Apprentissage actif sous contraite de budget en robotique et en neurosciences computationnelles : Localisation robotique et modélisation comportementale en environnement non stationnaire.

Nassim Aklil

Une Thèse de doctorat par

Nassim AKLIL

présentée à

Université Pierre et Marie Curie

École Doctorale Cerveau, Cognition et Comportement - ED3C

Institut des Systèmes Intelligents et de la Robotique - ISIR

pour obtenir le grade de

Docteur en Intelligence Artificielle et Robotique

Apprentissage actif sous contrainte de budget en robotique et en neurosciences computationnelles

Localisation robotique et modélisation comportementale en environnement non stationnaire

27/09/2017

Jury

Pr. Thierry Artières, LIF – Ecole Centrale Marseille	Rapporteur
Dr. Alain Dutech, LORIA – INRIA	Rapporteur
Pr. Mathias Quoy, ETIS – Univ. Cergy Pontoise	Examinateur
Dr. Celine Teuliere, Institut Pascal – Univ. Clermont Auvergne	Examinateur
Dr. Sio-Hoi Ieng, Institut de la Vision – UPMC	Examinateur
Pr. Ludovic Denoyer, LIP6 – UPMC	Co-encadrant
Dr. Benoît Girard, ISIR – UPMC	Co-encadrant
Dr. Mehdi Khamassi, ISIR – UPMC	Encadrant

Informations

Mots clefs

Apprentissage par Renforcement, Apprentissage budgétisé, Apprentissage Profond, Neurosciences Computationnelles, Compromis exploration/exploitation, *Policy Gradient*

Résumé

La prise de décision est un domaine vaste et très étudié en sciences, que ce soit en neurosciences pour comprendre les processus sous tendant la prise de décision chez les animaux, qu'en robotique pour modéliser des processus de prise de décision efficaces et rapides dans des tâches en environnement réel. Du point de vue des neurosciences, ce problème est résolu *online* avec des modèles de prises de décision séquentiels basés sur l'apprentissage par renforcement. Du point de vue de la robotique, l'objectif premier est l'efficacité, dans le but d'être déployés en environnement réel. Cependant, dans le cas de la robotique, ce que l'on peut appeler le *budget* et qui concerne les limitations inhérentes au matériel, comme par exemple les temps de calculs, les actions limitées disponibles au robot ou la durée de vie de la batterie du robot, ne sont, le plus souvent, pas prises en compte à l'heure actuelle.

Nous nous proposons dans ce travail de thèse d'introduire la notion de budget comme contrainte explicite dans les processus d'apprentissage robotique appliqués à une tâche de localisation en mettant en place un modèle basé sur des travaux récents développés en apprentissage statistique qui traitent les données sous contrainte de budget, en limitant l'apport en données ou en posant une contrainte de temps plus explicite.

Dans le but d'envisager à plus long terme un fonctionnement *online* de ce type d'algorithmes d'apprentissage budgétisé, nous discutons aussi certaines inspirations possibles qui pourraient être prises du côté des neurosciences computationnelles. Dans ce cadre, l'alternance entre recherche d'information pour la localisation et la décision de se déplacer pour un robot peuvent être indirectement liés à la notion de compromis exploration-exploitation. Nous présentons notre contribution à la modélisation de ce compromis chez l'animal dans une tâche non stationnaire impliquant différents niveaux d'incertitude, et faisons le lien avec les méthodes de bandits manchot.

Remerciements

Cette thèse a été menée grâce aux financements de l'Agence Nationale de la Recherche et le Labex SMART (ANR-11-LABX-65 Online Budgeted Learning Project).

J'aimerais tout d'abord remercier les membres du jury de m'avoir honoré de leur présence lors de la soutenance de thèse. Merci à Messieurs Alain Dutech et Thierry Artières d'avoir pris de leur temps pendant les vacances afin de remettre leurs rapports. Je remercie Madame Céline Teulière et Messieurs Sio Hoi Ieng et Mathias Quoy d'avoir accepté de faire partie du jury malgré leurs nombreuses charges de travail.

Je remercie mes encadrants et collaborateurs : Mehdi Khamassi, Benoît Girard et Ludovic Denoyer. Merci à Mehdi pour ses constants encouragements qui m'ont remonté le moral dans les périodes les plus difficiles. Merci à Benoît d'avoir toujours été présent et d'avoir toujours su se rendre disponible tout au long de cette thèse. Merci à Ludovic qui m'a beaucoup conseillé autour des problématiques du *machine learning* et qui m'a très souvent permis d'en apprendre plus. Je les remercie tous les trois pour leurs conseils et leur soutien tout au long de ce travail de thèse.

Je remercie mes collaborateurs avec qui j'ai travaillé en stage et en début de thèse : Virginie Fresno, Alain Marchand et Etienne Coutureau qui ont fourni les données utilisées dans les études du comportement des rats.

Merci aux collègues et amis de la J01, anciens et nouveaux, sans qui j'aurais probablement fini cette thèse en 3 ans au lieu 4 !! Merci aux anciens qui se sont tous retrouvés à Heuritech : Tony, CharlesT, CharlesO, Didier et Jean avec qui j'ai eu de très bonnes interactions pendant leurs thèses mais aussi après. Merci à Tony pour les séances sportives des plus intenses et toujours les bienvenues même si j'ai laissé tomber au commencement de ma 3eme année (désolé !!), merci à CharlesT d'avoir été la personne la plus désopilante du bureau et de m'avoir fait découvrir l'art du combat médiéval. Merci à CharlesO de m'avoir légué son bureau qui est, je dois bien le dire, le meilleur de toute la salle. Merci à Didier pour ses discussions musicales toujours des plus intéressantes et merci Jean d'avoir été là sans toi j'aurai été le seul petit de taille du bureau.

A ceux qui ne se sont pas retrouvés à Heuritech, j'aimerais remercier Alain qui nous a bien torturé avec sa présence en J01 de par ses trolls incessants, alors que son bureau

n'était pas la J01... Merci à Arthur avec qui j'ai pu montrer toute l'étendue de mon talent aux fléchettes. Merci à Antoine d'avoir été l'homme sage de toute les situation, sa disponibilité pour des discussions, son travail de représentant des doctorants et son apport de la tradition du petit déjeuner qui depuis a hélas été perdue. Merci à Florian qui ne s'est jamais gêné à nous humilier lors des session CS. Merci aussi à Erwan avec qui j'ai eu les plus belles discussions scientifiques (et non scientifiques) lors de mon séjour en J01 malgré ses quelques énervements lors de la fin de sa thèse.

J'aimerais remercier tout autant les actuels doctorants qui ont donné une ambiance si particulière à ce bureau. Merci Carlos d'avoir remplacé Antoine en tant qu'homme sage et d'avoir essayé de mettre un peu d'ordre dans le chaos qu'est ce bureau. Merci Pierre pour les nombreuses parties d'échecs qu'on a eu et qui m'ont permis de ne pas travailler lorsque je ne le voulais pas. Merci Leni avec qui la conférence à Taiwan est devenue un séjour des plus plaisants, merci pour les discussions et les débats animés qu'on a pu mener. J'aimerais remercier François d'avoir repris le travail sur la modélisation du comportement, cela m'a permis de bien alléger mon emploi du temps et de me consacrer à la deuxième partie de ma thèse. Enfin je remercierai particulièrement Thibaud avec qui les plus belles et les plus animées des discussions ont eu lieu qu'elles soient scientifiques ou autres et qui m'ont beaucoup aidé à m'améliorer autant d'un point de vue professionnel que personnel. En vrac je remercie aussi les stagiaires qui sont passés par la J01 : Paul, Florian, Quentin mais aussi Philippine et Lise qui ont apporté une touche de féminité dans ce bureau de mâles dominants.

Merci aussi à mes amis Bernard, Mad, Guillaume, Margaux, Nikos, Nancy et surtout Melissa d'avoir rendu la vie à Paris des plus agréables et avec qui j'ai passé certaines des soirées les plus mémorables à Paris. J'aimerais remercier mes amis qui sont toujours en Algérie : Walid, Hamza, Yacine, Abdelghani, Marwan, Yasmine, Lydia, l'autre Yasmine qui m'ont tous encouragé à entamer cette aventure des études en France.

Enfin je ne remercierais jamais assez ma petite famille, ma mère Rosa, mon père Hamid et ma sœur Sabrina sans qui je ne serais certainement pas là aujourd'hui. Je ne dois ce parcours qu'à leur éducation. Je les remercie de leur soutien infaillible tout au long de cette thèse et d'avoir cru en moi depuis le début de mes études à l'étranger.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Contexte

La prise de décision est une des problématiques centrale à l'heure actuelle dans plusieurs disciplines scientifiques allant de la biologie à l'informatique, en passant par la psychologie. On la retrouve par exemple du côté des neurosciences computationnelles, où des modèles sont utilisés pour la reproduction et l'étude de comportements observés dans les organisme vivants, ou du côté robotique, où le but de mettre en place des algorithmes efficaces de prise de décision en environnement réel.

Si les neurosciences computationnelles se donnent pour but d'expliquer le comportement observé chez les animaux, ils utilisent souvent des algorithmes d'apprentissage issus du domaine de l'intelligence artificielle tels que l'apprentissage par renforcement, ces modèles aidant à synthétiser une large quantité de données empiriques issus d'études comportementales. Ce domaine s'intéresse particulièrement à expliquer la capacité d'individus à interagir avec leur environnement, à prédire des événements futurs et finalement prendre des décisions et choisir des actions qui auront inévitablement des répercussions (qu'elles soient positives ou négatives) sur ces individus. Dans ce cadre, l'apprentissage par renforcement s'est imposé comme étant la solution la plus adéquate pour modéliser une grande part de ces comportements (Daw et O'Doherty 2013), de façon à ce que les agents prennent des décisions dans le but de maximiser une récompense ou minimiser une punition. Dans le cadre classique, ces modèles montrent des résultats prometteurs dans la modélisation des comportements (Dutech et al. 2011, Bellot et al. 2012) dans des environnements stationnaires i.e. des probabilités de récompense ou de punitions qui ne

changent pas au cours du temps. Or il apparaît clairement que les agents sont soumis de plus en plus à des environnements incertains et non stationnaires. Ce type d'environnements incite à la complexification de la modélisation de façon à pouvoir expliquer des comportements observés plus complexes trouvant une capacité à créer une balance entre l'exploitation d'actions apprises et l'exploration de nouvelles possibilités lors de changements brusques dans l'environnement.

D'un autre côté, dans le cadre de la robotique, il est de plus en plus mis en lumière que les robots doivent faire face à des tâches de plus en plus incertaines et changeantes. En effet, des premiers robots industriels, dont les instructions étaient clairement énoncées et invariables dans un environnement complètement stable, aux robots actuels dont les applications sont de plus en plus étendues vers des environnements incertains et changeants, la façon de prendre en charge ces tâches sont différentes (Khamassi et al. 2016). Une plus grande place est accordée à l'apprentissage dans les tâches robotiques actuelles, on pourra citer les problématiques levées par la robotique autonome, les applications militaires ou les robots compagnons. Or ces exemples, malgré leur efficacité, ne prennent pas en compte explicitement les limitations réelles des robots tels que les coûts computationnels ou la capacité limitée des robots à calculer certains aspects de la tâche de façon à maximiser le rendement du robot malgré ses limitations.

1.2 Problématique

Nous nous proposons dans ce travail d'introduire une notion de budget explicite dans les modèles de prise de décisions de façon à créer un compromis entre la possibilité d'exploiter des actions apprises et d'explorer l'environnement dans le cas de changements drastiques des perceptions et ce dans le cas de deux tâches complémentaires : une tâche en neurosciences où des rats doivent trouver par essai-erreur lequel parmi trois leviers donne la plus grande probabilité de récompense, cette probabilité pouvant changer abruptement de façon non signalée ; une tâche de robotique où un robot doit trouver une séquence efficace d'actions alternant entre prise d'information grâce aux senseurs et mouvement. Dans le premier cas nous étudions la capacité d'un modèle d'apprentissage par renforcement à utiliser un compromis entre exploration et exploitation grâce à une régulation active des fonctions de prise de décisions et dans le deuxième cas, à utiliser un budget explicite dans une tâche de localisation robotique en faisant apprendre au robot à alterner entre les actions d'utilisation de senseurs (i.e. relever une image ou une donnée laser) et des actions de mouvements (i.e. tourner, avancer, etc...).

1.3 Plan de la thèse

Ce manuscrit est divisé en 8 chapitres :

— Dans le chapitre 2 nous introduisons les algorithmes tirés de l'état de l'art de

l'apprentissage par renforcement qui sera la base des modèles présentés dans ce manuscrit. Nous présenterons une vue globale de la manière dont sont résolues des problématiques de renforcements. Nous évoquerons la problématique du compromis exploration/exploitation incidente à la manière de modéliser une tâche en utilisant l'apprentissage par renforcement.

— Dans le chapitre 3 nous introduisons les modélisations type réseaux de neurones profonds. Nous commençons par introduire les bases pour la construction d'architectures neuronales profondes, ainsi que le lien existant entre ces architectures et l'apprentissage par renforcement décrit dans le chapitre 2. Nous donnons quelques exemples révélateurs de l'efficacité de ce type de modèles utilisant l'apprentissage par renforcement pour apprendre des réseaux neuronaux.

— Dans le chapitre 4 nous nous intéresserons aux algorithmes d'apprentissage séquentiels, où les modèles de classification sont considérés comme des modèles de sélection séquentielle s'inspirant directement des algorithmes d'apprentissage par renforcements. Nous mettons aussi en avant la notion de budget et le compromis entre précision de prédiction et temps de calcul.

Dans les chapitres suivants nous présentons nos contributions :

— Dans le chapitre 5 nous introduisons d'abord l'état de l'art de l'apprentissage animal et des liens forts existants entre les comportements observés chez l'animal et l'apprentissage par renforcement. Nous présentons ensuite nos travaux pour la modélisation de l'apprentissage animal dans le cadre d'une tâche incertaine et non stationnaire, grâce aux algorithmes d'apprentissage par renforcement et aux algorithmes de bandits manchots. Nous introduisons aussi une manière de réguler le compromis exploration/exploitation en utilisant la récompense (travaux publiés dans Aklil et al. 2014).

— Dans le chapitre 6 nous introduisons une architecture de sélection séquentielle budgétisée modélisée en tâche de classification appliquée à une problématique de localisation en robotique. Nous évaluons ce modèle sur des données simulées simples puis transposées dans le cas d'une tâche réelle avec des données relevées en environnement réel (travaux publiés dans Aklil et al. 2017).

— Dans le chapitre 7 nous présentons une extension de ce modèle en testant sa capacité à apprendre des politiques de sélection efficaces qui sont réutilisables dans différents environnements. De ce fait nous testons la capacité de généralisation de l'apprentissage des politiques apprises tout en apprenant à sélectionner le budget le plus efficace du point de vue compromis exploration/exploitation dans les différents environnements (travaux publiés dans Aklil et al. Under Review).

Chapitre 2

Apprentissage par renforcement

2.1 Introduction

Dans ce chapitre nous nous intéresserons à décrire les mécanismes de l'apprentissage par renforcement (AR) comme outil pour la prise de décision d'agents artificiels. Ce formalisme étant à la base de toute les contributions présentées dans ce manuscrit (que ce soit dans la modélisation en neurosciences dans le chapitre 5 ou dans la localisation en robotique dans le chapitre 6).

Nous commencerons par décrire le formalisme des différents algorithmes de l'AR et la dichotomie entre les apprentissages direct et indirect ainsi que des algorithmes de bandits apprenant les compromis entre exploration et exploitation. Nous nous intéresserons à décrire le lien entre ces algorithmes d'apprentissage et le compromis entre l'exploration et

l'exploitation dans la prise de décision.

