Rapport De Projet : Apprentissage par renforcement – Snake

Yamis MANFALOTI

Exercice 1 : Tabular Q-learning Solo

Hyperparamètre (Par défaut) :

- Gamme = 0.95
- Epsilon = 0.2
- Alpha = 0.1

4. Résultats et évolution de l'apprentissage

- En fonction de la taille de la carte :
    - **Petite carte :**
        - Le score moyen se stabilise vers 9.2 à partir de la génération 800, ce qui est proche du score maximal possible dû à l'espace disponible sur la carte.
        - Cela montre que le serpent parvient rapidement à apprendre à se déplacer efficacement vers les pommes.
    - **Carte moyenne :**
        - Le score moyen chute et se stabilise vers 2.60 à partir de la génération 900 (epsilon = 0.2), ce qui indique que l'agent à plus de difficulté à apprendre un comportement optimal.
        - Avec plus d'exploration (epsilon = 0.5), le score se stabilise vers 3.51 à partir de la génération 1600.
        - Cela est dû au fait qu'il y a plus d'état possible, et donc le modèle met plus de temps à converger pour trouver les actions optimales. Et il arrive aussi que le modèle reste bloqué sur certains optimums locaux et ne découvre pas les solutions les plus optimales.
- Différence entre le mode train et test :
    - En mode train, l'exploration rend le comportement plus aléatoire (car epsilon > 0) et cela cause donc des scores plus faibles. Mais c'est aussi cela qui permet d'explorer différents états pour trouver les optimums. En mode test, l'agent suit son modèle avec les poids appris (epsilon = 0), et cela permet d'avoir des scores plus réguliers qui suivent les optimums découverts.

5. Efficacité selon la taille du plateau

Le Tabular Q-learning est beaucoup plus efficace sur des petits plateaux. L'espace des différents états est plus petit, ce qui permet un apprentissage

rapide. Sur des plateaux plus grands, la table Q devient trop vaste et l'exploration trop longue pour être réellement efficace.

Exercice 2 : Tabular Q-learning Duel

La version duel prend en compte la présence d'un second serpent et encode les serpents encore vivants.

Hyperparamètre (Par défaut) :

- Gamme = 0.95
- Epsilon = 0.2
- Alpha = 0.1

2. Tabular Q-learning Duel vs StrategyRandom

- Petite carte :
  - Le score moyen se stabilise autour de 8.05 à partir de la génération 1000

- Carte Moyenne :
  - Le score moyen se stabilise autour de 3.26 à partir de la génération 2200

- Analyse :
  - Le serpent utilisant StrategyRandom ayant un comportement très peu intelligent, il meurt souvent de lui-même. Ce qui fait que le serpent Tabular Q-learning Duel ne gagne pas fréquemment les +10 points bonus liés à l'élimination d'un adversaire. Dans un tel cas, il se contente donc de ramasser les pommes comme en mode solo, ce qui donne un score moyen similaire à celui obtenu en Tabular Q-learning Solo.
  - On remarque à nouveau une chute du score moyen sur des cartes plus grandes.

3. Tabular Q-learning Duel vs Tabular Q-learning Duel

- Petite carte :
  - Le score moyen se stabilise vers 6.2 à partir de la génération 1600, mais fluctue parfois jusqu'à 2.49 de score moyen.

- Carte Moyenne :
  - Le score moyen se stabilise vers 1.83 à partir de la génération 1500.

- Analyse :
    - Lorsque les deux serpents utilisent Tabular Q-learning Duel, les parties deviennent plus compétitives. Chaque serpent survit plus longtemps, ce qui augmente la probabilité d'un combat direct.

    - Lorsqu'un serpent élimine l'autre, il gagne +10 points, ce qui tire le score vers le haut. Cependant, si le serpent observé meurt tôt, car il se fait tué par l'autre serpent, son score devient très bas, ce qui fait baisser la moyenne générale.

    - Suite à ces fluctuations de score, on se retrouve avec un score moyen plutôt modéré et qui fluctue occasionnellement très bas.

    - Il y a toujours une chute du score moyen sur des cartes plus grandes.

Exercice 3 : Approximate Q-learning en mode solo

Hyperparamètre (Par défaut) :

- Gamme = 0.95
- Epsilon = 0.5
- Alpha = 0.1

1. Features implémentées

- Feature 0 : Biais
- Feature 1 : Présence d'une pomme dans la prochaine position (Boosté)
- Feature 2 : Distance à la pomme (Inversée pour favoriser la proximité)
- Feature 3 : Risque de mort

3. Test sur des petits niveaux

- Dans de nombreux cas, dès la première visualisation, le serpent atteint un score moyen de 7.44 et ne peut juste pas faire mieux, car il a déjà rempli la totalité de la carte. Mais même si ce n'est pas le cas, le serpent arrive très vite à un bon résultat sur les petites cartes.

