Université de Lille
Faculté des Sciences et Technologies

Offline Reinforcement Learning

Cheick Tidiane FOFANA

Master Informatique
Master mention Informatique

Département d'Informatique
Faculté des Sciences et Technologies

octobre, 2022

Ce mémoire satisfait partiellement les pré-requis du module de Mémoire de Master, pour la 2e année du Master mention Informatique.

Candidat: Cheick Tidiane FOFANA, Nº 11807017,
cheick.fofana2.etu@univ-lille.fr

Encadrant(e): Philippe PREUX, philippe.preux@univ-lille.fr

Département d'Informatique
Faculté des Sciences et Technologies
Campus Cité Scientifique, Bât. M3 extension, 59655 Villeneuve-d'Ascq

octobre, 2022

Je dédie ce travail à ma famille, elle qui m'a doté d'une éducation digne, son amour a fait de moi ce que je suis aujourd'hui : Particulièrement à mon père Boubacar FOFANA, pour le goût à l'effort qu'il a suscité en moi, de par sa rigueur. À toi ma mère Mari BA, ceci est ma profonde gratitude pour ton éternel amour, que ce mémoire soit le meilleur cadeau que je puisse t'offrir. À vous, mes frères (Mamadou Alpha et Amadou Mouctar) et sœur (Ramatoulaye) qui m'avez toujours soutenu et encouragé durant ces années d'études.

Remerciements

Ce présent mémoire est le fruit de nombreuses heures de travail, durant ces deux dernières années de master. Une période riche en partage, échanges formations/-d'informations, longues recherches et dont la finalité n'aurait pas été possible sans la participation de nombreuses personnes. Je souhaite ainsi adresser mes remerciements à toutes ces personnes pour tout ce qu'elles m'ont apporté dans la réalisation de mes travaux, et dans l'aboutissement de ce modeste mémoire.

Je souhaiterais en premier lieu, remercier monsieur Philippe PREUX, professeur d'informatique à l'univeristé de Lille et aussi mon directeur de mémoire, pour sa patience et son accompagnement. Ses judicieux conseils m'ont énormément aidé dans ma réflexion, et m'ont permis d'orienter encore plus facilement mes recherches. Je n'occulte par ailleurs pas sa disponibilité concernant le suivi de mes travaux pendant la durée de mes travaux.

Je tiens ensuite à remercier, monsieur Marc Tommasi, responsable du master Machine Learning pour sa disponibilité durant ces deux années. Mes sincères gratitudes vont également vers le corps académique du département informatique, pour la qualité de l'enseignement qu'ils prodiguent, les bons conseils durant ces cinq belles années académiques, mais surtout le travail qu'ils fournissent au quotidien pour créer cet environnement favorable à l'épanouissement académique de tous ceux qui y étudie.

J'adresse ma reconnaissance particulière à mes parents ainsi qu'à mes proches, frères, sœurs, cousins, amis, les autres familles, pour leur soutien, leurs efforts et leur aide quotidienne, tant moralement que matériellement ou financièrement, qui m'ont permis de mener à terme mes travaux, et ce, dans les meilleures conditions possibles.

En dernier lieu, je voudrais exprimer ma profonde gratitude à toutes les autres personnes qui, même sans être citées personnellement, par leurs conseils et leurs diverses compétences, ont tout autant contribué à la réalisation de ce mémoire.

Résumé

L'apprentissage par renforcement est une branche de l'apprentissage automatique qui vise à apprendre séquentiellement à un système autonome comment résoudre un problème à partir d'expériences. Le système est plongé au sein d'un environnement, et prend des décisions en fonction de son état courant. En retour, l'environnement procure une récompense à ce système qui adapte son comportement de façon à optimiser une récompense quantitative au cours du temps. Dans la majorité des problèmes du monde réel impliquant l'apprentissage par renforcement, interagir avec l'environnement peut être trop dangereux, trop coûteux voir impossible. L'apprentissage par renforcement hors ligne (*Offline Reinforcement Learning*) est une sous-catégorie de l'apprentissage par renforcement consistant à s'appuyer sur un jeu de données statique issu de précédentes interactions avec cet environnement. Il s'agit d'une solution de choix aux problèmes évoqués plus haut, cependant, il ne s'agit pas d'une tâche facile. Ce mémoire présente l'apprentissage par renforcement hors ligne et les principales difficultés que cela implique ainsi que les solutions proposées. Nous présenterons ensuite des résultats issus d'expériences réalisées dans les environnements à la fois continus et discrets afin d'appuyer les aspects théoriques mentionnés.

Mots-clés : Apprentissage automatique, Apprentissage par renforcement, Apprentissage par renforcement hors ligne, Intelligence Artificielle.

Abstract

Reinforcement learning is a branch of machine learning that aims to sequentially teach an autonomous system how to solve a problem based on experiments. The system is immersed in an environment, and makes decisions based on its current state. In return, the environment provides a reward to this system, which adapts its behavior in order to optimize a quantitative reward over time. In most real-world problems involving reinforcement learning, interacting with the environment may be too dangerous, too expensive or even impossible. Offline Reinforcement Learning is a subcategory of reinforcement learning that relies on a static data set from previous interactions with the environment. It is a solution of choice for the problems mentioned above, however it is not an easy task. This paper presents offline reinforcement learning and the main difficulties as well as the proposed solutions. We will then present results from experiments performed in both continuous and discrete environments to support the theoretical aspects discussed.

Keywords: Machine Learning, Reinforcement Learning, Offline Reinforcement Learning, Batch Reinforcement Learning, Artificial Intelligence.

Indice

Introduction

Les avancées récentes de l'apprentissage par renforcement (RL), notamment avec l'utilisation de fonctions d'approximations plus expressives avec les réseaux de neurones ont permis d'obtenir des résultats très prometteurs dans différents domaines. Nous pouvons citer non exhaustivement les domaines de la robotique ou des jeux vidéos. Cependant, l'apprentissage par renforcement appliqué aux problèmes du monde réel est beaucoup moins répandu, contrairement à l'apprentissage supervisé qui est largement adopté de nos jours du fait qu'il soit plus axé sur des données (*data-driven methods*) en constante augmentation. Cela s'explique en partie par le fait que l'apprentissage par renforcement est classiquement considéré comme un processus d'apprentissage actif, où chaque cycle d'apprentissage nécessite une interaction active avec l'environnement, ce qui en pratique pose problème. Les interactions avec le monde réel sont généralement impossible car, elles sont soit trop dangereuses/coûteuses comme dans les domaines des véhicules autonomes et de la médecine. L'apprentissage par renforcement hors ligne (*Offline RL*) offre une alternative afin de réduire l'écart entre l'apprentissage par renforcement (*online*) et les méthodes axées sur l'apprentissage supervisé à partir de grands volumes de données (*data-driven learning*). Cette approche permet ainsi d'utiliser de grands volumes de données recueillis dans le but d'entraîner des systèmes autonomes. Cette approche permet en principe d'exploiter de grands ensembles de données, mais dans la pratique, les méthodes d'apprentissage par renforcement hors ligne rencontrent des difficultés, plus particulièrement, le décalage distributionnel (*distributional shift*). Dans ce papier, nous étudierons dans un premier temps l'apprentissage par renforcement hors ligne proprement dit, les problèmes qu'il implique et quelques solutions proposées dans la littérature. Ensuite, nous passerons sur une seconde partie plus expérimentale où nous évaluerons les performances de plusieurs algorithmes d'apprentissage hors ligne sur différents environnements.

Chapitre 1

Apprentissage par Renforcement Hors-Ligne : Notions Indispensables et Domaines d'Application

Dans cette section, nous allons brièvement présenter les bases de l'apprentissage par renforcement tout en prenant le soin de bien définir les notations. Cela nous permettra ensuite d'introduire l'apprentissage par renforcement hors ligne et de mettre en évidence sa principale difficulté. Enfin, nous présenterons quelques exemples d'utilisation de l'apprentissage par renforcement hors-ligne dans différents domaines.

1.1 Apprentissage par Renforcement

Nous définissons tout d'abord un processus de décision de Markov (MDP) par un tuple $(S, A, P, R, \gamma, d_0)$. S, A représentant respectivement les espaces d'états et d'actions. $P : S \times A \times S \rightarrow [0, 1]$ représente la fonction de transition exprimant la dynamique du système et $R : S \times A \rightarrow R$, la fonction de retour associée à chaque paire état-action. γ et d_0 représentent respectivement le discount factor et la distribution initiale des états. L'objectif de l'apprentissage par renforcement est de maximiser la somme des récompenses attendues le long d'une trajectoire τ en trouvant la politique optimale π^* associé à un MDP :

$$\pi^* = \underset{\pi}{argmax} \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{1.1}$$

Plusieurs approches d'apprentissage par renforcement existent afin de résoudre l'équation (1.1). Nous présentons brièvement chacune de ces approches. Les implémentations de ces différentes approches suivent globalement la même boucle d'apprentissage. Nous avons un agent qui interagit avec le MDP en utilisant une politique π et qui, en observant l'état actuel s_t, sélectionne une action a_t. Cette action est ensuite exécutée et entraîne l'obtention d'un état suivant s_t défini par la fonction de transition P et d'un gain r associé à l'exécution de a_t en s_t. Cette boucle est itérée plusieurs fois et les transitions observées (s_t, a_t, s_t+1, r_t) sont ensuite (sauvegardées) utilisées pour mettre à jour la politique de l'agent. Dans la suite, nous utiliserons $\mathcal{D} = (s_t^i, a_t^i, s_{t+1}^i, r_t^i)$ pour faire référence à l'ensemble des transitions disponibles par l'agent.

Policy Gradient. La première approche vise à directement apprendre une politique π_θ paramétrée par un ensemble de paramètres θ. Ces paramètres peuvent typiquement être représentés par les poids d'un réseau de neurones. Ainsi, en faisant varier θ, nous pouvons estimer le gradient de notre politique π_θ(Équation 1.3) et minimiser la fonction d'objective l'équation de (1.2).

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{1.2}$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s_t \sim d_t^\pi, a_t \sim \pi_\theta(a_t|s_t)} \left[\gamma^t \nabla_\theta \log \pi_\theta(a_t|s_t) \hat{\mathrm{A}}(s_t, a_t) \right] \tag{1.3}$$

Programmation Dynamique. Cette seconde approche consiste à apprendre des estimations des fonctions valeur (V_π) et qualité (Q_π) d'une politique π afin d'en extraire une politique optimale. La fonction valeur $V_\pi(s_t)$(Equation 1.4) associée à un état s et une politique π correspond aux gains cumulés que l'on espère recevoir en suivant la politique π en démarrant en s_t. Quant à la fonction qualité $Q_\pi(s_t, a_t)$(Equation 1.5), elle correspond aux gains cumulés espérés en commençant en s_t, puis en effectuant l'action a_t avant de suivre la politique π. Plus informellement, la valeur associée à un état indique s'il est bon de passer par cet état pour optimiser la fonction objective et la qualité d'une paire *état-action* indique s'il est bon d'exécuter une certaine action dans un certain état.

$$V_\pi(s_t) = \mathbb{E}_{\tau \sim p_\pi(\tau|s_t)} \left[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_t, a_t) \right] \tag{1.4}$$

$$Q_\pi(s_t, a_t) = \mathbb{E}_{\tau \sim p_\pi(\tau|s_t, a_t)} \left[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_t, a_t) \right] \tag{1.5}$$

Nous pouvons aussi exprimer V_π et Q_π en utilisant l'opérateur de Bellman $\mathcal{B}_\pi$. Cet opérateur de Bellman admet un unique point fixe et permet ainsi de garantir la convergence des estimations faites pour V ou Q vers leur vraie valeur en appliquant plusieurs fois cet opérateur. Cette approche de programmation dynamique est à la base des algorithmes de Q-learning visant à apprendre la fonction Q et d'en extraire une politique. L'approche d'apprentissage de la politique peut être combinée avec l'apprentissage de la fonction valeur, cela donne naissance aux méthodes acteur-critique. Ces méthodes optimisent à la fois un approximateur de fonction (acteur) qui représente la politique en sélectionnant les actions à effectuer, et une fonction valeur (critique) qui va évaluer les actions prises par cet acteur.

