Rapport Morpion

Projet Etude de Cas
Matlab Avancé

M. Chaussard

2016

AUGER Romain	BERRICH Maxime	LAGUBEAU Gabriel
RIO Charles	YILMAZ Adem	
INFO1	ENER1	MACS1

Ingénieurs 1ère année, Institut Sup Galilée

Sommaire

Introduction

Comment Jouer

Lancer le programme
Utiliser l'interface

Implémentation

Représentation de l'IA
Comment joue l'IA
Gestion de l'apprentissage

Commentaires

Conclusion

Introduction

Le projet consiste à crée un jeu de Morpion en Matlab. Ce jeu proposera trois types de parties : 'Joueur vs Joueur', 'Joueur vs IA', 'IA vs IA'.
Il faudra donc créer une IA et surtout, cette IA devra non pas maîtriser le jeu, mais le découvrir et apprendre au fur et à mesure des parties jouées.

Nous allons tout d'abord expliquer comment jouer, comment manipuler l'interface que nous avons conçu. Ensuite nous expliquerons nos choix et méthodes d'implémentation concernant l'IA, ses représentation, façon de jouer, et méthode d'apprentissage.
Pour plus de détails sur le déroulement du jeu, vous pouvez consulter le code dans le fichier 'Morpion.m', celui-ci est entièrement commenté.
Enfin, nous ferons nos commentaires : des observations et critiques sur le projet que nous avons réalisé avant de conclure.

Comment Jouer

Lancer le programme

Tout d'abord, il vous faut extraire les fichiers 'Morpion.m' et 'Morpion.fig' de l'archive .zip.
Lancer alors Matlab dans le dossier les contenant et taper simplement la commande 'Morpion'.
(Nous avons joint le fichier 'save.mat' afin que vous puissiez tester l'IA entraînée sans avoir à lui faire faire des milliers de parties.)

Utiliser l'interface

Une fenêtre s'ouvre, elle contient le bouton de lancement du jeu avec l'intitulé 'Jouer', ainsi qu'un menu déroulant vous permettant de choisir le mode de jeu. Vous pouvez aussi voir une zone de texte pour l'instant vide au bas de la fenêtre.
Le mode de jeu est initialement réglé sur 'Joueur vs Joueur'. Si vous choisissez le mode 'Joueur vs IA', un second menu déroulant apparaît afin de choisir l'ordre de jeu. Si vous choisissez le mode 'IA vs IA', une zone de texte apparaît dans laquelle vous pouvez entrer le nombre de parties que l'IA jouera d'une traite au lancement de la partie.
Lorsque vous avez choisi le mode et les options, appuyez sur 'Jouer' et le plateau apparaîtra. Vous pouvez à tout moment modifier ces options et relancer une partie. Pour jouer, cliquer simplement sur une case (libre) de votre choix.
Lorsque la partie est terminé, la zone de texte que nous avons observée au bas de la fenêtre affiche le résultat.

Comme vous pourrez le constater, notre interface est extrêmement intuitive.
Après la première utilisation du jeu, un fichier 'save.mat' sera créé dans le dossier, il contient l'apprentissage de l'IA, si vous le supprimez, celle-ci sera réinitialisée, sinon elle continuera à évoluer.

Implémentation

Vous trouverez dans le fichier 'Morpion.m' le code du jeu entièrement commenté.

Représentation de l'IA

Nous représentons l'IA par une matrice (19683, 9) soit (3^9, 9).
Chaque ligne représente un état du plateau y incluant les états irréalisables dans une partie, tel que le plateau rempli uniquement de X. Cela n'est donc pas optimal en terme d'utilisation de la mémoire, cependant cette matrice occupera moins de 3Mo et ce uniquement pendant l'utilisation du programme. Le fichier de sauvegarde est lui inférieur à 100ko, même après plusieurs milliers de parties rendant la compression moins efficace ; à son initialisation il est inférieur au ko.
Pour associer un état du plateau à son indice, on verra le plateau comme un nombre écrit en base 3 dont les cases sont les chiffres avec X=2, O=1, vide=0. Le plateau est numéroté de gauche à droite et de haut en bas (coin supérieur gauche = case 1, coin inférieur droit = case 9).

Les colonnes associées à chaque ligne représente la valeur de chacune des 9 cases du plateau dans l'état considéré.

Comment joue l'IA

L'IA jouera différemment selon le mode de jeu choisi, nous parlerons ici de son mode de jeu standard utilisé lors des parties 'Joueur vs IA'.
Lorsque l'IA doit jouer, elle récupère dans sa matrice la ligne correspondant à l'état actuel du plateau. Appelons v ce vecteur ligne.
Elle recopie ensuite dans un vecteur les indices des cases autant de fois que la valeur indiquée à l'indice correspondant. Appelons w ce nouveau vecteur.
Ex : v = [2, 0, 4] => w = [1, 1, 3, 3, 3, 3]
On va ensuite choisir un indice i aléatoire possible dans le vecteur w et jouer sur la case w(i).
Ainsi plus la valeur de la case sera élevée dans v, plus on aura de chance de jouer dessus car elle apparaîtra d'autant plus dans w.
Soit P(x) la probabilité de jouer sur la case x.
Dans notre exemple : P(1)= 2/6 = 1/3
P(2) = 0
P(3) = 4/6 = 2/3

Gestion de l'apprentissage

Pour apprendre à jouer à notre IA, nous conservons un historique des coups joués dans une matrice. Chaque ligne représente un coup et les colonnes sont les informations nécessaires pour effectuer l'apprentissage.
Sur les colonnes on aura : case jouée, état du plateau et tour de jeu au moment du coup.
La case et l'état du plateau nous donnent les coordonnées dans la matrice de l'IA auxquelles il faut accéder.
Le tour de jeu nous servira à savoir quel joueur à joué ce coup et donc à savoir s'il faut le récompenser ou le punir.
Lorsque la partie prend fin, on parcourt cette matrice d'historique des coups et on applique à chacun une modification de sa valeur dans la matrice de l'IA.

