Université de Carthage
Ecole Supérieure de la Statistique et de l'Analyse de l'Information

Examen de Machine Learning

3 ème année du cycle de formation d'ingénieurs

Durée de l'épreuve : 1 heure 30 - Documents non autorisés
Nombre de pages : 3 - Date de l'épreuve : 9 janvier 2024

Exercice 1 : On considère le scripte donné en **Annexe 1** conçu pour utiliser l'alogorithme PPO dans l'environnement `CartPole-v1` de `gym`.

1. Compléter les 20 parties manquantes du code (une partie manquante étant signalée par un numéro entre parenthèse).
2. Pensez-vous que l'on devrait entraîner d'avantage le modèle ou que l'on devrait l'arréter ? Justifier votre réponse.

Execrcice 2 : On considère les pseudo-codes des deux algorithmes suivants :

Pseudo-code 1 : REINFORCE

(a) Initialiser θ
(b) Pour chaque épisode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\}$ effectué selon la politique π_θ faire :
Pour $t = 1, ..., T-1$ faire :
$\theta \leftarrow \theta + \alpha \Delta_\theta log\pi_\theta(s_t, a_t) Q^{\pi_\theta}(s_t, a_t)$

$Q = Q + \alpha(R + \gamma max Q - Q)$

Pseudo-code 2 : PPO

(a) Initialiser θ et fixer le paramètre clip ϵ
(b) Pour $k = 0, 1, 2, ...$ faire :
i. Collecter un ensemble de trajectoires (s, a, r, s') selon la politique $\pi_k = \pi_\theta$ et les stocker dans le *replay buffer*
ii. Estimer la fonction avantage $\hat{A}_t^{\pi_k}$
iii. Mettre à jour, en utilisant la méthode de déscente de gradient stochastique, la politique

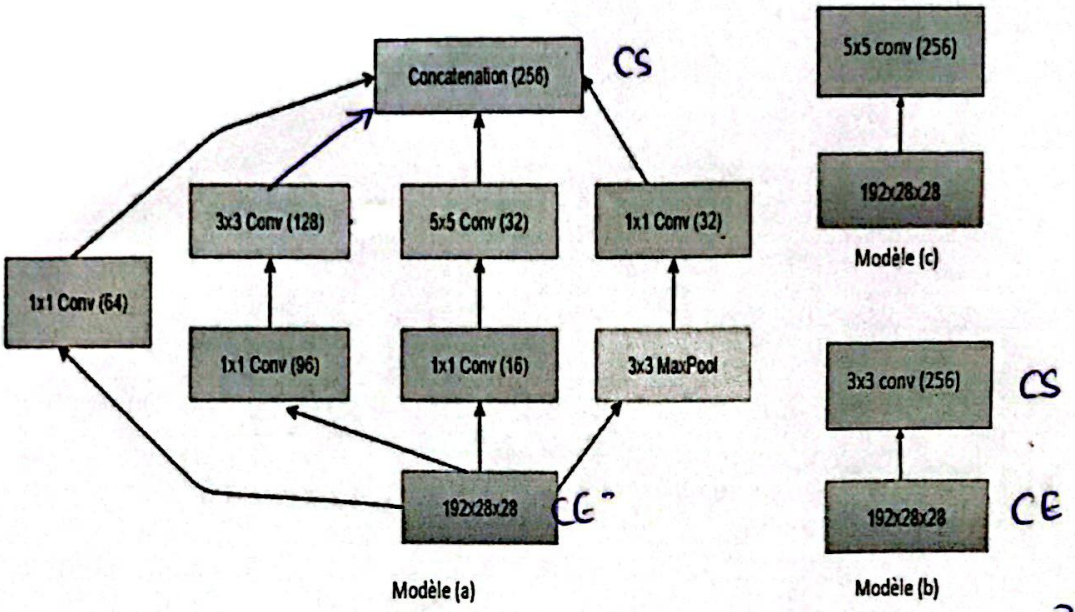
$$\theta_{k+1} = arg \max_a L_{\theta_k}^{CLIP}(\theta)$$

où

$$L_{\theta_k}^{CLIP}(\theta) = \hat{\mathbb{E}}_{\pi_k}[\sum_{t=0}^{T} min(r_t(\theta)\hat{A}_t^{\pi_k}, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^{\pi_k})]$$

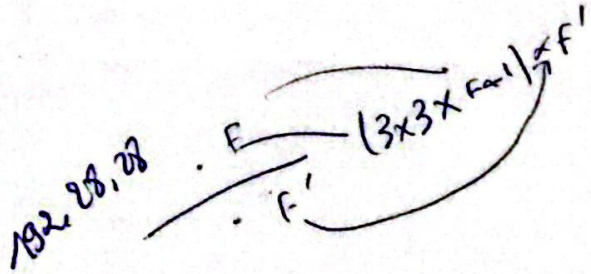
1. Quelle est le principal inconvénient du premier algorithme ?
2. Indiquer les principales différences du second algorithme par rapport au premier.

Execrcice 3 : On considère les 3 modèles d'architecture suivants. On rappelle que **"fxf Conv (K)"** signifie couche de convolution de K filtres dont le filtre est de taille f.



3x3 x

1. A quoi correspond l'architecture du modèle (a) ?
2. Calculer le nombre de paramètres de chacun des 3 modèles puis commenter les résultats obtenus.



Annexe 1

```
%%capture
!pip install .(1).

import .(2).
from stable_baselines3 import .(3).
from stable_baselines3.common.evaluation import evaluate_policy

%load_ext .(4).

env = gym.make(.(5).)
model = PPO("MlpPolicy", .(6)., verbose=1, tensorboard_log=.(7).)
model.learn(total_timesteps=100000)
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 500         |
|    ep_rew_mean          | 500         |
| time/                   |             |
|    fps                  | 1034        |
|    iterations           | 41          |
|    time_elapsed         | 81          |
|    total_timesteps      | 83968       |
| train/                  |             |
|    approx_kl            | 0.005896367 |
|    clip_fraction        | 0.0529      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.401      |
|    explained_variance   | 0.0288      |
|    learning_rate        | 0.0003      |
|    loss                 | 0.0061      |
|    n_updates            | 400         |
|    policy_gradient_loss | -0.00189    |
|    value_loss           | 7.88e-05    |
-----------------------------------------

%tensorboard --logdir ./cartpole_tensorboard/

# Evaluer le modèle
evaluate_policy(.(8)., env, n_eval_episodes=.(9)., render=.(10).)

# Sauvegarder le modèle en l'appelant PPO_model
model.save(.(11).)

# Supprimer le modèle
del .(12).

# Charger le modèle sauvegardé
model = PPO.load(.(13)., env=.(14).)

# Utiliser le modèle
obs = .(15).
while True:
    action, _states = model.predict(.(16).)
   .(17)., .(18)., .(19)., info = env.step(.(20).)
    env.render()
```