Model-Based Reinforcement Learning. Cette dernière approche englobe l'ensemble des méthodes qui visent à apprendre itérativement la dynamique de l'environnement en approximant une fonction de transition P_θ paramétrable. Cette fonction pourra ensuite être utilisée pour résoudre un problème de plannification associé au MDP. Les méthodes à base de modèle peuvent parfois associées avec des méthodes de programmation dynamique.

1.1.1 Apprentissage par Renforcement Hors-Ligne

L'apprentissage par renforcement hors ligne, également appelé *Batch Reinforcement Learning*, est une variante de l'apprentissage par renforcement qui consiste à apprendre à partir d'un jeu de données fixe sans exploration. En d'autres termes, comment exploiter au maximum un ensemble de données statiques ? Cette méthode apprentissage consiste donc à déterminer la meilleure politique possible et de pouvoir comprendre les dynamiques de l'environnement (capacité de généralisation) à travers d'un ensemble de données que nous noterons par $\mathcal{D} = (s_t^i, a_t^i, s_{t+1}^i, r_t^i)$. Ces données peuvent être générées par une politique ou plusieurs politiques que nous noterons π_β(*behavior policy*).

Cependant, plusieurs algorithmes d'apprentissage *off-policy* utilisant des données précédemment générées lors de leur apprentissage, comme DQN, échouent lorsqu'il s'agit d'apprendre uniquement à travers des données statiques sans exploration, ce qui fait tout de même de cet apprentissage une tâche difficile. L'objectif étant de faire mieux que la politique génératrice, cela peut impliquer que l'agent prenne des décisions non observées dans D. L'agent n'aura cependant pas de retour sur la réelle qualité de ses actions et ne pourra pas corriger sa fonction d'approximation.

En effet, la distribution des états visités par l'agent dépend fortement de sa politique génératrice. Lorsque l'agent visite des états qui sont loin de ceux observés dans la distribution d'apprentissage, même avec de fortes bases inductives, il n'y a aucune garantie de convergence. Cela a pour conséquence en pratique que l'agent se retrouve dans des états jamais visités en effectuant des actions non optimales, récursivement, cela conduit à de mauvaises performances lors de l'évaluation. Il s'agit de la principale difficulté de cet apprentissage connu sous le nom de *distributional shif* ou *Boostrapping Error Accumulation*. Le problème de l'exploration ne sera pas abordé, car il n'y a aucune solution à ce problème. Dans la suite, nous allons voir les différentes solutions proposées dans la littérature.

1.2 Domaines d'Applications

Ici, nous allons présenter quelques domaines d'applications pour lesquelles l'apprentissage par renforcement hors ligne est prometteur. Comme évoqué précédemment, l'adoption de l'apprentissage par renforcement est principalement freiné par le fait que l'entraînement nécessite des interactions avec l'environnement, ce qui est potentiellement dangereux. Nous citerons les domaines suivants :

Médical. L'apprentissage par renforcement hors ligne a été utilisé afin d'entraîner un agent à stimuler en temps réel certaines zones du cerveau d'une souris dans le but de réduire les crises épileptiques [1].

Langage humain. Une discussion entre individus peut être modélisée comme un MDP. Ensuite les données de ces échanges manuscrits ou audios entre plusieurs individus pourraient constituer des données afin d'entraîner un algorithme d'apprentissage par renforcement hors ligne pour remplacer les humains sur certaines tâches, comme l'assistance sur un site web [2].

Recommandations. Les moteurs de recommandations aussi sont un domaine dans lequel l'apprentissage par renforcement hors ligne est utilisé. Les données de navigations (like, favoris) sont utilisées pour apprendre les centres d'intérêt des utilisateurs. Cela permet d'éviter des grosses dépenses de publicité à suggérer aux utilisateurs en attendant leur *feedback* [3].

Robotique. L'apprentissage par renforcement actif est souvent utilisé en robotique. Dans certains cas, on peut être amené à apprendre des politiques pour une variété de compétences robotiques qui se généralisent efficacement dans différents environnements et contextes. Par exemple, apprendre à un robot à ramasser des objets dans des environnements différents (salle de cuisine de différentes maisons) [4].

Véhicule autonome. L'un des domaines les plus prometteurs est le domaine de la conduite autonome de véhicule. Une quantité non négligeable de données visuelles sont disponibles représentant la conduite de véhicule par des humains. Les véhicules autonomes de nos jours reposent sur l'apprentissage par imitation. L'apprentissage par renforcement pourrait être une alternative à l'apprentissage par imitation qui est plus fastidieuse.

Chapitre 2

Adaptation de l'Évaluation Off-Policy via l'Échantillonage Préférenciel

Dans cette section, nous aborderons les différentes méthodes de la littérature qui visent à estimer les retours espérés d'une politique. Elles utilisent l'échantillonnage préférentiel ou *Importance Sampling* afin d'évaluer une politique ou d'en estimer le gradient. Ces méthodes peuvent être utilisées afin d'estimer $J(\pi_\theta)$ (*off-policy evaluation*) à travers les échantillons de $\mathcal{D}$ généré par π_β(*behavior policy*), ou d'en estimer le gradient pour obtenir version une hors ligne des méthodes à base de gradient (*Policy Gradient*). Nous verrons comment ces méthodes qui sont initialement destinées à l'apprentissage par renforcement *online* sont utilisées dans la configuration Offline et quelles en sont les limites.

2.1 Évaluation Off-Policy via l'Échantillonage Préférenciel

Pour rappel, l'évaluation de la politique consiste à estimer les récompenses futures attendues en suivant une politique π. Ces retours sont utilisé dans la majorité des méthodes basées sur la politique telle que *policy iteration*. Classiquement, ils sont estimés en interagissant avec l'environnement selon π. Cependant, lorsqu'il est impossible d'utiliser directement cette politique π pour déterminer ces récompenses cumulées , il est possible d'utiliser des échantillons collectés par une autre politique π_β en utilisant la méthode de *importance sampling*. On parle donc de *off-policy evaluation*(OPE).

Importance sampling permet d'estimer l'espérance d'une variable aléatoire x distribuée selon *f(x)* par des échantillons tirés d'une autre distribution *g(x)*. Ainsi, nous pouvons déterminer un estimateur de $J(\pi_\theta)$ suivant :

$$\begin{aligned} J\left(\pi_\theta\right) &= \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\frac{\pi_\theta(\tau)}{\pi_\beta(\tau)}\sum_{t=0}^{T}\gamma^t r(\mathbf{s},\mathbf{a})\right] \\ &= \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\left(\prod_{t=0}^{T}\frac{\pi_\theta\left(\mathbf{a}_t \mid \mathbf{s}_t\right)}{\pi_\beta\left(\mathbf{a}_t \mid \mathbf{s}_t\right)}\right)\sum_{t=0}^{H}\gamma^t r(\mathbf{s},\mathbf{a})\right] \approx \sum_{i=1}^{n} w_T^i \sum_{t=0}^{T}\gamma^t r_t^i \end{aligned} \tag{2.1}$$

avec $w_t^i = \frac{1}{n}\prod_{t'=0}^{t}\frac{\pi_\theta\left(\mathbf{a}_{t'}^i|\mathbf{s}_{t'}^i\right)}{\pi_\beta\left(\mathbf{a}_{t'}^i|\mathbf{s}_{t'}^i\right)}$ et $(s_0^i, a_0^i, r_0^i, s_1^i, ...)_{i=1}^n$, n étant un nombre de trajectoire généré par π_β et accessible dans $\mathcal{D}$. w_t^i correspond au *importance weight* associé à l'estimation i. Ce terme agit comme un régulateur en augmentant l'importance des gains cumulés pour les trajectoires plus susceptibles d'être obtenues avec la politique cible, et diminue celles qui sont moins probables d'être empruntées. Ces multiplications de poids entraînent une grande variance. Deux solutions ont été proposées par [5] dont la première consiste à utiliser le fait que r_t ne dépend pas de $s_{t'}$ et $a_{t'}$ pour $t' > t$, ainsi les poids $w_{t'}^i$ peuvent être ignorés (*per-decision importance sampling estimator*). La seconde solution, quant à elle consiste à normaliser ces poids (*weighted importance sampling*). Cependant, cette solution réduit la variance au détriment du biais. Ces deux solutions fusionnées génèrent l'estimateur *weighted per-decision importance estimator*. Lorsque ces estimateurs précédents sont couplés avec un approximateur de $\hat{Q}_\pi(s_t, a_t)$, ils permettent d'obtenir des estimateurs plus consistants (*doubly robust estimator*[6]).

Au-delà de la cohérence ou des estimations non biaisées, l'obtention d'une garantie (à forte probabilité) sur la performance d'une politique est souvent souhaitable. Des garanties de confiance peuvent être mises en place en faisant des hypothèses probabilistes ou avec du *Bootstrapping*.

2.2 Off-Policy Policy Gradient avec L'échantillonage préférenciel

L'estimation du gradient de la politique peut aussi être adaptée à la configuration *Off-Policy/Offline* en utilisant *Importance Sampling*. L'estimation ∇_θ (Equation 1.3) est classiquement obtenue avec les trajectoires générées par la politique courante. Dans la configuration *Off-Policy*, comme précédemment, des poids sont ajoutés afin d'équilibrer les gradients.

$$\begin{aligned}\nabla_\theta J(\pi_\theta) &= \mathbb{E}_{\tau\sim\pi_\beta(\tau)}\left[\frac{\pi_\theta(\tau)}{\pi_\beta(\tau)}\sum_{t=0}^{T}\gamma^t\nabla_\theta\log\pi_\theta(\mathbf{a}_t\mid\mathbf{s}_t)\,\hat{A}(\mathbf{s}_t,\mathbf{a}_t)\right]\\ &\approx\sum_{i=1}^{n}w_T^i\sum_{t=0}^{H}\gamma^t\nabla_\theta\log\pi_\theta\left(\mathbf{a}_t^i\mid\mathbf{s}_t^i\right)\hat{A}\left(\mathbf{s}_t^i,\mathbf{a}_t^i\right),\end{aligned} \tag{2.2}$$

avec $w_t^i = \frac{1}{n}\prod_{t'=0}^{t}\frac{\pi_\theta\left(\mathbf{a}_{t'}^i|\mathbf{s}_{t'}^i\right)}{\pi_\beta\left(\mathbf{a}_{t'}^i|\mathbf{s}_{t'}^i\right)}$ et $\hat{A}$ la fonction d'avantage. Comme précédemment, cet estimateur du gradient est sujet à une grande variance. Les solutions proposées plus haut sont aussi applicables à cet estimateur. Malgré ces améliorations, l'estimateur est toujours sujet à une variance élevée. Ceci s'explique par le fait que dans une configuration *Offline*, la politique cible π_θ puisse être éloignée de la politique génératrice π_β à l'origine $\mathcal{D}$. L'ajout d'un terme de régularisation $\lambda\left(\log\sum_{i=1}^{n}w_T^i\right)$ à l'équation (2.2) est courant afin d'éviter que la politique cible π_θ puisse trop s'éloigner de π_β.

