MATH60623 Prise de décision séquentielle sous incertitude
Prof. Maryam Daryalal
Devoir No. 3
À rendre pour 21 Avril 2024, 23h55
Soumission: Soumettez vos réponses dans un seul fichier **.pdf** sur ZoneCours. Soumettez vos codes séparément.

Indiquez clairement les noms des membres de l'équipe et leurs numéros d'étudiant ci-dessous:

Nom	Matricule
1.	
2.	
3.	
4.	
5.	

Présentation, équipes et date limite:

- La présentation générale des devoirs est importante et doit être soignée. Vous pouvez taper ou écrire à la main vos devoirs tant qu'ils sont lisibles. Les problèmes doivent être clairement séparés, et vos réponses doivent être identifiées. Des devoirs avec une présentation inadéquate peuvent être pénalisés ou rejetés.
- Les devoirs doivent être réalisés en équipes de pas plus de cinq étudiants.
- Tous les devoirs doivent être remis pour la date et l'heure indiquées. Aucun retard ne sera accepté.

---

1 Objectif

L'objectif principal de ce devoir est de fournir une expérience pratique avec les algorithmes d'apprentissage par renforcement, en particulier le Q-learning et le Q-learning profond. Vous accomplirez cela en concevant, implémentant et comparant ces deux algorithmes sur une application réelle en utilisant le robot éducatif Codey Rocky. Vous rédigerez ensuite un rapport détaillant votre définition du problème, votre approche pour le résoudre, votre code et vos résultats. À la fin de ce devoir, vous devriez avoir une solide compréhension de la manière d'appliquer le Q-learning et le Q-learning profond pour résoudre des problèmes, ainsi que les différences entre ces deux approches. Vous acquerrez également des compétences pratiques en programmation, résolution de problèmes, rédaction technique et présentation.

2 Introduction à Codey Rocky

Codey Rocky est un robot éducatif développé par Makeblock conçu pour enseigner la programmation et la robotique aux enfants. Il combine le matériel avec le logiciel, permettant aux enfants d'apprendre la programmation par le jeu et la créativité. Il se compose de deux éléments:

- Codey: Un contrôleur avec plus de 10 modules programmables, Codey peut être utilisé indépendamment. Il peut également agir comme le « cerveau » de Rocky, fournissant des commandes pour se déplacer ou jouer.
- Rocky: Une voiture qui peut se déplacer sur diverses surfaces pour différents scénarios. Rocky suit les ordres de Codey et se déplace librement au sol.

Codey Rocky est compatible avec le logiciel de programmation graphique de Makeblock, `mBlock 5`. Ce logiciel est basé sur Scratch 3.0, offrant une interface de programmation par glisser-déposer qui est conviviale pour les débutants. Codey Rocky prend également en charge la programmation Python pour une programmation plus avancée.

2.1 Pour commencer

- Téléchargez la dernière version de l'application `mBlock PC version` depuis le site web mblock.
- Obtenez le fichier `RL-demo.mblock` depuis ZoneCours. Il comprend les parties de la mise en œuvre du Q-learning que vous pouvez compléter.
- Regardez le video `Demo-ProgrammingCodey` que j'ai posté sur ZoneCours, où je détaille le fichier `RL-demo.mblock`.
- Regardez le video `Codey-WiFi-Demo` que j'ai posté sur ZoneCours, où j'explique comment vous pouvez imprimer les informations de votre Codey.

Attention: Pour mBlock sur Windows:

- Installez l'application sur Windows en tant qu'administrateur en cliquant avec le bouton droit sur l'icône d'installation de l'application et en sélectionnant `Exécuter en tant qu'administrateur`.
- Si vous travaillez avec Mac, vous pouvez connecter votre Codey via Bluetooth directement. Si vous êtes sous Windows, vous devriez le connecter via un câble USB.

Packages externes: Dans les contextes pratiques, il est courant d'utiliser des packages externes de Python pour écrire notre code, et cela est parfaitement acceptable. Mais l'objectif de ce projet est également d'approfondir votre compréhension des algorithmes sous-jacents, plutôt que de les utiliser simplement comme des outils boîte noire. De plus, de nombreux robots éducatifs ne prennent pas en

charge les packages tiers. Dans notre cas, Codey prend en charge les bibliothèques disponibles dans MicroPython. Pour pallier à cela, dans la page de l'Assignment 3, j'ai publié une implémentation complète d'un simple DQN et d'un algorithme du Q-learning qui résout un exemple jouet avec 3 états et 2 actions. Vous pouvez utiliser ces implémentations au lieu de packages tiers. N'hésitez pas à y apporter des modifications/améliorations selon ce que vous jugez approprié. (Ces implémentations sont celles que j'ai utilisées pour mes propres tests avec Codey afin d'éviter les obstacles. J'ai juste changé le fichier pour un exemple numérique simple.)

