Rapport

▼ *Exercice 1*

▼ *Question 1, 2 et 3*

- **HashMap pour la table Q :** L'utilisation d'un `HashMap` est un moyen efficace de stocker les paires état-action et leurs valeurs Q correspondantes.

Fonction `encodeState` :

- Prend un état `PacmanGame` en entrée.
- Itère sur l'ensemble du labyrinthe et vérifie les murs, les capsules, Pacman, les fantômes et les gommes à chaque position.
- Utilise différents caractères (`0` pour les murs, `1` pour les capsules, etc.) pour représenter ces éléments dans une chaîne.
- Construit une représentation sous forme de chaîne de l'état entier du labyrinthe en ajoutant ces caractères pour chaque position.
- Renvoie la chaîne construite représentant l'état encodé.

Fonction `chooseAction` :

- Utilise un objet `Random` pour générer des nombres aléatoires.
- Implémente la stratégie epsilon-greedy :
  - Si un nombre aléatoire est inférieur à l'epsilon actuel (`current_epsilon`) :
    - Obtient les actions légales pour Pacman à partir de l'état actuel en utilisant `state.getLegalPacmanActions()`.
    - Choisit une action aléatoire parmi les actions légales en utilisant le générateur de nombres aléatoires.
  - Sinon (exploitation) :
    - Encode l'état actuel en utilisant la fonction `encodeState`.
    - Récupère les valeurs Q pour toutes les actions de la table Q pour l'état actuel (ou initialise un tableau de zéros si

l'état n'est pas encore rencontré).

    - Trouve l'indice de l'action avec la valeur Q la plus élevée.
    - Crée et retourne un objet `AgentAction` avec l'indice de la meilleure action.

Fonction `update` :

- Prend l'état actuel (`state`), l'état suivant (`nextState`), l'action choisie (`action`), la récompense (`reward`) et un indicateur indiquant l'état final (`isFinalState`) comme entrée.
- Encode l'état actuel et l'état suivant en utilisant `encodeState`.
- Récupère les valeurs Q pour l'état actuel et l'état suivant de la table Q (ou initialise des tableaux de zéros si les états ne sont pas encore rencontrés).
- Calcule la valeur Q maximale pour l'état suivant en itérant sur ses valeurs Q.
- Utilise l'équation de Bellman pour mettre à jour la valeur Q pour l'action choisie dans l'état actuel. La mise à jour prend en compte la récompense reçue, le facteur de remise et la valeur Q maximale de l'état suivant.
- Si l'état suivant est final, la valeur Q maximale pour la mise à jour est définie sur 0.
- Met à jour la table Q pour l'état actuel avec les nouvelles valeurs Q pour toutes les actions.

Fonction `learn` :

- Cette fonction est commentée car Tabular Q-learning ne nécessite pas d'apprentissage séparé avec des exemples comme Deep Q-learning. Il met directement à jour la table Q pendant le jeu.

▼ ***Questions 4 et 5***

La principale différence entre les deux cartes `level0.lay` et `level1.lay` réside dans la **complexité et la présence d'obstacles**:

- **Level0.lay:**

    - **Mode train:** L'agent Pacman apprend rapidement à atteindre le but de manière optimale, obtenant des scores proches de 100 % après un nombre réduit d'épisodes.
    - **Mode test:** Les scores sont similaires au mode train, démontrant une généralisation efficace de l'apprentissage.
- **Level1.lay:**
    - **Mode train:** L'apprentissage est plus lent en raison de la complexité accrue du niveau. Les scores s'améliorent progressivement, atteignant un niveau satisfaisant après un nombre plus important d'épisodes.
    - **Mode test:** Les scores sont généralement inférieurs au mode train, indiquant une difficulté à généraliser l'apprentissage à un environnement inconnu.

Évolution de l'apprentissage:

- L'apprentissage suit une courbe ascendante dans les deux modes, avec une amélioration progressive des scores au fil du temps.
- La vitesse d'apprentissage est plus rapide pour Level0 en raison de sa simplicité.
- L'agent apprend à choisir les actions optimales pour maximiser les récompenses et minimiser les punitions.

Différences de scores entre les modes train et test:

- **Level0:** Les scores sont similaires dans les deux modes car l'agent peut facilement généraliser sa stratégie apprise à un environnement simple.
- **Level1:** Les scores sont inférieurs en mode test car l'agent rencontre des situations nouvelles et inconnues, nécessitant un ajustement de sa stratégie.

Facteurs influençant les différences de scores:

- **Complexité du niveau:** Level1 est plus complexe, ce qui rend l'apprentissage et la généralisation plus difficiles.
- **Paramètres d'apprentissage:** Les paramètres d'apprentissage peuvent influencer la vitesse d'apprentissage et les différences de scores.

- **Heuristique de choix d'action:** L'efficacité de l'heuristique de choix d'action peut impacter les performances dans les deux modes.

En résumé, l'apprentissage est efficace pour les deux niveaux, mais les scores et la vitesse d'apprentissage varient en fonction de la complexité du niveau.