Une seconde approche [7] consiste à déterminer une approximation de gradient de politique en se basant sur la distribution des états $d^{\pi_\beta}(s)$ induit par la politique génératrice π_β. En effet, l'évaluation de la politique consiste à estimer $V_\pi(s)$ pour chaque état en suivant la politique π. La fonction objective correspondante est $J(\pi_\theta) = \mathbb{E}_{s\sim d^{\pi\theta}(s)}[V_{\pi_\theta}(s)]$. Cependant, n'étant pas possible d'utiliser la distribution des états $d^\pi(s)$, nous pouvons la remplacer par la distribution $d^{\pi_\beta}(s)$. Il en résulte un gradient biaisé en raison de l'inadéquation des distributions d'états, lequel fournisse tout de même des performances raisonnables en pratique. La fonction objective mise à jour devient alors $J(\pi_\theta) = \mathbb{E}_{s\sim d^{\pi_\beta}(s)}[V_{\pi_\theta}(s)]$. Les retours estimés avec cette nouvelle fonction objective peuvent être calculés directement à partir de $\mathcal{D}$ et permettent de se passer de l'échantillonage préférentiel. En conséquence, l'approximation de l'estimation ∇_θ se formule ainsi :

$$\begin{aligned}\nabla_\theta J_{\pi_\beta}(\pi_\theta) &= \mathbb{E}_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s}),\mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s})}\left[Q^{\pi_\theta}(\mathbf{s},\mathbf{a})\nabla_\theta\log\pi_\theta(\mathbf{a}\mid\mathbf{s})+\nabla_\theta Q^{\pi_\theta}(\mathbf{s},\mathbf{a})\right]\\ &\approx\mathbb{E}_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s}),\mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s})}\left[Q^{\pi_\theta}(\mathbf{s},\mathbf{a})\nabla_\theta\log\pi_\theta(\mathbf{a}\mid\mathbf{s})\right]\end{aligned} \tag{2.3}$$

Dans cette section, nous avons vu que les méthodes présentées posent un certain nombre de problèmes majeurs. Premièrement, l'échantillonnage préférentiel souffre d'une variance élevée, qui est exacerbée par la multiplication des *importance weights* au fur des itérations. La méthode d'approximation (utilisant la distribution des états) et des techniques de normalisation/régularisation atténuent ce problème en limitant ces multiplications. En revanche, lorsque la politique apprise π_θ est trop divergente de la politique génératrice π_β, la variance des estimateurs est trop élevée et devient inutilisable, en particulier dans les espaces d'état et d'action de haute dimension ou pour les problèmes à long horizon. Ces estimateurs sont plus pratiques dans la configuration *Off-Policy*, car la politique est mise à jour itérativement par rapport à la politique génératrice qui, elle aussi, évolue dans le temps, contrairement à la configuration *Offline* où la politique peut être éloignée.

Chapitre 3

Apprentissage par renforcement hors ligne : Programmation Dynamique

Dans le chapitre précédent, les méthodes permettant d'estimer les retours moyens et le gradient à partir d'un jeu de données statiques ont été présentées ainsi que leurs limites. Dans cette section, nous présenterons les approches de programmation dynamiques appliquées à la configuration *Offline*.

Les méthodes de programmation dynamique, telles que les algorithmes basés sur l'apprentissage de la fonction qualité, peuvent en principe offrir une option plus intéressante pour l'apprentissage par renforcement hors ligne plutôt que l'estimation des gradients de politique. Ces méthodes de programmation dynamique visent à apprendre une fonction de valeur d'état ou d'état-action, puis d'utiliser cette fonction de valeur pour extraire directement la politique optimale, ou dans le cas des méthodes *actor-critic*, d'utiliser cette fonction de valeur pour estimer un gradient pour les retours d'une politique. Pour des raisons de complétude, nous présenterons une première approche d'estimation de la fonction de valeur avec des fonctions linéaires. Ensuite, nous présenterons comment le décalage distributionnel (*distributional shift*) affecte les méthodes classiques comme le Q-learning et les différentes solutions proposées, soit les méthodes de restriction de la politique apprise et les méthodes basées sur l'estimation de l'incertitude.

3.1 Estimation Off-Policy de la fonction de valeur par des fonctions linéaires

Alors que les méthodes modernes d'apprentissage par renforcement évitent généralement les estimateurs linéaires (en faveur des estimateurs non linéaires plus expressifs) , les méthodes linéaires constituent tout de même une classe importante d'algorithmes d'apprentissage par renforcement hors ligne. Ci-dessous, nous présenterons leur fonctionnement global.

Ces algorithmes visent à approximer la fonction de qualité Q par une fonction linéaire paramétrée par ϕ qui satisfait l'équation de Bellman. Plus formellement, cette fonction peut être définie ainsi : $Q_\phi \approx f(s,a)^T\phi$, avec $f(s,a) \in \mathbb{R}^d$ représentant le vecteur de *features* associé à une paire d'état-action. Q^π_ϕ peut ensuite être estimée pour extraire une meilleure politique. La principale difficulté consiste à déterminer les paramètres ϕ afin de se rapprocher de la vraie fonction de Q. La solution consiste à utiliser les propriétés du problème. Lorsque l'approximateur de fonction utilisée pour représenter Q_ϕ est linéaire, Q peut être représenté comme la solution d'un système linéaire [8]. En exprimant Q_ϕ avec l'opérateur de Bellman $\mathcal{B}$ (qui est linéaire), on peut ainsi déterminer ϕ avec la méthode de minimisation des moindres carrés (minisation de l'erreur de bellman). Par conséquent, la solution obtenue sous forme vectorielle est la suivante :

$$\mathbf{F}\phi \approx \mathcal{B}^\pi \mathbf{F}\phi = \vec{R} + \gamma P^\pi \mathbf{F}\phi \Longrightarrow (\mathbf{F} - \gamma P^\pi \mathbf{F})\,\phi \approx \vec{R}. \tag{3.1}$$

$$\phi = \left((\mathbf{F} - \gamma P^\pi \mathbf{F})^T (\mathbf{F} - \gamma P^\pi \mathbf{F})\right)^{-1} (\mathbf{F} - \gamma P^\pi \mathbf{F})^T \vec{R}. \tag{3.2}$$

Une seconde approche consiste à itérer l'opérateur de Bellman, plutôt que de minimiser directement l'erreur de Bellman. Nous savons que $Q_{k+1} \to \mathcal{B}^\pi Q_k$ convergence vers la vraie valeur de Q lorsque $k \leftarrow \infty$. Cela n'est pas directement applicable à $Q_{k+1} = \mathcal{B}^\pi F_{\phi_k}$, puisqu'il se peut qu'il n'y ait aucun ϕ tel que cette équation soit vraie. Nous devons donc appliquer l'opérateur de Bellman puis projeter le résultat dans le *span* de F pour déterminer Q_{k+1}. Par soucis de concision, nous ne développerons pas plus cette approche.

Il n'y a pas de consensus clair sur la meilleure approche entre les deux. Elles donnent différentes solutions en général lorsque la vraie fonction Q^π n'est pas dans le *span* de F (c'est-à-dire qu'elle ne peut pas être représentée par un vecteur de paramètres ϕ).

L'algorithme LSTD-Q est une méthode d'estimation de Q à partir de données statiques. Il vise à déterminer itérativement l'opérateur de projection évoqué plus haut, et ensuite d'extraire ϕ de la fonction Q obtenue après convergence. Notons que l'algorithme LSTD-Q n'est pas directement applicable à l'estimation de Q^* ,puisque l'équation de Bellman optimale pour Q^* n'est pas linéaire en raison de l'opération de maximisation. LSTD-Q est utilisé dans LSPI (*Least Squared Policy Iteration*), un algorithme d'apprentissage par renforcement hors ligne qui est un algorithme d'itération de la politique. Il utilise LSTD-Q pour l'évaluation de la politique courante puis procède de manière gloutonne pour déterminer la politique suivante.

3.2 Impact du Décalage distributionnel sur les méthodes de programmation dynamique

Les méthodes de programmation dynamique, linéaire et non linéaire présentées jusqu'à présent peuvent en principe apprendre à partir de données hors ligne, collectées selon une politique génératrice différente, a priori inconnue. Cependant, en pratique, les résultats de ces méthodes ne sont pas aussi bons qu'espérés, en raison des problèmes de décalage distributionnel.

Afin d'illustrer le décalage distributionnel, nous allons étudier les performances de l'algorithme SAC (*Soft Actor-critic*) avec des démonstrations d'experts sur l'environnement *HalfCheetah-v2* réalisé par [9]. La figure de gauche ci-dessous montre la courbe d'apprentissage, et, à première vue, cela ressemble à du sur-apprentissage, car la performance d'évaluation se détériore avec plus d'entraînements (courbe verte), mais l'augmentation de la taille de l'ensemble de données ne rectifie pas le problème (orange vs vert), ce qui suggère que le problème est plus complexe.

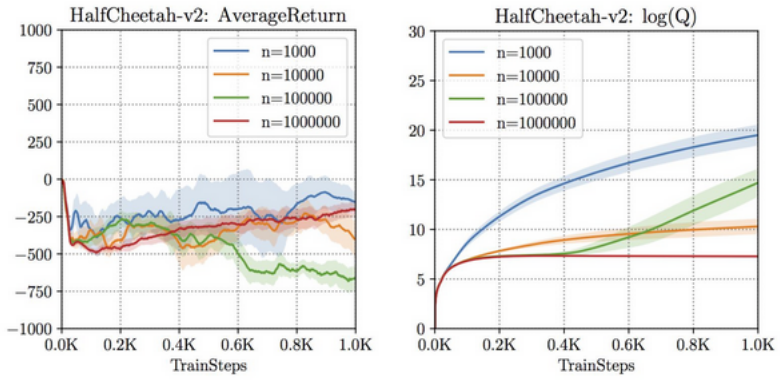


FIGURE 3.1 – Performance de l'algorithme SAC dans une configuration *Offline*

Afin d'expliquer comment le décalage distributionnel impacte les méthodes de programmation dynamique, nous allons rappeler le concept global de ces méthodes. Les algorithmes d'apprentissage par renforcement basés sur la programmation dynamique maintiennent une fonction Q paramétrique $Q_\theta(s,a)$ et, éventuellement, une politique paramétrique $\pi_\phi(a|s)$. Les méthodes basées sur le Q-learning apprennent une fonction Q en appliquant itérativement l'opérateur d'optimalité de Bellman $\mathcal{B}^*Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(s'|s,a)}[max_{a'} Q(s',a')]$, puis génère la politique suivante en sélectionnant l'action optimale pour chaque état (*greedy policy*). Dans les méthodes *actor-critic*, une politique séparée est en plus apprise pour maximiser les valeurs de Q. Ces méthodes alternent ainsi entre le calcul de Q_π via l'évaluation de la politique, en itérant l'opérateur d'optimalité de Bellman, et l'amélioration la politique $\pi(a|s)$ en l'actualisant afin d'obtenir des actions qui maximisent la valeur Q. Nous obtenons ainsi les deux équations suivantes pour l'évaluation et la mise à jour de la politique :

$$\hat{Q}^{k+1} = \arg\min_Q \mathbb{E}_{s,a,s' \sim \mathcal{D}} \left[\left((r(s,a) + \gamma \mathbb{E}_{a' \sim \hat{\pi}^k(a',s')}[\hat{Q}^k(s',a')] - Q(s,a) \right)^2 \right] \quad (3.3)$$

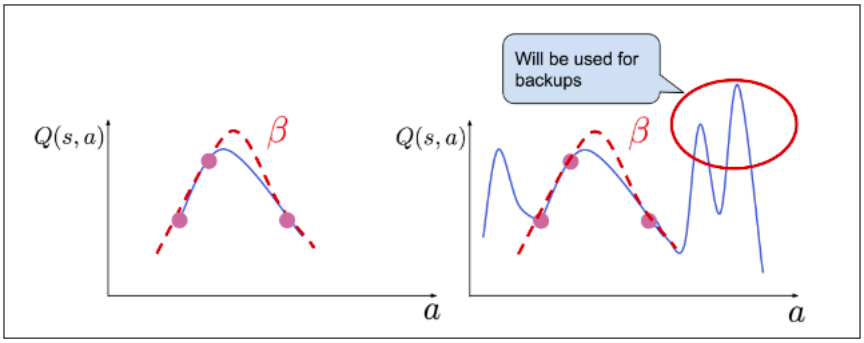
$$\hat{\pi}^{k+1} = \arg\max_\pi \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi^k(a|s)} \left[\hat{Q}^{k+1}(s,a) \right] \quad (3.4)$$

Dans la configuration *Offline*, cette erreur de Bellman est minimisée en utilisant les transitions de $\mathcal{D}$. Même si l'équation (3.4) correspond à un problème de régression supervisée, les *targets* pour cette minimisation sont calculées avec l'estimation de la fonction Q. Ceci permet de comprendre la source de décalage ditributionnel.

