UE 456 : traitement de l'image et intelligence artificielle

Apprentissage par renforcement : note du cours de Michel Kieffer

THOMAS BOULANGER

10 AVRIL 2023

Table des matières

Table des matières **1**

1 Bandit à bras multiples **2**
1.1 Méthode reposant sur la valeur de l'action 3
1.2 Exemple d'un MAB à 10 bras 3
1.3 Implantation incrémentale 3
1.4 Bibliographie 5

2 Processus de décision Markoviens **6**
2.1 Interface entre agent et environnement 6
2.2 Récompenses et épisodes 7
2.3 Politique et fonctions valeur 7
2.4 Politique optimale 8
2.5 Algorithme SARSA 9
2.6 Q-learning 10

3 Apprentissage par renforcement profond **12**
3.1 Approximation de la fonction $Q_\pi(S,A)$ 12
3.1.1 Gradient stochastique 12
3.1.2 Replay Buffer 13
3.2 Neural Fitted Q-learning 13
3.3 Deep Q learning (DQN) mnih et al (2015), Nature 14

1 Bandit à bras multiples

On considère le problème d'apprentissage suivant :

Objectif de l'apprentissage par renforcement : Trouver une succession d'action qui maximise la récompense accumulée.

Modèle est simple car pas de retour d'un état de l'environnement

Pour le problème MAB (multi-arms bandit) à k bras :
- A_t : action choisie à l'instant (distinct) t (aléatoire)
- R_t : récompense obtenue (aléatoire)

A chaque action, on peut associer l'espérance de la récompense

$$q^*(a) = E(R_t | A_t = a)$$

Si $q^*(a)$ était connue pour tout a, il suffirai de choisir a maximisant q^*.

En pratique :
- q^* n'est pas connue
- une estimée de $Q_t(a)$ de $q^*(a)$ peut être disponible à t

Ensuite, à t
- l'action "$a_t = \text{argmax}_a(Q_t(a))$" est appelée action gloutonne (greedy). Cela revient à **exploiter** la connaissance courante sur le MAB
- l'action "$a_t \neq \text{argmax}_a(Q_t(a))$" consiste à **explorer** d'autres choix possibles

Exploitation : Maximise la récompense à **1 pas** en avant (compte tenu de la connaissance sur le système).

Exploration : Se donner la possibilité d'obtenir une récompense accumulée supérieure **à long terme**.

Le choix entre exploration et exploitation dépend :
- de l'incertitude sur $Q_t(a)$
- du nombre d'épisodes à jouer

Les stratégies reposent sur deux hypothèses :
- la stationnarité des récompenses associées à chaque bras
- informations à priori sur le modèle statistique ... Difficile a vérifier en pratique

1.1 - Méthode reposant sur la valeur de l'action

$q^*(a)$ est la fonction mesurant la valeur de l'action a. Estimée à l'instant t de $q^*(a)$:

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

Où

$$\mathbb{1}_{A_i=a} = 1 \quad si \quad A_i = a, \quad 0 \quad sinon$$

Si $\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a} = 0$, $Q_t(a)$ peut prendre une valeur arbitraire, par exemple 0.

A condition que chaque action soit choisie un nombre tendant vers ∞ de fois

Stratégie gloutonne : Choisir $A_t = \text{argmax}_a Q_t(a)$ avec tirage aléatoire lorsque plusieurs actions donnent le même maximum.

Stratégie ϵ-gloutonne :
- Stratégie gloutonne avec un probabilité de 1 - ϵ.
- Action choisie au hasard avec une probabilité ϵ.

Avec la stratégie ϵ-gloutonne, chaque action est choisie un infinité de fois.

$$Q_t(a) \xrightarrow[t\to\infty]{} q^*(a)$$

1.2 - Exemple d'un MAB à 10 bras

On considère un MAB à 10 bras.
- A chaque bras, on associe une récompense moyenne tirée aléatoirement suivant $\mathcal{N}(0,1)$ (c'est $q^*(a)$)
- A chaque tirage du bras a, R_t sera distribué suivant $\mathcal{N}(0,1)$
- On considère 1000 tirages successifs

1.3 - Implantation incrémentale

L'évaluation de

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

Nécessite :
- de stocker toutes les récompenses et actions choisies
- un calcul de complexité croissante