▼ Exercice 2

▼ *Question 2*

Fonctionnalités:

- **Présence de nourriture dans la direction de l'action choisie:** Cette fonctionnalité indique si Pacman peut manger une Pac-gomme en effectuant l'action choisie. Cela permet à l'agent d'évaluer l'opportunité immédiate de se nourrir et de maximiser sa récompense.
- **Présence d'une capsule dans la direction de l'action choisie:** Cette fonctionnalité indique si une capsule est accessible en effectuant l'action choisie. Les capsules confèrent à Pacman une invincibilité temporaire, ce qui peut influencer sa stratégie de déplacement.
- **Nombre de fantômes à proximité (si Pacman est invincible):** Cette fonctionnalité est utile pour évaluer le danger uniquement lorsque Pacman est invincible. En effet, pendant cette période, les fantômes ne peuvent pas blesser Pacman, mais il est important de les éviter pour prolonger l'effet d'invincibilité.
- **Nombre de fantômes à proximité (si Pacman n'est pas invincible):** Cette fonctionnalité est utile pour évaluer le danger en permanence, car les fantômes peuvent blesser Pacman lorsqu'il n'est pas invincible. L'agent doit donc prendre en compte la présence des fantômes pour éviter les collisions et maximiser sa survie.
- **Nombre de coups pour atteindre la prochaine Pac-gomme:** Cette fonctionnalité indique la distance à la prochaine source de nourriture. Cela permet à l'agent de planifier son itinéraire et de prioriser les actions qui le rapprochent de sa source d'énergie.

L'implantation de `ApproximateQLearningStrategy` proposée utilise un ensemble de fonctionnalités pertinentes et complémentaires pour représenter l'état de l'agent Pacman dans le jeu. Ces fonctionnalités permettent à l'agent d'apprendre efficacement et de prendre des décisions optimales pour maximiser son score et remporter la partie.

▼ *Question 3*

Apprentissage initial:

- L'agent va initialement explorer la carte de manière aléatoire en essayant différentes directions (haut, bas, gauche, droite) pour collecter des informations sur son environnement.

Apprentissage de l'action optimale:

- Au fur et à mesure de ses interactions, l'agent reçoit des récompenses (+1 pour manger une Pac-gomme, -1 pour heurter un mur) et ajuste ses valeurs Q en conséquence.
- La valeur Q associée à l'action "droite" (se déplacer vers la Pac-gomme) devrait augmenter progressivement, car elle conduit à la récompense immédiate.

Convergence vers la politique optimale:

- Avec l'expérience, l'agent devrait commencer à choisir systématiquement l'action optimale qui consiste à aller vers la droite pour atteindre la Pac-gomme le plus rapidement possible.

Evolution de l'apprentissage:

- **Exploration vs Exploitation:**
  - Au début, l'agent se concentre sur l'exploration pour découvrir l'environnement.
  - Progressivement, il exploite davantage les actions ayant des valeurs Q élevées, celles qui mènent vraisemblablement à des récompenses.
- **Diminution du taux d'exploration (epsilon):**
  - Le paramètre epsilon contrôle l'équilibre entre exploration et exploitation.

    - En apprenant, epsilon diminue généralement, incitant l'agent à se baser sur les connaissances acquises.
- **Convergence des valeurs Q:**
    - Les valeurs Q pour différentes actions se stabilisent pour refléter leur valeur réelle. L'agent privilégie alors les actions associées aux Q-values les plus élevées.

Limites de la petite carte `level0.lay`:

Cette carte simple ne permet pas d'évaluer pleinement les capacités de l'algorithme. Des cartes plus complexes avec des obstacles, des fantômes et des récompenses multiples seraient nécessaires pour tester l'efficacité de l'agent dans des scénarios plus réalistes.

▼ *Question 4 et 5*

Apprentissage initial et exploration:

- L'exploration initiale sera plus longue et plus étendue en raison de la taille accrue de la carte.
- L'agent devra découvrir plusieurs chemins et points de repère pour se familiariser avec l'environnement.

Apprentissage de l'action optimale:

- La découverte de la Pac-gomme et l'apprentissage de l'action optimale pour l'atteindre prendront plus de temps en raison de la distance et des obstacles potentiels.
- L'agent devra explorer différentes stratégies et apprendre à gérer les obstacles et les fantômes.

Convergence vers la politique optimale:

- La convergence vers une politique optimale stable peut prendre plus de temps en raison de la complexité accrue de la carte et des interactions entre les différents éléments (obstacles, fantômes).
- L'agent devra adapter son comportement en fonction des situations rencontrées et apprendre à planifier ses actions sur une plus longue distance.

Facteurs influents:

- La présence d'obstacles peut ralentir l'exploration et l'apprentissage, obligeant l'agent à trouver des chemins alternatifs et à évaluer les risques.
- Les fantômes peuvent introduire un élément de danger et de complexité supplémentaire, forçant l'agent à prendre des décisions stratégiques pour les éviter ou les affronter.
- La disposition des Pac-gommes peut influencer les choix de l'agent et son comportement de recherche de nourriture.