Étant donné que le calcul des *targets* dans l'équation (3.3) nécessite d'évaluer $\hat{Q}^\pi(s_{t+1}, a_{t+1})$, où $a_{t+1} \sim \hat{\pi}(a_{t+1}|s_{t+1})$, la précision des *targets* calculées dépend de l'estimation de $\hat{Q}$ pour des actions a_{t+1} qui peuvent être en dehors de la distribution des actions sur lesquelles la fonction Q a été entraînée. Lorsque la politique courante $\hat{\pi}(a|s)$ (basé sur l'estimation de $\hat{Q}$) est vraiment différente de la politique génératrice $\hat{\pi}\beta(a|s)$, les *targets* calculées sont erronées. Ce problème est exacerbé par le fait que la politique courante $\pi(a|s)$ est explicitement optimisée pour maximiser $\mathbb{E}_{a \sim \hat{\pi}(a|s)}[\hat{Q}^\pi(s,a)]$. Cela signifie que, si la politique peut produire des actions hors distribution pour lesquelles la fonction Q apprise produit à tort des valeurs excessivement grandes, elle apprendra à le faire. Ceci est d'autant plus vrai pour les méthodes *actor-critic* que les méthodes basées de Q-learning. Dans l'apprentissage par renforcement en ligne standard, ces problèmes sont corrigés naturellement lorsque la politique interagie avec l'environnement, en tentant les transitions qu'elle croit (à tort) être bonnes, et en observant qu'en fait elles ne le sont pas. Cependant, dans la configuration *offline*, la politique ne peut pas corriger de telles valeurs, Q est

trop optimistes, et ces erreurs s'accumulent à chaque itération de l'apprentissage, ce qui entraîne des mauvais résultats lors de la phase de test.

L'estimateur de la fonction Q n'est fiable que sur les actions observables dans l'ensemble d'entraînement. Par conséquent, la maximisation naïve de la valeur Q dans $Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(s'|s,a)}[max_{a'} Q(s',a')]$ peut amené l'estimateur Q a être évalué sur des actions qui se trouvent en dehors de la distribution d'entraînement (Image 3.2), ce qui par conséquent aboutit des valeurs *targets* démesurées. Nous appelons ces actions, *out-of-distribution action* (OOD). Ainsi, utiliser des valeurs de Q incorrectes pour calculer les targets entraîne des erreurs qui se propagent au cours des itérations, ce phénomène est connu sous le nom de **Boostrapping error accumulation**.



• : action visible dans le dataset $\mathcal{D}$
- - : Q(s,a) estimer par la politique génératrice π_β
– : Q(s,a) estimer par la politique courante π

FIGURE 3.2 – Évaluation de Q pour un état s et un intervalle d'action continue

Dans la suite, nous allons aborder les différentes solutions proposées afin de restreindre l'utilisation des actions hors distribution pour l'opération de backup dans l'équation (3.3). Ces solutions peuvent être scindées en plusieurs catégories que nous aborderons ci-dessous.

3.3 Différentes approches contre le problèmes des actions hors Distribution (OOD)

Nous avons vu plus haut comment le décalage distributionnel affecte les méthodes de programmation dynamique en itérativement accumulant des erreurs d'estimations. En effet, pour implémenter un algorithme efficace d'apprentissage par renforcement hors ligne, il est crucial de traiter le problème des actions hors distribution. Plusieurs travaux ont été effectués dans la littérature, nous pouvons les regrouper dans les 3 catégories présenter ci-dessous.

3.3.1 Les méthodes à base de contrainte

Les méthodes à base de contrainte [9] [10] [11] consistent à contraindre la politique apprise π à être proche de la politique génératrice π_β en restreignant la distribution des actions de $\pi(a|s)$ à être proche de celle de $\pi_\beta(a|s)$, cela permet ainsi de minimiser les erreurs d'accumulation lors de l'étape de *backup* dans l'équation 3. Il existe plusieurs méthodes qui diffèrent en termes de métrique utilisée pour définir la "proximité" entre les politiques et d'intégration de contraintes aux algorithmes.

Ces contraintes peuvent être directement ajoutées durant la mise à jour de l'agent (*explicit policy contrains / implicit policy contrains*) ou être appliquées sous forme de pénalités ajoutées à la fonction Q. Quant aux métriques utilisées pour définir la proximité, nous pouvons citer KL-divergence, $\mathcal{X}^2$-divergence ou encore MMD (*maximum mean discrepancy*). Il est aussi important de noter que le degré de liberté de ces contraintes n'est pas à négliger. Considérons un problème où la politique génératrice $\pi_\beta(a|s)$ est aléatoire, dans ce cas, l'apprentissage par renforcement hors ligne devrait en principe être très efficace. Cependant, sachant que les contraintes induites obligent la distribution apprise à être proche de celle générée, des contraintes peu malléables empêcheraient la politique apprise à trop dévier de la politique génératrice. Ainsi, une alternative proposée serait d'adapter le degré de liberté de la politique (en fonction des données) en restreignant le support de la politique apprise $\pi(a|s)$ au support de la distribution génératrice $\pi_\beta(a|s)$.

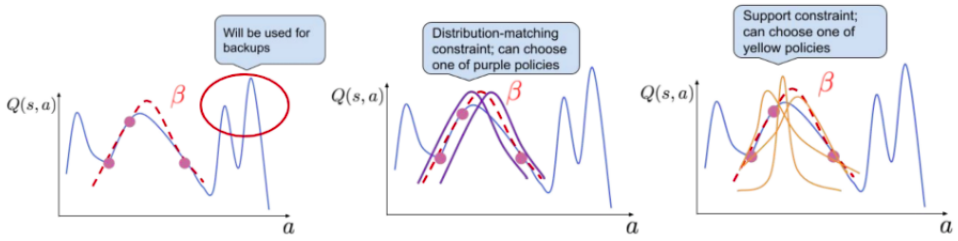


FIGURE 3.3 – Performance de l'algorithme SAC dans une configuration *Offline*

3.3.2 Les méthodes à base d'estimation d'incertitude

Précédemment, nous avons vu comment les méthodes à base de contrainte permettaient de prévenir le décalage distributionnel induit par les actions hors distributions. Une seconde approche consiste à apprendre itérativement une estimation de la certitude attribuée à l'estimation de la fonction Q pour une paire état-action. Plus formellement, cela revient à apprendre une distribution représentant l'incertitude de l'estimation faite à travers $\mathcal{D}$. Cette distribution peut être modélisée explicitement comme une distribution paramétrable (Distribution Gaussienne). Ces estimations de l'incertitude sont utilisées afin de proportionnellement réduire la sélection de ces OOD.

3.3.3 CQL : Concervative Q-learning

La dernière approche [12] consiste à utiliser directement une expression de régularisation afin d'éviter la surestimation de Q. L'avantage de cette approche est qu'elle est directement applicable sur les variantes du Q-learning et des méthodes *actor-critic* même lorsque la politique n'est pas représentée explicitement. L'objectif est d'apprendre une fonction Q conservative en ajoutant un terme de régularisation (Equation 3.5) qui minimise les valeurs de Q(s,a) sous une distribution adversariale $\mu(a|s)$ (correspondant aux actions OOD) et maximise les valeurs de Q(s,a) pour les couples état-action sous D. Cette expression de régularisation devrait permettre de pousser vers le bas les valeurs de Q pour les actions hors distribution pour lesquelles les valeurs de Q sont (potentiellement) élevées, tout en assurant des valeurs de Q correctes pour les actions *in-distribution* en minimisant l'erreur de Bellman.

$$\hat{Q}^{k+1} = \arg\min_{Q} \mathbb{E}_{s,a,s' \sim \mathcal{D}} \left[\left(Q(s,a) - \mathcal{B}^{\pi} \hat{Q}^{k}(s,a) \right)^2 \right] + \alpha \cdot \mathbb{E}_{s \sim \mathcal{D}, a \sim \mu(a|s)} [Q(s,a)] \quad (3.5)$$

Dans cette section, nous avons vu l'ensemble des approches existantes afin de prévenir et d'éviter la surestimation de Q pour les actions hors distribution. Nous avons vu les méthodes basées sur l'utilisation de contraintes, d'estimation de l'incertitude et de la régularisation.

Les méthodes à base de contrainte ont tout de même quelques limitations. Elles doivent avant tout modéliser la politique génératrice π_β, ce qui implicitement signifie que les performances de ces méthodes sont limitées par l'exactitude de la modélisation réalisée, ce qui est une tâche compliquée lorsque la politique génératrice est multimodale.

Quant aux méthodes qui estiment une fonction de valeur conservatrice, elles présentent un ensemble de compromis quelque peu différents. Bien qu'elles évitent les problèmes associés à l'estimation de la politique génératrice et permettent d'éviter la surestimation, elles peuvent souffrir tout de même de sous-estimation de Q et d'une forme de surapprentissage lorsque l'échantillon est limité.

Chapitre 4

Apprentissage par Renforcement Hors Ligne : Model Based

L'utilisation de modèles prédictifs peut être un outil puissant pour permettre un apprentissage par renforcement hors ligne efficace. Ces algorithmes d'apprentissage par renforcement reposent principalement sur leur capacité à estimer $\mathrm{P}(s_{t+1}|s_t, a_t)$ via un modèle paramétré $\mathrm{P}_\theta(s_{t+1}|s_t, a_t)$, plutôt que d'utiliser la programmation dynamique ou un échantillonnage préférentiel. Ils utilisent de méthodes d'apprentissage supervisés pratiques et puissants, qui leur permettent d'utiliser efficacement des ensembles de données importants et diversifiés. Cependant, comme les méthodes de programmation dynamique, ces méthodes ne sont pas à l'abri des effets du changement de distribution. Dans cette section, nous verrons brièvement les différentes approches appliquées à la configuration *Offline*. Par soucis de ressources suffisantes dans la littérature, cette section sera plus courte que les autres et serait peut-être un point de départ afin de pousser la recherche vers cette approche beaucoup moins commune.

L'apprentissage par renforcement *model-based* consiste à modéliser l'environnement à travers sa fonction de transition P en utilisant un modèle prédictif. Ce modèle peut ensuite être utilisé pour des tâches de planification ou pour générer une politique associée à un MDP. Dans les deux cas, le modèle fournit des prédictions précises pour les tuples état-action et souffre du décalage distributionnel induit par la distribution des états et des actions. Il souffre également d'un second effet plus subtil ; étant donné que la politique est optimisée afin de maximiser ces gains futurs en utilisant l'estimation de la fonction de transition courante P_θ, le procédé d'optimisation de la politique peut exploiter le modèle en produisant des actions hors distribution menant à des gains élevés.

Comme avec la programmation dynamique, les algorithmes *model-based* peuvent directement être utilisés dans la configuration *Offline* en adaptant légèrement les algorithmes existants. En effet, ces méthodes ont déjà prouvé leur efficacité dans la configuration *off-policy*, plus conventionnelle. Elles utilisent des estimations de l'incertitude comme précédemment afin de limiter le problème d'exploitation du modèle évoqué plus haut. L'algorithme MBPO [13] utilisé naïvement sans aucune modification donne de meilleurs résultats que les méthodes de programmation dynamique utilisées naïvement sur certains problèmes, ce qui suggère que ce type d'approche soit peut-être prometteur dans la configuration *Offline*.

Afin d'atténuer les effets du décalage de distribution, plusieurs méthodes équivalentes à celles du chapitre précédent ont été utilisées. Nous pouvons citer les algorithmes DIMs (*Deep Imitative models*) qui utilisent des contraintes sur la politique dans le but de la conserver dans des régions sûres dans l'espace d'états. Une approche plus conservative consiste à utiliser les estimations des valeurs afin de fournir des garanties de confiance aux modèles. Cette approche a été implémentée à travers deux algorithmes, MoREL[14] et MOPO [15].

Cependant, il est important de noter que ces approches *model-based* présentent des difficultés notamment dans la modélisation de certains MDP complexes à cause de la multidimensionnalité des problèmes (vidéos, images) ou encore aux longs horizons.