2.2 Problématique de l'apprentissage par renforcement

L'idée d'apprendre par interaction avec l'environnement est une des premières idées à apparaître lorsque l'on aborde la question de l'apprentissage. Un enfant apprenant des gestes de la vie quotidienne n'apprend pas grâce à la présence d'un enseignant, mais bien grâce à son interaction avec l'environnement. L'AR est un des formalismes qui permet à des agents d'apprendre justement de leurs interactions avec leur environnement en forme de suites d'actions. Comme défini dans Sutton et Barto 1998, l'AR permet d'apprendre par essai-erreur les actions adéquates dans un environnement précis dans le but de maximiser un critère de récompense. Ceci mène à des questionnements centraux que l'on peut résumer ainsi :

1. Est-il plus efficace pour l'agent de choisir l'action la plus récompensante à un temps donné parmi les actions déjà choisies auparavant ou bien de choisir une nouvelle action (i.e. explore) dans le but de trouver une nouvelle action plus récompensante ? On parle ici de compromis exploration/exploitation qui se pose dés lors que l'agent apprend par interaction directe avec son environnement (*online*) plutôt que d'expériences pré-acquises (*offline*) ?
2. La récompense est-elle obtenue directement après interaction avec l'environnement (*single-step* ou tâches bandit manchot en référence aux machines à sous) ou bien après une longue séquence (*multi-step*) ?
3. Dans le cas d'une tâche *multi-step*, est-il préférable de mémoriser les transitions et les fonctions récompenses (on parle de modèle indirect où l'apprentissage se fait sur un modèle de la tâche qui peut être efficace mais coûteux en temps de calcul), ou bien est il préférable d'évaluer par expérimentation directe (on parle d'apprentissage direct ou l'agent apprend par interaction directe avec l'environnement sans passer par un modèle) ?
4. Enfin, en plus de la récompense, peut-on également considérer le coût computationnel de façon à forcer l'agent à apprendre en tenant compte d'un budget explicite qui limitera par exemple le nombre d'actions disponibles à l'agent avant d'atteindre la récompense, ou qui limitera le nombre d'accès à des senseurs pour un robot ou encore limiter le temps de calcul ce qui n'autorisera pas l'agent à explorer tout l'environnement.

2.2.1 Problèmes du bandit manchot

Ce qui diffère entre l'AR et l'apprentissage classique, est le fait que le *feedback* est "évaluatif" et ne fait qu'indiquer la performance d'une action (i.e bonne ou mauvaise), contrairement à l'apprentissage "instructif" qui lui donne l'information que l'action soit

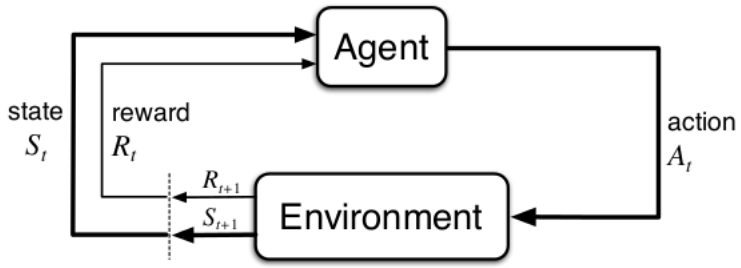

Figure 2.1 – *Schématisation de la problématique de l'AR : L'agent sélectionne une action a_t qui à son tour influencera son environnement, qui lui-même retournera un nouvel état s_{t+1} retournant potentiellement une récompense, ce qui améliorera la politique au temps suivant.*SUTTON *et* A *1998*

la meilleure ou la pire dans un cas donné. Le cas instructif donne la bonne action sans prendre en compte l'historique des actions passées ni de l'expérience de l'agent apprenant, à l'inverse de l'apprentissage purement évaluatif qui apprend selon l'expérience subie par l'agent apprenant et dépend entièrement de sont interaction avec son environnement. Dans le cadre évaluatif de l'apprentissage il existe une partie qui est *non associative* et qui n'apprend pas de l'expérience mais apprend tout de même de l'interaction avec l'environnement, que l'on nomme le problème de bandit manchot (du nom du jeu de hasard qui est la machine à sous). Dans notre travail nous nous intéressons particulièrement au cas du *multi-armed bandit*.

Le problème du *multi armed bandit* consiste à maximiser une récompense attendue en faisant face à plusieurs choix récompensants mais de probabilités différentes de distribution de la récompense. Les solutions sont multiples pour résoudre ce type de problèmes dont la majorité consiste à estimer la moyenne de la récompense pour chaque levier et de choisir le levier avec la moyenne la plus grande pour maximiser sa récompense. Or ce type d'approches n'est d'aucune utilité dans le cadre d'un problème bandit non stationnaire (dont la distribution de probabilités change au cours du temps). Dans ce dernier cas il est sensé de considérer les récompenses récentes comme étant plus importantes que les récompenses passées.

Effectivement de cette manière le modèle favorise une certaine exploration de l'environnement qui dépendra de la façon dont la récompense est perçue. Par exemple, si nous considérons $Q_t(a)$ comme étant l'estimation de la récompense moyenne de l'action a au temps t, au temps $t+1$, l'estimation devient :

$$Q_{t+1}(a) = Q_t(a) + \alpha \left[R_t - Q_t\right] \tag{2.1}$$

où R_t est la moyenne de la récompense cumulée depuis le début de l'expérience et α est un paramètre de taille de mémoire. De cette façon $Q_{t+1}(a)$ devient une moyenne pondérée des récompenses passées. L'utilité réelle de cette exploration est dans le fait que

les estimations des valeurs des actions sont incertaines. Les actions considérées comme meilleures sont celles qui semblent les meilleures à ce temps précis, or il est possible qu'une des autres actions soit meilleure. Quand on considère la problématique non stationnaire du bandit manchot, il est donc important de garder une information prenant en compte l'incertitude des actions. Dans cette optique des algorithmes appelés *Upper Confidence Bound* (UCB) ont été proposés de façon à prendre en compte à quel point l'estimation est proche du maximum et les incertitudes de cette estimation. Une façon de faire connue comme UCB-1 (2.2) est de sélectionner l'action de cette manière :

$$A_t = argmax_a \left[Q_t(a) + \rho \sqrt{\frac{logt}{N_t(a)}} \right] \tag{2.2}$$

où $N_t(a)$ est le nombre de fois où l'action a a été exécutée avant l'instant t et ρ une constante qui contrôle le degrés de l'exploration. Le second terme de l'équation 2.2 est considéré comme une mesure de la variance de l'estimation de l'action a.

D'autres variantes de cet algorithme sont apportées de façon à donner une plus grande importance aux récompenses récentes. Le *Discounted Upper Confidence Bound* (D-UCB) est proposé par Kocsis et Szepesvári 2006 où les politiques de sélection de l'UCB-1 sont augmentés avec un facteur d'atténuation qui donnera donc une importance plus grande aux nouvelles observations et oubliera les récompenses passées au fur et à mesure des expériences. D'une autre façon dans les travaux de Moulines 2008, les auteurs proposent une manière plus abrupte de valoriser les récompenses les plus récentes en introduisant une fenêtre glissante (version appelée *Sliding Window Upper Confidence Bound*, SW-UCB), qui, au lieu d'atténuer les récompenses sur tout le temps de la tâche, considérera la moyenne empirique de la récompense sur une fenêtre de taille τ sur les dernières expériences observées. Les trois variantes présentées ici sont implémentées telles quelles dans une tâche comportementale en neuroscience décrite dans le chapitre 5 pour tester si ces algorithmes sont pertinents pour expliquer l'adaptation comportementale chez l'animal en environnement incertain et non-stationnaire.

2.2.2 Processus de décisions markoviens

Les bandits manchots présentés précédemment sont pertinents dans les tâches impliquant des choix répétés parmi N actions dans une même situation de la tâche qu'on pourrait nommer un seul et même état du monde. Or dans le cas général, les agents peuvent faire des actions qui les amènent dans d'autres états de la tâche où ils auront à choisir parmi d'autres actions. La récompense peut alors n'être délivrée qu'après une séquence d'actions. Les méthodes pertinentes sont alors regroupées dans le formalisme dit de l'apprentissage par renforcement (Sutton et A 1998). Le problème de l'AR est généralement formalisé dans le cadre d'un Processus de Décision Markovien (MDP, Kaelbling et al. 1998). Un MDP est défini avec un n-uplet $< S, A, T, R >$ où :

— S est un espace d'états définissant les différentes observations que l'agent peut faire de son environnement.
— A est l'ensemble des actions pour faire évoluer l'agent dans l'environnement.
— T est la fonction de transition décrivant les états d'arrivées données les états de départ et les action effectuées. Cette fonction peut être décrite formellement par $T : S \times A \times S \to [0, 1]$, ce qui représente les probabilités de passer d'un état à un autre grâce aux action A.
— R est la fonction de récompense qui décrit les récompenses obtenues dans les états S. Elles peuvent aussi être associées aux transition T. Cette fonction est décrite par $R : S \to \mathbb{R}$ si l'obtention de la récompense est associée à un état ou $R : S \times A \to \mathbb{R}$ si elle est associée à une transition. Cette récompense peut être négative ou positive. Dans le cas général la fonction T est inconnue et l'agent doit donc l'apprendre (apprentissage indirect) soit l'approximer par essai-erreur (apprentissage direct)

A ce processus définissant un MDP nous ajouterons une fonction de politique, notée π, qui décrira le modèle du comportement de l'agent dans l'environnement en définissant les probabilités de choisir telle ou telle action dans les différents états de l'environnement de la part de l'agent. Cette fonction est définie par $\pi : S \to A$ si la politique est déterministe et $\pi : S \times A \to [0, 1]$ si elle est définie stochastique. La fonction π choisis les actions appliquées à l'environnement et de ce fait joue un rôle dans les fonctions de transition et de récompense.

Dans les cas les plus simples, les ensembles S et T sont représentés par des description simples et ponctuelles du problème modélisé. Par exemple, dans le cadre d'une modélisation en neurosciences, ces représentations sont souvent données à priori à l'agent : un état représente un stimulus (son, lumière, ...) et une action représente une réponse comportementale (pression de levier, avancer, tourner etc ...). D'autres travaux mettent en valeur l'apprentissage de représentation comme un moyen plus subtil de représenter les états et les actions, en permettant à un modèle de créer ses propres représentations de l'environnement (Droniou 2015).

2.2.3 Buts et récompenses

Comme souligné précédemment les algorithmes d'AR tendent à optimiser un certain critère, la récompense cumulée. L'agent doit donc maximiser sa récompense cumulée au cours du temps imparti pour une expérience de son environnement. C'est grâce à cette récompense cumulée que l'agent pourra choisir l'action qui le mènera à la meilleure solution. Généralement cette récompense cumulée est définie de cette façon :

$$R_t = \sum_{i=1}^{+\infty} \gamma_i . r_{t+1+i} \tag{2.3}$$

où γ représente un facteur d'atténuation qui donnera plus ou moins d'importance aux récompenses futures. Si par exemple $\gamma = 0$ les seules actions mises en avant seront celles

qui retourneront des récompenses immédiates à l'agent. Plus γ tend vers 1 et plus l'agent considérera les récompenses plus lointaines et donc pourra planifier des politiques à plus long terme.

Dans le cas où la tâche est modélisée avec un état final, tâches que l'on nommera épisodiques, l'agent effectuera une séquence d'actions jusqu'à atteindre l'état final et le problème sera réinitialisé en replaçant l'agent à son état initial. Dans ce cas l'équation précédente peut être réécrite en :

$$R_t = \sum_{i=0}^{T} \gamma_i . r_{t+i+1} \tag{2.4}$$

avec T étant le moment où l'agent atteint l'état final.

2.2.4 Fonctions de valeur

Nous avons défini dans ce qui précède que l'optimalité d'une politique va dépendre de la maximisation de la récompense cumulée. Pour ce faire, l'agent doit aussi estimer des valeurs $V^{\pi}(s)$ associées aux états afin de pouvoir juger de la meilleure action à appliquer dans un contexte donné (ces valeurs peuvent être aussi des valeurs de transition $Q^{\pi}(s,a)$). Ces valeurs seront définies respectivement par :

$$V^{\pi}(s) = E_{\pi}\left[R_t | S_t = s\right] = E_{\pi}\left[\sum_{i=1}^{+\infty} \gamma_i . r_{t+1+i} | S_t = s\right] \tag{2.5}$$

$$Q^{\pi}(s,a) = E_{\pi}\left[R_t | S_t = s, A_t = a\right] = E_{\pi}\left[\sum_{i=1}^{+\infty} \gamma_i . r_{t+1+i} | S_t = s, A_t = a\right] \tag{2.6}$$

En développant l'espérance E, on obtiendra que l'équation de la valeur soit réécrite de la façon suivante :

$$V^{\pi}(s) = \sum_{s' \in S} T(s, \pi(s), s')(R(s,a,s') + \gamma V^{\pi}(s')) \tag{2.7}$$

où s' est l'état suivant s. La valeur de l'état s est donc une somme pondérée de la récompense obtenue grâce à l'action rendue par π, ce qui en pratique équivaut à ce que la valeur de l'état le plus proche de la récompense sera la plus élevée.

A partir de là, il est donc possible d'évaluer la politique optimale π^* en maximisant le retour attendu. Si on considère deux politiques π et π', π sera meilleure dans le cas où $\forall s \in S, V^{\pi}(s) \geqslant V^{\pi'}(s)$. Les meilleures politiques seront donc associées aux valeurs d'états optimales notées V^* avec :

$$V^*(s) = \max_{\pi} V^{\pi}(s) = \max_{a} \sum_{s' \in S} T(s,a,s')(R(s,a,s') + \gamma V^*(s')) \tag{2.8}$$

et

$$Q^*(s,a) = \underset{\pi}{max} Q^\pi(s,a) = \sum_{s' \in S} T(s,a,s')(R(s,a,s') + \underset{b}{max}\gamma Q^*(s',b)) \tag{2.9}$$

L'agent doit donc apprendre à retrouver la politique optimale en suivant la fonction de valeur optimale estimée au fur et à mesure de ses interactions avec son environnement.

2.3 Programmation dynamique

Les algorithmes de programmation dynamique sont un ensemble d'algorithmes utilisées dans l'estimation des fonctions de valeur d'un modèle parfait c'est à dire que l'agent a une connaissance parfaite des fonctions de transition et de récompense. On dira que l'agent a un modèle de la tâche qu'il doit résoudre. Il s'agira donc de planifier pour trouver la meilleure politique qui maximise le critère d'optimalité, en fonction des récompenses données par cette politique selon le modèle de l'environnement.

Les algorithmes principaux dans cette catégorie, que l'on décrira dans ce qui suit, sont le *Policy Iteration* et le *Value Iteration*.

2.3.1 *Policy iteration*

L'algorithme de *policy iteration* (Howard 1960) est composé de deux étapes : l'évaluation de la politique (*policy evaluation*) et l'amélioration de la politique (*policy improvement*). La phase d'évaluation permet d'évaluer la fonction de valeur $V^\pi(s)$ pour une politique définie π. La phase d'amélioration détermine la nouvelle politique qui maximise la valeur de chaque état. Il s'agit donc de démarrer d'une politique π_0 et de mettre à jour les valeurs puis la politique pour converger vers la politique optimale π^* correspondant aux valeurs optimales V^*. L'algorithme complet est décrit dans ce qui suit :

Algorithm 1 Policy Iteration

1 : Initialize V^π and π randomly
2 : **repeat** ▷ *Policy Evaluation*
3 : $\Delta \leftarrow 0$
4 : **for all** $s \in S$ **do**
5 : $v \leftarrow V^\pi(s)$
6 : $V^\pi(s) \leftarrow \sum'_s T(s, \pi(s), s') \left[R^\pi(s) + \gamma V^\pi(s')\right]$
7 : $\Delta \leftarrow max(\Delta, |v - V^\pi(s)|)$
8 : **end for**
9 : **until** $\Delta < \epsilon$ ▷ With $0 < \epsilon < 1$

10 : $stable \leftarrow true$ ▷ *Policy Improvement*
11 : **for all** $s \in S$ **do**
12 : $b \leftarrow \pi(s)$
13 : $\pi(s) \leftarrow argmax_a \sum_{s'} T(s, \pi(s), s') \left[R^\pi(s) + \gamma V^\pi(s')\right]$
14 : **if** $b \neq \pi(s)$ **then**
15 : $stable \leftarrow false$
16 : **end if**
17 : **end for**
18 : **if** non stable **then** back to *Policy Evaluation*
19 : **end if**
return V^π and π

2.3.2 Value Iteration

Dans le cas de *value iteration* (Bellman 1957) l'amélioration se fait après une évaluation partielle de la politique. Cette méthode aura donc pour effet de converger plus rapidement vers la politique et la valeur optimale. Les algorithmes de *value iteration* font une étape d'amélioration à chaque étape d'évaluation. La mise à jour de la valeur se fait directement sur le retour maximal estimé sur l'état s'. La convergence de cet algorithme reste de même type que celui de *PI*, mais la vitesse de convergence peut être améliorée en faisant un certain nombre d'évaluations avant l'amélioration.

