Université Pierre et Marie Curie

Cahier des charges

PIAD de Master1 d'Informatique en Intelligence Artificielle et Décision

Semi-supervised Learning Agents

Auteurs :
Lan Zhou
Matthieu Zimmer

Superviseurs :
Paolo Viappiani
Paul Weng

9 avril 2013

Version 1.1

Table des matières

1 Objectif du PIAD

1.1 Présentation

Ce document représente le cahier des charges du PIAD portant sur un apprentissage semi-supervisé d'agents intelligents en partant de l'**apprentissage par renforcement**.

Il s'agit de concevoir et développer un système dans lequel un agent intelligent évoluera (surveillance automatique, ...). Il devra être capable d'apprendre à partir de **retours limités** d'un expert humain du domaine. Imaginons par exemple un agent apprenant à jouer au TETRIS : l'expert pourrait dire à l'agent où placer le prochain élément ou bien si l'agent l'a bien placé. On s'attend alors à ce que cette aide diminue grandement le temps d'apprentissage de l'agent.

1.2 Objectifs principaux

Nous devrons développer une **bibliothèque** indépendante du domaine dans lequel l'agent évolue. Elle fournira plusieurs algorithmes de base de l'apprentissage par renforcement (Q-Learning, SARSA, ...), plusieurs critères de performance, ainsi qu'une ouverture sur l'apprentissage semi-supervisé : c'est à dire avec les retours de l'expert pris en compte.

Dans un premier temps, l'expert pourra simplement compenser la fonction de récompense en précisant si l'agent a bien ou mal agit. Dans un second temps, si le temps le permet, l'expert pourra également agir sur le choix de l'action à entreprendre lors de l'exploration de l'agent, ou encore dire à l'agent s'il est temps d'exploiter ou d'explorer.

Parallèlement au développement de la bibliothèque, afin d'avoir une application pratique de la théorie, on utilisera **TORCS** qui simule des courses de voiture en 3D dans lequelles on intégrera nos propres conducteurs. Nous modifierons également l'interface TORCS pour intégrer les retours positifs ou négatifs de l'expert.

1.3 État de l'art

Apprentissage non supervisé L'apprentissage par renforcement fait référence à une classe de problèmes d'apprentissage automatique, dont le but est d'apprendre, à partir d'expériences, ce qu'il convient de faire en différentes situations, de façon à optimiser une récompense numérique au cours du temps.

L'agent cherche, au travers d'expériences itérées, un comportement décisionnel optimal, en ce sens qu'il maximise la somme des récompenses au cours du temps [Wikipédia, 2012].

Il existe déjà de nombreux algorithmes éprouvés qu'il conviendra simplement d'implémenter [Sutton and Barto, 1998].

Apprentissage semi-supervisé est une classe de techniques d'apprentissage automatique qui utilise un ensemble de données étiquetées et non-étiquetés. Le domaine est déjà assez développé avec plusieurs algorithmes existants [Wikipédia, 2013].

Pour ce qui est de l'apprentissage semi-supervisé par renforcement, il existe déjà aussi quelques recherches dont celles de [Ng et al., 1999],[Ng and Russell, 2000] ou [Abbeel and Ng., 2004].

TORCS : The Open Racing Car Simulator C'est un simulateur de course de voiture multi plate-forme. Il peut être utilisé comme un jeu de voiture ordinaire, ou comme plateforme de recherche en intelligence artificielle car il permet de définir facilement des robots qui conduiront les voitures. Son code source est sous licence GPL [TORCS Team, 2001].

1.4 Contraintes techniques

- Environnement de développement :
  - Plate-forme : PC
  - OS : Linux
  - Langage : C++, C
  - Outils : Doxygen, CodeBlocks, GCC, KDevelop, Cmake, Make
  - Simulateur : TORCS
  - Bibliothèques : Dépendances de TORCS
- Contraintes matérielles requises pour faire fonctionner le système :
  - 1GHz CPU
  - 512MB RAM
  - OpenGL 1.3 compatible graphics card with 64 MB RAM
- Contraintes matérielles : toutes les fonctionnalités du système doivent pouvoir fonctionner sans ralentissement dans un ordinateur de dernière génération :
  - Quad Core CPU
  - 8 GB RAM

2 Description de la solution demandée

De nombreux termes et concepts utilisés dans cette section se réfèrent au document : [Groupe PDMIA, 2008]. Il est necessaire d'être familier avec les bases du **processus de décision markovien** (MDP) et de celles de l'apprentissage par renforcement pour comprendre ce qui suit.

Nous allons maintenant détailler les differentes fonctionnalités de notre solution :

2.1 Fonctionnalités de la bibliothèque

Structures Nous proposerons des structures de données permettant de gérer des actions/états génériques, qu'ils soient continues ou discrétisés.

Algorithmes La bibliothèque sera composée de 2 algorithmes génériques, basés sur le critère de performance γ-pondéré à horizon infini :

- Q-learning
- Sarsa

déclinés sous 3 versions chaque fois :

- tabulaire, sans historique (version la plus simple)
- tabulaire, avec prise en compte de l'historique
- par fonction d'approximation, avec prise en compte de l'historique

Ils devront pouvoir être utilisés dans n'importe quel domaine d'application.

Fonction de récompenses - semi supervisée Il sera possible d'intégrer les retours d'un tuteur dans la fonction de récompense. En ajoutant/retirant une constante ou en multipliant la récompense par un facteur.