Chapitre 5

Partie Expérimentale : Benchmark d'algorithme d'Offline RL

Dans les chapitres évoqués plus haut, nous avons vu comment le décalage distributionnel affecte les différentes méthodes d'apprentissage appliquées à la configuration Offline. Nous avons également vu les différentes solutions proposées dans la littérature. Dans cette seconde section, nous présenterons les résultats d'expérimentations réalisés avec différents algorithmes sur différents datasets. Pour ce faire, nous présenterons les algorithmes ainsi que les datasets avant de présenter les résultats.

5.1 Caractéristiques des jeux de données

Dans l'apprentissage par renforcement hors ligne, l'apprentissage est réalisé sur un jeu de données statiques généré par une ou plusieurs politiques génératrices π_β. L'apprentissage étant uniquement réalisé sur des données statiques, aucune exploration n'est envisageable, ainsi, l'apprentissage se limite à trouver la meilleure politique pour le MDP associé aux jeux de données.

Les données générées par π_β sous la forme de tuples (s,a,s',r) peuvent être catégorisées selon deux caractéristiques : la qualité des trajectoires et la couverture des paires état-actions. Quantifier ces caractéristiques à travers le jeu de données n'est pas toujours trivial, mais il existe des proxys comme la distribution des récompenses, des actions (mesure de l'entropie de π_β). Les *datasets* devraient inclure des actions menant à des récompenses élevées (qualité des trajectoires), mais aussi à des actions sous-optimales que π_β n'aurait pas prise, menant l'agent dans d'autres paires état-action (state-action coverage). La couverture des paires état-action observées permet de pouvoir «explorer» dans la limite des données. Il existe cependant un compromis qui peut être induit par π_β entre la qualité des trajectoires et la couverture des paires états-actions. Lorsque π_β est purement aléatoire, cela conduit à un «state-action coverage» élevé, donc une bonne connaissance de l'environnement (à travers le *dataset*) et la politique apprise pourrait faire mieux que la politique π_β, sans aucune garantie que les données collectées soient assez riche pour pouvoir converger vers un agent efficace). Lorsque π_β est entièrement déterministe, le «state-action coverage» est plus faible et ainsi l'agent aura plus de mal à surpasser la politique génératrice, d'où l'importance d'avoir une politique génératrice π_β assurant des trajectoires de qualité (permettant l'exploitation) et une bonne couverture des paires état-action (permettant l'exploitation). En pratique, pour les problèmes du monde réel, les données peuvent être aussi aléatoires avec des logs d'activités, que déterministes avec des démonstrations d'experts. Pour les expérimentations dans des environnements virtuels, les *datasets* peuvent être générés en utilisant plusieurs politiques (half-trained, Q-learning, trained sarsa).

5.2 Expérimentations et Résultats

Afin d'appuyer certains aspects théoriques évoqués plus haut, des expériences ont été réalisées en entraînant et évaluant plusieurs algorithmes sur des données issues de plusieurs environnements virtuels. Dans cette section, nous présenterons les méthodes utilisées pour générer les *datasets* d'entraînement ainsi que les algorithmes utilisés avant d'analyser les résultats.

5.2.1 Environnements Virtuels et Datasets

Les expérimentations ont été réalisées sur les environnements de «**Cartpole**» (espace d'action discret) et «**Maze2D**» (espace d'action continu). «**Cartpole**», également connu sous le nom de pendule inversé, est un jeu dans lequel un agent apprend à faire tenir en équilibre un poteau en supposant qu'à la pointe du poteau, il y a un objet qui le rend instable. Pour générer le jeu de données, un agent a été entraîné jusqu'à convergence (Voir Image 5.1) en utilisant l'algorithme DQN. Les épisodes générés ont ensuite été répartis en 3 *datasets* selon la phase d'apprentissage de l'agent tout en s'assurant de l'équilibrage de la taille des *datasets*. Les datasets obtenus sont les suivants : «*Cartpole-early-stage-learning*» (episode 0-24), «*Cartpole-mid-stage-learning*» (episode 25-50), «*Cartpole-Expert-stage-learning*» (episode 51-76).

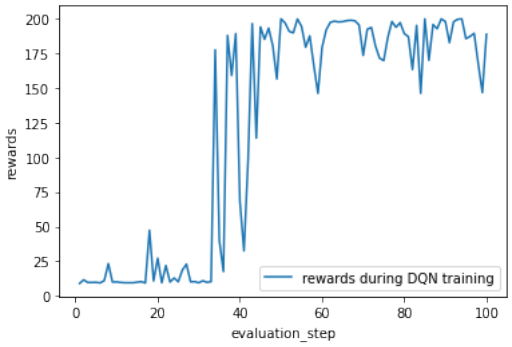


FIGURE 5.1 – Gains cumulés lors de l'entraînement de DQN sur Cartpole *Offline*

«**Maze2D**»[16] est un environnement virtuel en 2D représentant un labyrinthe dans lequel un agent doit trouver un chemin pour atteindre un emplacement donné. La tâche de navigation est conçue pour fournir un test simple de la capacité des algorithmes d'apprentissage par renforcement hors ligne à assembler des sous-trajectoires précédemment collectées pour trouver le chemin le plus court vers l'objectif d'évaluation. Le dataset est directement fourni par «D4RL», un benchmark open-source pour l'entraînement et le benchmarking dans la configuration *Offline*. Plusieurs variantes de **Maze2D** existent, et l'entraînement a été effectué sur les transitions de «*maze2d-medium-v1*».

5.2.2 Algorithmes d'Apprentissage

Deux types d'algorithmes ont été utilisés afin de mener les expérimentations. Les baselines, correpondant à DQN et BC, et les algorithmes d'Offline RL (algorithmes de programmation dynamique dérivés du Q-learning) proprement dit sont spécifiquement développés pour atténuer le décalage de distribution. L'ensemble des algorithmes sont implémentés dans la librairie «d3rly» utilisée pour nos expérimentations. Nous décrirons brièvement ces algorithmes :

Behaviour Cloning(BC). Il s'agit de la forme la plus simple d'apprentissage par imitation (imitation learning), car elle ne nécessite pas d'interactions avec l'environnement. En pratique, c'est la méthode la plus facile à mettre en œuvre puisqu'elle se contente de copier/imiter les actions du *training set*, donc aucun processus RL n'est impliqué. La méthode BC ne contrôle que les erreurs de distribution et n'aborde pas le problème des actions hors distribution.

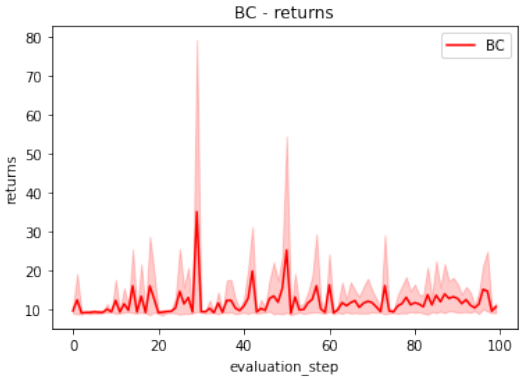
Batch Constrained Deep Q-learning(BCQ). L'idée derrière BCQ est d'exécuter un Q-learning normal, mais durant l'étape de maximisation lors du calcul de la cible (qui est normalement $max_{a'}Q(s,a)$), au lieu de considérer le max sur toutes les actions possibles, on ne considère que les actions a telles que les paires (s,a) sont observables dans les données d'apprentissage. Cela permet d'éliminer les actions qui sont peu susceptibles d'être sélectionnées par π_β.

Conservative Q-learning (CQL). L'algorithme CQL a été proposé par [12]. L'idée générale de cet algorithme est d'ajouter un terme de régularisation durant la phase de l'évaluation de la politique afin d'éviter de surestimer les valeurs pour les actions hors distribution, tout en maximisant celles observables dans les données d'apprentissage.

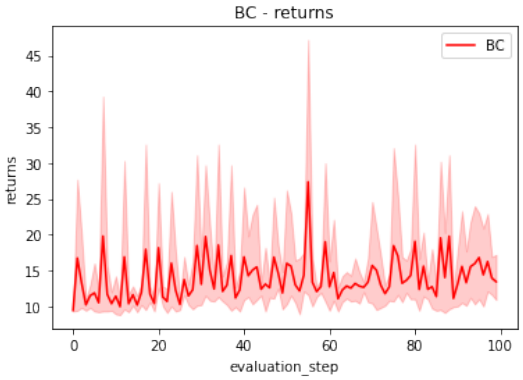
Soft Actor Critic (SAC). Il s'agit d'un algorithme *actor-critic off policy*, visant à maximiser la récompense attendue le long des trajectoires tout en maximisant l'entropie. L'acteur est ainsi incité à explorer plus tout en apprenant. Ceci permet de proposer plusieurs solutions optimales équivalentes.

5.2.3 Cartpole

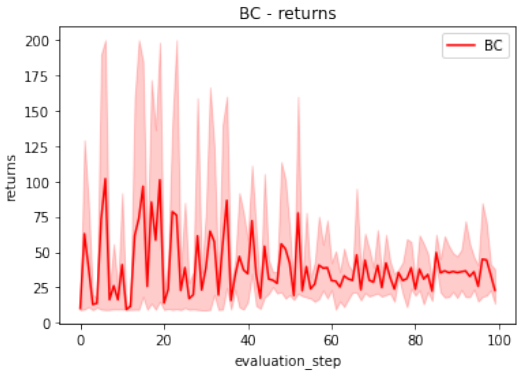
L'objectif des premières expériences était de comparer les résultats des différents algorithmes en fonction de la qualité des trajectoires et de la couverture de l'environnement par le *dataset*. Pour ce faire, les algorithmes ont été entrainés sur les 3 datasets générés « *Cartpole-early-stage-learning* », « *Cartpole-mid-stage-learning* » et « *Cartpole-Expert-stage-learning* » . « *Cartpole-early-stage-learning* » est généré par un agent peu entrainé agissant aléatoirement, ce qui permet d'obtenir une bonne couverture de l'environnement malgré une qualité de trajectoire moindre. Les deux *datasets* suivants « *Cartpole-mid-stage-learning* » et « *Cartpole-Expert-stage-learning* » sont caractérisés par une couverture de l'environnement décroissante au profit de la qualité croissante des trajectoires. Pour chaque algorithme, plusieurs apprentissages ont été réalisés sur chacun des *datasets* pour garantir la véracité des résultats tout en s'assurant de la reproductibilité des expériences. L'initialisation des algorithmes dépend de *seeds* fixés pour chaque expérience. Afin de mesurer la performance de chaque algorithme, nous avons mesuré les gains obtenus par l'agent après chaque itération en l'évaluant sur l'environnement de Cartpole. Nous avons aussi mesuré la dispersion des résultats dans le but d'estimer la variance des gains obtenus. Dans un intérêt de clarté, nous n'afficherons que des courbes d'apprentissage lissées (sauf cas particulier), les courbes réelles seront disponibles en annexe. Nous interprèterons globalement les résultats de chaque algorithme en fonction du *dataset* avant de les comparer.