- L'apprentissage évolue peu au fur et à mesure du temps dans ce cas parce que la stratégie est très performante dès le début et atteint un plafond dû à la taille de la carte

4. Test sur des niveaux moyens

- Les scores sont plus vairés sur des cartes de plus grandes tailles. Dans certain cas, il atteint de très hauts scores moyens, jusqu'à 15.29. Alors que d'en d'autres cas, il se bloque et donne des scores autours de 0.

- Ce problème vient surement du fait qu'il y a plus d'espace à explorer. Et si le serpent, ne trouve rencontre pas assez vite des situations ou il obtient des récompenses grâce aux pommes, il se retrouve juste à éviter la mort, et donc à faire des aller-retours entre deux case.

5. Proposition de nouvelles features

- Feature 4 : Distance des murs (Inversée pour favoriser les zone sûre)
- Feature 5 : Nombre de directions sûres au prochain coup
- Feature 6 : Direction de la pomme

6. Limites de l'algorithme

- Dépendance forte aux features : L'approximate Q-learning repose très fortement sur la qualité des features choisies. Si elles ne reflètent pas bien les aspects importants de l'environnement, l'algorithme n'arrivera pas à apprendre des stratégies efficaces. Cela fait que cette algorithme est difficile à mettre en place dans des situations où :
    - On manque d'expertise pour concevoir de bonnes features.
    - Les règles du jeu ou de l'environnement ne sont pas bien définies.
    - Les données ou les résultats sont limités pour et ne permettent pas de guider la sélection des features.

- Généralisation limitée : Certain features peuvent avoir du mal à capturer la complexité des environnements qui sont plus grand.

- Exploration insuffisante : L'algorithme peut converger vers des stratégies sécuritaires, mais non optimales, surtout quand il y a manque de récompenses fréquentes.

- Apprentissage lent dans les grands espaces : Sans être guidé plus précisément (par des features), le serpent a du mal à découvrir les stratégies efficaces par lui-même.

Exercice 4 : Approximate Q-learning en mode duel

2. Features intéressante à ajouter

- Features 5 : Distance avec un snake plus grand (à éviter)
- Features 6 : Distances avec un snakes plus petit (à focus)

3. TabularQlearning_duel vs ApproximateQlearning_duel

- Petite carte :
    - Le serpent avec la stratégie Tabular Q-learning Duel à des scores moyens entre 9 et 16
    - Avec la stratégie Approximate Q-learning Duel, l'autre serpent a des scores moyens entre 6 et 11

- Carte Moyenne :
    - Via la stratégie Tabular Q-learning Duel, le serpent a des scores moyens entre 1.96 et 4.58
    - Tandis que le serpent avec la stratégie Approximate Q-learning Duel a des scores moyens entre 8.68 et 17.97

- Analyse :
    - On peut voir que sur les petites cartes, le serpent avec la strategie Tabular Q-learning Duel est bien plus efficace que celui en Approximate Q-learning Duel. Cela est surement dû au fait que le Tabular Q-Learning est particulièrement adapté à ce genre de carte, et est aussi plus précis dans ces cas.
    - Tandis que pour les cartes moyennes, c'est cette fois-ci c'est le serpent avec la stratégie Approximate Q-learning Duel qui est bien meilleurs. Cela est dû au fait que le Approximate Q-learning Duel permet une meilleure généralisation, et est donc plus efficace sur des cartes moyennes, car il est beaucoup moins affecté par taille de la carte que le Tabular Q-learning. De plus les features ajouté permet de mieux cibler l'action de dévourer un joueur et de fuire les plus gros, alors que cela n'est pas spécifié dans Tabular Q-learning .

Exercice 5 : Deep Q-learning

Hyperparamètre :

- Epsilon : 1
- Réduction d'epsilon :
    - multiplicateur appliqué à chaque chooseAction en train : 0.995
    - minimum autorisé pour epsilon : 0.1
- Alpha : 0.001
- Gamma : 0.95
- NbEpochs : 1
- Batchsize : 32
- EncodedStateSize : 5 * 5 * 5 (largeur * hauteur * nb features)

Visualisation toutes les 20 itérations (dans main_batchMode)

1.1 Résultats en mode solo

- Petite carte :
    - Score moyen qui se stabilise autour de 9.02 à partir de l'itération 60

- Carte moyenne :
    - Le score moyen de la stratégie Deep Q-learning se stabilise autour de 11.85 à partir de la génération 140. Cependant, le score moyen arrive difficilement à dépasser 12, et cela est dû au nombre de tours qui est limité à 100. Ainsi, le serpent ne peut donc accumuler plus de pomme et le score dépend fortement de l'endroit où apparaissent les pommes.

- Analyse :
    - Le Deep Q-learning montre de bonnes performances sur les petites cartes, ce qui confirme sa capacité à généraliser et à s'adapter. Sur des cartes plus grandes, l'apprentissage est plus lent. Le réglage d'epsilon s'avère crucial, car une réduction trop rapide empêche l'agent d'explorer suffisamment au début.

1.2 Résultats en mode duel

- Petite carte :
    - Le serpent Deep Q-Learning obtient un score moyen qui se stabilise autour de 10.68 à partir de l'itération 100 contre un serpent avec une stratégie Tabular Q-learning Duel.

