ChatGPT

Environnements et fonctions de récompense

La bibliothèque **OpenAI Gym** propose un environnement *Soccer* (Half Field Offense) pour un agent offensif dans un jeu de football 2D. Dans la tâche *Soccer*, la récompense vaut +1 pour chaque but marqué, et 0 sinon [1]. La variante *SoccerEmptyGoal* fournit un signal plus fréquent : l'agent est récompensé lorsqu'il s'approche du ballon et lorsqu'il pousse le ballon vers le but [2]. Ces descriptions officielles (Gym-Soccer) clarifient comment sont définies les fonctions de récompense de base pour un jeu de foot 2D [1] [2].

Exemples d'implémentations et hyperparamètres

PA-DDPG multi-agent (Half Field Offense)

Le dépôt **pddpg-hfo** (par L. Zheng) illustre un entraînement multi-agent 2v2 (Offense vs Defense) en Half-Field Offense. Sa fonction de récompense est extrêmement simple : +1 pour un but marqué, 0 dans les autres cas [3]. Par exemple, les auteurs précisent une fréquence d'évaluation tous les 500 épisodes d'entraînement (avec des épisodes de 1000 pas lors du test) [4]. Ce code (PA-DDPG) fournit un exemple concret de projet multi-agent pour le soccer 2D, avec les métriques et paramètres d'entraînement spécifiés (bien que les hyperparamètres DDPG eux-mêmes ne soient pas détaillés dans le README) [3] [4].

PyTorch – Hybrid-SAC sur Soccer

Un autre exemple (PyTorch) est l'implémentation **hybrid-sac**, qui inclut l'environnement *Soccer* (état 59 dimensions) [5]. L'image ci-dessus montre l'agent (carré) et la balle (cercle) vus de dessus. La fonction de récompense y est dite *« informative »* : elle oriente l'agent pour atteindre d'abord le ballon puis diriger le tir vers le but [5]. Le script d'entraînement associé indique par exemple `--total-timesteps 3500000 --learning-starts 257` (3,5 millions de pas de jeu) [5]. Ces paramètres illustrent la configuration d'apprentissage (durée totale, phases d'exploration, etc.) utilisée dans ce cas d'étude avec un unique attaquant contre gardien.

Soccer Twos (Unity PPO)

Un projet multi-agent 2v2 basé sur Unity ML-Agents (*Soccer Twos*) détaille lui aussi des hyperparamètres précis. Par exemple, le README liste : taux d'apprentissage du gardien = 8e-5, du joueur de champ = 1e-4, *gamma* = 0.995, *batch size* = 32, *epsilon* = 0.1, poids de l'entropie = 0.001 [6]. Ces valeurs proviennent d'une implémentation PPO avec deux agents par équipe (attaquant + gardien). Elles constituent un exemple concret de configuration d'entrainement en PyTorch pour un jeu de foot 2D multi-agent [6].

Tutoriel Gym-HFO (TensorFlow)

Un article de blog (Amulya Konda, Medium) sur *gym-soccer (HFO)* donne également des hyperparamètres pour un entraînement simple. On y trouve par exemple `learning_rate = 0.01`, *discount_rate* = 0.95, avec 2 parties simulées par itération et 1000 pas maximum par partie [7] [8]. Ce tutoriel précise aussi la taille du réseau (59 entrées, 10 neurones cachés) et ces paramètres

d'entraînement concrets, illustrant ainsi un autre exemple de projet RL pour football 2D mono-agent [7] [8].

Sources : Documentation et code de Gym-Soccer (OpenAI) [1] [2] ; dépôt GitHub *pddpg-hfo* [3] [4] ; dépôt GitHub *hybrid-sac* (CleanRL) [5] ; projet Unity *Soccer Twos* [6] ; tutoriel Medium *Soccer HFO* [7] [8]. Ces références contiennent les paramètres d'entraînement (γ, lr, batch, etc.) et fonctions de récompense utilisés dans chaque cas.

---

[1] [2] GitHub - openai/gym-soccer
https://github.com/openai/gym-soccer

[3] [4] GitHub - ltzheng/pddpg-hfo: Half Field Offense in Robocup 2D Soccer with reinforcement learning
https://github.com/ltzheng/pddpg-hfo

[5] GitHub - nisheeth-golakiya/hybrid-sac: Single-file pytorch implementation of hybrid-SAC
https://github.com/nisheeth-golakiya/hybrid-sac

[6] GitHub - marcelloaborges/Soccer-PPO: Udacity Deep Reinforcement Learning Nanodegree Program
https://github.com/marcelloaborges/Soccer-PPO

[7] [8] Soccer HFO using Reinforcement Learning | by Amulya Reddy Konda | Medium
https://amulyareddyk97.medium.com/soccer-hfo-using-reinforcement-learning-a9a8aa465810