$Q_t(a)$ eut peut être calculé de manière itérative. Imaginons que la même action a été tout le temps choisie :

$$Q_{t+1}(a) = \frac{1}{t}\sum_{i=1}^{t} R_i = \frac{1}{t}(R_t + (t-1)Q_t) = Q_t(a) + \frac{1}{t}(R_t - Q_t(a))$$

$$\boxed{Q_{t+1}(a) = Q_t(a) + \frac{1}{t}(R_t - Q_t(a))}$$

- $Q_{t+1}(a)$ est la nouvelle estimée
- $Q_t(a)$ est l'ancienne estimée
- $\frac{1}{t}$ est le pas de correction (variable dans le temps)
- R_t est la mesure

Le pas de correction est aussi appelé α ou $\alpha(t)$ dns la suite du cours.

Algorithme : Bandit manchot à bras multiples

```
initialisation :
Pour a =1 à k
    $Q_1(a) = 0$
    $N(a)=0$
while true
    A = argmax Q(a) avec une proba 1- epsilon OU
    A = a choisi aléatoirement avec proba de epsilon
    R = bandit(A)
    N(A)+=1
    Q(A)=Q(A)+ 1/N(A) * (R - Q(A))
```

Dans le cas où les caractéristiques du MAB sont non stationnaires, $Q_t(A)$ doit pouvoir "suivre" les variations de récompenses moyennes. Si a est choisie à t :

$$\boxed{Q_{t+1} = Q_t(a) + \alpha(R_t - Q_t(a))} \quad avec \quad \alpha \in [0,1]$$

Pour l'analyse, on suppose que l'action a a tout le temps été choisie.

$$Q_{t+1}(a) = Q_t(a) + \alpha(R_t + Q_t(a))$$
$$Q_{t+1}(a) = \alpha R_t + (1-\alpha)Q_t(a)$$
$$Q_{t+1}(a) = \alpha R_t + (1-\alpha)(\alpha R_{t-1} + (1-\alpha)Q_{t-1}(a))$$
$$Q_{t+1}(a) = \alpha R_t + (1-\alpha)\alpha R_{t-1} + (1-\alpha)^2 \alpha R_{t-2} + (1-\alpha)^3 Q_{t-2}(a)$$

Avec un α "grand" (0.5), on s'adapte rapidement car le poids des mesures les plus anciennes est faibles. Les estimées $Q_t(a)$ varient beaucoup avec t. Avec α "petit" (0.01), l'adaptation est lente, les estimées $Q_t(a)$ vont peu fluctuer. Le pas α peut varier avec le nombre de sélections du bras a : $\alpha_n(a)$: valeur de α après n sélections du bras a. Dans le cas stationnaire, $Q_t(a)$ converge à condition que $\sum_{n=1}^{\infty} \alpha_n(a)$ converge

L'expression de $Q_{t+1}(a)$ en choisissant tout le temps a est :

$$\boxed{Q_{t+1}(a) = (1-\alpha)^t Q_1(a) + \sum_{i=1}^{t} \alpha(1-\alpha)^{t-i} R_i}$$

On peut favoriser l'exploration, même avec une stratégie gloutonne en faisant une initialisation optimiste de $Q_t(a)$ avec $Q_1(a)$ supérieur aux récompenses attendues, par exemple $Q_1(a) = 5$ lorsque $q^*(a) \approx 1$

1.4 - Bibliographie

— Sutton et Barto - Renforcement Learning

2 Processus de décision Markoviens

On cherche à estimer
- q(a)
- q(s,a) avec s état de l'environnement

2.1 - Interface entre agent et environnement

L'agent prend en compte l'état s_t et la Récompense r_t pour déterminer l'Action a_t à des instant discret.

En retour de cet action a_t, l'environnement fournit :
- Son état $s_{t+1} \in \mathscr{S}$ avec $\mathscr{S}$ l'ensemble des états.
- Une récompense numérique $r_{t+1} \in \mathscr{R} \in \mathbb{R}$

L'action $a_t \in \mathscr{A}(s_t)$, doit être choisie dans un ensemble $\mathscr{A}(s_t)$ qui peut dépendre de l'état.
Un processus de décision Markovien (PDM) $(\mathscr{S},\mathscr{R},\mathscr{A})$

Le PDM est fini lorsque $\mathscr{S},\mathscr{R},\mathscr{A}$ sont finis.