- Carte moyenne :

    - Sur une carte moyenne toujours contre un serpent en Tabular Q-learning Duel, le serpent en Deep Q-learning se stabilise autour d'un score moyen de 10.85 à partir de la génération 120.

2. Comparaison des stratégies

Les trois stratégies de Q-learning implémentées ont chacune leurs forces et leurs limites, qui se manifestent différemment selon la taille du plateau et le mode de jeu (solo ou duel).

Tabular Q-Learning :

Avantages :

- Très efficace sur les petits plateaux, où l'espace d'état est limité.
- A une convergence rapide vers des comportements optimaux quand les états sont peu nombreux.
- Est très facile à implémenter et à comprendre.

Limites :

- Ne généralise pas, car chaque état doit être appris individuellement.
- Il est plutôt inefficace sur les grandes cartes, puisque la table Q devient trop grande.
- En mode duel, il a performances instables dues au grand nombre d'états supplémentaire rajouté par la présence de deux joueurs.

Approximate Q-Learning :

Avantages :

- Il à bonne capacité de généralisation grâce à l'utilisation des features.
- Il est aussi plus scalable que le Tabular Q-Learning, car l'espace des états n'est plus une contrainte directe.
- De plus, il permet d'ajouter certaine logique via la conception des features, ce qui permet de mieux cibler ce qu'on attend de l'algorithme.
- Enfi, il est meilleur que Tabular Q-learning sur les plateaux moyens à grands et plus rapide que le Deep Q-learning.

Limites :

- Il est très dépendant de la qualité des features, car un mauvais choix peut complètement compromettre l'apprentissage.
- Il peut aussi converger vers des stratégies sous-optimales, surtout quand les récompenses sont rares.
- Puis, il est moins stable que Tabular Q-Learning sur des petites cartes à cause de la généralisation dû aux features.

Deep Q-Learning

Avantages :

- Il à la capacité d'apprendre automatiquement des représentations complexes à partir d'un état encodé. Sans avoir à définir des features manuellement.
- Il est un très bon compromis entre généralisation et performance brute. Et est moins sensible à la taille du plateau que les deux autres approches.
- Enfin, il produit des résultats solides et stables en mode duel et en mode solo, même sur des plateaux plus grands.

Limites :

- Mais il a un temps d'apprentissage qui est beaucoup plus long. Et peut nécessiter de nombreuses itérations et batchs pour converger.
- Il est aussi plus complexe à implémenter. Et les performances peuvent fortement varier selon les hyperparamètres choisies.

Conclusion générale :

- Pour des environnements plutôt simples et petits, le Tabular Q-Learning est bien suffisant et très performant.
- Dans des environnements plus grands, où l'espace d'état est trop vaste, la stratégie Approximate Q-Learning est une bonne solution, à condition d'avoir des features bien choisies.
- Enfin, lorsque la complexité devient plus élevée, le Deep Q-Learning devient l'approche la plus robuste et la plus prometteuse, bien qu'il ait un coût computationnel plus élevé.

Bonus

Pour le bonus, je suis partie sur ces 5 features :

```
// Feature 0 : Biais
features[0] = 1.0;

// Feature 1 : Présence d'une pomme dans la prochaine position (Boosté)
features[1] = isAppleInNextMove(state, action, snake) ? 1 : 0;

// Feature 2 : Distance à la pomme (Inversée pour favoriser la proximité)
features[2] = (1 - normalizedAppleDistance(state, snake, action));
```

```
// Feature 3 : Risque de mort (Pénalité renforcée)
features[3] = willDieNextMove(state, action, snake) ? -1 : 0.0;

// Features 4 : Distance avec un snake plus grand (à eviter)
features[4] = normalizedNeareastBiggerSnakeDistance(state, snake);

// Features 5 : Distances avec un snakes plus petit (à focus)
features[5] = (1 - normalizedNeareastSmallerSnakeDistance(state, snake));
```

Et j'ai initialisé cette stratégie avec ces paramètres :

- nbActions : 4
- epsilon : 0
- gamma : 0.95
- alpha 0.1
- fonction update vide
- w :
    - w[0] = -1.110835399526325,
    - w[1] = 0.07571323388807702,
    - w[2] = 4.82513943092903,
    - w[3] = 5.760921209750668,
    - w[4] = -0.27799864400873875,
    - w[5] = 2.2218426602927863

Avec cette stratégie, pour un nombre de tours maximum à 100 et contre la stratégie Advanced pour le duel, j'obtiens :

- Alone Small With Walls : ≈ 8.41
- Alone Small No Walls : ≈ 18.86
- Alone Medium With Walls : ≈ 14.85
- Alone Medium No Walls : ≈ 16.91

- Duel Small With Walls : ≈ 17.13
- Duel Small No Walls : ≈ 25.12
- Duel Medium With Walls : ≈ 21.74
- Duel Medium No Walls : ≈ 22.11