Apprentissage par Renforcement
Organisation des séances et Travaux Pratiques

Pierre Gérard
pierre.gerard@lipn.univ-paris13.fr

Master MICR
Université de Paris 13

Résumé

Ce cours est la première partie du cours « Apprentissage Numérique ». La seconde partie sera assurée par Younès Bennani, après quoi l'examen final aura lieu. Ce document détaille l'organisation et le contenu de chacune des 5 séances consacrées à l'AR. Les fichiers nécessaires à la réalisation des travaux pratiques sont disponibles à l'adresse `http://www-lipn.univ-paris13.fr/~gerard/cours/micr/`. Tous les travaux pratiques utilisent le framework RL-Glue (`http://rlai.cs.ualberta.ca/RLBB/top.html`.)

1 Problème d'Apprentissage par Renforcement

1.1 Cours

- **Introduction**
- **Formalisation du problème**

1.2 Travaux pratiques

Téléchargez l'archive BlacjackEval.tar.gz et décompressez là sur votre compte. Cette archive contient RL-Glue et un dossier Exercice.[1]

Exercice : *Faites fonctionner l'agent Rd4Agent sur l'environnement Blackjack.*

Exercice : *Modifiez le code de l'agent Rd4Agent pour qu'il puisse évaluer la politique qu'il est en train de suivre :*

- *Faites en sorte qu'au fil des tours d'une partie – cad des pas de temps d'un essai – l'agent mette à jour le cumul des récompenses obtenues (retour) au cours de la partie ;*
- *Faites en sorte que plusieurs parties successives puissent être lancées à la suite (par exemple 100 parties), et calculez la moyenne des cumuls de récompense obtenus au fil des parties.*

Exercice : *En vous inspirant de Rd4Agent, créez deux nouveaux agents*[2] *:*

- *TireAgent qui choisit toujours de tirer une carte ;*
- *StopAgent qui choisit toujours de s'arrêter.*

[1] Dans le dossier Exrecice, on pourra créer un nouveau sous-dossier pour chaque expérimentation (ou exercice). Pour l'instant, le dossier Exercice ne contient qu'un seul dossier d'expérimentation : BlackjackRandom.

Chaque dossier d'expérimentation comporte une paire de fichiers pour le code de l'agent (ici Rd4Agents.*) et une autre pour le code de l'environnement (ici BlackJack.*). Les autres fichiers sont des fichiers du framework ou de configuration permettant de faire automatiquement le lien entre l'agent et l'environnement (config.txt, BlackjackCommon.h). Après paramétrage, on compile en utilisant le script build.bash et on exécute le fichier RL_main produit.

Votre travail consistera la plupart du temps à modifier des fichiers « Agent »de manière à implémenter de nouveaux algorithmes. Vous interviendrez peu sur les autres fichiers. En première approximation, considérez qu'une fois les fichiers de configuration correctement renseignés, le framework RL-glue fait en sorte que les méthodes agent_init, agent_start, agent_step et agent_end (dans Rd4Agents.cpp) sont lancées automatiquement au moment opportun, avec des arguments adéquats.

L'expérimentation proposée dans l'archive fait intervenir un agent Rd4Agent qui choisit ses actions de manière aléatoire pour tenter de résoudre le problème du Blackjack. Le Balckjack est un jeu de cartes dans lequel le joueur doit obtenir un score plus important que la banque, mais sans jamais dépasser une valeur totale de cartes de 21. A chaque tour, il peut choisir de s'arrêter ou de tirer une nouvelle carte, au risque de dépasser 21.

[2] Créez deux nouveaux dossiers BlackjackTire et BlackjackStop – copies de BlackjackRandom – puis modifiez les fichiers Agent de chaque nouveau dossier. Pour réutiliser certains fichiers d'un exercice à l'autre, les liens symboliques sont vos amis.

Exercice : *Vous disposez maintenant de trois agents avec des politiques différentes : utilisez la fonctionnalité d'évaluation que vous avez développé pour déterminer empiriquement quelle politique est la meilleure*

Exercice : *Créez un nouvel agent dont vous écrirez vous-même le comportement de manière à ce qu'il soit meilleur que les trois autres.*

2 Programmation dynamique et AR

2.1 Cours

- **Programmation dynamique**
- **Monte Carlo**

2.2 Travaux pratiques

Téléchargez l'environnement Gridworld et l'agent aléatoire associé RandAgent.

Exercice : *Mettez en place une évaluation de la politique de l'agent avec facteur de dépréciation.*

3 AR par Différence Temporelle et AR indirect

3.1 Cours

- **Différence Temporelle**
- **AR indirect**

3.2 Travaux pratiques

Remplacez l'ancien fichier main.c par le fichier main_stats.c que vous téléchargerez. Ce nouveau fichier permet de profiter de l'affichage de statistiques pour l'évaluation des politiques. Téléchargez aussi l'agent SarsaAgent et le problème Catmouse.

Exercice : *Testez Sarsa sur le problème Gridworld.*

Exercice : *Inspirez-vous de SarsaAgent pour produire un nouvel agent Q-learning de manière à résoudre autrement le même problème.*

Exercice : *Testez Sarsa et Q-learning sur le problème Catmouse.*

4 Unification des méthodes d'AR direct

4.1 Cours

- **Traces d'éligibilité**

4.2 Travaux pratiques

Reprenez l'agent SarsaAgent qui vous avait été fourni au TP précédent.

Exercice : *Créez un agent SLambdaAgent à partir de l'agent SarsaAgent, afin d'implémenter les traces d'éligibilité.*

Exercice : *Testez le nouvel agent sur les problèmes Gridworld et Catmouse en faisant varier λ. Retrouvez ainsi les cas particuliers de TD et de Monte Carlo.*

5 Approximation de fonctions et perspectives

5.1 Cours

- **Approximation de fonctions**
- **POMDP**
- **Continuité du temps**

5.2 Travaux pratiques

Exercice : *Après avoir téléchargé les fichiers nécessaires, testez Tile Coding sur le problème de Mountaincar.*

Exercice : *Modifiez le code source de manière à visualiser les différentes Tiles*

Les exercices suivants nécessitent un petit retour en arrière. Reprenez l'agent Q-learning que vous aviez développé lors de la troisième séance (ou téléchargez la correction). Reprenez aussi les environnements Gridworld et Catmouse.

Exercice : *Inspirez-vous de l'agent QAgent pour créer un nouvel agent DynaQ appliqué aux mêmes problèmes.*

Exercice : *Testez cet agent sur le problème Catmouse. Que pourrait-on faire pour améliorer les résultats de DynaQAgent pour ce problème ?*