L'évolution du PDM est décrit par la probabilité de transition

$$p(s^{'},r|s,a) = Pr(S_{t+1} = s^{'}, R_{t+1} = r|S_t = s, A_t = a)$$

invariant dans le temps.

$$p(s^{'}|s,a) = \sum_{r \in \mathscr{R}} p(s^{'},r|s,a)$$

$$r(s,a) = \sum_{s^{'} \in \mathscr{S}, r \in \mathscr{R}} r.p(s^{'},r|s,a)$$

Exemple : Robot collecteur de canettes de soda → Graphe des actions possibles

2.2 - Récompenses et épisodes

En appliquant une succession d'action à partir de t, on obtient $R_{t+1}, R_{t+2}, \ldots$ En supposant que le processus se termine à T>t, on peut calculer le revenu (gain) :

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T$$

Dans certaines situation, le processus peux ne pas se terminer. Dans ce cas, G_p peux diverger. Pour éviter ça, on affecte aux récompenses un poids favorisant les récompenses immédiates plutôt que celles qui arriveront dans très longtemps. On introduit donc le taux d'escompte $\gamma \in]0;1[$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma \check{s} R_{t+3} + \ldots$$

$$\boxed{G_t = \sum_{i=t+1}^{\infty} \gamma^{i-1} R_{t+i}}$$

G_t satisfait une relation de récurrence :

$$\boxed{G_t = R_{t+1} + \gamma G_{t+1}}$$

Valable pour des processus se terminant à T si on prends $G_T = 0$

2.3 - Politique et fonctions valeur

Une politique est une fonction $\Pi(a|s)$, donnant la probabilité de choisir a lorsque l'état est s.

C'est ce que l'on essaie de chercher. On introduit pour tout $s \in \mathscr{S}$ la fonction valeur de l'état s sous la politique Π :

$$v_\Pi(s) = E_\Pi(G_t|S_t = s) = E_\Pi\left(\sum_{i=1}^{\infty} \gamma^{i-1} R_{t+i}|S_t = s\right)$$

On peut également évaluer l'espérance du revenue partant de $S_t = s$ en appliquant $A_t = a$, puis la politique Π

$$q_\Pi(s,a) = E_\Pi(G_t|S_t = s, A_t = a)$$
$$q_\Pi(s,a) = E_\Pi\left(\sum_{i=1}^{\infty} \gamma^{i-1} R_{t+i}|S_t = s, A_t = a\right)$$

Les fonctions $v_\Pi(s)$ et $q_\Pi(s,a)$ satisfaisant l'équation de Bellmann :

$$v_\Pi(s) = E_\Pi(G_t|S_t = s) = E_\Pi\left(R_{t+1} + \gamma G_{t+1}|S_t = s\right)$$

$$v_\Pi(s) = \sum_a \Pi(a|s) E\left(R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a\right)$$

$$\boxed{v_\Pi(s) = \sum_a \Pi(a|s) \left[\sum_{s',r} p(r,s'|s,a)(r + \gamma v_\Pi(s'))\right]}$$

Pour tout $s \in \mathscr{S}$

cf Backup diagram

<u>Exemple :</u> Grid world, calculer $v_\Pi(s)$ nécessite de résoudre un système d'équations linéaires.

Relation entre $v_\Pi(s)$ et $q_\Pi(s,a)$?

$$v_\Pi(s) = E_\Pi(G_t|S_t = s)$$
$$q_\Pi(s,a) = E_\Pi(G_t|S_t = s, A_t = a)$$

On a donc :

$$\boxed{v_\Pi(s) = \sum_a \Pi(a|s) q_\Pi(s,a)}$$

On peut alors obtenir l'équation de Bellmann satisfaite par $q_\Pi(s,a)$

$$q_\Pi(s,a) = E_\Pi\left(R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a\right)$$
$$q_\Pi(s,a) = \sum_{s',r} p(r,s'|s,a)(r + \gamma v_\Pi(s'))$$

$$\boxed{q_\Pi(s,a) = \sum_{s',r} p(r,s'|s,a)\left(r + \gamma \sum_{a'} \Pi(a'|s') q_\Pi(s',a')\right)}$$

