# Examen
# Apprentissage profond par renforcement

Université Lyon 1 — M2IA — 2 mars 2020

# 1 Réseaux convolutionnels et apprentissage profond par renforcement

## 1.1 Réseaux convolutionnels (2 points)

Voici ci-dessous une image I légèrement bruitée et un masque F. Après chaque application d'un opérateur, vous redessinerez la table.

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |

| | |
|---|---|
| -1 | 1 |
| -1 | 1 |

TABLE 1 – A gauche l'image I, à droite le masque F.

**Question 1. (1 pt)** Appliquez la cross-corrélation du masque F sur l'image I. Appliquez :

- Un *padding* de 1 sur chaque côté de l'image.
- Un *striding* vertical et horizontal de 2.

**Réponse** :
On rajoute une ligne de 0 en haut et en bas, une colonne de 0 à gauche et à droite(padding 1). Ensuite on applique la convolution en se déplacant de deux cases en deux cases (striding 2). Cela donne :

| | | | |
|---|---|---|---|
| 0 | 1 | -1 | 0 |
| 0 | -2 | 2 | 0 |
| 0 | -1 | 2 | 0 |
| 0 | 1 | -1 | 0 |

**Question 2. (0.5 pt)** Appliquez un max-pooling 2x2.

**Réponse** :
On prend le maximum des carrés de deux, cela donne :

| | |
|---|---|
| 1 | 2 |
| 1 | 2 |

**Question 3. (0.5 pt)** Appliquez la fonction d'activation ReLU.

**Réponse :** ReLU : $x \rightarrow max(0, x)$. Ici toutes les valeurs sont positives donc ca ne change rien.

## 1.2 Apprentissage profond par renforcement (5 points)

Dans cette partie, vous allez devoir modéliser un problème, avec quelques contraintes imposées, et proposer une architecture d'apprentissage par renforcement addressant ce problème. Vous pouvez choisir entre deux problèmes ; choisissez donc celui que vous préférez et répondez aux questions. Notez toutefois qu'il n'y a pas de solution parfaite, , mais nous attendons de vous des choix justifiés et une idée des problèmes induits par vos choix.

### 1.2.1 Problème 1.

Dans un environnement de simulation, notre agent est coincé dans un labyrinthe similaire à celui illustré par la Figure 1a. Dans cet environnement, il y a des portes fermées à clé (chaque couleur correspond à une clé), des portes non fermées à clés, des balles (vertes) qu'il peut tirer et des boites (grises) qu'il peut ouvrir pour obtenir une clé. Entre chaque épisode, le labyrinthe ne change pas, et nous aimerions que notre agent apprenne à atteindre la balle bleue le plus rapidement possible. On a :

**Espace d'observation** : La grille de cases où chaque contenu possible de case correspond à un chiffre (16x16). À cela s'ajoute un booléen indiquant la couleur des clés qu'il a déjà trouvé (11 booléens).

**Espace d'action** : L'agent peut se déplacer dans les 4 directions cardinales, ouvrir une boîte, essayer d'ouvrir une porte et tirer une boule.

### 1.2.2 Problème 2.

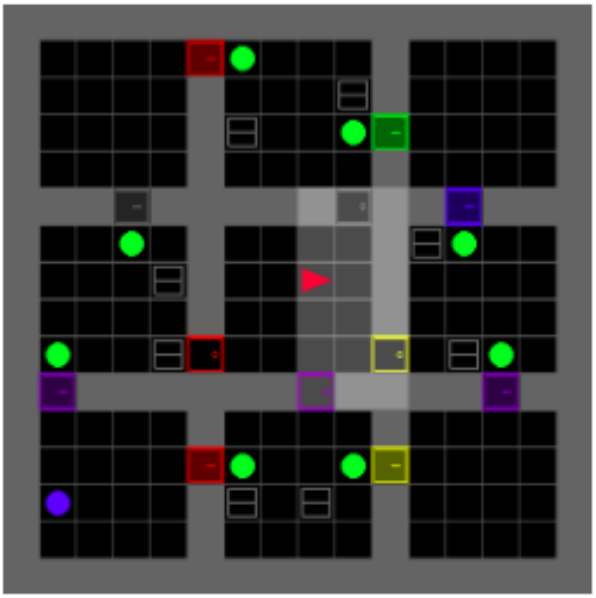
Le département informatique vient de recevoir un drône réel (Figure 1b et souhaite lui apprendre à atteindre des coordonnées de géolocalisation. Les coordonnées visées sont disponibles mais peuvent varier selon la personne qui l'utilise. Il étudie ses données d'entrées et choisit une nouvelle action toutes les 500ms. On a :

