Sorbonne Université

2I013 : Projet (application)

Sujet :

IA Football

Fangzhou Ye
Angelo Ortiz

Licence d'Informatique
Année 2017/2018

Table des matières

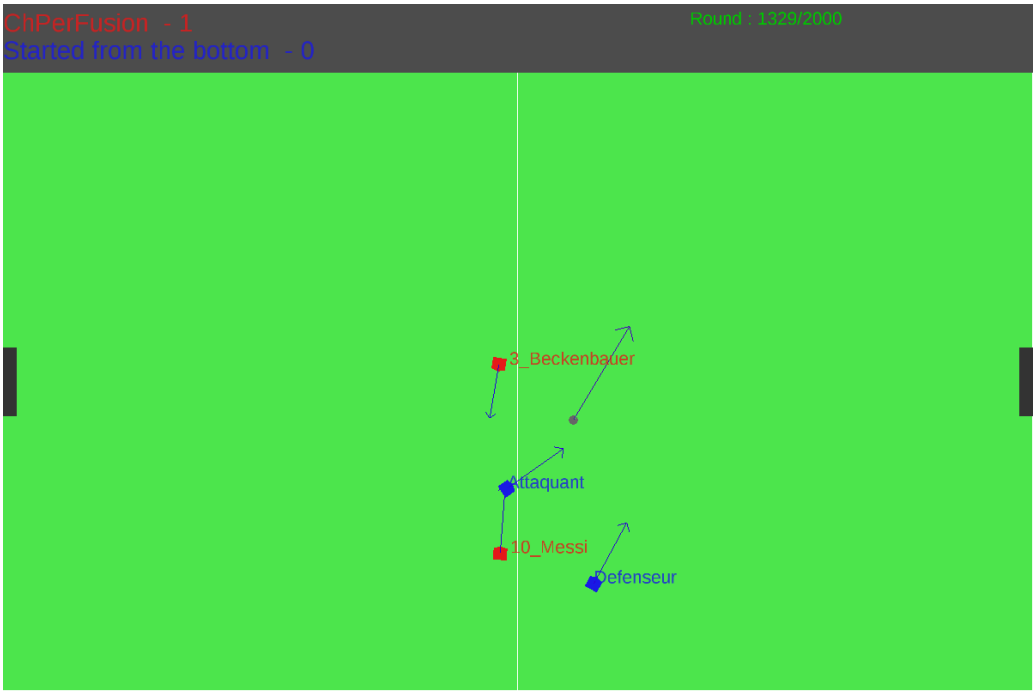
Introduction

Présentation

Le projet a consisté en le développement d'intelligences artificielles de joueurs de football. Le simulateur du jeu étant déjà fourni, notre travail a été d'implémenter des stratégies de jeu pour mieux réussir les matches. L'objectif de ce travail a été d'apprendre à bien mener un projet collaboratif long sur un nouvel environnement de développement, à savoir Python.

Aperçu

Voici un aperçu du simulateur de football lors d'une partie de notre équipe *ChPerFusion* en rouge face à l'équipe d'un de nos camarades de classe.



Arborescence

Le projet a été organisé en un module **ia** dirigeant le comportement de nos joueurs et en divers fichiers de test.

ia/tools

Ce fichier contient une classe représentant l'état du jeu qui comporte des méthodes nous facilitant l'accès aux diverses caratéristiques de cet état, ainsi que plusieurs fonctions utilitaires.

ia/conditions

Comme le nom l'indique, ce fichier a été dédié aux diverses conditions qui déclenchent les différents types de comportement.

ia/actions

Ce fichier comporte les actions de base que l'on peut réaliser à chaque instant du jeu.

ia/behaviour

En ce qui concerne ce fichier, ici se trouvent les actions complexes, i.e. composées de plusieurs actions de base. En effet, pour une certaine phase de jeu, un joueur peut prendre différentes décisions.

ia/strategies

C'est dans ce fichier que les stratégies ont été réunies. Ces stratégies dirigent le comportement des joueurs tout au long d'un match.

ia/gene_optimisation

Ce fichier contient les classes nécessaires pour la mise en œuvre d'une implémentation des algorithmes génétiques.

ia/gs_optimisation

On retrouve ici le code utilisé pour suivre la méthode de recherche en grille.

ia/

init ? ML_strats ? DT_strats ?

Il reste encore des fichiers

Première partie

Architecture logicielle

Dans cette première partie, on vous expliquera les choix faits concernant l'architecture logicielle.

Deuxième partie

Stratégies

Dans cette deuxième partie, on vous détaille le comportement de nos différents joueurs.

Compte tenu de la taille du terrain de jeu, le nombre maximum de joueurs par équipe est limité à quatre. Ainsi, il y a trois catégories de matches selon le nombre de joueurs par équipe : un, deux ou quatre.