Algorithm 2 Value Iteration

Initialize V^{π} randomly
repeat ▷ Update the values
    $\Delta \leftarrow 0$
    for all $s \in S$ **do**
        $v \leftarrow V^{\pi}(s)$
        $V^{\pi}(s) \leftarrow \sum_{s'} T(s,a,s') \left[R^{\pi}(s) + \gamma V^{\pi}(s')\right]$
        $\Delta \leftarrow max(\Delta, |v - V^{\pi}(s)|)$
    end for
until $\Delta < \epsilon$ ▷ With $0 < \epsilon < 1$

for all $s \in S$ **do** ▷ Computing a deterministic policy
    $\pi(s) \leftarrow argmax_a \sum_{s'} T(s,a,s') \left[R_{\pi}(s) + \gamma V(s')\right]$
end for
    return V^{π}

Ces deux méthodes permettent de trouver des solutions optimales dans des problèmes stationnaires avec connaissance du modèle. Cependant, dans la plupart des cas réels, ce genre de connaissances n'est pas pas à disposition de l'agent. Il est rare que l'agent, par exemple, ait un accès direct aux fonctions de transitions et de récompense. Pour obtenir une estimation, une interaction est nécessaire avec l'environnement. Par ailleurs, la dimension de l'espace des états et des actions peut rendre l'utilisation de ces méthodes computationnellement intraitables. La compléxité combinatoire de ces algorithmes croit exponentiellement avec le nombre d'états et d'actions, ce qui limite leur applicabilité à des cas complexes.

Dans le premier cas on peut gérer ces problèmes avec des méthodes d'AR présentés dans ce qui suit. Dans le deuxième cas on utilise des fonctions d'approximation et des méthodes continues que l'on développera plus loin dans ce manuscrit (chapitre 3).

2.4 Apprentissage par renforcement

Avant de nous lancer dans les explications des algorithmes d'AR, nous ferons une brève parenthèse pour expliquer ce que sont les méthodes de Monte Carlo.

Les méthodes de Monte Carlo sont un ensemble de techniques pour l'estimation des fonctions valeurs $V^{\pi}(s)$. C'est une manière de résoudre les problèmes d'AR en moyennant des échantillons de trajectoires ou de politiques. Lorsque l'on suppose que l'expérience est épisodique et qu'un épisode atteint un état final cette méthode permet à l'agent d'avoir assez d'expérience pour bien estimer sa fonction de valeur de façon à améliorer sa politique.

Les méthodes de Monte Carlo échantillonnent et moyennent les retours de chaque paire d'état-action. Pour une politique donnée, les méthodes MC, en moyennant les retours

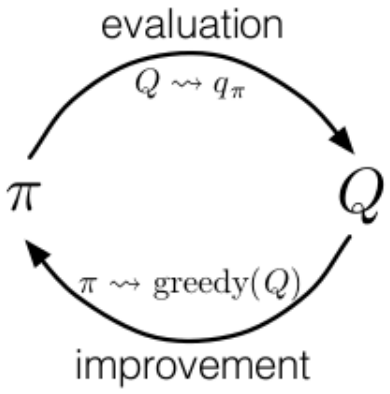

Figure 2.2 – *Méthode de Monte Carlo : A chaque étape la politique permet d'évaluer la fonction de valeurs Q qui elle-même améliorera la politique π. Dans ce cas, tout se fait grâce à une approximation discrète.*Sutton *et* A *1998*

espérés d'un ensemble de trajectoires, convergerons de ce fait vers les valeurs optimales de la fonction valeur V.

2.4.1 Apprentissage direct

Lorsque l'agent interagit directement avec l'environnement sans avoir de modèle défini, il obtient un retour d'information sur le problème qu'il peut utiliser pour estimer la fonction valeur. L'agent mettra à jour localement sa fonction valeur qui dépendra uniquement de l'état présent s, l'action effectuée par l'agent a et l'état suivant s', il devra par ce fait explorer la totalité des états pour estimer globalement la fonction de valeur.

Ce type de mise à jour est appelé *apprentissage par différence temporelle* (ou *TD learning* pour *Temporal Difference*). Il s'agira encore d'estimer les valeurs des états ou paires d'état-action connues.

$$V_{k+1}(s) = V_k(s) + \alpha\delta_k \tag{2.10}$$

avec $\alpha \in [0, 1]$ étant un taux d'apprentissage et δ_k une erreur de prédiction de la valeur de l'état s à l'itération k.

Dans le cadre d'une tâche stationnaire, on fait décroître α pour converger rapidement aux valeurs optimales.

TD(0)

Cet algorithme consiste à mettre à jour la valeur de l'état s une fois arrivé à l'état s' en fonction de l'erreur entre le retour reçu à l'état s' et l'estimation de la valeur à l'état

s. On pourra de ce fait évaluer une politique π au fur et à mesure de son déroulement :

$$V_{k+1}(s) = V_k(s) + \alpha \underbrace{\left(\overbrace{r(s) + \gamma V_k(s')}^{\text{retour perçu}} - \overbrace{V_k(s)}^{\text{prédiction}} \right)}_{\delta_k \text{ : erreur de prédiction}} \tag{2.11}$$

Cette méthode ne prend en compte que la transition effectivement expérimentée, indépendamment des autres options dans l'état s'. La modification de la politique π est donc impossible sans une information en plus qui dit de quelle façon on passe d'un état à l'autre et donc d'avoir un modèle. Cependant, au lieu d'avoir un modèle détaillé on transformera notre fonction en valeur sur les paires d'états-actions ($Q(s,a)$) en lieu et place des valeurs d'états $V(s)$.

SARSA

L'algorithme du SARSA (*State-Action-Reward-State-Action*) a été proposé par Rummery et Niranjan 1994 et permet d'apprendre une fonction Q qui affine l'estimation de la valeur de l'état en une estimation de la transition. La mise à jour de la valeur $Q(s,a)$ nécessite l'information de récompense instantanée $r(s,a)$ reçue en passant de l'état s à l'état s' et l'information de récompense reçue en exécutant l'action a' à l'état s' pour atteindre l'état d'après s'' afin d'évaluer la valeur de $Q(s',a')$:

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha \left(r(s,a) + \underbrace{\gamma Q_k(s',a')}_{\equiv V^\pi(s')} - Q_k(s,a) \right) \tag{2.12}$$

Il est ainsi toujours possible d'évaluer une politique π à l'aide des valeurs $Q^\pi(s,a)$ mais aussi de la modifier en choisissant une nouvelle action dans s. L'algorithme du SARSA apprend donc sur politique : la mise à jour de la transition courante est faite en se basant sur la l'évaluation de la transition suivante.

Q-Learning

Pour rendre indépendantes les valeurs de transition et la politique suivie par l'agent, l'apprentissage peut se faire hors politique. Pour ce faire, Watkins et Dayan 1992 proposent de remplacer l'estimation de la valeur de la paire état action $Q(s',a')$ par la valeur correspondante à l'action optimale à l'état s c.à.d l'action qui maximise la valeur à s'.

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha \left(r(s,a) + \underbrace{\gamma max_b Q_k(s',b)}_{\equiv V^\pi(s')} - Q_k(s,a) \right) \tag{2.13}$$

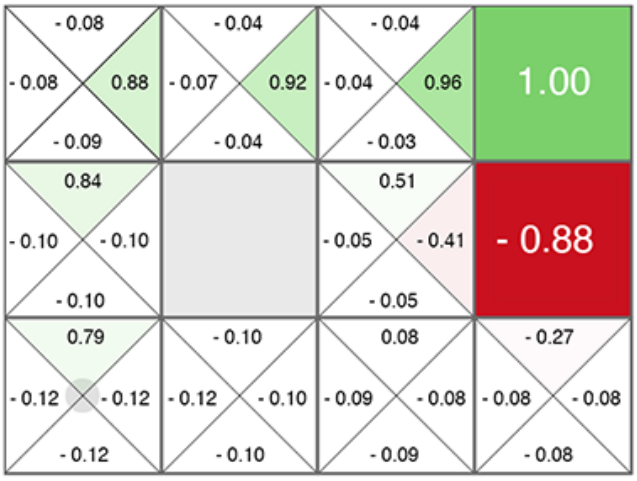

Figure 2.3 – *Exemple d'un algorithme Q-learning dans le cadre d'un labyrinthe. La récompense est située en haut à droite, le point de départ en bas à gauche. On observer un gradient depuis l'emplacement de la récompense jusqu'au point de départ. Ce gradient est une schématisation de l'atténuation de la prédiction de la récompense dépendamment des actions. Figure reproduite de Q-learning 2016*

L'algorithme se libère de la procédure exploratoire de l'agent puisqu'il évalue les transitions selon les actions qui lui semblent optimales selon sa politique courante π. La façon dont l'algorithme du *q-learning* estime les valeurs $Q(s, a)$ est montré dans la figure 2.3.

Actor-Critic

Ce type d'algorithmes a été proposé par (Witten 1977 ; Barto et al. 1983 ; Konda et Tsitsiklis 2000) où il s'agit de séparer la fonction valeur V (*Critic*) et la politique π (*Actor*). Le critique aura pour fonction d'évaluer l'erreur de prédiction et de mettre à jour la fonction valeur et l'acteur aura pour but de calculer les préférences $p(s, a)$ indiquant la probabilité de choisir l'action a à l'état s.

2.4.2 Apprentissage indirect

La deuxième manière d'apprendre serait d'estimer directement un modèle transitionnel. Au lieu d'apprendre les valeurs des état ou des couples état-action, l'agent estime les conséquences qu'auront ses décisions dans la matrice de transition de ses états. Cet apprentissage se fait par interaction avec l'environnement comme dans les techniques décrites précédemment.

Chaque expérience $(s, a, s', r(s, a))$ est intégrée aux expériences précédentes pour construire un modèle que l'agent pourra utiliser pour estimer les conséquences des actions exécutées dans un état donné. Si le résultat d'une action est stochastique, et que le modèle qui en

résulte prend en compte tous les états but possibles, on parle de modèle de distribution. S'il ne renvoie qu'une seule possibilité choisie selon la stochasticité des transitions, on parle de modèle échantillonné.

Une fois le modèle estimé, il devient possible de l'utiliser pour déterminer une politique, par exemple en appliquant les algorithmes de programmation dynamique présentés plus haut, mais également à affiner la fonction valeur comme présenté dans Dyna-Q (Sutton et al. 1992).

L'intérêt d'apprendre un modèle est souligné par Sutton et al. 1992 comme justifié par le besoin d'un raisonnement "cognitif" ou de pouvoir raisonner sur les causes et conséquences des actions. Il met en avant que la différence primaire existant entre cette approche et les approches à différence temporelle réside dans l'information résidant dans l'interaction entre les états actions est plus riche que l'information unidimensionnelle qu'est la récompense utilisée par les algorithmes TD.

La construction du modèle est dépendante du problème traité : dans des cas discrets, une méthode simple serait d'évaluer la statistique des expériences de l'agent où pour chaque couple (s, a) nous tiendrons un compte $C(s, a, s')$ pour le nombre de fois où s' a été atteint. De là, on peut déduire les probabilités de transition (s, a, s') :

$$C(s, a) = \sum_{u \in S} C(s, a, u) \tag{2.14}$$

$$P(s'|s, a) = T(s, a, s') = \frac{C(s, a, s')}{C(s, a)} \tag{2.15}$$

De la même manière l'agent peut construire un modèle qui estime en moyenne la fonction récompense :

$$R(s, a) = \frac{\sum_t r_t(s, a)}{C(s, a)} \tag{2.16}$$

Cette approche est utilisée dans (Kearns et Singh 2002) et (Brafman et Tennenholtz 2001), où un nouvel état est d'abord "évalué" sur ces statistiques avant son intégration complète au modèle.

2.4.3 Compromis exploration/exploitation et sélection de l'action

Dans ce qui a été présenté précédemment, le calcul de la politique et de la sélection de l'action à faire à l'état s est en général déterministe, en prenant l'action a qui maximise le retour. Lorsque le MDP est entièrement connu, cette pratique garantit de trouver la politique optimale, mais quand le MDP est estimé ou que l'on s'appuie sur des méthodes

directes, ce n'est plus le cas. En effet, les connaissances de l'agent sont formées par l'expérience de ses interactions avec l'environnement, et ne représentent donc potentiellement qu'un sous-espace du problème.

Dans ce cas, l'agent a parfois intérêt à choisir non pas l'action qui semble optimale a^* mais une action *à priori* sous-optimale qui lui permettra d'affiner sa connaissance du problème. On parle alors de "compromis exploration/exploitation". Cette variabilité dans le choix est d'autant plus importante si l'agent est confronté à un problème non-stationnaire, dont la structure peut évoluer au cours du temps, et pour lequel ses connaissances passées (modèle comme fonction de valeur) deviennent fausses.

Répondre à ce compromis revient à sélectionner stochastiquement l'action que l'agent effectue dans un état donné. Une écriture plus générale d'une prise de décision stochastique est la règle de sélection ϵ-*greedy*, décrite par :

$$a = \begin{cases} \text{action aléatoire}, & \text{si } X \leq \epsilon \\ argmax_a Q(s,a), & \text{sinon} \end{cases} \tag{2.17}$$

où ϵ est une probabilité faible et X une variable aléatoire suivant une loi uniforme sur $[0,1]$. Le problème de cette méthode est qu'elle choisit de manière équiprobable lorsque l'agent explore. Une action connue comme de faible valeur (parce qu'elle amène vers des récompenses négatives, par exemple) pourra autant être sélectionnée qu'une action inconnue mais potentiellement informative. Afin de prendre en compte l'estimation de la valeur des actions connues par l'agent tout en conservant la possibilité d'explorer, la règle de sélection *softmax* transforme la distribution des valeurs en une distribution de probabilités P à l'aide d'une exponentielle normalisée ou *fonction softmax* :

$$P(a|s) = \frac{exp(Q(s,a)/\tau)}{\sum_{b \in A} exp(Q(s,b)/\tau)} \tag{2.18}$$

τ étant la température de sélection : plus sa valeur est faible, plus l'agent tend à sélectionner l'action qu'il pense optimale, plus elle est grande, plus l'action sera choisie selon une distribution quasi-uniforme. On peut également voir τ comme un réglage de la sensibilité de la sélection au contraste des valeurs d'actions. La plupart des travaux considèrent des températures fixes mais certains travaux se sont intéressés à l'évolution dynamique de la température (Khamassi et al. 2011, Aklil et al. 2014, Tokic 2010, Cinotti et al. 2017) ainsi que d'autres paramètres (Schweighofer et Doya 2003) comme le taux d'apprentissage. Effectivement, dans ces travaux, il est observé que la modulation des paramètres d'exploration ou des paramètres d'apprentissage permettent une amélioration des performances de ces algorithmes, reproduisant ainsi de manière plus fidèles des comportements observés dans le cadre de la modélisation de tâches en neurosciences comportementales. Cette approche qui consiste à contrôler le compromis entre exploration et exploitation est le cœur des algorithmes de bandit manchots décrits au début de ce chapitre, mais n'ont qu'un apport très vague à la littérature de la modélisation en

neurosciences, surtout que les algorithmes d'AR ont un lien très fort avec l'apprentissage observé dans les études de modélisation en neuroscience (ceci est discuté dans chapitre 5).