**Espace d'observation** : Vélocité (x,y,z), géolocalisation, altitude, 20 données laser détectant sa distance aux objets autour de lui et les valeurs actuelles de ses actions (voir ci-dessous) (30 données).

**Espace d'action** : 4 valeurs continues qui gèrent la puissance, le tangage, le roulis et le lacet du drône à un instant donné.

### 1.2.3 Questions

**Question 1. (1pt)** Proposez une fonction de récompense, *i.e.* la récompense donnée à l'agent à chaque étape.
**Question 2. (0.5pt)** Proposez un paramètre de durée maximale d'un épisode (en nombre d'itération). Justifiez.
**Question 3. (0.5pt)** Proposez une valeur pour le paramètre d'atténuation $\gamma$. Quelle est l'utilité principale d'un $\gamma$ proche de 1 ? Inversement quel est son principal inconvénient ?



(a) Illustration d'un labyrinthe possible pour le problème 1.

(b) Exemple de drône utilisé dans le problème 2.

**Question 4. (1pt)** Choisissez un algorithme d'apprentissage adéquat au problème (Q-learning, Q-learning linéaire, DQN, A2C ...). Vous pouvez aussi proposer et expliquer des améliorations.

**Question 5. (1pt)** En prenant en compte votre choix à la question précédente, proposez une architecture du/des potentiel(s) réseau(x) de neurone(s). Indiquez les entrées, le type, le nombre et la taille des couches cachées et la taille de la couche de sortie. Si vous utilisez un réseau de neurone convolutionnel, vous n'êtes pas obligés de donner la taille exacte de sa couche de sortie (taille que l'on nommera N).

**Question 6. (1pt)** Considérant les précédentes réponses, discutez votre proposition. Quels sont les avantages/inconvénients ? Avez-vous des pistes pour l'améliorer ? Est-ce que l'apprentissage par renforcement vous semble adéquat ?

**Réponses** Plusieurs solutions sont possibles tant qu'elles sont détaillées, justifiées que les différentes réponses sont cohérentes entre elles. La correction présente **une** solution possible.

Problème 1

Q1)

- 10 lorsque l'agent atteint la boule bleue.
- -0.01 le reste du temps.

L'avantage de cette récompense très simple est qu'on met relativement peu de connaissances expertes ; la fonction fonctionnera naturellement sur un labyrinthe similaire et le peu de connaissances expertes ajoutées évite des comportement indésirables. Typiquement récom-

penser le tirage de boule verte peu amener l'agent à faire que tirer les boules.

Q2) 1000 itérations devrait suffire pour parcourir les salles tout en permettant une exploration de type $\epsilon - greedy$.

Q3) $\gamma = 0.99$ permet d'agir en prenant en compte les récompenses sur le très long terme. La récompense d'une bonne action au début sera récompensée potentiellement plusieurs centaines d'itérations plus tard. En contrepartie, un important $\gamma$ ralentit la stabilisation de la Q-valeur, puisqu'il faut rétro-propager les erreurs sur beaucoup d'états.
Q4)

- L'espace d'état est trop grand pour un Q-learning ( 16x16x2$\hat{1}$1).
- Il est difficile de construire de bonnes features non linéaires à la main.
- A2C pourrait être utilisé, mais est plus compliqué et lourd sur le DQN, sans présenter de réel avantage.
- On va donc choisir le DQN, qui permet d'approximer efficacement sur l'espace d'état.