2.4 - Politique optimale

On considère deux politique Π et Π'. On dit que Π' est une meilleur politique que Π lorsque $v_{\Pi'}(s) > v_\Pi(s)$ pour tout $s \in \mathscr{S}$

la politique Π^* optimale est :

$$\Pi^*(s) = arg\ \max_{\Pi}(v_{\Pi}(s))$$

Pour tout $s \in \mathscr{S}$

La fonction valeur de l'état optimale :

$$v_{\Pi^*}(s) = \max_{\Pi}\ v_{\Pi}(s)$$

Pour tout $s \in \mathscr{S}$

La fonction valeur de l'action optimale :

$$q_{\Pi^*}(s,a) = \max_{\Pi}\ q_{\Pi}(s,a)$$

Pour tout $s \in \mathscr{S}$ et $a \in \mathscr{A}$

$v^*_{\Pi}(s)$ et q_{Π^*} sont uniques.

Il peut y avoir plusieurs politiques optimales associées.

Ces deux fonctions valeurs satisfont l'équation de Bellmann suivante :

$$\boxed{v^*(s) = v_{\Pi^*}(s) = \max_{a \in \mathscr{A}} \sum_{s',r} p(r,s'|s,a)\Big(r + \gamma v^*(s')\Big)}$$

$$\boxed{q^*(s,a) = \sum_{s',r} p(r,s'|s,a)\Big(r + \gamma\ \max_{a'} q^*(s',a')\Big)}$$

Sinon on dispose de $q^*(s,a)$, la ou les actions optimales à appliquer satisfaisant :

$$a^* \in arg\ \max_{a \in \mathscr{A}} q^*(s,a)$$

$$\Pi^*(s) = arg\ \max_{a \in \mathscr{A}} q^*(s,a)$$

2.5 - Algorithme SARSA

Rappel du cours 1 : $Q_{t+1} = Q_t(a) + \alpha(R_t - Q_t(a))$

Transposition à la fonction valeur de l'état :

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

En supposant que G_t puisse être obtenu... ce qui est loin d'être aisé. En appliquant une action $A_t = a$ suivant la politique $\Pi(a|s)$, on récupère R_{t+1} et S_{t+1} On approxime G_{t+1} par

$$G_{t+1} \approx V(s_{t+1})$$

d'où

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1} - V(S_t))$$

Mise à jour appliquée par l'algorithme SARSA d'estimation de $v^*(s)$ ou de $q^*(s,a)$

Algorithme SARSA de calcul de q^* Paramètres d'entrée

- α taux d'apprentissage
- ϵ probabilité d'exploration

Initialiser Q(s,a) arbitrairement pour $s \in \mathscr{S}$ et $a \in \mathscr{A}$ sauf pour s terminal où l'on a Q(s,a) = 0 pour tout $a \in \mathscr{A}$. Pour chaque épisode :

- Initialiser S
- Faire
  - Déterminer A à partir de S par exemple en appliquant une politique ϵ gloutonne dérivée de Q.
  - Appliquer A et récupérer R et S'
  - On choist une action A' à partir de S' en appliquant une stratégie ϵ gloutonne déduite de Q
  - $Q(S,A) \leftarrow Q(S,A) + \alpha\left(R + \gamma Q(S',A') - Q(S,A)\right)$
  - $S \leftarrow S'$, $A \leftarrow A'$
- jusqu'à la fin de l'épisode

SARSA permet d'apprendre la fonction Q associée à une politique $\epsilon - greedy$

2.6 - Q-learning

L'objectif du Q-learning est d'apprendre la fonction $Q*(s_t, A_t)$ associée à une politique optimale.

$$Q(s_t, A_t) \leftarrow Q(s_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, A_t))$$

$R_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')$ est l'estimée par bootstrap de $Q*(s_t, A_t)$

Algorithme Q-learning

Initialisation de Q(s,A) arbitrairement sauf pour Q(S= terminal,.) = 0

Pour chaque épisode :

- Initialisation de S

- tant que s $\neq$ Terminal
  - [p]

    Choisir A à partir de S suivant avec une stratégie ϵ-greedy déduite de Q
  - Appliquer A et observer S' et R
  - $Q(s_t, A_t) \leftarrow Q(s_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(s', a') - Q(s_t, A_t))$
  - S $\leftarrow$ S'

L'estimée Q(S,A) va converger vers Q*(S,A). On estime la fonction Q* associée à une politique optimale (politique cible). On utilise pour cela des mesures obtenues avec une politique différente, ici, ϵ-greedy

Q-learning est une approche dite "off-policy" d'apprentissage par renforcement, SARSA est une approche "on-policy".