Fonction d'approximation Lorsqu'il y a une explosition combinatoire trop importante d'états/actions dans un domaine, la bibliothèque fournira des solutions pour approximer les $Q(s, a)$ en sommant des features (pouvant être binaire) :

$$Q(s, a) = \theta_0 + \sum_{i=1}^{n} \theta_i \times f_i(s, a)$$

Permettant à la fois de réduire l'espace nécessaire et de généraliser l'apprentissage sans passer par tous les couples d'états/actions.

Sauvegarder/Restaurer l'apprentissage Étant donné la nature des algorithmes et la nécessité d'apprendre sur plusieurs épisodes, nous permettrons de sauvegarder et restaurer les paramètres requis permettant le bon déroulement de l'apprentissage.

2.2 Fonctionnalités du simulateur

Robots Les utilisateurs pourront visualiser les robots conduirent au terme ou durant leur apprentissage. Il sera possible de visualiser les 6 différents algorithmes de la bibliothèque.

Mesure de performance Différentes mesures de performance pour les agents seront disponibles tel que : la somme de récompenses additionnées, la durée du temps de sortie de route, dommages perçus par le véhicule, le temps pour effectuer un tour.

Fonction de récompenses Il sera également possible de choisir la fonction de récompense tel que le fait de rester au centre de la route, la distance parcourue, ne pas sortir de la route.

Réglage des paramètres Les paramètres tel que le taux d'apprentissage, le réglage entre exploitation et exploration, le taux de discrétisation des états/actions, le taux d'influence de l'historique seront modifiables.

Interface retour expert Dans le cadre de l'apprentissage semi-supervisé, l'interface de TORCS sera modifiée pour permettre à un tuteur de donner des retours positifs ou négatifs durant l'apprentissage de l'agent. Il pourra presser sur une touche pour dire que l'agent se comporte bien, et une seconde touche pour dire que l'agent n'a pas fait le bon choix. Un retour sera affiché sur la console pour permettre à l'expert de savoir que son action a bien été prise en compte.

2.3 Contraintes de réalisation

- Il est primordial que les algorithmes soient **facilement interchangeables** et **réutilisables** en dehors de TORCS : il doit donc y avoir une nette distinction entre le code de la librairie et l'interfacement avec TORCS.
- Durant la phase de conception et de développement, les documents et le code devront être maintenus en ligne (dans une forge type github) pour mettre aux encadrants de facilement suivre l'évolution du projet.
- Les fonctionnalités citées précédemment constituent le **noyau central** du projet et devront être à réaliser absolument. Les fonctionnalités à venir sont facultatives et leurs réalisations dépendront de l'avancement du projet.

2.4 Fonctionnalités additionnelles de la bibliothèque

Algorithmes Un algorithme de **différence temporelle** pourra être ajouté.

Fonction de récompenses - semi supervisée L'expert pourra intégrer des feedbacks plus complexes tel que la prochaine action à choisir ou le renseigner sur sa stratégie exploration/exploitation.

2.5 Fonctionnalités additionnelles du simulateur

Robots L'algorithme de **différence temporelle** sera également disponible.

Interface retour expert L'interface de TORCS sera modifiée avec toute une partie indépendante pour permettre au tuteur de voir les détails de l'exécution de l'algorithme et intégrer des retours plus complexes (choix d'action, stratégie exploitation/exploration).

Fichier configuration Le robot et l'ensemble de ses paramètres pourront être définis dans un fichier permettant alors de le simuler sans avoir à re-compiler des sources.

Course multijoueur Il sera également possible d'aborder les courses à plusieurs conducteurs. En ce sens, on pourra alors fournir de nouvelles fonctions de récompense et mesures de performance tel que la position dans la course.

3 Composition du délivrable

Dans le cadre de la réalisation du projet, nous nous engageons à livrer les produits suivants :

- La bibliothèque implantant toutes les fonctionnalités prévues
- Le simulateur modifié de TORCS avec :
  - Les robots implémentant les algorithmes d'apprentissage
  - Les modifications de l'interface pour intégrer les retours de l'expert
- Le code source entièrement commentés (hormis celui de TORCS que nous n'aurons pas modifié)
- Un ensemble de script facilitant la compilation et l'installation
- La documentation complète du projet à savoir :
  - Cahier des charges
  - Plan de développement
  - Analyse et conception
  - Manuel d'installation
  - Manuel de l'utilisateur
  - Manuel du programmeur
  - Rapport d'expériences décrivant les résultats complets en fonction des différentes routes, algorithmes et critères
  - Rapport final

Références

[Abbeel and Ng., 2004] Abbeel, P. and Ng., A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Sixteenth International Conference on Machine Learning*.

[Groupe PDMIA, 2008] Groupe PDMIA (2008). *Processus Décisionnels de Markov en intelligence artificielle*. Laboratoire Informatique Paris 6, Equipe MAIA - INRIA (LORIA).

[Ng et al., 1999] Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations : Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*.

[Ng and Russell, 2000] Ng, A. Y. and Russell, S. (2000). Algorithms for inverse reinforcement learning. In *Proceedings of the Sixteenth International Conference on Machine Learning*.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning : An Introduction*. MIT Press, Cambridge, MA.

[TORCS Team, 2001] TORCS Team (2001). The open racing car simulator. `http://torcs.sourceforge.net/`.

[Wikipédia, 2012] Wikipédia (2012). Apprentissage par renforcement. `http://fr.wikipedia.org/wiki/Apprentissage_par_renforcement`.

[Wikipédia, 2013] Wikipédia (2013). Semi-supervised learning. `http://en.wikipedia.org/wiki/Semi-supervised_learning`.