si on utilise la meieur acitions pour cette état tous sa est multiplier par must_bootstrap[1].int() un tablaux de booléen qui indique si un episode est finis de cette mainier on ne calcule pas la fin d'un episode et le debus d'un nouvaux comme une trensitions.

torche.squeeze() : cette methode suprime toute les dimentions de taille 1.
puis on aplique un desente de gradient classique.

4 Calcule de la loss de l'actor

Pour calculer la loss de l'actor il nous faux re caluler les meieur actions puis a nouvaux le critique car on veux recuperer les q_values que on va passer au en paramete de la loss foncitons.

```
ddpg.t_actor(rb_workspace, t=0, n_steps=1)
ddpg.t_critic(rb_workspace, t=0, n_steps=1)

q_values = rb_workspace["critic/q_value"]
```

puis on envois les q_values dans la loss fonctions $\mathbb{E}_{a_t \sim \pi_\theta(\cdot)}\left[\nabla_a \hat{Q}_\phi^{\pi_\theta}(s_t, a_t)\, \nabla_\theta \pi_\theta(s_t)\right]$

pour calculer celle la sais trais simple il sufis de faire moin la moyenne car nous cherchons a la maximiser est pythorche ne fais des desente de gradient que pour des probléme de minimisations.