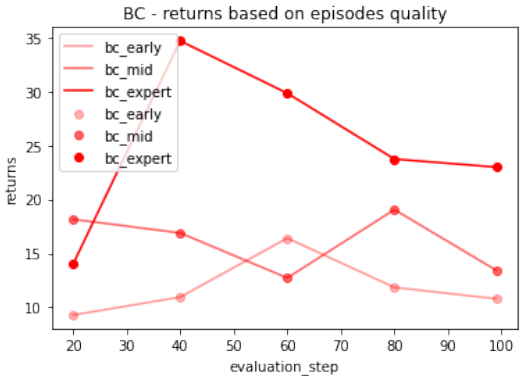
(a) - apprentissage sur « *Cartpole-early-stage-learning* »



(b) - apprentissage sur « *Cartpole-mid-stage-learning* »

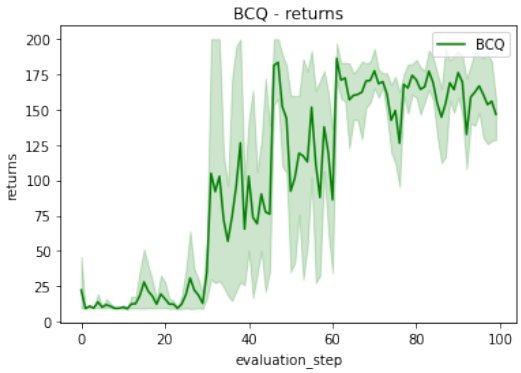


(c) - apprentissage sur « *Cartpole-expert-stage-learning* »

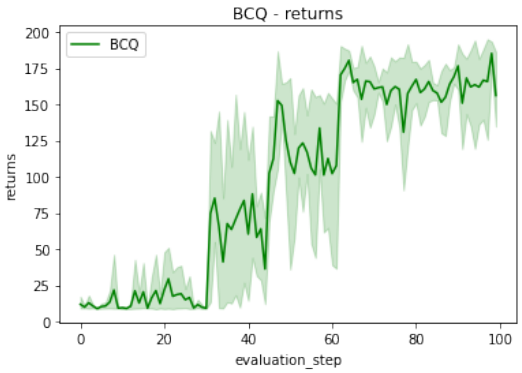


(d) - Comparaison des résultats selon le dataset

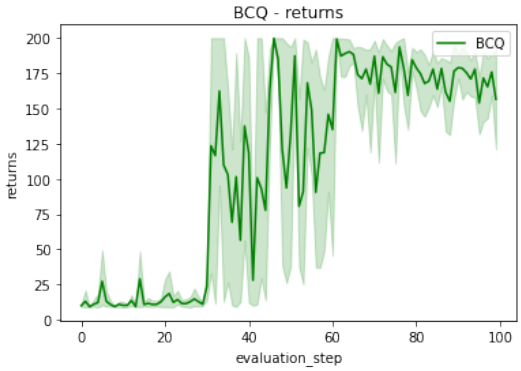
Behavior Cloning (BC). Ci-dessus, nous observons les résultats obtenus avec BC. Nous observons que les gains moyens obtenus varient entre 0-80 pour les datasets « *Cartpole-early-stage-learning* », « *Cartpole-mid-stage-learning* » et atteignent 200 (gain maximal possible sur Cartpole) pour *Cartpole-expert-stage-learning* ». Ces résultats sont plutôt cohérents puisque BC ne fait que mimer la politique génératrice (DQN), qui dans notre cas devient plus performante avec le temps. Ainsi, BC est plus performante sur des *datasets* expert et donne de mauvais résultats sur des *datasets* générés par un agent aléatoire.



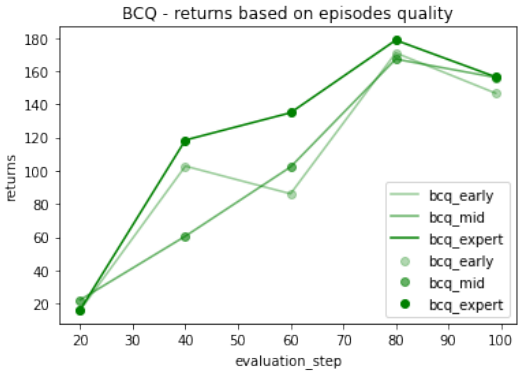
(a) - apprentissage sur « *Cartpole-early-stage-learning* »



(b) - apprentissage sur « *Cartpole-mid-stage-learning* »

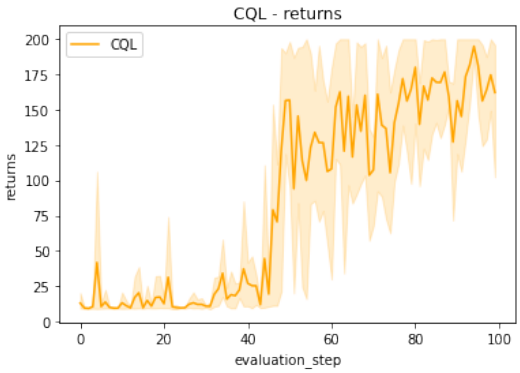


(c) - apprentissage sur « *Cartpole-expert-stage-learning* »

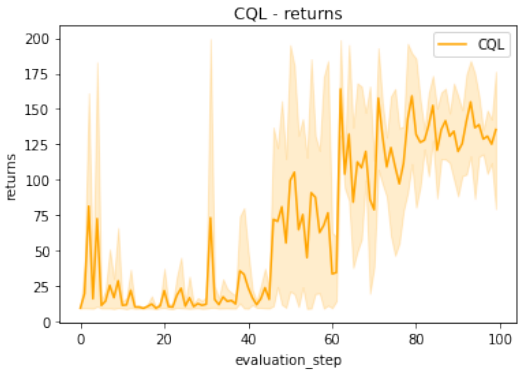


(d) - Comparaison des résultats selon le dataset

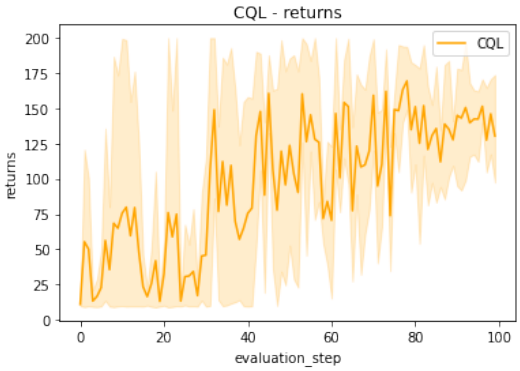
Batch Constrained Q-learning (BCQ). Nous observons que sur les 3 *datasets*, BCQ atteint le gain maximal posssible pour Carpole. Il n'y a pas de différences majeures induit par le jeu de données, nous pouvons cependant ajouter que BCQ converge plus vite sur les données expertes. Nous remarquons aussi que la variance des gains de BCQ diminue avec l'apprentissage, ce qui présage la stabilité de l'algorithme.



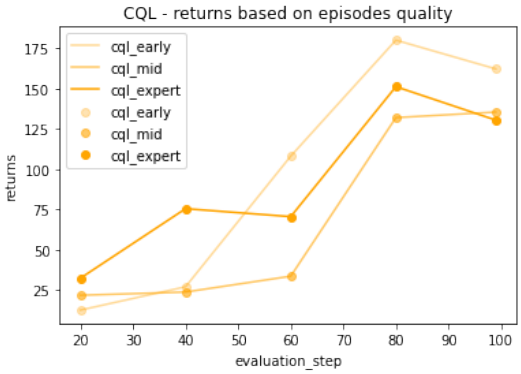
(a) - apprentissage sur « *Cartpole-early-stage-learning* »



(b) - apprentissage sur « *Cartpole-mid-stage-learning* »

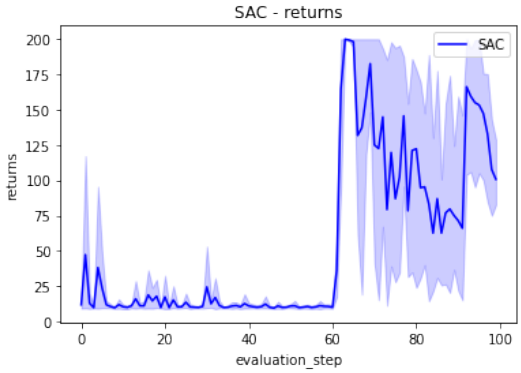


(c) - apprentissage sur « *Cartpole-expert-stage-learning* »

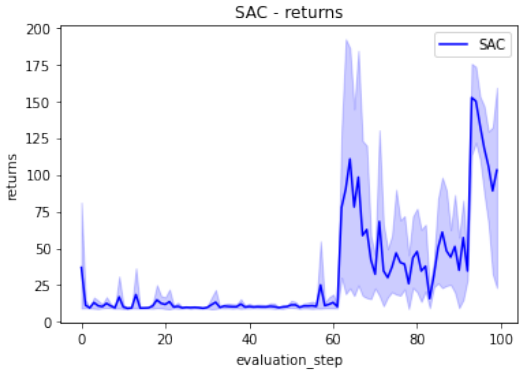


(d) - Comparaison des résultats selon le dataset

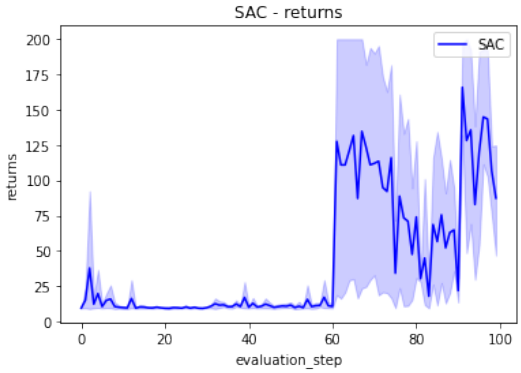
Concervative Q-learning(CQL). Nous remarquons que CQL obtient de bons résultats sur l'ensemble des datasets, plus spécifiquement sur « *Cartpole-early-stage-learning* » avec une bonne couverture de l'environnement. On remarque également qu'on a des résultats beaucoup plus dispersés sur le *dataset* expert. Pour rappel, CQL correspond à un Q-learning avec un terme de régularisation afin de limiter le décalage distributionnel. Il est paramétré par α correspondant au poids accordé au terme de régularisation. CQL avec un petit $\alpha(= 0.01)$ est plus performant sur des données contenant beaucoup d'aléas, mais échoue sur des données expertes. Le contraire est vrai pour de grandes valeurs de $\alpha(= 1)$. Ainsi, avec CQL, plus les données sont expertes et plus la valeur de α doit décroître. Utiliser les proxys évoqués dans la section (5.1) permet de déterminer le type de données. La valeur de α lors de notre entraînement est automatiquement déterminé par la librairie utilisée, d'où l'impossibilité de vérifier l'assertion précédente.



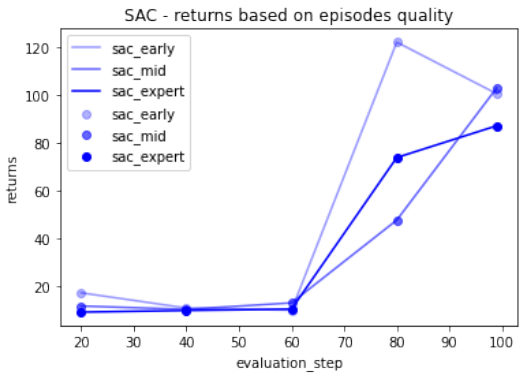
(a) - apprentissage sur « *Cartpole-early-stage-learning* »



(b) - apprentissage sur « *Cartpole-mid-stage-learning* »

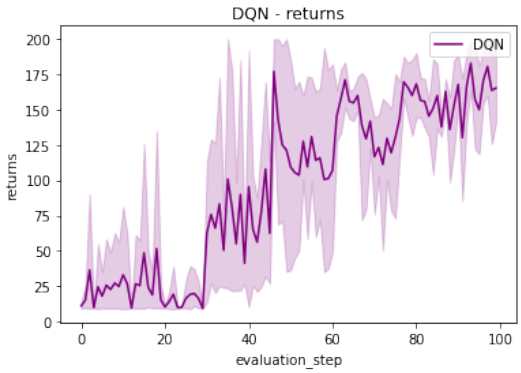


(c) - apprentissage sur « *Cartpole-expert-stage-learning* »

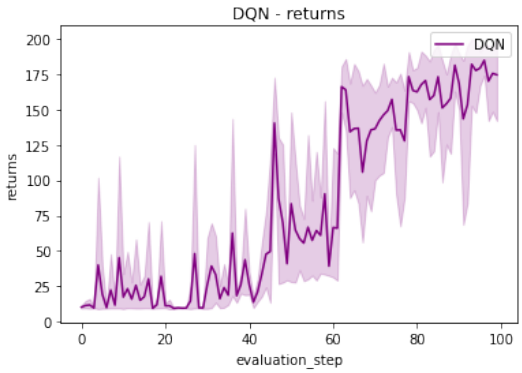


(d) - Comparaison des résultats selon le dataset

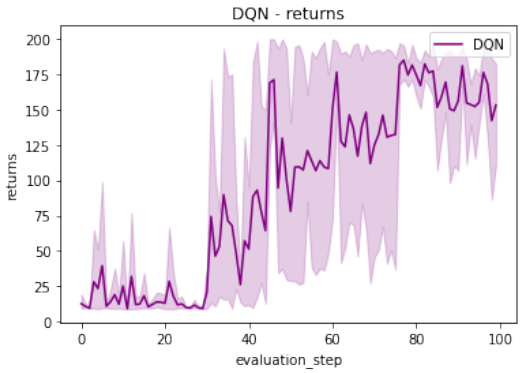
Soft Actor-Critic (SAC). Nous observons que SAC converge après la 60 itération pour (a). Sur l'ensemble des datasets, nous remarquons aussi que la variance des gains diminue avec les itérations. Les gains cumulés moyens plus élevés pour (a) peuvent s'expliquer par le fait que l'algorithme de SAC incite son acteur à explorer plus, ce qui plus est adapté pour un dataset avec beaucoup d'aléas comme dans (a). Cependant, il n'y a pas d'explications pour la croissance subite des gains à la soixantième itération.