2.2 Ressources d'apprentissage

Pour vous aider à commencer avec Codey Rocky, voici quelques ressources:

- Makeblock Education: Le centre de ressources officiel de Makeblock Education offre une variété de guides et de ressources pour utiliser Codey Rocky.
- Chaîne YouTube Officielle de Makeblock: Vous y trouverez de nombreux tutoriels vidéo qui peuvent vous aider à démarrer avec Codey Rocky.
- Commencer avec Codey Rocky: Un guide de démarrage rapide de Makeblock, qui fournit une introduction à Codey Rocky.
- Utiliser mBlock 5 pour programmer Codey Rocky: Un guide fourni par Makeblock, décrivant comment utiliser mBlock 5 pour programmer Codey Rocky.
- Documentation de l'API Python Codey Rocky: Cette documentation fournit des informations détaillées sur la manière de contrôler Codey Rocky en utilisant Python.

3 Description du devoir

3.1 Étape 1: Définition du problème

Dans ce devoir, vous avez pour mission de définir un problème stimulant et captivant que Codey Rocky devra résoudre en utilisant des techniques d'apprentissage par renforcement. Le problème devra impliquer une forme de navigation et de prise de décision basée sur les entrées des capteurs. Voici quelques lignes directrices et considérations pour vous aider à définir votre problème:

Environnement Considérez l'environnement dans lequel Codey Rocky va opérer. Cela pourrait être un labyrinthe physique, une pièce avec des obstacles ou une scène construite sur mesure avec des repères spécifiques. L'environnement devrait être suffisamment complexe pour représenter un défi, mais suffisamment gérable pour qu'une solution soit réalisable. N'oubliez pas que votre environnement définira l'espace d'état que les algorithmes d'apprentissage par renforcement devront naviguer.

Tâches Définissez les tâches que Codey Rocky doit effectuer pour résoudre le problème. Cela pourrait impliquer de se rendre à un endroit spécifique ou d'éviter certaines zones. Soyez clair et précis sur ce qui constitue l'achèvement réussi de la tâche.

Entrées des capteurs Considérez quelles entrées de capteurs Codey Rocky utilisera pour comprendre son environnement et prendre des décisions. Codey Rocky est équipé de capteurs pour la détection de la lumière, du son et de la couleur, ainsi que d'un accéléromètre, d'un gyroscope et d'un récepteur/émetteur infrarouge. Vous pouvez choisir d'utiliser certains de ces capteurs dans le cadre de votre problème.

Espace d'action Pensez aux actions que Codey Rocky peut entreprendre en réponse à ses entrées de capteurs. Cela pourrait inclure avancer, tourner, s'arrêter ou d'autres actions en fonction de votre définition du problème. L'ensemble des actions possibles constituera l'espace d'action pour les algorithmes d'apprentissage par renforcement.

Récompenses Définissez un système de récompenses pour les tâches. Le système de récompenses est crucial en apprentissage par renforcement car il guide le processus d'apprentissage. Vous devriez concevoir un système de récompenses qui encourage Codey Rocky à effectuer les tâches souhaitées et à éviter les résultats indésirables. Cela pourrait inclure des récompenses positives pour l'achèvement des tâches ou des récompenses négatives (pénalités) pour heurter des obstacles ou dévier de la piste.

Contraintes Enfin, considérez toutes contraintes qui pourraient affecter la solution. Les contraintes pourraient être liées à l'environnement (par exemple, taille, espace de navigation disponible), aux tâches (par exemple, limites de temps, ordre des tâches) ou aux capacités de Codey Rocky lui-même (par exemple, vitesse, portée des capteurs).

Après avoir pris en compte tous ces aspects, rédigez une description claire et détaillée de votre problème. Cette description devrait inclure l'environnement, les tâches, les entrées des capteurs, l'espace d'action, le système de récompenses et toutes les contraintes. Cette définition du problème formera la base de votre mise en œuvre des algorithmes du Q-learning et du Q-learning profond.

Exemples de problèmes

Navigation dans un labyrinthe

Dans ce problème, Codey Rocky est placé dans un labyrinthe et doit trouver son chemin vers la sortie.

Pour commencer: Commencez par concevoir et construire un labyrinthe physique. L'espace d'état pourrait être défini comme les différentes positions que Codey Rocky pourrait occuper dans le labyrinthe, tandis que l'espace d'action serait les mouvements possibles que Codey Rocky pourrait faire (par exemple, avancer, tourner à gauche, tourner à droite). Le système de récompenses pourrait inclure une récompense positive pour atteindre la sortie et une récompense négative pour avoir heurté un mur.

Évitement d'obstacles

Dans ce problème, Codey Rocky doit naviguer à travers une pièce remplie d'obstacles et atteindre un emplacement spécifié.