Q5) On a en entrée une grille 2d structurée, on peut donc utiliser un réseau convolutionnel. L'idée est juste de donner une architecture cohérente. Par exemple :

1. Entrées : Grille 16x16
2. 10 filtres de taille 4x4, striding 2
3. Max-pooling 2x2
4. ReLU
5. 10 filtres de taille 4x4, striding 2
6. Max-pooling 2x2
7. ReLU
8. Mise à plat : N
9. Concaténation avec les 11 booléens
10. Couche linéaire de 256 neurones
11. Sigmoid
12. Couche linéaire vers 7 neurones (1 pour chaque action)

Q6) Dans la proposition faite, le gros problème est qu'il va être difficile d'atteindre la boule bleue avec de l'exploration classique. Avec $\epsilon - greedy$, l'agent va errer aléatoirement et il y a peu de chance qu'il ne trouve ne serait-ce qu'une clé. Quand bien même il trouve la boule bleue une seule fois, cela ne suffira pas pour apprendre correctement, l'agent a besoin d'être beaucoup guidé par la récompense pour apprendre un comportement intéressant.

Il y aurait plusieurs possibilités d'améliorations : 1-On pourrait rajouter des méchanismes de curiosité pour justement explorer bien plus intelligemment l'environnement ; 2-On pourrait découpler l'apprentissage de la représentation de la grille en utilisant un auto-encodeur ;

3-On pourrait créer un curriculum sur l'apprentissage de l'agent en éloignant petit à petit l'agent de la boule bleue ; 4- Plus simplement il est aussi possible de créer une récompense plus intelligente, mais on perd alors de la généricité.

Dans tous les cas, l'apprentissage prendra beaucoup de temps malgré la simplicité du problème.

Problème 2

Q1)

- -dist(agent,objectif).
- -10 s'il heurte un objet.

On veut que l'agent atteigne l'objectif sans subir de heurts.

Q2) 1000 itérations devrait suffire pour être capable d'atteindre l'objectif, même avec de l'exploration.

Q3) $\gamma = 0.95$ permet d'agir en prenant en compte les récompenses sur le long terme. La récompense d'une bonne action au début sera récompensée potentiellement plusieurs centaines d'itérations plus tard. Un gamma plus faible pourrait accélérer l'apprentissage, mais selon la topologie du terrain et sa hauteur, l'heuristique de distance peut l'amener vers un optimal local (il pourrait être obligé de reculer pour contourner un immeuble). Il vaut mieux donc utiliser un gamma relativement haut.

Q4) L'espace d'action est continu, donc toute solution à base de Q-learning est impossible. Il faut dans ce cas utiliser un algorithme Actor-critic, on va donc choisir A2C.

Q5) L'actor : 30 neurones en entrée, deux couches cachées de 128 neurones suivies de Leaky ReLU, une couche linéaire donnant sur les données de variance et de moyenne des actions (2x4).
Le critic : 30+4(action) données en entrée, deux couches cachées de 128 neurones suivies de Leaky ReLU, une couche linéaire donnant sur un neurone estimant la Q-valeur de l'action.

Q6) C'est plutôt une mauvaise idée d'appliquer de l'apprentissage profond par renforcement dans ce cas là. L'apprentissage serait beaucoup trop long pour du temps réel, le drône se casserait beaucoup trop souvent et il serait même dangereux pour les passants.

Éventuellement, on pourrait lui faire apprendre à voler sur un simulateur et faire du transfert d'apprentissage Sim2real. Même ainsi, il faudrait assurer une surveillance. Il pourrait être aussi intéressant de lui faire des démonstrations de vol.