1 Un joueur

Pour l'équipe composée uniquement d'un joueur, on a développé un seul joueur que l'on appelle *Fonceur*. Au tout début du projet, il faisait tout au maximum : il doit s'approcher de la balle à toute vitesse lorsqu'il ne la contrôle pas et il frappe avec toute sa puissance dans le cas contraire. On a décidé de garder le même nom, même si son comportement diffère. En effet, à présent il avance avec la balle de petites distances et lorsqu'un adversaire lui fait opposition il essaie de le dribbler ; il frappe vers la cage adverse s'il se trouve dans la surface de réparation.

2 Deux joueurs

Cette catégorie a été celle sur laquelle on a consacré la plupart du projet. On compte trois joueurs fonctionnels que l'on appelle *Attaquant*, *Gardien* et *Défenseur*.

Attaquant

Il a une vocation plutôt offensive. Il se comporte différemment selon s'il contrôle la balle ou pas. Dans la première situation, il essaie systématiquement d'avancer sur le terrain avec la balle et de faire un tir vers la cage dès qu'il se trouve dans la surface de réparation adverse. Cependant, s'il rencontre un adversaire lui bloquant le chemin, il essaie de faire une passe vers son coéquipier s'il est sans marquage, il dribble le joueur sinon. Lorsqu'il se trouve sans le contrôle de la balle ...

Gardien

Ce joueur se comporte exactement pareil que l'attaquant lorsqu'il s'agit d'attaquer. En revanche, en situation défensive, il reste à une distance de sa cage lorsque l'équipe adverse contrôle la balle et essaie d'intercepter la balle lorsque le porteur du ballon se trouve dans sa surface de réparation. S'il voit que son coéquipier va bientôt contrôler la balle ou il en a déjà le contrôle, il monte dans le terrain pour lui proposer une possibilité de passe.

Défenseur

Le défenseur n'a par contre qu'une vocation défensive. En effet, s'il a la balle, il s'en défait le plus rapidement possible soit à l'aide d'une passe vers son coéquipier si démarqué, soit via le dégagement le moins risqué possible. Quant à son effort défensif, il demeure à une certaine distance radial à partir de sa cage et s'approche de la balle en vu d'une interception s'il en suffisamment proche.

3 Quatre joueurs

Il s'agit de la catégorie que l'on a trouvé la plus intéressante en raison de la difficulté de la gestion des espaces et la répartition des fonctions. Tout comme pour la catégorie précedente, on compte trois joueurs fonctionnels que l'on repère d'ailleurs par les mêmes noms.

Attaquant

Il y a deux différences en son comportement par rapport à son pair de la catégorie ci-dessus et celles-ci concernent toutes les deux la situation où il se trouve dans sa surface de réparation. La première fait référence à la récupération de la balle : il ne se précipite plus et se comporte tel qu'il le fait dans le milieu du terrain. La deuxième correspond à l'effort de marquer les adversaires sans marquage.

Gardien

Ce joueur combine le comportement de son pair de la catégorie ci-dessus et de l'attaquant de cette catégorie. En effet, il suit les mêmes indications que l'attaquant lorsqu'il contrôle et réalise les mêmes mouvements sans ballon que le défenseur de l'équipe à deux joueurs.

Défenseur

Ce défenseur prend les mêmes décisions que celui de l'équipe à deux joueurs, à l'exception de sa prise de risque lors d'une interception. Effectivement, si un coéquipier se trouve dans une meilleure position pour intercepter la balle, i.e. il en est plus proche que le défenseur, celui-ci continue de positionner radialement à une certaine distance de sa cage.

Troisième partie

Méthodes d'optimisation

Dans cette troisième partie, on vous détaille les différents algorithmes appris tout au long du semestre pour l'amélioration des stratégies initialement proposées.

4 Recherche en grille

La première méthode d'optimisation preésentée en cours a été la recherche en grille. Celle-ci est utilisée pour l'optimisation d'une action.

Tout d'abord, on doit définir l'action à optimiser et le critère d'optimalité, et repérer les paramètres associés. Puis on obtient un ensemble de valeurs pour chaque paramètre. Pour les paramètres discrets, il suffit de prendre toutes les valeurs possibles, alors que pour les paramètres continus il faudra faire en amont une discrétisation, i.e. on divise l'intervalle délimité par des bornes définies selon un pas de précision. On compte ainsi des vecteurs dont les coordonnées correspondent chacune à un paramètre. On fait ensuite une recherche exhaustive sur tous les vecteurs obtenus : on teste chaque vecteur sous différents conditions environnementales et on moyenne les évaluations du critère dans lesdites conditions. Finalement, on prend la valeur optimale et son vecteur associé.