Pour commencer: Installez une pièce avec divers obstacles. L'espace d'état pourrait être défini comme les lectures des capteurs de Codey Rocky (par exemple, la distance à l'objet le plus proche), tandis que l'espace d'action serait les mouvements possibles que Codey Rocky pourrait réaliser. Le système de récompenses pourrait inclure une récompense positive pour avoir atteint l'emplacement cible et une récompense négative pour une collision avec un obstacle.

Suivi de lumière

Dans ce problème, Codey Rocky doit suivre une source lumineuse à travers une pièce.

Pour commencer: Installez une pièce avec une source de lumière mobile. L'espace d'état pourrait être défini comme les lectures du capteur de lumière de Codey Rocky et sa localisation, tandis que l'espace d'action serait les mouvements possibles que Codey Rocky pourrait effectuer. Le système de récompenses pourrait inclure une récompense positive pour se rapprocher de la source lumineuse et une récompense négative pour s'en éloigner.

Rappelez-vous, ce ne sont que des exemples. N'hésitez pas à créer votre propre problème qui respecte les lignes directrices et représente un défi pour les algorithmes d'apprentissage par renforcement.

3.2 Étape 2: Implémentation du Q-learning

Implémentez un algorithme du Q-learning pour résoudre le problème que vous avez défini. Votre implémentation devrait inclure:

- Une représentation appropriée de l'espace d'état et de l'espace d'action
- Une méthode permettant à Codey Rocky d'interagir avec l'environnement et de recevoir des récompenses
- Une méthode de mise à jour de la table-Q en fonction des récompenses reçues

Voici des étapes détaillées pour vous guider tout au long de ce processus:

Définir les espaces d'état et d'action La première étape de la mise en œuvre du Q-learning consiste à définir les espaces d'état et d'action. L'espace d'état représente l'ensemble de tous les états possibles dans lesquels Codey Rocky peut exister, tandis que l'espace d'action représente toutes les actions possibles que Codey Rocky peut effectuer. Par exemple, si votre problème implique de naviguer sur une grille de 4x4, votre espace d'état pourrait être les 16 cellules de la grille, et votre espace d'action pourrait être les quatre mouvements possibles (haut, bas, gauche, droite).

Initialiser la table-Q Ensuite, initialisez une table-Q. Il s'agit d'une table où chaque ligne représente un état, chaque colonne représente une action et chaque cellule représente la récompense future attendue pour cette action dans cet état. Au début, cette table est généralement initialisée à zéro.

Interagir avec l'environnement À présent, Codey Rocky peut commencer à interagir avec l'environnement. Il commence dans un état initial, choisit une action basée sur la table-Q actuelle, exécute l'action, puis reçoit une récompense et se retrouve dans un nouvel état.

Mettre à jour la table-Q Après chaque interaction, mettez à jour la table-Q en utilisant la récompense reçue et la récompense future maximale attendue pour le nouvel état. La règle de mise à jour du Q-learning est la somme pondérée de l'ancienne valeur-Q et de la valeur-Q *temporaire*:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha * \left(r + \gamma * \max_{a'}\{Q(s',a')\}\right)$$

où:

- s est l'état actuel

— a est l'action effectuée
— r est la récompense reçue
— s' est le nouvel état
— a' est l'action qui serait choisie dans le nouvel état en fonction de la table-Q actuelle
— α est le taux d'apprentissage (une valeur entre 0 et 1 qui détermine la quantité de nouvelles informations que nous intégrons)
— γ est le facteur de remise (une valeur entre 0 et 1 qui détermine l'importance des récompenses futures)

Continuez à laisser Codey Rocky interagir avec l'environnement et mettre à jour la table-Q jusqu'à ce que la table-Q converge (c'est-à-dire que les valeurs ne changent plus significativement d'une interaction à l'autre) ou qu'un nombre maximum d'interactions ait été effectué.

Tester la politique apprise Enfin, testez la politique apprise en laissant Codey Rocky suivre les actions suggérées par la table-Q à partir d'un état initial jusqu'au but. Enregistrez les résultats pour les comparer avec l'algorithme du Q-learning profond.

Rappelez-vous, l'objectif de lu Q-learning est d'apprendre une politique, qui est une correspondance des états aux actions. Lorsque l'algorithme est réussi, la politique devrait permettreà Codey Rocky de résoudre le problème de manière efficace.

3.3 Étape 3: Implémentation du Q-learning profond

Le Q-learning profond combine le Q-learning avec des réseaux de neurones profonds. Au lieu de stocker les valeurs-Q dans une table, un réseau de neurones est entraîné pour prédire les valeurs-Q. Les entrées du réseau sont les représentations d'état, et les sorties sont les valeurs-Q prédites pour chaque action.