2.5 Conclusion

Dans ce chapitre, nous avons décrit les techniques principales utilisées pour la modélisation de tâches discrètes en utilisant l'AR. Nous avons décrit les principaux algorithmes de bandit manchot ayant pour but de maximiser une récompense dans un environnement incertain mais aussi la résolution de ce type de problématiques à l'aide de processus de décision markoviens où deux types de solutions peuvent être utilisées : 1) la programmation dynamique qui permet, étant donné le modèle du problème, d'évaluer et d'améliorer la politique d'un agent et 2) le modèle direct, où l'interaction directe avec l'environnement permet d'estimer le modèle. Dans les deux cas l'important est de trouver une solution optimale, qui dans ce cas, s'avère être la maximisation de la récompense cumulée.

Nous avons aussi vu qu'une problématique de compromis entre exploration et exploitation émergeait selon la manière de modélisér le problème. Ce questionnement est d'autant plus important que lors de la modélisation de comportements dans des environnements non stationnaires, il nous faille prendre en compte les changements externes qui surviennent dans l'environnement. De plus cette problématique peut être observée dans le cas de la robotique où il peut être important de prendre des décisions rapides en prenant en compte une façon d'équilibrer entre des actions d'exploitation de données et des actions d'exploration de l'environnement en un temps fini.

Chapitre 3

Deep Reinforcement Learning

3.1 Introduction

Dans ce chapitre nous abordons des méthodes de l'apprentissage statistique qui traitent du cas où les états et les actions ne sont pas représentés de façon parfaite et discrète (ou tabulaire), mais qui sont approximés par des fonctions principalement décrites par des réseaux de neurones. Cette approche est utilisée dans la mise en place d'un modèle de prise de décision dans le chapitre 6. Nous commençons par montrer le lien existant entre les neurones artificiels et l'inspiration biologique qui a conduit à la création de modèles computationnels. Nous décrivons ensuite, brique par brique, la modélisation d'architectures de type réseaux de neurones en commençant par décrire la capacité computationnelle d'un neurone artificiel, de l'architecture première de classifieurs supervisés qu'est le perceptron et les perceptrons multicouche. Nous continuerons ensuite par décrire les architectures de réseaux de neurones profonds et des variantes récurrentes et à base de mémoire tels que les LSTM et GRU. Enfin nous décrirons les architecture qui mixent les méthodes d'apprentissage à base de réseaux de neurones et des méthodes d'apprentissage par renforcement décrites dans chapitre 2 et la manière d'apprendre ce type de modèles grâce aux tech-

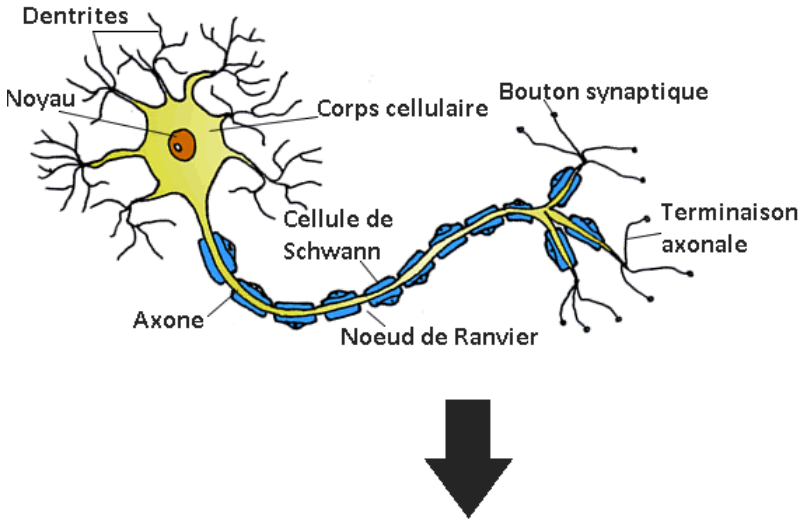

Figure 3.1 – *Exemple de neurone biologique. Les dendrites récupèrent les potentiels d'actions depuis les terminaisons axonales de neurones voisins. Ces potentiels d'actions seront "sommés" dans le corps cellulaire dont la résultante sera conduite le long de l'axone. Arrivé au terminaisons axonales, ce signal sera transmis vers d'autres neurones voisins.*

niques de *policy gradient*. Nous décrirons aussi quelques exemples d'applications utilisant ces techniques qui ont eu des résultats importants dans la littérature de l'apprentissage statistique de ces dernières années.

3.2 Réseaux de neurones

Les réseaux de neurones sont les premiers algorithmes d'apprentissage qui se sont inspirés de la caractéristique connexionniste des neurones biologiques dont l'exemple le plus basique est le perceptron. Avant de décrire en détail l'algorithme du perceptron et le modèle neuronal, nous commencerons par introduire brièvement les notions de neurone biologique et la règle de Hebb (Hebb 1949).

3.2.1 Neurone biologique

Le neurone est la cellule fondamentale du système nerveux animal. Chaque neurone est composé d'un corps cellulaire avec noyau, de dendrites pour conduire l'influx nerveux de la périphérie vers le corps cellulaire et d'un axone qui conduit le potentiel d'action émis par le corps du neurone vers les dendrites de neurones voisins. Le signal est transmis de neurone en neurone au niveau de la synapse par processus chimique. Un exemple est montré dans la figure 3.1

Grossièrement, chaque neurone reçoit en entrée des signaux venant de différents neurones voisins qu'il somme dans le corps du neurone. L'importance des signaux est dépendante de la longueur de l'axone et de l'efficacité de la liaison synaptique entre l'axone pre-synaptique et les dendrites post-synaptiques. Si le signal dépasse un certain seuil on appelle cela un potentiel d'action qui sera lui même transmis à travers l'axone vers d'autres neurones. On dit que ce neurone décharge.

En 1949, Donald Hebb émet l'hypothèse que la capacité d'apprentissage du cerveau dépend de la temporalité de décharge des neurones. Plus précisément, cette règle peut être résumée ainsi : "les neurones déchargeant en même temps renforceront leurs connexions". Ainsi, plus un neurone A décharge régulièrement juste avant un neurone B, le mécanisme biochimique accroît l'efficacité du neurone A à induire un potentiel d'action dans le neurone B, ce que l'on qualifie de plasticité synaptique. La temporalité est essentielle car si le neurone A décharge après le neurone B, l'événement inverse arrive où la connexion sera affaiblie. Dans le cas des modélisations de neurones formels cette problématique temporelle sera facilitée et on considère généralement que les neurones associés sont actifs régulièrement aux mêmes pas de temps de la simulation.

3.2.2 Neurone Formel

La première proposition de neurone formel est donnée par McCulloch et Pitts 1943. Si l'on considère un ensemble d'entrées $s \in \mathbb{R}^n$ et une sortie $y \in \mathbb{R}$ le neurone formel associe des poids w_i à chaque entrée x_i, calcule la somme pondérée de leurs poids respectifs à laquelle s'ajoute un biais b et transforme le résultat par un fonction d'activation non linéaire σ :

$$y = \sigma\left(\sum_i w_i x_i + b\right) = \sigma(\mathbf{w}^T\mathbf{x} + b) \tag{3.1}$$

De nombreux modèles de neurones formels ont été développés depuis. D'un coté ont été développé des modèles qui visent à étudier les phénomènes neuronaux précis, tels que les modèles de Hodgkin et Huxley 1953. D'autres visent à rendre compte du comportement de groupes de neurones (Eggert et Hemmen 2001). Ces modèles revendiquent une certaine plausibilité biologique et non de développement d'algorithmes d'intelligence artificielle. D'un autre coté, De nombreux modèles de neurones formels ont été développés pour servir de brique de base à la conception d'algorithmes d'apprentissage en intelligence artificielle. Les plus simples sont des prolongements du modèle de McCulloch et Pitts en utilisant différentes fonctions de combinaisons (3.2) non-linéaires, comme une sigmoïde ou une tangente hyperbolique. Plus récemment, des fonctions comme le "softplus", la fonction linéaire rectifiée (ReLU), ou le maximum ont été l'objet d'un intérêt soutenu (Nair et Hinton 2010 ; Glorot et al. 2011 ; Goodfellow et al. 2013). La philosophie de ces modèles consiste à identifier la valeur de sortie de ces neurones au taux de décharge d'un neurone biologique (ou d'un groupe de neurones). Des modèles reflétant plus finement la dimension temporelle de la réponse des neurones ont également été proposés, comme le

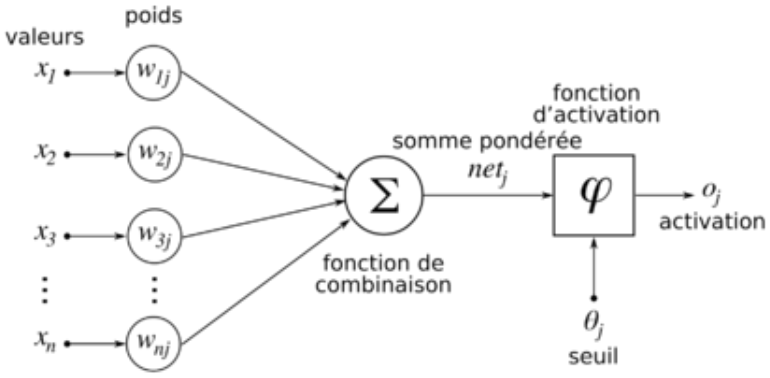

Figure 3.2 – *Exemple de neurone formel. Les entrées x_i sont multipliés par les poids w_{ij} grâce à la fonction décrite dans 3.1 pour donner en sortie la donnée transformée o_j. Figure tirée de Neurone Formel 2006*

modèle intègre et décharge (*integrate and fire*) (Burkitt 2006), qui reste confiné aux modèles bio-mimétiques ainsi qu'à certains réseaux spécifiques (par exemple dans Maass et al. 2002) du fait du coût computationnel élevé.

3.2.3 Perceptron et réseaux *feedforward*

Le premier algorithme d'apprentissage à base de réseaux de neurones s'inspirant de l'apprentissage hebbien appelé perceptron a été proposé par Frank Rosenblatt dans Rosenblatt 1958, à ceci prés que la différence post-synaptique est remplacée par l'erreur entre l'activité post-synaptique souhaitée y et celle obtenue en sortie de réseau $\hat{y}$. L'apprentissage se fait par comparaison d'une sortie post-synaptique souhaitée y et celle obtenue en sortie $\hat{y}$:

$$\Delta w_i \propto (y - \hat{y}) x_i \tag{3.2}$$

qui permet d'apprendre un classifieur linéaire.

Des architectures plus complexes permettant de classifier des données non linéairement séparables ont ensuite fait leur apparition. Ces réseaux de neurones plus complexes ont une architecture qui empile plusieurs couches à la suite avant de prédire la sortie $\hat{y}$.

A partir de ce classifieur linéaire, la complexité croissante des données ont montré les limites du perceptron. Les chercheurs ont pallié à ce problème en complexifiant les architectures des modèles en empilant des couches de neurones en plus d'un nombre croissant de neurones par couches. On obtient dès lors un perceptron multi-couches (3.3). Ce qui permet d'avoir des partitions plus complexes de l'espace et de ce fait classifier des données non linéairement séparables.

Ce type d'algorithmes est entraîné par rétro propagation de gradient (Werbos 1975 ; Parker et al. 1985 ; Le Cun 1986). Ce principe d'apprentissage consiste à minimiser

une erreur de prédiction entre une sortie prédite que l'on notera $\hat{y}$ et une sortie désirée notée y, on prendra donc un critère de comparaison que l'on notera $E(D, \Theta)$ où D est l'ensemble des données et Θ sont les paramètres du réseau. Ce critère est en général défini comme une marge simple ou une marge carrée. Il s'agit donc de trouver les paramètres Θ qui minimisent E tel que :

$$\Theta = \underset{\Theta}{\operatorname{argmin}} E(D, \Theta) \tag{3.3}$$

Dans ce qui suit nous noterons la matrice de poids entre la couche i et j par $W_{i,j}$ et la résultante h_i tel que $h_i = \sigma(W_{i-1,i}.h_{i-1})$ et $\hat{y} = f(x, \Theta)$ où f est la fonction composée $f(x, \Theta) = h_n \circ h_{n-1} \circ ... \circ h_1(x)$ qui correspond à l'ensemble des transformations de x à travers le réseau avec les paramètres Θ. En dérivant le critère E respectivement aux poids W nous minimiserons le critère par descente de gradient en utilisant un taux d'apprentissage α :

$$\Delta W_{i,j} = -\alpha \frac{\partial E}{\partial W_{i,j}} \tag{3.4}$$

On peut calculer le gradient pour l'erreur sur tout l'ensemble des données d'apprentissage D qu'on appellera apprentissage *batch*, ou pour chaque donnée séparément qu'on appellera apprentissage *online* ou bien de façon intermédiaire que l'on appellera apprentissage par *mini batch* qui consiste à créer des sous ensembles de l'ensemble total d'apprentissage.

3.3 *Deep Learning*

L'apprentissage profond (ou *Deep Learning*, DL) consiste à composer plusieurs couches de traitement de données successives dans un seul système dans le but d'obtenir des représentations de ces données à des niveaux de complexité moindres. Il s'agira d'empiler plusieurs couches de type perceptron pour un traitement plus long des données ce qui impliquera que des poids sont existants entre chaque unité de chaque couche. Après calcul des représentations jusqu'à la dernière couche, l'apprentissage se fait en calculant les dérivées de l'erreur qui sera rétro propagée dans le système jusqu'à la première couche de façon à modifier les poids dans l'optique d'optimiser ladite erreur.

On peut décrire plusieurs types d'architectures utilisés dans le cadre du DL. Un perceptron multi-couches par exemple (3.3) va créer un *mapping* entre une entrée et une sortie en utilisant une fonction composée de fonctions plus simples comme mentionnées ci-dessus. Dans le cas d'un réseau convolutionnel (CNN pour *Convolutional Neural Network*) le réseau est composé de couches de convolutions (qui sont des couches traitant des régions de pixels dans les images par des neurones), des couches de *pooling* (représentant un sous échantillonnage des images) et des couches totalement connectées (qui sont des réseaux de neurones classiques). Ces réseaux sont souvent utilisés pour l'extraction de de données complexes telles des images en couleur ou des spectrogrammes audio ou

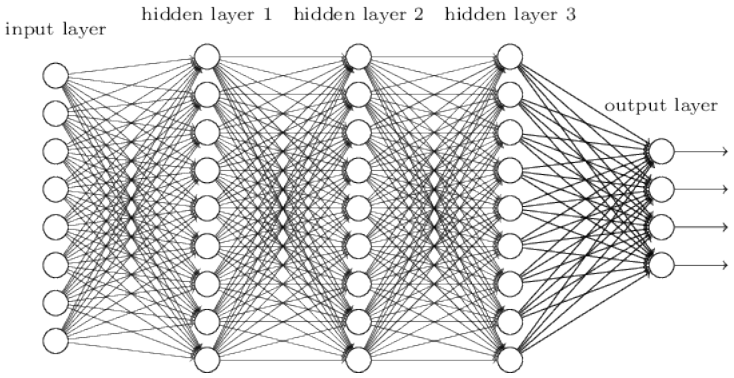

Figure 3.3 – *Exemple de neurone profond empilant des couches de perceptron les unes sur les autres avec plusieurs neurones par couche. Chaque couche devient une nouvelle projection de l'espace précédent.*

vidéo. Un réseau récurrent (RNN pour *Recurrent Neural Network*) utilisera des unités pour traiter des données séquentielles dont la connaissance contextuelle est importante et dont l'utilisation d'une mémoire interne est indispensable, on peut citer notamment la problématique du traitement intelligent du langage. Ce type d'architecture a une capacité limitée à stocker des informations pour de longues périodes en plus de la problématique de l'évanouissement du gradient (*gradient vanishing*) qui est le problème de la baisse de la valeur du gradient au fur et à mesure du temps et qui aura pour conséquence un mauvais apprentissage. Pour pallier à ce type de problèmes certaines structures de mémoire ont été proposées telles que les LSTM (*Long Short Term Memory*, Hochreiter et Urgen Schmidhuber 1997) et les GRU (*Gated Recurrent Units*, Chung et al. 2014, Cho et al. 2014).