Il est important de remarquer que la plupart de nos paramètres sont continus. Pour obtenir des valeurs plus précises et un meilleur comportement, on a besoin d'un pas de discrétisation très petit. Ainsi, on réalise qu'une augmentation du nombre de paramètres à optimiser a une très forte influence sur le temps d'exécution de l'algorithme mis en place. On en déduit que le minuscule gain relatif nombre de paramètres–temps d'exécution ne convient pas au moyen terme.

5 Algorithmes génétiques

Dans un deuxième temps, on est passés aux algorithmes génétiques. On en a mis en place une implémentation pour optimiser les équipes composées d'un ou deux joueurs.

Cette classe d'algorithmes se base sur la théorie de l'évolution. En effet, la notion de sélection naturelle est le mécanisme permettant de choisir des solutions potentielles de plus en plus meilleures.

La toute première chose à faire est la définition de la fonction $f : \mathbb{R}^n \to \mathbb{E}$ à optimiser, où n est le nombre de paramètres concernés. Un élément $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ est appelé *candidat* ou *chromosome*, et chacune de ses coordonnées x_i avec $i = 1, \ldots, n$, appelée *propriété* ou encore *gène*, correspond à un paramètre à optimiser. Elles forment le *génotype*. Autrement dit, un candidat correspond à une équipe définie par les valeurs de ses propriétés.

Dans le cadre du projet, on a défini $f(x) = (V, N, D, P, C)$ comme le bilan après avoir disputé plusieurs matches avec les mêmes valeurs des paramètres données par x, i.e. le tuple composé du nombre de victoires, matches nuls, défaites, buts marqués et buts encaissés. La relation d'ordre suit les règles du classement d'un tournoi de football.

Cette classe d'algorithmes comportent quatre étapes.

1. Initialisation : Tout d'abord, il faut générer un ensemble de candidats $\{x_1, \ldots, x_m\}$. Le but est d'obtenir une population initiale la plus diverse possible de sorte à pouvoir atteindre le plus de maxima relatifs. C'est pourquoi cette première génération est obtenue de manière aléatoire.

2. Évaluation : Ensuite, on évalue la fonction en chacun des candidats, i.e. on calcule $\{f(x_1), \ldots, f(x_m)\}$.
3. Sélection : Suivant la notion de sélection naturelle, on ne conserve que les candidats avec les meilleurs résultats. Autrement dit, on dresse un classement des équipes associées aux candidats et n'en garde qu'une partie.
4. Reproduction : Finalement, on attribue les places libres aux candidats nés du brassage génétique des candidats les plus performants. Pour ce faire, on compte sur deux méthodes, à savoir le croisement et la mutation. Étant donné deux parents, on commence par les cloner en deux enfants. Puis on choisit une ligne de coupe divisant le génotype en deux parties. Ensuite, les enfants s'échangent une partie de leur génotype. Le croisement s'arrête à ce stade-là, tandis que la mutation ajoute du bruit aléatoirement dans un gène pour chacun des nouveaux chromosomes. On a décidé de faire une petite modification à cette étape-là lors de l'implémentation pour avoir plus de diversification : une minorité des places libres est attrribuée à des candidats générés aléatoirement.

Cet algorithme étant itératif, on recommence à la deuxième phase avec la nouvelle génération. Au bout d'un nombre raisonnable d'itérations, on se retrouve avec une génération composée de chromosomes très proches des maxima relatifs. On finit donc l'algorthme par prendre le candidat optimal de la dernière population.

6 Apprentissage automatique

Dans la dernière partie du semestre, on s'est concentré sur l'apprentissage automatique. On nous a présenté les deux grandes familles d'algorithmes d'apprentissage : supervisés et non supervisés, ainsi que l'apprentissage par renforcement. Deux méthodes ont été implémentées lors du projet.

Arbres de décision

Il s'agit d'une technique d'apprentissage supervisé.

Q-learning

C'est un algorithme d'apprentissage par renforcement. Le principe est le suivant : on apprend une *politique* comportementale qui optimise l'espérance des *récompenses*. Cette politique est tout simplement une fonction qui associe une action à entreprendre à un état de jeu donné.

Cet algorithme répose sur un ensemble d'états S, un ensemble d'actions A, et une fonction $Q : S \times A \to \mathbb{R}$, appelée politique, calculant la valeur d'une paire état-action. Il utilise la notion d'*épisode* que l'on associe à un match dans le cadre du projet.

La première phase consiste à initialiser Q à une valeur fixe arbitraire, ici 0. La deuxième phase concerne un épisode, ou encore un match, et est répétée autant de fois qu'il y a d'épisodes. On commence par choisir un état initial s_0, sauf qu'il est toujours le même pour un match de football : ceci correspond au coup d'envoi. Puis on choisit une action a_t selon la politique Q et observe la récompense associée à cette paire. Cette action génère un nouvel état s_{t+1}. Formule de Q

Conclusion

...