Implémentez un algorithme du Q-learning profond pour résoudre le même problème que précédemment. Cela devrait impliquer:
— La définition d'une architecture de réseau de neurones pour approximer la fonction-Q
— L'entraînement de ce réseau sur la base des récompenses que Codey Rocky reçoit de l'environnement

Voici les étapes détaillées que vous devez suivre:

Définir l'architecture du réseau de neurones Commencez par définir l'architecture du réseau de neurones. Le réseau devrait prendre en entrée la représentation de l'état et sortir une valeur-Q pour chaque action possible. Vous pourriez commencer avec un réseau de neurones simple avec une ou deux couches cachées, puis envisager des architectures plus complexes si nécessaire.

Préparer les données d'entraînement Les données d'entraînement pour le réseau consisteront en des tuples état-action-récompense-état suivant (s, a, r, s') (souvent appelés un ≪tuple SARSA≫) générés par les interactions de Codey Rocky avec l'environnement. Ces tuples sont typiquement stockés dans un tampon de rejeu. Chaque fois que Codey Rocky effectue une action, le tuple résultant est ajouté au tampon.

Entraîner le réseau Entraînez le réseau en utilisant les tuples du tampon de rejeu. Pour chaque tuple, calculez la valeur-Q cible en utilisant la récompense et la valeur-Q maximale prédite pour l'état suivant

(cela est similaire à la règle de mise à jour du Q-learning). Ensuite, effectuez une rétro-propagation pour mettre à jour les poids du réseau en fonction de la différence entre les valeurs-Q prédites et cibles.

Mettre à jour la politique Après chaque tour d'entraînement, mettez à jour la politique en fonction des poids actuels du réseau. La politique devrait dicter que l'action avec la valeur-Q prédite la plus élevée soit choisie pour chaque état. Continuez à laisser Codey Rocky interagir avec l'environnement et à entraîner le réseau jusqu'à ce que la politique converge ou qu'un nombre maximum d'interactions ait été effectué.

Tester la politique apprise Enfin, testez la politique apprise de la même manière que pour l'algorithme du Q-learning. Laissez Codey Rocky suivre les actions suggérées par la politique à partir d'un état initial jusqu'au but, et enregistrez les résultats pour les comparer.

3.4 Étape 4: Comparaison

Comparez la performance de vos implémentations du Q-learning et du Q-learning profond. Considérez des facteurs tels que:

- La rapidité avec laquelle chaque algorithme apprend une politique efficace
- La qualité de la politique apprise (par exemple, est-ce qu'un algorithme trouve systématiquement de meilleures solutions que l'autre?)
- Le temps de calcul requis par chaque algorithme

4 Livraisons

1. **Code Python**: Soumettez votre code Python pour les algorithmes du Q-learning et du Q-learning profond. Votre code doit être bien commenté, en expliquant comment chaque fonction fonctionne et comment les différentes parties de votre code se rapportent aux concepts du Q-learning et du Q-learning profond.
2. **Rapport**: Rédigez un rapport détaillant votre projet. Ce rapport doit inclure:
   - **Introduction**: Expliquez le problème que vous avez défini pour que Codey Rocky le résolve.
   - **Méthodes**: Décrivez votre mise en œuvre des algorithmes du Q-learning et du Q-learning profond. Cela devrait inclure une explication de vos représentations d'état et d'action, de votre fonction de récompense et de l'architecture de votre réseau de neurones (pour le Q-learning profond).
   - **Résultats**: Présentez les résultats de votre comparaison entre le Q-learning et le Q-learning profond. Incluez tous les graphiques ou tableaux pertinents.
   - **Discussion**: Discutez de tout défi que vous avez rencontré pendant ce projet, de ce que vous avez appris et de toute observation intéressante que vous avez faite concernant la performance du Q-learning et du Q-learning profond. Vous pourriez également suggérer des améliorations basées sur votre travail.
3. **Présentation**: Préparez une présentation de 10 minutes qui comprend:
   - Un résumé de 5 minutes des informations de votre rapport, en se concentrant sur les points clés de votre définition du problème, méthodes, résultats, et discussion ;
   - Un enregistrement de 5 minutes qui montre la phase d'entraînement et la politique finale apprise adoptée par votre robot.

Optionnel: Si vous êtes prêt pour notre dernière séance le 10 avril, vous pourrez présenter votre projet à la classe.

5 Critères d'évaluation

— Exactitude et efficacité du code Python
— Exhaustivité et clarté du rapport et de la présentation
— Compréhension du Q-learning et du Q-learning profond telle que démontrée dans le rapport, la présentation et la mise en œuvre
— Qualité de la comparaison entre le Q-learning et le Q-learning profond: Les métriques de comparaison sont-elles pertinentes et bien expliquées? Les résultats ont-ils du sens?