3 Apprentissage par renforcement profond

3.1 - Approximation de la fonction $Q_\pi(S,A)$

On considère S et/ou A de grande taille, éventuellement continues. On utilise un modèle $Q_\theta(S,A)$ de $Q_\pi(S,A)$ paramétrisé en θ

exemple :

$$Q_\theta(S,A) = \sum_{i=1}^{N} \phi_i(S,A)\theta_i = \Phi^T(S,A)\Theta \qquad \text{avec} \qquad \Phi(S,A) = \left(\phi_1(S,A) \ \dots \ \phi_n(S,A)\right)^T$$

$$A \in \mathbb{R}_{\mathbb{1}} \quad \Phi_1(S,A) = S, \quad \Phi_2(S,A) = A, \quad \Phi_3(S,A) = SA$$

Exemple : $Q_\theta(S,A)$ peut également être implanté par un réseau de neurones dont les poids sont θ

3.1.1 - Gradient stochastique

Pour ajuster θ, il faut un critère. On considère une politique Π. On recherche la valeur de θ qui minimise :

$$J(\theta) = \frac{1}{2} E_{(S,A)\approx\Pi}((Q_\Pi(s,a) - Q_\theta(s,a))^2)$$

$E_{(S,A)\approx\Pi}$ est l'espérance considérant que la politique Π est appliquée sur le système. On cherche θ à l'aide d'un algorithme de descente du gradient.

$$\nabla_\theta J(\theta) = \frac{dJ(\theta)}{d\theta} = \frac{1}{2}\nabla_\theta E_{(S,A)\approx\Pi}((Q_\Pi(s,a) - Q_\theta(s,a))^2)$$
$$\nabla_\theta J(\theta) = -E_{(S,A)\approx\Pi}((Q_\Pi(s,a) - Q_\theta(s,a))\nabla_\theta Q_\theta(S,A))$$

Itération de l'algorithme de descente :

$$\theta_{n+1} = \theta_n - \alpha \nabla_\theta J(\theta)$$

Il y a plusieurs problèmes à résoudre :
- $Q_\Pi(s,a)$ n'est pas connu.
- Calculer $E_{(S,A)\approx\Pi}$ nécessite la dynamique de l'environnement (inconnue)
- comment choisir α

Dans l'approche du gradient stochastique, on considère une réalisation au lieu de considérer l'espérance :

$$\nabla_\theta J(\theta) = -((Q_\Pi(s,a) - Q_\theta(s,a))\nabla_\theta Q_\theta(S,A))$$

Comme on ne connaît pas $Q_\Pi(s,a)$, on l'estime par bootstrap.
- $Q_\Pi(s,a) = r + \gamma \sum_{a'} \pi(a'|s') Q_\theta(s',a')$ (approximation de la fonction valeur de l'action associée à π)
- $Q^*(s,a) \approx r + \gamma \max_{a'} Q_\theta(s',a')$ (approximation de la fonction valeur de l'action associée à $\pi*$)

3.1.2 - Replay Buffer

Le replay buffer va stocker des episodes, c'est-à-dire des successios (s,a,r,s') observées via des interactions avec l'environnement. On note $\mathscr{D}$ le replay buffer.