3.3.1 Réseaux neuronaux récurrents

Les réseaux neuronaux récurrents sont une classe spécifique de réseaux de neurones où les données sont traitées par des modules ayant les mêmes paramètres de poids et biais. Lorsque le RNN est déplié comme dans la figure 3.4, il revient à une structure ressemblant aux réseaux *feedfoward*. De cette façon l'apprentissage d'une telle structure revient à utiliser un descente de gradient comme celle utilisée pour des réseaux de neurones classiques. Cette architecture, contrairement aux réseaux *feedforward*, permet de créer un état interne du réseau qui a la capacité d'engendrer des comportements temporels ce qui autorise un traitement séquentiel des données.

L'un des intérêts des RNN est qu'ils peuvent être capables de prendre en compte les informations précédentes à la tâche actuelle, comme par exemple utiliser les images précédentes d'une vidéo pour pouvoir comprendre les images actuelles. Les RNN sont performants dans le cas où ces données sont temporellement proches les unes des autres mais si les données de contextes sont séparés par un long espace temporel nous observons, en

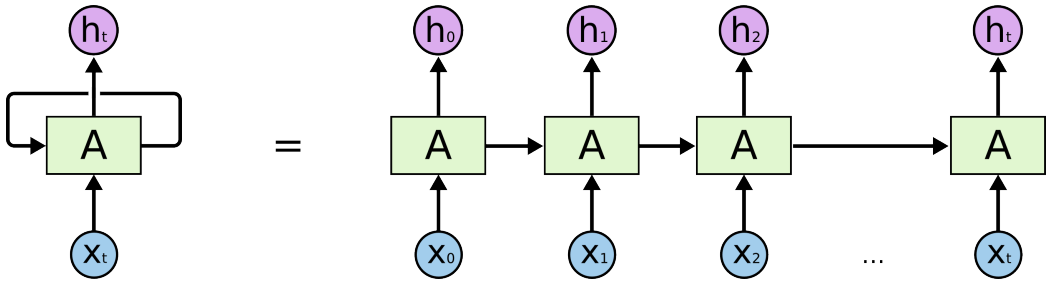

Figure 3.4 – *Exemple de réseau récurrent déroulé sur un nombre t de pas de temps. La donnée d'entrée x est aussi déroulée tel que* $x = \{x_1, x_2, ..., x_t\}$ *de façon à obtenir une sortie* $h = \{h_1, h_2, ..., h_t\}$

pratique, que les RNN sont incapables de capturer une telle dépendance temporelle à cause de l'évanouissement du gradient. Pour palier ce problème de nouvelles architectures ont été proposées.

3.3.2 *Long Short Term Memory* et *Gated Recurrent Networks*

Les réseaux Long Short Term Memory (LSTM) sont des architectures complexes qui sont utilisées pour palier au problème d'évanouissement du gradient et du problème de l'apprentissage de la dépendance temporelle, en faisant en sorte d'apprendre des réseaux avec des mémoires. Ils ont donc la possibilité d'apprendre des dépendances à long terme sur les données d'entrées.

Formellement parlant, la différence entre les RNN classiques et les LSTM est non négligeable. Habituellement, les cellules RNN sont décrites par :

$$h_{t+1} = tanh(f(x_t) \circ g(h_t)) \tag{3.5}$$

où la seule opération non linéaire se trouve dans la transformation en tangente hyperbolique. Les données qui traversent une cellule LSTM 3.5 subit plusieurs transformations que l'on pourra écrire dans la suite des équations suivantes.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{3.6}$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{3.7}$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{3.8}$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{3.9}$$
$$h_t = o_t \circ \sigma_h(c_t) \tag{3.10}$$

où σ_g est une fonction sigmoïde, W et b sont les paramètres de la cellule. La clé de la LSTM se trouve dans le c qui sera une information convoyée entre les cellules récurrentes et qui représente l'information amassée précédemment sans être atténuée par des opérations non linéaires. f_t sera la *forgetting gate* qui sera le poids de la représentation

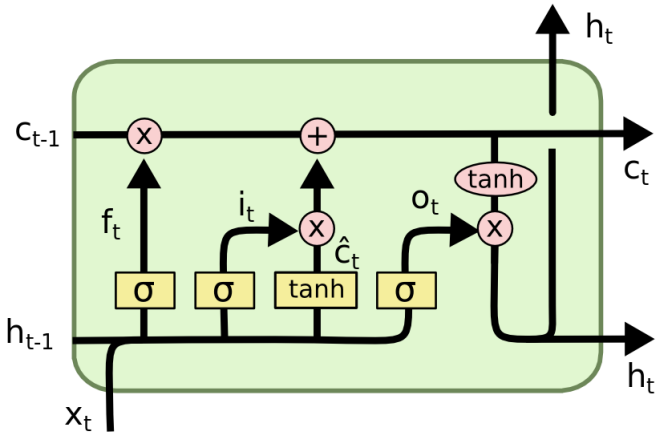

Figure 3.5 – *LSTM : La cellule reçoit en entrée h_{t-1} et x_t. f_t est appelée forget gate qui contient l'information à oublier, l'information retenue dans la cellule est composée de deux étapes i_t est appelée input gate qui décide de quelles valeurs seront mises à jour et $\hat{c}_t$ qui seront les valeurs candidates qui seront ajoutées à l'état, C_t sera la la mise à jour de l'état précédent C_{t-1} avec les information récoltées, enfin o_t et h_t seront les sorties de la cellule.*

des informations précédentes, i_t sera la *input gate* de la nouvelle information acquise et o_t sera la *output gate* qui représente la valeur de la sortie.

En 2014 Chung et al. 2014 introduisent une version simplifiée de la LSTM nommée GRU (3.6) qui fusionnera les *input et forgetting gates* en une *update gate* et que le modèle devienne :

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{3.11}$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{3.12}$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \tag{3.13}$$

Ces fonctions sont les fonctions les plus utilisées pour crée des structures complexes de traitement de données en apprentissage supervisé.

3.4 Deep learning et apprentissage par renforcement

Récemment, de nouvelles techniques d'apprentissage de politiques ont émergé. Plusieurs travaux ont en effet mis en avant l'apprentissage par renforcement en utilisant des réseaux de neurones pour approximer des fonctions valeurs ou des politiques. L'apprentissage de ce type de réseaux se fait par ce qui est appelé le *Policy Gradient*. La principale différence entre l'AR décrit dans le chapitre 2 et l'apprentissage par renforcement utilisé dans ce cadre réside dans la façon dont la politique est choisie. Au lieu de se baser sur des valeurs

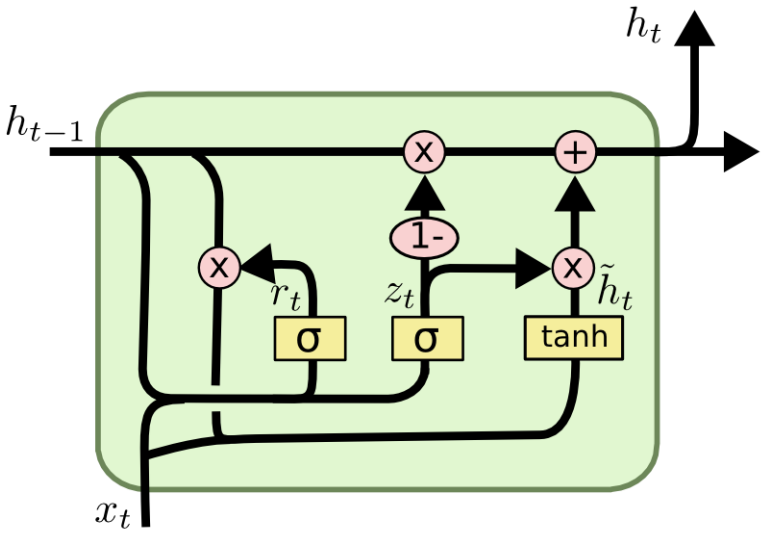

Figure 3.6 – *GRU : La cellule est simplifiée de façon à calculer uniquement r_t qui est une reset gate, z_t une update gate qui combine la forgetting gate et la input gate et la sortie h_t*

d'états actions pour sélectionner les actions, on apprend une politique paramétrée qui sélectionne les actions sans passer par des valeurs de couples états-actions.

Cette fonction d'approximation sera paramétrée par θ, ce qui impliquera que la selection d'action ne sera plus paramétrée par $\pi(a_t|s_t)$ mais par $\pi(a_t|s_t, \theta_t)$, donc l'action sélectionnée dépendra non seulement de l'état s à l'instant t mais aussi des paramètres θ de la fonction d'approximation.

Le schéma général de l'apprentissage pas *Policy gradient* est généralement tout type d'apprentissage qui suit la règle

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \tag{3.14}$$

où $J(\theta_t)$ est une critère stochastique dont l'espérance mesure la performance des paramètres θ_t. On peut aussi apprendre la politique en même temps qu'une fonction de valeurs Q, on appelle cet apprentissage simultané *actor-critic* où *actor* fait référence à la politique et le *critic* à la fonction de valeurs.

La fonction d'approximation de la politique peut être paramétrée de n'importe quelle façon tant que $\pi(a|s, \theta)$ est dérivable respectivement à ses poids, qui veut dire que $\nabla_\theta \pi(a|s, \theta)$ existe et est fini. Généralement les fonctions d'approximations sont des réseaux neuronaux plus ou moins profonds selon la tâche modélisée. L'avantage de cette approche se trouve principalement dans la capacité de généralisation lorsque plusieurs actions sont disponibles ou bien dans un cas continu.

Considérons S un ensemble d'états, un ensemble d'actions A, une politique π et un fonction de transition $P(s|a)$. On notera r_t la récompense obtenue au temps t qui dépend de l'état s_t. A chaque interaction avec l'environnement le modèle obtient une observation que l'on notera o_t. Le but reste de maximiser la récompense cumulée, au long de trajectoires que l'on notera T, mesurée par un critère de performance $J(\pi) = E_\pi [R_0] = E\left[\sum_{t=0}^{T-t} \gamma^t r_t\right]$. Les T trajectoires sont les suites de séquences d'actions et d'observations obtenues au long interactions de l'agent avec l'environnement jusqu'au temps t notée $T_t = \{a_0, o_0, ..., a_t, o_t\}$. La politique π prendra des décisions en fonction de la trajectoire observée T_t où $\pi(a|T_t)$ donnera une distribution de probabilités sur les possible actions de l'agent d'où la future action est tirée ($a_t \sim \pi(a|T_{t-1})$).

L'algorithme d'apprentissage de modèles basé sur des fonctions d'approximations se fait grâce aux algorithmes dits de *policy gradient*. Le but étant d'estimer les meilleurs paramètres θ de façon à maximiser la récompense cumulée. On notera $R(T)$ comme étant la récompense cumulée au long de la trajectoire T. On posera $P(T|\theta)$ la probabilité de l'existence de la trajectoire en fonction des paramètres θ. Avec ces notations, le critère de performance J peut être réécrit sur l'ensemble des trajectoire de façon :

$$J = \int_T P(T|\theta)R(T)dT \tag{3.15}$$

Le gradient de ce critère par rapport aux paramètres θ est devient :

$$\nabla_\theta J = \int \nabla_\theta P(T|\theta)R(T)dT \tag{3.16}$$

En utilisant une technique du ratio de vraisemblance (*likelihood ration trick*) cela devient :

$$\nabla_\theta J = \int \frac{P(T|\theta)}{P(T|\theta)} \nabla_\theta P(T|\theta)R(T)dT = \int P(T|\theta)\nabla_\theta log P(T|\theta)R(T)dT \tag{3.17}$$

En appliquant les approximation de Monte Carlo en tirant M trajectoires on réécrit le gradient de cette façon :

$$\nabla_\theta J = E_T [\nabla_\theta log P(T|\theta)R(T)] \approx \frac{1}{M}\sum_{k=1}^{M} \nabla_\theta log P(T_k|\theta)R(T_k) \tag{3.18}$$

Cette estimation est dépendante du terme de $P(T|\theta)$ inconnu du système. Cependant, les probabilités des actions sont connues par l'agent, il suffit de décomposer ce terme en fonction des probabilités des actions de façon :

$$P(T) = P(o_0) \prod_{i=1}^{|T|} P(o_i|T_{i-1}, a_{i-1})\pi(a_{i-1}|T_{i-1}) \tag{3.19}$$

qui est le produit de toutes les probabilités d'actions et d'observations dans une trajectoire T. En utilisant la capacité de la dérivé de la fonction logarithme on pourra réécrire le critère J comme :

$$\nabla_\theta J \approx \frac{1}{M}\sum_{i=1}^{M}\sum_{t=0}^{T} log\pi(a_t|T_t^i)R_t^i \tag{3.20}$$

où R_t^i est la récompense définie plus haut.

Avec ce gradient l'incrémentation des paramètres θ se fera à chaque pas de temps proportionnellement à la récompense obtenue et ira dans la direction de l'action choisie au temps t. Plus le retour est grand, plus la probabilité de l'action dont la conséquence a été l'obtention de cette récompense en est augmentée, de ce fait l'action sera de plus en plus souvent sélectionnée. En itérant ce processus à chaque pas de temps on obtiendra une politique π approchant une politique optimale π^*.

3.5 Conclusion

Dans ce chapitre nous avons parlé des réseaux de neurones comme modèles d'apprentissage utilisés dans le cadre de l'apprentissage supervisé. Nous avons montré les origines qui ont conduit à l'utilisation de ce type d'algorithmes dans le cas de l'apprentissage statistique. Ces approches montrent aussi le mérite de l'apprentissage basé sur des données et des approches statistiques qui sont meilleures que des systèmes basés sur des règles explicites.

Ces approches ont montré leurs performances empiriquement dans des applications réelles tels que dans de récents travaux de *DeepMind* dans lesquels les auteurs ont réussi à mettre en place des modèles apprenant des tâches considérées complexes tels que jouer à des jeux *Atari* (Mnih et al. 2015) ou du jeu au Go (*AlphaGo*, Silver et al. 2016).

Les principes décrits dans ce chapitre seront la base de notre travail développé dans les chapitres 6 et 7, précisément les notions concernant le *policy gradient* qui sera la principale technique utilisée pour l'apprentissage des tâches modélisées.

Chapitre 4

Méthodes séquentielles budgétisées pour la classification

4.1 Introduction

Dans ce chapitre nous introduisons les algorithmes de l'état de l'art utilisant des budgets explicites lors de traitements de données dans le cadre de l'apprentissage statistique. Nous commencerons par développer l'intuition derrière l'inclusion d'un budget dans le traitement de données, nous continuerons par introduire les algorithmes de l'état de l'art dans le cadre de traitement de textes et d'images sous contrainte de budget. Nous finirons par introduire l'utilité d'une telle approche dans le cadre de la robotique et du lien entre l'apprentissage budgétisé et ses possibles utilisations dans la problématique du compromis entre exploration et exploitation.

4.2 Classification séquentielle

Dans le cas de la classification supervisée, généralement les données sont représentées comme étant atomiques, ce qui veut dire que les caractéristiques (i.e les composantes des vecteurs des données d'entrée) sont données d'un bloc, mais il est avéré que dans les cas réels les données ne sont pas disponibles en totalité mais amassées au fur et à mesure

que l'agent amasse des données conditionnellement aux actions qu'il aura exécutées. Ceci reviendrait donc à un type de prise de décision séquentielle ressemblant très fortement à la façon d'apprendre par renforcement qui est décrite dans le chapitre 2.