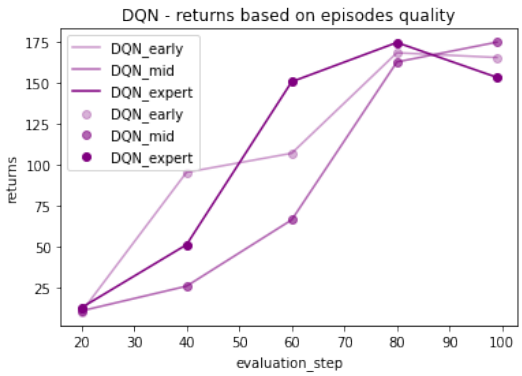
(a) - apprentissage sur « *Cartpole-early-stage-learning* »



(b) - apprentissage sur « *Cartpole-mid-stage-learning* »

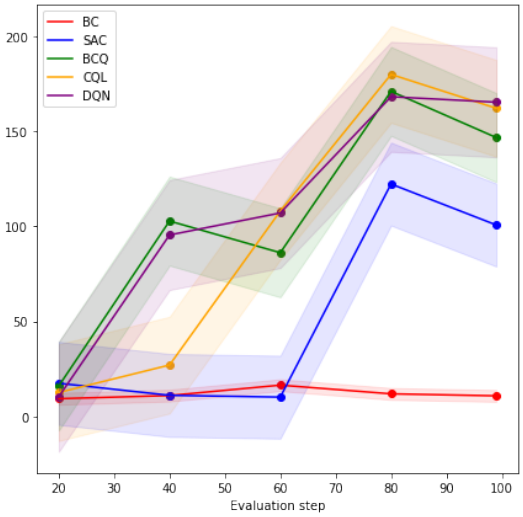


(c) - apprentissage sur « *Cartpole-expert-stage-learning* »

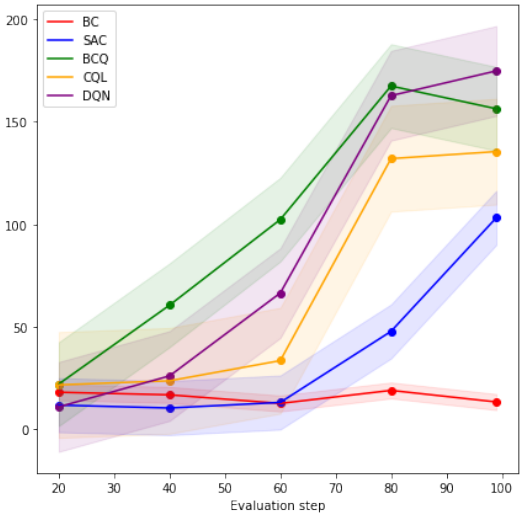


(d) - Comparaison des résultats selon le dataset

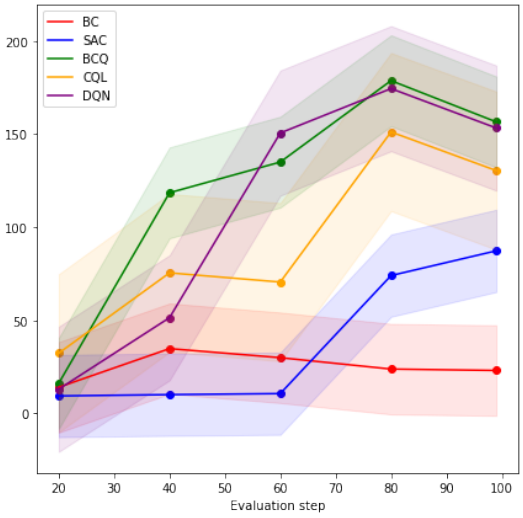
Deep Q-Network (DQN). Les données d'entraînement ont été générées par un algorithme DQN. On remarque un phénomène intéressant, on s'aperçoit que DQN converge en (a) malgré que l'ensemble d'apprentissage « *Cartpole-early-stage-learning* » ne soit composé que des 24 premiers épisodes de la politique génératrice dont la récompense cumulée atteinte ne dépasse pas 50 (voir Image 5.1). On remarque aussi que la variance diminue avec la convergence de l'algorithme sur les 3 *datasets*.



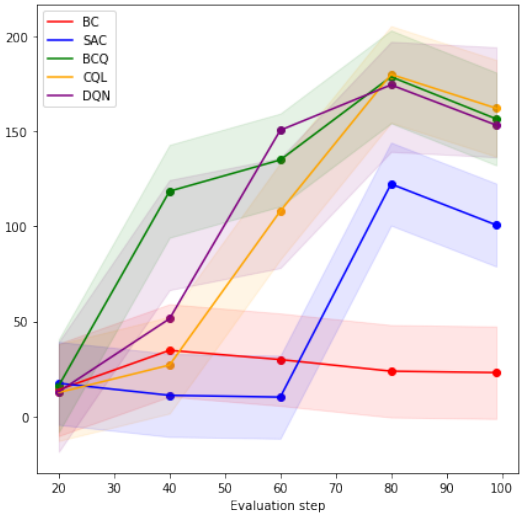
(a) - benchmark sur « *Cartpole-early-stage-learning* »



(b) - benchmark sur « *Cartpole-mid-stage-learning* »



(c) - benchmark sur « *Cartpole-expert-stage-learning* »

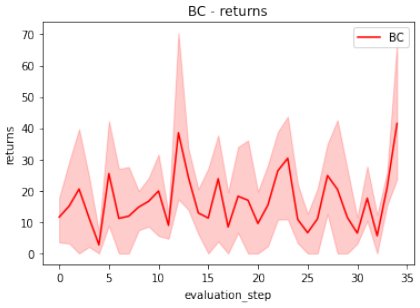


(d) - Comparaison des meilleurs modèles

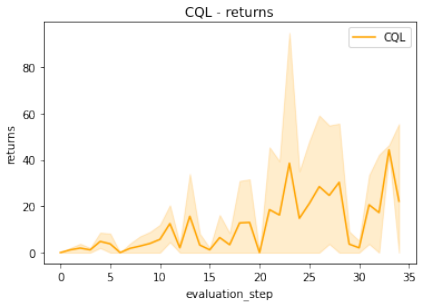
Les images ci-dessus montrent les performances de l'ensemble des algorithmes en fonction du datasets. Les visualisations sont lissées afin de faciliter l'interprétation des graphiques (voir Annexe pour graphique brute). Nous remarquons que sur (a) (b) et (c), l'ensemble des algorithmes à base de Q-learning convergent, suivis de SAC, un algorithme *actor-critic*. Quant à BC, il ne converge pas sur (a), ce qui était prévisible, mais pas sur (b) et (c). (d) illustre la comparaison des résultats des algorithmes en sélectionnant le modèle le plus performant pour chaque algorithme. On remarque que les algorithmes à base de Q-learning sont les plus performants sur le problème de Cartpole. Il faut cependant garder à l'esprit que les données ont été générées par un algorithme à base de Q-learning, ce qui pourrait peut-être en partie expliquer ces résultats.

5.2.4 Maze

Les expérimentations sur les labyrinthes ont été réalisées sur 3 environnements virtuels (2 discrets et 1 continu). Cependant, nous ne présenterons que les résultats sur **Maze2D** car les algorithmes entraînés sur les données des labyrinthes discrets (10x10 et 20x20) n'ont pas réussi à converger. Les algorithmes employés pour l'entraînement utilisent des réseaux de neurones contrairement à la politique génératrice (Q-learning tabulaire) de ces labyrinthes discrets, ce qui peut peut-être expliquer la non-convergence. Quelques caractéristiques de l'entraînement de la politique génératrice de ces labyrinthes discrets sont tout de même disponibles en annexe. Les données d'entraînement issues de **Maze2D** sont fournies par D4RL (*Datasets for Deep Data-Driven Reinforcement learning*), un dépôt *github* permettant d'avoir accès à un ensemble de données adapté pour l'apprentissage par renforcement hors-ligne. Les résultats de l'entraînement et de l'évaluation des algorithmes sur **Maze2D** sont disponibles ci-dessous :

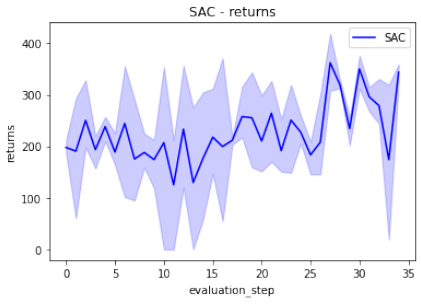


(a) - Behaviour Cloning (BC) sur Maze2D



(b) - Conservative Q-learning (CQL) sur Maze2D

(c) - Batch Contrained Q-learning (BCQ) sur Maze2D



(d) - Soft Actor-Critic (SAC) sur Maze2D

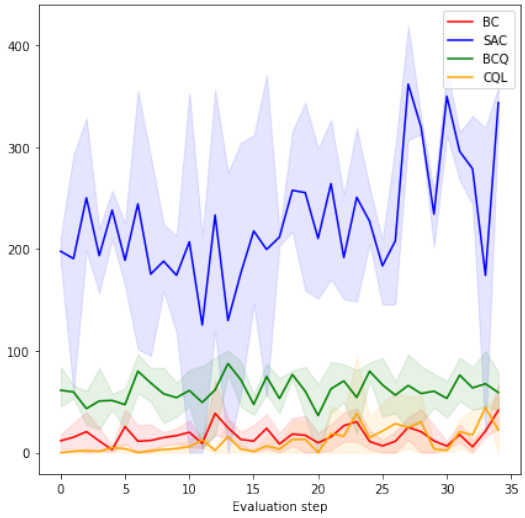


FIGURE 5.2 – Courbes d'évaluation lissées sur Maze2D

Nous observons sur la figure 5.2 que seul SAC converge vraiment avec un gain cumulé avoisinant 400. CQL (b) et BC (c) obtiennent des gains compris 0 et 40 et BCQ (c) avoisine 100. Il faut cependant prendre en compte le nombre d'itérations pour l'entrainement. Il est de 35 itérations pour chaque algorithme en raison de l'aspect chronophage (plusieurs jours) de l'apprentissage. Le nombre d'itérations n'est probablement pas suffisant pour faire converger BCQ ou CQL.