Lorsqu'une mise à jour de θ doit être faite, on sélectionne un sous ensemble $\mathscr{D}' \in \mathscr{D}$ d'experiences et on calcule

$$\nabla_\theta J(\theta) \approx \sum_{(s,a,r,s')\in\mathscr{D}} \left[\left(r + \gamma \max_{a'} Q_\theta(s',a') - Q_\theta(s,a)\right) \nabla_\theta Q_\theta(s,a)\right]$$

3.2 - Neural Fitted Q-learning

Cet algorithme estime Q* à l'aide
- d'un replay buffer
- de bootstrap

<u>Algorithme NFQ</u>

- Initialisation : On considère $Q_\theta(s,a)$ avec θ aléatoire, replay buffer $\mathscr{D}$ vide.
- Pour chaque épisode faire :
  - $s = s_0$
  - faire tant que s n'est pas terminal :
    - choisir a avec une stratégie ϵ-gloutonne déduite de Q_θ appliquer a et observer s' et r.
    - $\mathscr{D} = \mathscr{D} \cup \{(s,a,r,s')\}$

- Fabrication d'un apprentissage supervisé :
- Pour chaque $(s,a,r,s') \in \mathscr{D}'$ faire :
  - $x^{(i)} = s$
  - Si s' n'est pas terminal alors :
    - $y^{(i)}(a) = r$
    - $y^{(i)}(a') = Q_\theta(s,a')$ pour tout $a' \in A$ et $a \neq a'$
  - Sinon
    - $y^{(i)}(a) = r + \gamma \max_{a'} Q_\theta(s',a')$
    - $y^{(i)}(a') = Q_\theta(s,a')$
  - i = i + 1
- mettre à jour θ à l'aide de $T = \{(x^{(i)}, y^{(i)})\}_{i=1,\dots,|\tau|}$
- s= s'

3.3 - Deep Q learning (DQN) mnih et al (2015), Nature

Voir Rethmiller et al. 2015 DQN utilise deux réseaux de neurones

- $Q_{\theta-}(s,a)$: mis à jour avec une période importante
- $Q_\theta(s,a)$: mis à jour plus fréquament

$Q_{\theta-}(s,a)$ est utilisé pour réaliser le bootstrap. les mises à jour de θ se feront à l'aide de :

$$\nabla_\theta J(\theta) = -\sum_{(s,a,r,s')\in\mathscr{D}} \left[\left(r + \gamma \max_{a'} Q_\theta(s',a') - Q_\theta(s,a)\right) \nabla_\theta Q_\theta(s,a)\right]$$

<u>Algorithm DQN</u>

Initialisation : On considère $Q_\theta(s,a)$ avec θ aléatoire, $Q_{\theta-}(s,a)$ avec $\theta-$ aléatoire et $\mathscr{D} = \emptyset$
Pour chaque épisode faire :
- $s = s_0$
- faire tant que s n'est pas terminal :
  - choisir a avec une stratégie ϵ-gloutonne déduite de Q_θ appliquer a et observer s' et r.
  - $\mathscr{D} = \mathscr{D} \cup \{(s,a,r,s')\}$
- Fabrication d'un apprentissage supervisé :
- Pour chaque $(s,a,r,s') \in \mathscr{D}'$ faire :
  - $x^{(i)} = s$
  - Si s' n'est pas terminal alors :
    - $y^{(i)}(a) = r$
    - $y^{(i)}(a') = Q_\theta(s,a')$ pour tout $a' \in A$ et $a \neq a'$
  - Sinon
    - $y^{(i)}(a) = r + \gamma \max_{a'} Q_\theta(s',a')$
    - $y^{(i)}(a') = Q_\theta(s,a')$
  - i = i + 1
- mettre à jour θ à l'aide de $T = \{(x^{(i)}, y^{(i)})\}_{i=1,\dots,|\tau|}$
- s= s'

Tous les p épisodes : $\theta- = \theta$

$\mathscr{D}'$ est construit en sélectionnant aléatoirement les expériences de $\mathscr{D}$. Avec le prioritised replay, on associe une probabilité de selection plus élevée aux meilleurs experiences, c'est à dire celles pour lesquelles $r + \gamma \max_{a'} Q_\theta(s', a') - Q_\theta(s, a)$ est faible. On associe à chaque experience une priorité p_i, proportionnelle à l'erreur. La probabilité de selection est

$$\frac{p_i^\beta}{\sum_{j=1}^m p_j^\beta} \qquad \text{avec } \beta \text{ réglant le niveau de priorité}$$

$\beta = 0 \Longleftrightarrow$ selection uniforme.