Certains travaux adaptent de ce fait les algorithmes de classification classiques, utilisant du *batch* habituellement, à des algorithmes de traitement séquentiel des données comme par exemple des tâches de diagnostic médical, les interfaces utilisateurs ou les données à flux continu (*streaming data*). Ces problématiques sont traitées avec des modèles que l'on nommera Modèles de Prédiction Séquentielle (PSM en anglais pour *Prediction with Sequential Models*).

Il existe deux classes de ce type de problèmes : les problèmes de caractéristiques fixes (*fixed features problems*) et les problèmes de caractéristiques libres (*free features problems*). Dans le cas du premier les caractéristiques ou *features* sont fixées et l'algorithme apprend de lui même a quel moment arrêter sa collection de données pour émettre une décision. Dans le deuxième cas les *features* sont libres d'accès et l'algorithme apprend donc simultanément dans quel ordre acquérir les données et à quel moment s'arrêter. La deuxième classe de problèmes est plus complexe que la première du fait de la complexité combinatoire de l'ordre d'acquisition des données.

4.2.1 Problèmes à caractéristiques fixes

Cette problématique a été introduite en premier dans Wald 1945 qui considérait déjà une méthode séquentielle d'analyse avec un critère d'arrêt, ce qui a montré que l'on pouvait considérer un arrêt précoce de l'acquisition si ce qui a été déjà acquis est suffisant pour pouvoir prendre une décision.

L'idée générale est que chaque donnée x est composée d'un ensemble ordonné de caractéristiques qui est fixé durant l'inférence. Le but du classifieur est d'être capable d'itérativement considérer chaque caractéristique et de la classifier le plus tôt possible des qu'assez d'information ont été amassée pour chaque donnée x particulière.

Classifieurs en cascade

L'un des moyens d'apprendre ce type de tâches est d'intégrer un classifieur en cascade. Un exemple de cette architecture est donné dans la figure (4.1) tiré du papier Viola et Jones 2001. Ce classifieur est utilisé sur des images qui contiennent des visages. A chaque couche la donnée est soit rejetée en tant que non-visage soit transmise à la couche suivante pour être traitée. Le fait que le modèle soit entraîné sur les données acceptées par les classifieurs précédents conduit au fait que les classifieurs sont de plus en plus complexes. Ce classifieur est d'ailleurs considéré comme l'un des meilleurs pour la détection des visages. Les deux problèmes d'un tel classifieur en cascade sont 1) le fait que le seuil

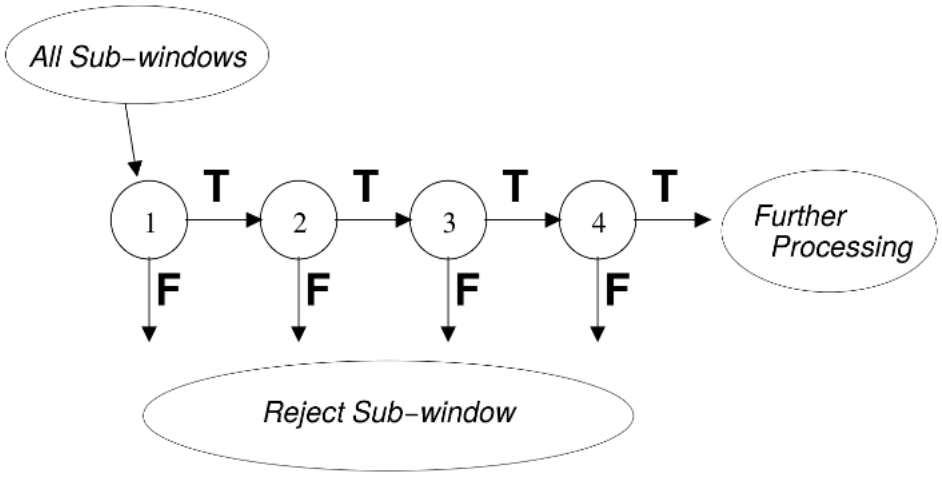

Figure 4.1 – *Classifieur en cascade tiré de* Viola *et* Jones *2001, les quatre premiers éléments du classifieur en cascade sont représentés comme un graphe acyclique dirigé. T étant une fonction de transformation et F la fonction évaluant la donnée rejetée*

de décision d'acceptation ou de rejet d'une image est réglé à la main et 2) le coût réel de chaque niveau du classifieur n'est pas pris en compte.

D'autres contributions utilisent des coûts à chaque couche du classifieur, comme par exemple dans Raykar et al. 2010, où les auteurs associent des coûts pour chaque caractéristique et le classifieur est ensuite arrangé selon leurs coûts avec un paramètre explicite de compromis entre precision et coût. Nous pouvons aussi citer Chen et al 2012 qui utilise toute les caractéristiques initialement et pénalise durant l'entraînement l'acquisition des données ce qui conduit le classifieur à s'ordonner naturellement en une architecture qui prend en compte le compromis coût/précision. Dans Poczos et al 2009 les auteurs ont modélisé le critère d'arrêt en utilisant un classifieur en forme de MDP qui apprend grâce à une récompense distribuée conditionnellement à la capacité du classifieur à une bonne prédiction, cette approche inclut un apprentissage basé sur du *policy gradient*.

Direct Acyclic Graphs (DAG) classifiers

La contrainte implicite des modèles décrits ci dessus est le fait que les données qui arrivent à la dernière couche du classifieur traversent la totalité des couches intermédiaires. De ce fait les classifieurs en cascade sont des DAG avec un seul chemin. Mais en généralisant cette vision en ayant un DAG à plusieurs chemins, la rapidité de traitement des données est optimisée, par exemple, certaines données ne seront traitées qu'au début puis à la fin du DAG uniquement sans traverser la totalité de l'architecture. Par exemple, dans D. Benbouzid 2012, les auteurs ont mis en place un modèle de classifieur de type DAG qui autorise une flexibilité quant à la façon de traiter les données. Les données ne

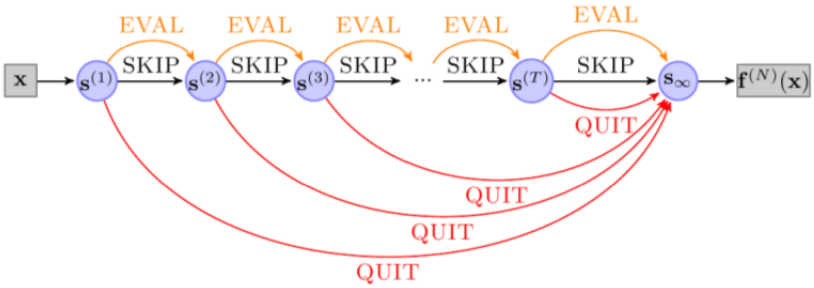

Figure 4.2 – *Schématisation du modèle de classification MDDAG développé dans* D. Benbouzid *2012. A chaque niveau, le modèle autorise 1) de transformer la donnée d'entrée avec EVAL, 2) de passer à la couche suivante sans opération dans l'actuelle couche (fonction SKIP) où 3) passer directement à la classification avec la fonction QUIT*

sont pas simplement rejetées ou acceptées mais elles peuvent aussi être transmises vers les prochains classifieurs telles quelles sans être traitées par l'actuel classifieur (4.2).

Cette architecture permet dès lors d'obtenir différentes manières de classifier les données. Par exemple, pour deux données de même classe, le modèle peut retourner deux différents chemins pour les classifier, direct dans un cas en passant à la classification aprés une opération et indirect dans un autre cas passant par plusieurs transformations intermédiaires avant de classifier.

4.2.2 Problèmes à caractéristique libre

Ce cas ci est une généralisation du cas précèdent. Dans ce problème on considère que le classifieur a un accès libre aux caractéristiques des données et est capable de choisir la meilleure caractéristique dans l'ensemble des caractéristiques disponibles et a la possibilité d'arrêter le processus de classification à tout moment. A chaque fois qu'une caractéristique est utilisée, elle est retirée de l'ensemble des caractéristiques que le classifieur utilise. Ce qui conduit à un processus plus complexe vu que n'importe quel chemin est possible ce qui revient à apprendre une politique de sélection qui rend la problématique plus complexe.

Les arbres de décision

Les premiers classifieurs de cette catégorie ont été développés par Breimann 1993. L'arbre de décision considère itérativement les données une par une en décidant à chaque étape de quelle caractéristique utiliser ensuite. Le processus de décision à chaque nœud ne prend généralement en compte que les caractéristiques associés à ce nœud et la classification finale est faite lors de la réception d'une caractéristique dans un nœud feuille qui aura une classe associée. Par exemple dans Quinlan 1993, l'arbre est construit itérativement en séparant l'ensemble des données sur les caractéristiques les plus discriminantes.

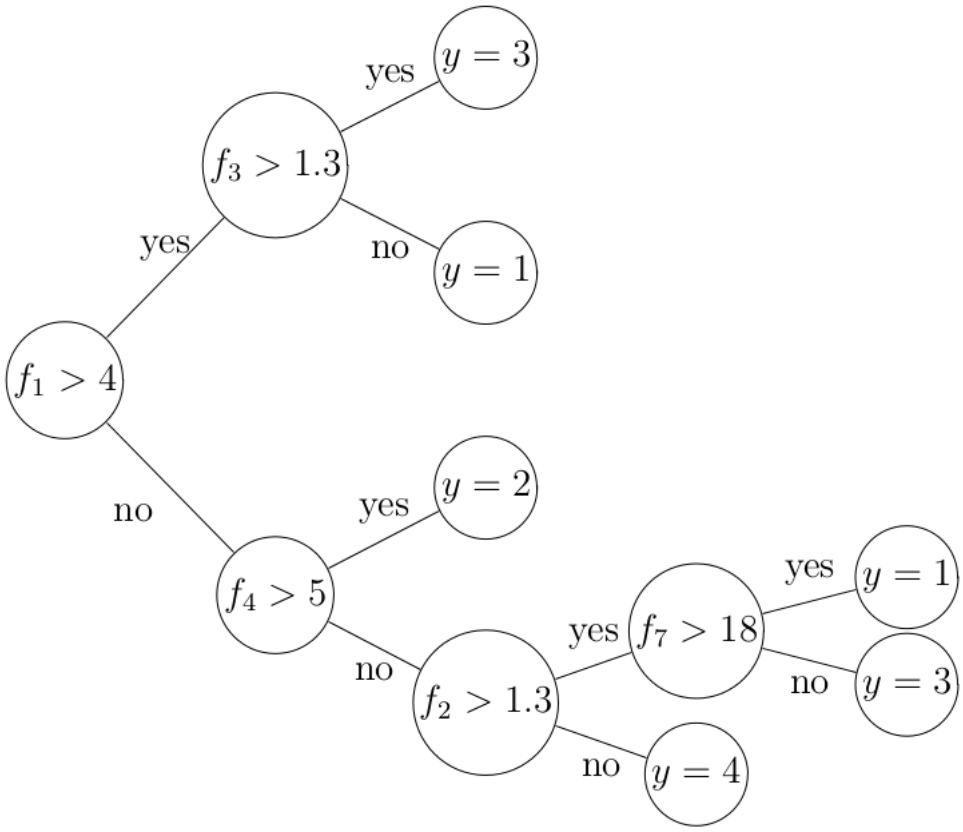

Figure 4.3 – *Exemple d'un arbre de décision à 4 classes. Chaque nœud est un nœud arborant une condition qui sépare les classes qui se trouvent dans les feuilles.*

Cette architecture initiale généralement fait du sur-apprentissage et est donc élagué pour trouver une balance entre la précision et la généralisation.

Cette approche est une solution efficace aux problèmes à caractéristiques libres, il existe malgré tout plusieurs problèmes :

- L'ordre d'acquisition des caractéristiques est basée sur une heuristique, ce qui peut conduire à un ordre qui ne serait pas nécessairement le meilleur pour la classification finale. Ajoutons à cela que le fait d'utiliser une heuristique à l'aveugle peut conduire à un sur-apprentissage.
- A coté de cela chaque nœud ne prend en compte que les caractéristiques associées uniquement à ce même nœud et ne prend pas en considération les valeurs exacts des caractéristiques des nœuds précédents, et n'utilise qu'un seuil pour les discriminer.
- Du fait de l'absence d'une fonction de coût, l'application d'une telle approche à un problème plus complexe, éventuellement multi objectif, nécessiterait l'utilisation d'algorithmes *ad hoc*. Bien que des arbres de décision qui prennent en compte un coût existent (Turney 1995), ils nécessitent malgré tout un algorithme d'appren-

tissage spécifique (algorithmes génétiques dans le cas de Truney), ou incluent des heuristiques ad-hoc qui ne sont globalement pas optimales (Dredze et Elias-bachrach 2007).

Certains travaux récents proposent des arbres de décisions avec des fonctions de coûts (Xu et al. 2013) qui imitent l'idée développée dans Chen et al. 2012 pour les classifieurs en cascade et les appliques à des arbres de décision. En mettant en place une fonction de coût paramétrée par une fonction de perte (*loss function*), cette approche permet d'avoir des seuils plus flexible et adaptatifs plutôt qu'une heuristique directe. De cette façon on obtient un classifieur capable de classifier efficacement, malgré un algorithme d'apprentissage complexe.

Active learning

Il existe un large spectre dans la littérature sur l'apprentissage actif pour de la classification budgétisée (Greiner et al. 2002, Kapoor et Greiner 2005). Ces approches modélisent l'apprentissage avec un *PAC-based learner* (PAC pour *Probably Approximately Correct*), et ainsi être capable de trouver un classifieur actif qui trouverait un équilibre entre éviter des caractéristiques coûteuses tout en prenant en compte leur impact positif attendu sur la classification.

Greiner et al. 2002 est l'un des premiers à avoir introduit les classifieurs actifs sensibles au coût : des classifieurs qui peuvent demander certaines caractéristiques plutôt que d'autres durant le processus de classification, avec un budget sur ces caractéristiques. Ce travail considère le cas d'un classifieur centré sur une table de consultation avec des données représentées en binaire. Les auteurs concluent qu'apprendre un classifieur exact et optimal dans ce type de tâches est généralement intraitable. Ce travail est très centré sur des appreneurs PAC et n'est pas implémenté dans un quelconque système. Les auteurs évoquent aussi la similarité de cette approche de problèmes formalisés sous forme de MDP mais ne développent pas le lien entre leur méthode et une approche RL complète. Globalement ce travail est intéressant car il introduit la problématique mais ne donne pas d'implémentations ou d'applications. Il montre cependant la difficulté de mettre en place de bonnes politiques de sélection de caractéristiques dans un tel problème.

Un autre exemple plus récent (Kapoor et Greiner 2005), utilise des contraintes de budget lors de la phase d'apprentissage et de la phase de test avec une approche en MDP qui apprendra des classifieurs budgétisés, en mettant un budget b_L sur les données acquises en phase d'apprentissage pour produire, lors de la phase de test, des prédictions fiables avec un budget différent b_C. Le modèle en apprentissage apprend dés lors les meilleures caractéristiques à utiliser pour chaque donnée traitée dans le temps imparti.

4.3 Approches par apprentissage par renforcement

Ici nous mettrons en lumière les approches d'apprentissage budgétisé basées sur des algorithmes d'apprentissage pas renforcement. Ces approches sont les plus développées dans le cadre de la classification sous contraintes de budgets. Nous pourrons citer par exemple l'article de Ji et Carin 2007 où les auteurs modélisent la tâche de la sélection de caractéristiques en processus de décision markovien partiellement observable POMDP (Kaelbling et al. 1998) et sont rapidement confrontés à la difficulté à trouver une bonne politique de sélection dans un POMDP due à la grande complexité de la modélisation des aspects partiels du POMDP. La modélisation en POMDP ne permet pas à leur agent de garder facilement trace de quelles caractéristiques ont été sélectionnées précédemment, un problème qui requiert l'ajout d'informations supplémentaires pour pouvoir être traité.