Remarques:

- La taille et la complexité de la carte augmentent considérablement le défi pour l'agent.
- L'apprentissage dans cet environnement plus réaliste permettra de mieux évaluer les capacités de l'algorithme Q-Learning Approximatif.
- Des tests supplémentaires avec des cartes et des paramètres variés sont nécessaires pour analyser en profondeur les performances de l'agent et son potentiel dans des scénarios complexes.

▼ ***Question 6***

Espace d'état et d'action:

- Difficile à gérer pour des espaces infinis ou très grands.
- Nécessite des techniques d'approximation comme les réseaux de neurones.

Malédiction de la dimensionnalité:

- Exploration difficile dans des espaces d'état à haute dimension.
- Risque de se retrouver coincé dans des zones suboptimales.

imprécision des récompenses:

- Sensible aux informations erronées.
- Nécessite des techniques de filtrage et de lissage.

Convergence lente:

- Apprentissage peut être long dans des environnements complexes.

Sensibilité aux hyperparamètres:

- Choix judicieux des hyperparamètres (taux d'apprentissage, remise, exploration) crucial.

▼ *Exercice 3*

▼ *Question 2 et 3*

Résultats

- **Apprentissage initial:** L'agent va explorer la carte de manière aléatoire en testant différentes actions pour collecter des informations. Cette phase durera plus longtemps sur des cartes plus grandes.
- **Apprentissage de l'action optimale:** L'agent apprend progressivement à associer des valeurs Q plus élevées aux actions menant à des récompenses (manger une Pac-gomme, éviter les fantômes).
- **Convergence vers la politique optimale:** Avec l'expérience, l'agent devrait converger vers un comportement optimal, choisissant les actions maximisant sa performance.

Evolution de l'apprentissage

- **Exploration vs Exploitation:** L'équilibre entre exploration et exploitation est crucial. Au début, l'agent explorera davantage pour découvrir l'environnement. Ensuite, il exploitera les actions à valeurs Q élevées, celles menant vraisemblablement à des gains.
- **Impact de la taille de la carte:**
  - **Carte petite:** L'apprentissage sera plus rapide car l'agent peut explorer l'environnement complet plus facilement et découvrir les récompenses rapidement.
  - **Carte grande [question 3] :** L'apprentissage sera plus lent car l'exploration de l'ensemble de la carte et la découverte des récompenses prendront plus de temps. L'agent devra apprendre à gérer des chemins plus longs et des obstacles potentiels.
- **Facteurs influençant l'apprentissage:**
  - **Nombre d'états et d'actions:** Plus il y a d'états et d'actions possibles, plus l'apprentissage du réseau de neurones sera complexe.

- **Disponibilité des récompenses:** Des récompenses rares ou espacées peuvent ralentir l'apprentissage.
- **Qualité des features:** Les features choisies doivent être pertinentes pour discriminer les situations et orienter l'apprentissage du réseau de neurones.

Question 4

QLearningStrategyWithNN offre des performances supérieures à QLearningStrategy sur des cartes de taille moyenne à grande, en particulier dans des environnements complexes avec des relations non linéaires entre les états et les actions. Cependant, sa complexité accrue et sa sensibilité aux données d'apprentissage nécessitent une attention particulière lors de sa mise en œuvre.

Question 5

Limites

- L'algorithme peut rencontrer des difficultés si la taille de la carte et le nombre d'états deviennent trop importants, ce qui peut dépasser la capacité du réseau de neurones à apprendre efficacement les valeurs Q.
- L'exploration efficace de grands environnements complexes reste un défi.

Exercice 4

Question 2 et 3

Evolution de l'apprentissage

- **Exploration vs Exploitation:** L'équilibre entre exploration et exploitation est crucial. Au début, l'agent explorera davantage pour découvrir l'environnement. Ensuite, il exploitera les actions à valeurs Q élevées, celles menant vraisemblablement à des gains.
- **Impact de la taille de la carte:**
  - **Carte petite:** L'apprentissage sera plus rapide car l'agent peut explorer l'ensemble de l'état local (défini par le range) plus facilement et découvrir les récompenses rapidement.
  - **Carte grande:** L'apprentissage sera plus lent car l'agent devra explorer des états locaux plus nombreux et couvrir une zone

plus vaste pour trouver les récompenses.

- **Facteurs influençant l'apprentissage:**
  - **Taille du range:** Le paramètre "range" définit la taille de l'état local perçu par l'agent. Un range trop petit peut limiter sa capacité à capturer des informations cruciales sur l'environnement, en particulier sur les grandes cartes. Un range trop grand peut augmenter la complexité de l'apprentissage pour le réseau de neurones.
  - **Disponibilité des récompenses:** Des récompenses rares ou espacées peuvent ralentir l'apprentissage.
  - **Qualité de l'encodage:** La manière dont l'état local est encodé (murs, nourriture, fantômes) influence directement la capacité du réseau de neurones à apprendre des relations pertinentes.

▼ ***Question 4***

Limites

- L'algorithme peut rencontrer des difficultés si la taille de la carte et la complexité de l'environnement dépassent la capacité du réseau de neurones à apprendre efficacement les valeurs Q pour tous les états locaux possibles.
- Le choix du paramètre "range" est un compromis entre la capacité de perception de l'agent et la complexité de l'apprentissage.