Nous avons choisi de modifier les coefficients donnés dans l'énoncé afin d'améliorer les performances de l'IA. Ainsi nous initialisons la totalité des coefficients à 20 et non 10 et les coups perdants subiront un malus de -2 et non -1.

Cette méthode est appliquée quel que soit le type de partie, ainsi l'IA profite même des parties entre joueurs. Cependant, pour faire progresser l'IA, on utilisera le mode 'IA vs IA' afin de lui faire faire un grand nombre de parties.

Nous avons remarqué que si dans ce cas nous la faisions jouer uniquement de la façon décrite précédemment, l'IA n'apprenait que très peu car elle ne maîtrisait qu'un type de partie. En effet cette méthode oriente très rapidement sa façon de jouer et elle va finir par rejouer la même partie. Elle n'acquerra alors aucune expérience sur les parties qui ne correspondent pas aux « meilleures coups » pour elle. Nous avons alors choisi de modifier la fonction de jeu de l'IA dans le cas des parties 'IA vs IA'. Elle va maintenant alterner entre jeu intelligent, qui tient compte de l'apprentissage, et jeu bête, qui joue aléatoirement sur les cases disponibles. Cela lui permet de rencontrer infiniment plus de situations et donc de développer son apprentissage.

Commentaires

Nous avons pu faire plusieurs observations concernant l'IA.

Premièrement, son niveau est bien meilleur lorsqu'elle commence. Elle parviendra toujours à l'égalité et, si on lui laisse une chance, elle gagnera.
En revanche, si le Joueur commence, l'IA est bien plus faible.
Si l'on suit le schéma de partie qu'elle maîtrise en tant qu'attaquant, elle réussira à nous contrer.
Sinon, elle jouera quelques coups intelligents mais fera ensuite un coup inadapté, ne respectant pas les priorités essentielles du jeu (gagner avant de contrer, contrer avant de continuer).

Nous avons plusieurs explications et pistes de réflexion pour améliorer notre IA.

Tout d'abord, lorsque la partie est terminée, on récompense tous les coups gagnant et on punit tous les coups perdants. Pour les coups gagnants, ce système n'est pas problématique, cependant, punir l'ensemble des coups du perdant n'est pas adapté. En effet, l'IA peut avoir joué plusieurs coups adaptés au début du jeu puis jouer un coup qui la fera perdre, seul ce coup devrait donc être puni.
Nous avons alors émis l'idée d'ajouter un coefficient qui punirait d'autant les coups perdants qu'ils sont joués tard dans la partie.
Mais ce système ne rendrait pas non plus justice au perdant puisque l'issue peut être décidée dès le premier coup du défenseur s'il n'est pas adapté, alors que les coups suivants peuvent être joués au mieux des possibilités qui lui restent.
Nous pensons donc que ce système, même s'il améliorerait notre IA à ce niveau, n'est pas adapté.

Nous pourrions aussi revoir notre système d'apprentissage pour les parties 'IA vs IA'. Pour ces parties nous faisons jouer une partie sur deux par une IA avec l'apprentissage ou aléatoire. Cette alternance est essentielle pour le joueur 1 pour rencontrer différentes situations et ne pas apprendre une seule série de coups. Cependant, l'intérêt de cette alternance pour le joueur 2 est contestable. En effet, le défenseur devrait toujours être en train d'apprendre à réagir aux coups de l'attaquant, ce n'est pas lui qui mène le jeu, il semble inutile qu'il joue aléatoirement une partie sur deux.

Enfin, il serait préférable d'initialiser la matrice IA à une valeur plus élevée dès le début, cela combiné avec d'autres méthodes d'apprentissage et modifications des coefficients.

Conclusion

Nous avons réussi à créer un jeu de Morpion avec une interface intuitive permettant toutes les fonctionnalités demandées. Nous avons pu implémenter les fonctions qui encadrent le jeu et surtout nous avons pu créer une IA qui apprenait à jouer sans avoir aucune expérience au début.

L'important à retenir de ce projet est tout de même la difficulté de créer une IA. En effet, même si nous avons pu lui apprendre à jouer et clairement améliorer son niveau par apport à tout aléatoire, il apparaît que les méthodes utilisées ne sont pas suffisantes même pour un jeu aussi simple que Morpion. De nombreux paramètres entrent en jeu et imposent de complexifier l'apprentissage.

Sur un autre plan, ce projet nous a permis de nous améliorer en matlab et de bien prendre en main le guide. Celui-ci est vraiment facile d'utilisation pour créer nos interfaces.