Dans Gao et Koller 2011, les auteurs considèrent un modèle consistant en un ensemble intelligent de classifieurs où seuls certains classifieurs seront utilisés lors de l'inférence d'une certaine donnée. Ils associent un coût de traitement à chaque classifieur et tentent d'inférer sur un ensemble de données en essayant de minimiser le coût de calcul global. Bien qu'ils ne structurent pas explicitement leur problème en termes de caractéristiques individuelles, rien n'empêche de remplacer les classifieurs de base par des caractéristiques extraites directement d'une donnée x. Les auteurs n'ont pas explicitement modélisés le problème sous forme de MDP, mais en pratique ils ont bien défini une fonction de récompense et une politique bayésienne dont le but est de maximiser la récompense relative à la classification, conditionnellement aux classifieurs précédemment sélectionnés. Contrairement à certaines approches qui pénalisent les caractéristiques individuelles utilisés par des ensembles de classifieurs, cette méthode ne pénalise pas les caractéristiques mais le temps de calcul des classifieurs.

Dans ces approches budgétisées basées sur de l'apprentissage par renforcement, Dulac-Arnold et al. 2011 propose de considérer la classification de texte comme un processus de prise de décision séquentielle. Dans ce processus un agent apprend à classifier les documents en parcourant les phrases dans les textes séquentiellement et apprend à arrêter la lecture quand il considère que les informations actuelles sont suffisantes pour pouvoir classifier le document. L'agent commence à lire à une position donnée du document et décide entre plusieurs actions : soit classifier le document, lire une autre phrase ou bien arrêter le processus considérant le document comme bien classifié. L'apprentissage se fait en estimant une fonction $Q(s, a)$ qui minimisera la fonction de perte associée à la bonne classification du document. Cette $Q - fonction$ sera le critère qui permettra de choisir les meilleurs actions à chaque état s (s étant une phrase).

Dans Dulac-Arnold et al. 2014 cette approche séquentielle est appliquée aux images de façon à n'utiliser que certaines fenêtres (*patchs*) d'une image donnée de façon à minimiser le nombre d'opérations. Les auteurs proposent un modèle séquentiel qui, étant donné une image, sélectionne un sous ensemble de régions dans cette image et la classifie. Le modèle inclut un budget explicite qui sera la taille du sous ensemble de régions utili-

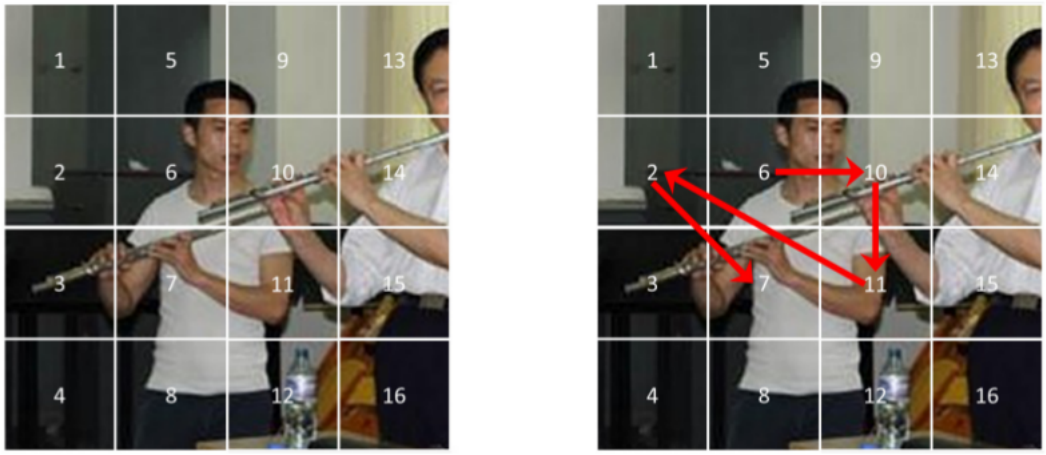

Figure 4.4 – *Exemple de politique de sélection de fenêtres reproduite de* Dulac-Arnold *et al. 2014*

sées pour classifier la donnée. Le modèle apprend d'abord une fonction f_θ qui apprendra les représentations et la classification des régions qui seront dépendantes du budget B donné en amont. A partir de la région du milieu, les possibilités de groupes de régions de taille B selon une politique exploratoire π. L'idée serait d'apprendre en amont une bonne fonction de classification pour chaque sous ensemble de B régions. Une fois cette fonction apprise, le modèle apprend une politique efficace qui sélectionne la meilleure trajectoire dans l'image qui donnera la meilleure performance au classifieur. Le modèle proposé dans cet article a pour but de trouver l'ensemble des régions qui minimisent l'erreur de classification d'une image, cependant, les auteurs entraînent la fonction de classification (appelée f) sur des sous ensembles de fenêtres et ce n'est qu'une fois f apprise que le modèle apprend la politique qui à sélectionner les ensembles de fenêtres les plus efficaces. Dans le travail présenté dans cette thèse, nous proposons un modèle proche de celui présenté dans ce travail avec la différence que le modèle apprend la fonction de classification en même temps que la politique.

Dans Contardo et al. 2016, les auteurs proposent d'associer un budget à chaque caractéristique un coût particulier d'acquisition. Le modèle proposé acquiert des caractéristiques d'une manière adaptative et peuvent être acquises par blocs, ce qui autorise le traitement de données des données à grande dimension. Le modèle est aussi basé sur de l'apprentissage de représentations.

Dans Mnih et al. 2014 les auteurs proposent un algorithme d'attention pour de la reconnaissance d'images. Les auteurs justifient cette approche par le coût dépensé pour un traitement de données par des réseaux de neurones de convolution, qui malgré leurs performances, ont l'inconvénient d'être très coûteux en temps de calcul. Leur proposition de résumé à un modèle qui permet de sélectionner une "zone d'attention" dans les images

pour les traiter plus rapidement. Ces zones d'attentions seront des séquences limitées d'images (ou vidéos) dans les données de plus grande échelle. Le problème de l'attention est considéré comme étant un problème séquentiel, où à chaque pas de temps l'agent observe une partie de l'environnement avec un senseur limité. L'agent peut contrôler la manière de déployer ce senseur (en choisissant la location du senseur) et vu que l'information n'est que partiellement observable, l'agent doit apprendre à intégrer l'information au cours du temps pour pouvoir déployer ses senseurs le plus efficacement possible. A chaque pas de temps l'agent reçoit une récompense qui dépend de l'action exécutée. L'agent à droit à deux types d'actions : la translation de la "rétine" qui est la fenêtre pour capturer une partie de la donnée et une action influençant l'état de l'environnement.

Dans Graves 2016 les auteurs proposent un modèle d'arrêt automatique pour des réseaux de neurones récurrents adaptatif en terme de temps de calcul. Les auteurs présentent un réseau neuronal augmenté d'une unité de *halting* qui décidera à quelle étape le réseau doit arrêter ses opérations. Le réseau résultant est une augmentation du réseau neuronal où à chaque donnée d'entrée une séquence de transformations s_t^i sera amorcée mais qui sera susceptible d'être interrompue suite à la décision de la *halting unit*. L'apprentissage d'une telle structure se fait grâce à une rétro propagation du gradient classique ce qui permet de ne pas perdre les propriétés des réseaux de neurones et de l'efficacité de leur apprentissage.

Dans cette thèse nous nous sommes basés particulièrement sur le travail de Denoyer et Gallinari 2014, dans lequel les auteurs ont développé un modèle de sélection séquentielle pour des réseaux profonds. Dans cet article, les auteurs considèrent le processus de classification comme un processus séquentiel où chaque couche de la structure est composée de nœuds qui représentent des fonctions de transformation. La transformation de la donnée à l'entrée sera dépendante de la trajectoire que sélectionnera la politique de sélection de nœuds.

Plus précisément à chaque couche une fonction de transformation sera sélectionnée parmi celles disponibles sur l'ensemble de la couche et accessible par le nœud actuel. Lorsqu'une donnée x est présentée en entrée de la structure, elle suivra une trajectoire dans le DAG, comme montré dans 4.5, jusqu'à obtenir une sortie y. La trajectoire est sélectionnée à chaque niveau par une fonction de sélection p qui sera apprise par les algorithmes de *policy gradient* alors que les fonctions de transformation seront apprises grâce à une rétro propagation du gradient classique.

On peut voir cette architecture comme une extension des réseaux de neurones profonds classiques, qui au lieu de considérer le réseau comme une opération atomique, le considère comme un enchaînement de transformations différentes. Pour preuve, si l'on considère qu'à chaque couche il n'existe qu'une seule fonction de transformation, on obtiendra de ce fait un réseau de neurones classique. Le processus d'inférence peut être décrit ainsi :

— Étant donné une entrée x, le modèle choisira parmi les différentes fonctions de transformation possibles, posons f_i.

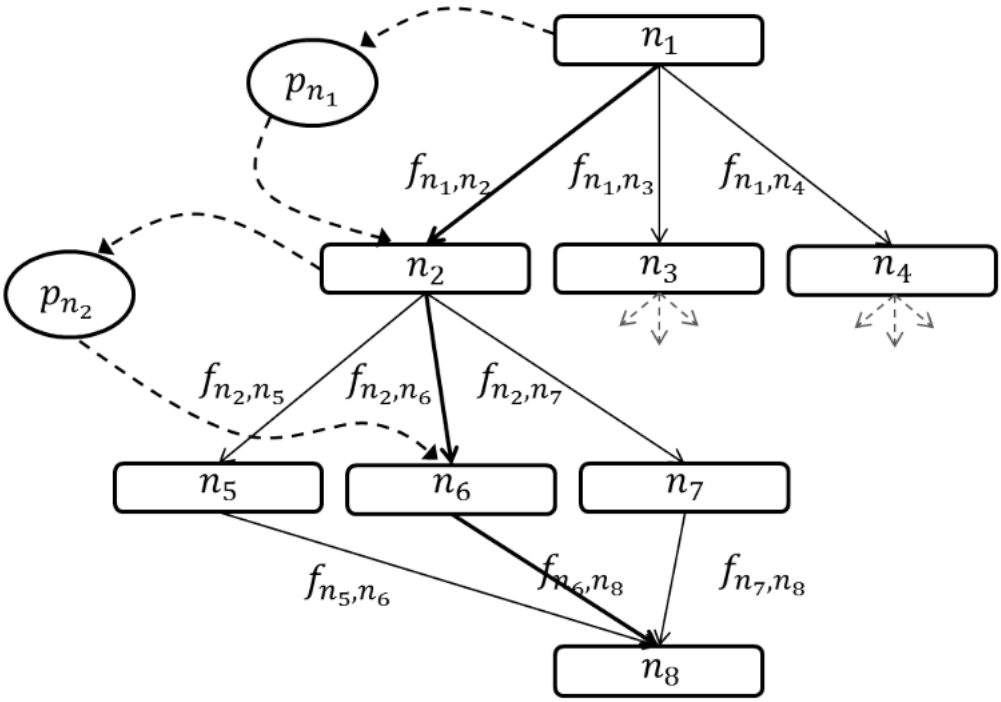

Figure 4.5 – *DSNN (*Denoyer *et* Gallinari *2014) : Cette figure représente l'architecture en graphe du DSNN. L'entrée est traitée séquentiellement selon les probabilités de sélection de l'une ou l'autre des fonctions de transformations. Ici par exemple la trajectoire sélectionnée par les experts de sélection de politique est de faire passer l'entrée de n_1 vers n_2 avec p_{n_1} puis n_6 avec p_{n_2} pour enfin aller au dernier nœud n_8 ce qui donnera à a fin du processus une prédiction $\hat{y} = f_{n6,n8}(f_{n_2,n_6}(f_{n_1,n_2}(x)))$*

— x est alors transformé dans une nouvel espace de représentation en $x' = f_i(x)$
— Avec x' une nouvelle fonction de transformation est sélectionnée et ce jusqu'à atteindre la dernière couche du réseau qui représente la sortie prédite.

Les apprentissages des fonctions de transformations et des fonctions de sélections sont faits simultanément et non l'un après l'autre. Au lieu d'apprendre les séquences de représentation successives cette architecture apprendra à sélectionner la fonction de représentation la plus appropriée aux données.

4.4 Conclusion

Dans ce chapitre nous avons présenté des modèles venant de l'état de l'art de l'apprentissage statistique, qui considèrent le processus de classification comme un processus séquentiel d'acquisition de données. Ces modèles, qui s'inspirent de la littérature de

l'apprentissage par renforcement, y ajoutent des contraintes explicites de budget, pour un traitement efficace et limité des données, de façon à optimiser un compromis entre précision et temps de calcul, que l'on pourra facilement rapprocher du compromis exploration/exploitation défini dans le chapitre 2 sur l'apprentissage par renforcement.

Ces modèles ont l'avantage de traiter des données complexes et variées, mais peu de travaux ont été menés en prenant en compte la multimodalité des données ou le cas d'actions retournant des observations nulles. En effet les travaux cités dans ce chapitre prennent en compte uniquement des données de même nature, par exemple, le classifieur peut être spécialisé sur des données images ou texte mais n'utilise pas une multimodalité pour classifier. D'un autre coté, ces modèles sont formalisés de façon à recevoir une donnée entre chaque action, par exemple dans le cas du modèle présenté dans les travaux de Dulac-Arnold les actions sont des actions de déplacement dans l'images et les observations sont les fenêtres et à chaque action le modèle est nourri d'une fenêtre.

Cette approche est celle adoptée dans les contributions décrites dans les chapitre 6 et chapitre 7 pour la mise en place d'un modèle d'apprentissage séquentiel appliqué à la localisation en robotique. En effet, il apparaît utile de mettre en place un budget explicite dans le cas d'une application robotique de façon à limiter les robots dans leurs manière de résoudre les différentes tâches assignées.

Résumé de l'état de l'art

Dans l'introduction et dans les chapitres précédents nous avons mis en valeur plusieurs domaines différents qui s'intéressent néanmoins à la même problématique qui est de modéliser et de comprendre la prise de décision chez des agents autonomes. Nous avons vu que cette problématique engendrait un questionnement fondamental que l'on peut résumer en l'étude du compromis entre exploration et exploitation, que ce soit dans le cas de la prise de décision en apprentissage par renforcement dans le chapitre 2, dans le cadre de l'étude du comportement animal (discuté dans le chapitre 5) ou dans l'acquisition de données dans le cadre de la classification séquentielle en apprentissage supervisé dans le chapitre 4.

Nous avons vu que dans le cadre de l'apprentissage par renforcement, cette problématique est centrale lors de l'implémentation de différents systèmes de prise de décision en mettant en valeur la dichotomie entre modèle direct et modèle indirect. Ces problématiques sont posées dans le cadre de la robotique autonome par les limitations matérielles. Dans ce cadre, plusieurs travaux ont proposé de faire coordonner des modèles de prises de décision multiples tels que (Renaudo et al. 2014, Chavarriaga et al. 2005, Girard et Khamassi 2016 pour une revue de l'état de l'art). Cependant, ces modèles ne prennent pas en compte les limitations physiques tels que le coût computationnel d'une prise de décision, la capacité de calcul des processeurs embarqués ou bien les limites que l'on peut donner au robot au niveau des actions possibles.

Nous avons ensuite introduit les algorithmes de classification séquentielle dans lesquels les modèles sont très inspirés de la littérature de l'apprentissage par renforcement. Ces modèles sont basés sur des budgets explicites avec lesquels les limitations en terme de données sont prises en compte, par exemple en coût computationnel ou en nombre d'action autorisées. Ces algorithmes attaquent la problématique du compromis exploration/exploitation de manière différente, où au lieu de considérer plusieurs modèles d'apprentissages et d'ensuite en sélectionner le meilleur, tente au contraire d'introduire des limitations explicites aux modèles de façon à intégrer ces limitations aux processus d'apprentissage. Néanmoins, ces méthodes de classification séquentielle fonctionnent pour le moment principalement de façon *offline*, et il est donc important d'étudier dans quelle mesure elles peuvent être pertinentes pour la robotique, qui traite la plupart du temps de problématique de prise de décision *online*.