Conclusion

Dans cette étude, nous avons présenté les principales techniques d'apprentissage par renforcement existantes ainsi que l'apprentissage par renforcement hors Ligne. Nous avons également montré les principales difficultés qu'impliquait ce type d'apprentissage, notamment avec le décalage distributionnel. En effet, le décalage distributionnel affecte l'ensemble des méthodes d'apprentissage du moment où l'algorithme d'apprentissage est évalué sur une distribution différente de celle de l'apprentissage. Plusieurs solutions ont été proposées dans la littérature afin de limiter ce décalage distributionnel. Des approches utilisant l'échantillonnage préférentiel pour l'évaluation hors politique ont été adaptées à la configuration hors ligne pour les méthodes *Policy-Gradient*. Il faut cependant noter que ces approches souffrent d'une grande variance induite par la divergence entre la politique apprise et la politique génératrice. Quant à la programmation dynamique, le décalage distributionnel intervient lors de l'estimation de la fonction valeur/qualité, lorsque les estimations sont basées sur des actions hors distribution. Afin de limiter l'implication de ces actions hors distribution, un certain nombre d'améliorations ont été proposées ; notamment les méthodes à bases de contrainte qui visent à contraindre la politique apprise à ne pas trop s'éloigner de la politique génératrice, ce qui constitue aussi sa principale limite. Nous pouvons aussi citer les méthodes à base d'incertitude et de régularisation évoquées dans la section **??**. L'apprentissage par renforcement hors ligne basé sur des modèles présentent également ses propres défis : alors que certains MDPs sont faciles à modéliser avec précision, d'autres peuvent être extrêmement difficiles. Des solutions à base d'incertitude et de contraintes ont également été proposés dans la littérature. Dans une seconde partie, nous avons comparé les résultats de plusieurs algorithmes d'apprentissage sur des données associées à des environnements discrets et continus. Nous avons évoqué quelques caractéristiques importantes des *datasets* plus particulièrement la couverture de l'espace et la qualité des transitions qui affectent l'apprentissage. Ces expérimentations ont montré que les algorithmes les plus récents n'obtiennent pas unanimement les meilleures performances par rapport aux algorithmes de référence comme BC. En effet, le fait de consacrer beaucoup de ressources à la collecte de données et/ou à la personnalisation des algorithmes ne garantit pas de bonnes performances pour des tâches complexes. De futurs travaux pourraient se diriger vers le *Model-Based Offline reinforcement learning* qui n'est pas assez exploité dans la littérature malgré des résultats probants.

Références

[1] A. Raghu, M. Komorowski, I. Ahmed, L. Celi, P. Szolovits, and M. Ghassemi, "Deep reinforcement learning for sepsis treatment," *arXiv preprint arXiv :1711.09602*, 2017. [Cité en page 6]

[2] N. Jaques, A. Ghandeharioun, J. H. Shen, C. Ferguson, A. Lapedriza, N. Jones, S. Gu, and R. Picard, "Way off-policy batch deep reinforcement learning of implicit human preferences in dialog," *arXiv preprint arXiv :1907.00456*, 2019. [Cité en page 6]

[3] A. Swaminathan, A. Krishnamurthy, A. Agarwal, M. Dudik, J. Langford, D. Jose, and I. Zitouni, "Off-policy evaluation for slate recommendation," *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Cité en page 6]

[4] G. Kahn, P. Abbeel, and S. Levine, "Badgr : An autonomous self-supervised learning-based navigation system," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1312–1319, 2021. [Cité en page 6]

[5] D. Precup, "Eligibility traces for off-policy policy evaluation," *Computer Science Department Faculty Publication Series*, p. 80, 2000. [Cité en page 8]

[6] Z. Tang, Y. Feng, L. Li, D. Zhou, and Q. Liu, "Doubly robust bias reduction in infinite horizon off-policy estimation," *arXiv preprint arXiv :1910.07186*, 2019. [Cité en page 8]

[7] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*, pp. 387–395, PMLR, 2014. [Cité en page 9]

[8] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *Proceedings of the 26th annual international conference on machine learning*, pp. 993–1000, 2009. [Cité en page 12]

[9] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," *Advances in Neural Information Processing Systems*, vol. 32, 2019. [Cité en pages 13 et 16]

[10] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International conference on machine learning*, pp. 2052–2062, PMLR, 2019. [Cité en page 16]

[11] N. Y. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller, "Keep doing what worked : Behavioral modelling priors for offline reinforcement learning," *arXiv preprint arXiv :2002.08396*, 2020. [Cité en page 16]

[12] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020. [Cité en pages 17 et 23]

[13] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model : Model-based policy optimization," *Advances in Neural Information Processing Systems*, vol. 32, 2019. [Cité en page 19]

[14] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, "Morel : Model-based offline reinforcement learning," *Advances in neural information processing systems*, vol. 33, pp. 21810–21823, 2020. [Cité en page 19]

[15] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma, "Mopo : Model-based offline policy optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14129–14142, 2020. [Cité en page 19]

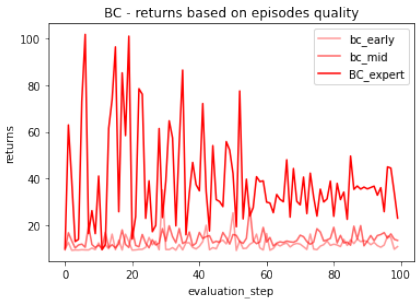
[16] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl : Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv :2004.07219*, 2020. [Cité en page 22]

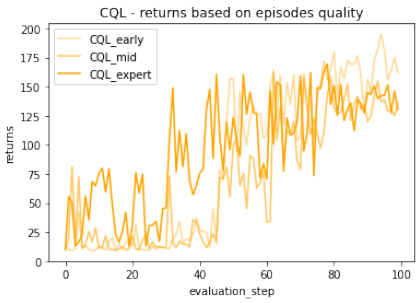
Annexe A

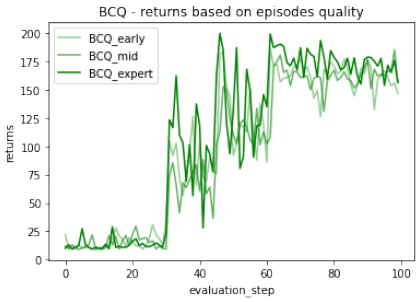
Expériences

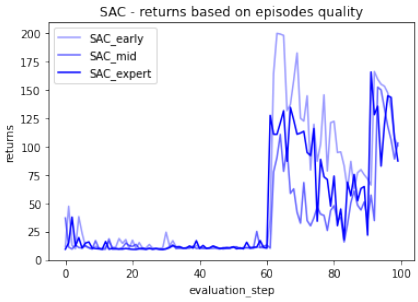
A.1 Expérience Cartpole

A.1.1 Courbe d'apprentissage brute des algorithmes sur Cartpole

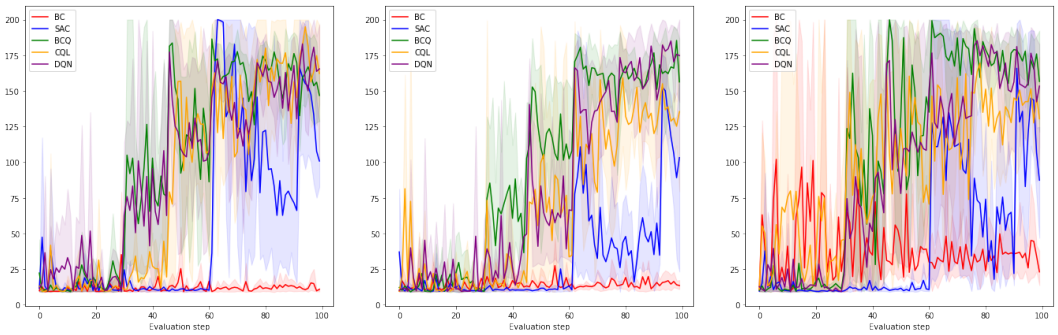






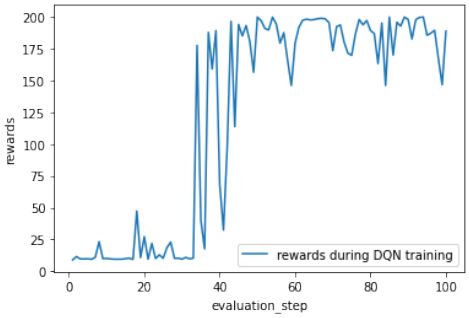


A.1.2 Courbe d'apprentissage brute des algorithmes sur Cartpole selon le dataset

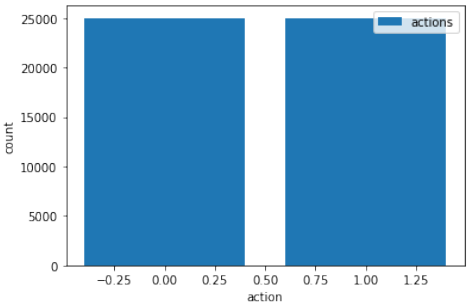


Early-stage-learning *Mid-stage-learning* *Expert-stage-learning*

A.1.3 DQN - Politique génératrice



(a)- Retours épisodique de π_β

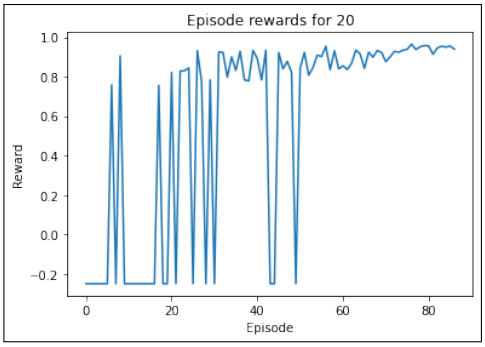


(b)- Distribution des actions

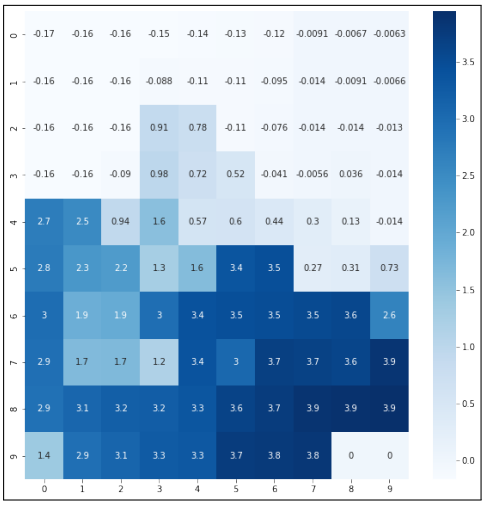
A.2 Expérience Maze

A.2.1 Maze Discret

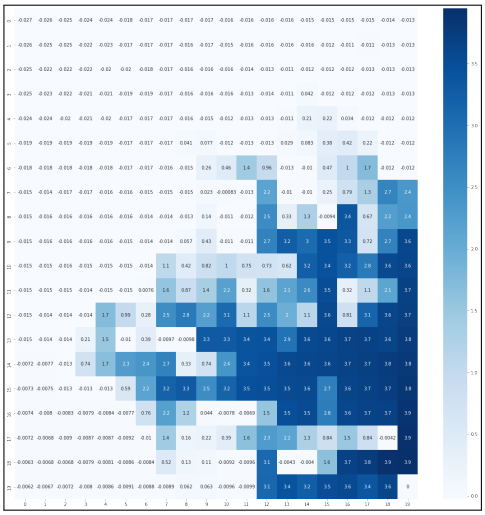
Retours épisodique sur *Maze10x10*



Retours épisodique sur *Maze20x20*



Valeur des états de *Maze10x10*



Valeur des états de *Maze20x20*

A.2.2 Maze2D

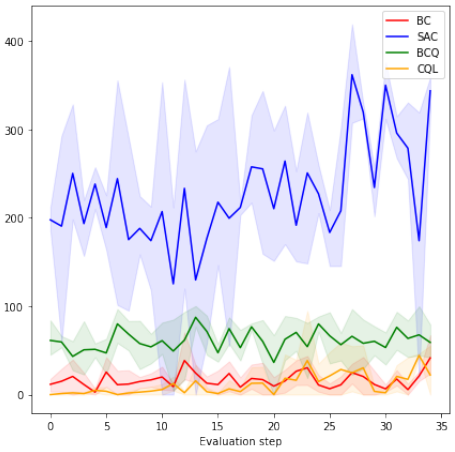


FIGURE A.1 – Retours épisodique brut des algorithmes durant l'apprentissage sur Maze2D