L'objectif principal à long terme auquel cette thèse vise à contribuer est de faire converger les différents travaux, qu'ils soient en neurosciences computationnelles ou en robotique/apprentissage statistique, vers des modèles de traitement en ligne de l'apprentissage budgétisé. Dans le cas des neurosciences computationnelles, nous avons étudié un modèle de régulation du compromis exploration/exploitation (présenté dans le chapitre 5) dans le but de proposer des pistes futures pour la conception d'un modèle d'apprentissage budgétisé en ligne. D'un autre coté, le but de l'étude des modèles budgétisés (présenté dans le chapitre 6) était d'introduire un budget explicite dans les modèles de neurosciences computationnelles et de robotique, et voir dans quelle mesure cela peut permettre à un robot de naviguer de façon autonome tout en cherchant à minimiser le budget qui lui est attribué en terme de nombre d'actions qui lui est permis pour acquérir de l'information et ainsi se localiser.

Dans cette optique, nous avons mené deux types de travaux qui sont décrits dans les chapitres qui suivent. Le chapitre 5 introduit d'abord l'état de l'art des différentes études comportementales dans la prise de décision chez les mammifères et le lien fort existant entre cette littérature et les algorithmes d'apprentissage par renforcement introduits dans le chapitre 2. Nous détaillons ensuite les modèles utilisés dans le cadre d'une tâche incertaine et non stationnaire, dans laquelle nous utilisons des algorithmes de bandits, qui, à notre connaissance, n'ont jamais été confrontés à des données réelles de ce genre, contrairement aux algorithmes par renforcement.

Nous poursuivons ensuite avec un modèle d'apprentissage séquentiel budgétisé appliqué à la navigation en robotique grâce à des algorithmes budgétisés mais traitant les données en hors ligne pour le moment. Dans ce cas on force le modèle avec un budget explicite en utilisant les méthodes d'apprentissage montrés dans les chapitre 3 et chapitre 4, et l'on teste dans quelle mesure la politique d'actions permettant au robot de se localiser peut être généralisée à d'autres environnements ayant des propriétés similaires (e.g. taille similaire, nombre similaire d'obstacles).

Nous reviendrons enfin dans la discussion de ce manuscrit, à la lumière des résultats obtenus, sur des pistes possibles pour intégrer la notion de budget dans des algorithmes en ligne, de façon utile pour les neurosciences et la robotique.

Chapitre 5

Modélisation de la prise de décision chez l'animal en environnement incertain et non-stationnaire

5.1 Bref État de l'art

Dans ce chapitre nous nous concentrerons sur les travaux en biologie de la prise de décision et de sa modélisation. Nous introduirons les notions d'apprentissage par essai-erreur et la notion de compromis entre exploration et exploitation dans la prise de décision chez l'animal. Nous pointerons aussi le lien entre l'apprentissage par renforcement et les modèles du comportement. Nous détaillerons ensuite une de nos contributions dans ce domaine, publiées dans Aklil et al. 2014.

5.1.1 Études du comportement

Les travaux fondateurs de l'étude du comportement des êtres vivants datent de la fin du $XIXe$ siècle - début XXe, avec de nombreuses expériences sur les mammifères. Ce type d'expérimentations ont pour but de montrer l'apprentissage de ces animaux dans

des cas expérimentaux contraints. Ainsi l'expérience de Thorndike 1911 met en lumière un aspect important de l'apprentissage, et celui qui nous intéressera le plus dans le suite de ce travail : la notion d'essai-erreur. Dans cette expérience, un chat ayant faim est placé dans une boîte dont la porte peut être ouverte en tirant une chaîne et en pressant un levier. De la nourriture est placée à la vue de l'animal mais à l'extérieur de la boîte. Au début l'animal exhibera des comportements simples comme griffer la porte où donner des coups de pattes. Après un certain nombre de comportements exploratoires l'animal arrivera, par chance, à tirer la chaîne et à appuyer sur le levier, et la porte s'ouvrira. Après un certain temps, l'animal est remis dans la boîte et au fur et à mesure on observe que l'animal mettra de moins en moins de temps à sortir de la cage.

Thorndike émettra l'hypothèse que la nourriture joue le rôle d'une récompense qui renforce le comportement qui a permis d'y accéder. Cette hypothèse est exprimée ainsi :

Parmi les différentes réponses à une même situation, celles qui permettent (...) de satisfaire les désirs de l'animal seront, toutes choses égales par ailleurs, connectées plus fermement avec cette situation, de sorte que, lorsqu'elle se présente de nouveau, ces réponses réapparaissent plus probablement ; celles qui (...) sont suivies par une gêne pour l'animal (...) verront leur connexion réduite, de sorte à ce qu'il soit moins probable qu'elles réapparaissent. Plus la satisfaction ou la gêne expérimentées seront importantes, plus le renforcement ou l'affaiblissement sera important. (Thorndike 1911, p244)

En parallèle, Pavlov (Pavlov 1927) montre qu'il est possible de faire apprendre, avec suffisamment de répétitions, l'association entre un stimulus et une récompense à un animal. Ses travaux sont connus sous le nom de conditionnement Pavlovien, et avec ceux de Thorndike, posent les bases des travaux sur le comportement.

Conditionnement classique et instrumental

Dans les travaux qui s'intéressent aux comportements des individus nous distinguons deux types de conditionnements : le conditionnement classique ou pavlovien, célèbre grâce à l'expérience du chien de Pavlov (Pavlov 1927) et le conditionnement instrumental ou opérant (Skinner 1938). Le premier représente l'apprentissage d'une réponse automatique et involontaire à un stimulus initialement neutre. Si nous reprenons l'exemple du chien cité précédemment qui est très représentatif d'un tel apprentissage : nous observons que le chien salive à la vue de nourriture (réponse réflexe à un stimulus inconditionnel) et que cette réponse peut être transférée à un stimulus initialement neutre (dans ce cas un son de cloche) après plusieurs étapes de conditionnement. Si nous présentons le stimulus neutre avant chaque présentation du stimulus inconditionnel, nous observons qu'à la fin de l'apprentissage le chien a une réponse réflexe à chaque présentation du stimulus neutre même en l'absence du stimulus inconditionel.

Le second type de conditionnement diffère par le fait que le sujet participe activement à sa tâche de conditionnement, par exemple en interagissant avec un levier pour obtenir

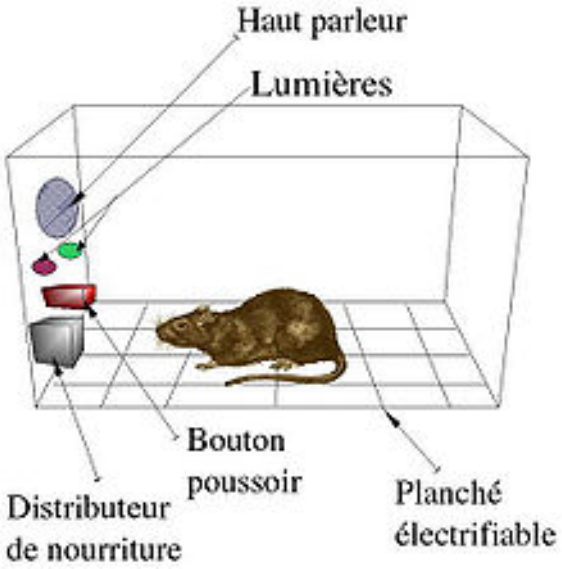

Figure 5.1 – *Une schématisation d'une boîte de Skinner (*Skinner *1938) utilisée pour l'étude du comportement instrumental. Le dispositif est utilisé pour montrer différent types de conditionnements. 1) Le renforcement positif : Le rat reçoit de la nourriture à l'appui sur le levier. L'observation est l'augmentation de la fréquence d'appui sur le levier. 2) Le renforcement négatif : Le sol de la cage est électrifié, à chaque appui du levier la décharge disparaît. L'observation sera la même que précédemment. 3) La punition positive : A l'appui du levier le rat reçoit une décharge électrique. On observe une diminution de la fréquence d'appui. 4) La punition négative : De la nourriture est disponible pour le rat et à l'appui du levier la nourriture disparaît. Même observation que pour 3)*

une récompense. La réponse du sujet est donc volontaire et non un simple réflexe. Dans l'expérience de Skinner (5.1), l'expérimentateur met un rat dans une boîte, le sujet est soumis à 2 types de stimuli : un levier qui lui sert de la nourriture lorsqu'il est appuyé et un plancher électrifié qui servira à punir l'animal. On peut de ce fait combiner différents cas de renforcements comportementaux : il peut recevoir un renforcement positif où il recevra la récompense au moment de l'appui du levier, un renforcement négatif où des décharges continues sont stoppées par l'appui du levier, une punition positive où il reçoit une décharge à l'appui du levier ou bien une punition négative où on retire la nourriture lors de l'appui du levier. Dans les deux premiers cas nous observons que la fréquence de l'action en question augmente alors qu'elle baissera dans les deux autres cas.

Hypothèses sur le comportement instrumental

A partir des observations et travaux décrits dans le paragraphe ci dessus, plusieurs travaux, principalement en navigation, ont mis en opposition les théories héritées de la psychologie où les réponses dans un cadre instrumental ne sont qu'un apprentissage d'associations entre stimulus et réponse motrice, et la vision de Tolman p.d. pour qui l'animal apprend une carte de son environnement héritée des travaux de Blodgett 1929.

Pour Tolman, l'observation de comportements VTE (*Vicarious Trial and Error*, ou essai-erreur simulé) est également une indication de l'utilisation d'une carte qui pour lui est un modèle des enchaînements entre lieux successifs visités et directions de déplacements associées, similaire aux modèles de transitions de l'apprentissage par renforcement. Le comportement VTE correspond à un animal qui hésite ; il est interprété comme l'indication que l'animal évalue les conséquences de plusieurs options. Cette hypothèse est confirmée par Adams et Dickinson 1981 qui montrent qu'un rat ayant appris à presser un levier, motivé par l'obtention de sucrose, stoppe dès lors que ce sucrose rend l'animal malade. Cependant, le degré auquel l'animal est sensible à la conséquence de son action dépend de son degré d'entraînement (Adams 1982), le surentraînement le rendant complètement insensible.

Cela conduit à l'hypothèse que les deux manières de créer le comportement existent en parallèle, et qu'il y a un transfert de contrôle entre elles, ce qui est confirmé par Dickinson (1985) d'un point de vue comportemental et renforcé par des études de lésions (Balleine et Dickinson 1998).

Caractérisation des comportements

Cette distinction de comportements se fait entre deux manières de prendre une décision. On dit que l'animal a un comportement *dirigé vers un but* si il remplit deux conditions : si l'action reflète un choix "informé" des conséquences de cette action, et si la conséquence de l'action est désirable à l'agent. Le comportement qui en résulte est flexible et s'adapte aux changements survenant dans l'environnement. Cette flexibilité de la prise de décision

aura un coût qui sera le temps imparti pour prendre cette décision.

A l'inverse le sujet est dit avoir un comportement *habituel* lorsqu'il prend des décisions indépendamment de ces conséquences. Ce qui veut dire que la réponse de l'animal se rapproche plus d'une réponse réflexe exécutée à cause d'un stimulus perceptuel que d'une décision choisie. Au fur et à mesure des expériences, le sujet apprend à associer des stimuli à des actions et à leurs conséquences. Ce qui a pour effet de ralentir drastiquement les changements d'actions lors de changement de l'environnement mais est rapide lors de la prise de décision.

De ces deux systèmes de prise de décision, découle une problématique de compromis entre exploration et exploitation. En effet, les décisions d'un comportement *habituel* sont rapidement prises mais peu flexibles alors que les décisions d'un comportement *dirigé vers un but* sont très flexibles mais lentement prises. Ce qui implique qu'à un point donné il est nécessaire de savoir si la décision à prendre doit être rapide même si incertaine où certaine mais lente.

Pour identifier ces comportements chez les sujets expérimentalement, il existe deux manières dans la littérature : la dévaluation des résultats et la dégradation de contingence. La première méthode consiste à dévaluer la récompense fournie lors de l'apprentissage en la rendant neutre ou répulsive lorsque le sujet arrive à satiété par exemple. Si le taux d'exécution de l'action faite jusqu'à maintenant baisse rapidement nous en déduisons que le sujet appliquait un comportement *dirigé vers un but* mais si le taux baisse lentement au cours de la dévaluation il est considéré que le sujet usait d'un comportement *habituel*. La seconde méthode consiste à modifier les conséquences des actions, par exemple en supprimant la délivrance de la récompense ou bien en la rendant non-conditionnée à l'action. Comme pour la méthode précédente si le sujet persiste et désapprend lentement son comportement est *habituel* et dans le cas où il adapte son comportement et désapprend rapidement son comportement est *dirigé vers un but*.

Ces résultats sont étendus aussi à l'humain. Valentin et al. 2007 montre que l'on retrouve l'effet de la dévaluation de la récompense dans le cortex orbito-frontal chez l'homme, tandis que Tricomi et al. 2009 parvient à montrer la même désensibilisation à la conséquence de l'action que chez les animaux.

Du point de vue substrat neuronal, des études ont montré que les comportements *dirigés vers un but* ont pour substrat neural le *striatum dorsomédial* (DMS) tandis que le comportement *habituel* le *striatum dorsolatéral* (DLS) qui sont toute deux des sous parties des ganglions de la base (voir Yin et Knowlton 2006 et Lienard 2013 pour une représentation détaillée des ganglions de la base). Par ailleurs, la partie ventromédiale du Cortex Préfrontal (vmPFC) semble refléter la représentation que l'animal aurait de la tâche, soit l'équivalent d'un modèle Hampton et al. 2006. De la même manière, on observe des transferts d'activités entre le DMS et le DLS (Yin et al. 2009, Thorn et al. 2010), ce qui peut être rapproché d'observations similaires dans des tâches spatiales (Pa-

ckard et McGaugh 1996).

Des résultats plus récents montrent également que s'il y a bien une évaluation des valeurs dans les deux parties du striatum, le cortex préfrontal joue également un rôle d'intégration des informations pour la décision (Bornstein et Daw 2013 ; Glascher et al. 2010 ; Wunderlich et al. 2012).

Connexion entre apprentissage par renforcement et comportement

Le développement de l'apprentissage par renforcement a été fortement influencé par les études du comportement animal. L'idée d'essai-erreur et de renforcement développées dans les études comportementales développées plus haut sont les point clés de l'apprentissage par renforcement décrites dans le chapitre 1. Par exemple les algorithmes de *TD learning* ont été proposés comme extension du modèle de Rescorla et al. (étendu à des mises à jours en un essai plutôt qu'entre les essais).

D'un point de vue neurobiologique, plusieurs études ont montré l'importance du rôle de la dopamine dans l'évaluation de la récompense dans les tâches de conditionnement (Ljungberg 1992). Aussi, ce neuromodulateur a un rôle tout aussi important lors du processus d'apprentissage (Montague et al. 2004, Wise 2004). Il est également montré qu'il existe un lien très fort entre la dopamine et l'erreur de prédiction de l'apprentissage par renforcement : Schultz et al. montrent une corrélation très forte entre l'activité phasique des neurones dopaminergiques et l'erreur de prédiction calculée dans les algorithmes d'AR direct (Schultz et al. 1997 ; Hollerman et Schultz 1998 ; Schultz 1998). En effet il est observé qu'une augmentation de la décharge des neurones dopaminergiques d'un animal lorsqu'il reçoit une récompense non prédite (ce qui correspond algorithmiquement à un retour supérieur à la valeur prédite). Lorsque l'association entre un stimulus donné et la récompense est faite, nous observons cette activité dopaminergique au moment du stimulus et non plus au moment de la récompense (la prédiction est équivalente au retour reçu, l'erreur est quasi-nulle). Enfin lorsque la prédiction est fausse, nous observons une inhibition de l'activité au moment où la récompense aurait du être obtenue (équivalent à un retour reçu inférieur à la valeur prédite). Cette hypothèse est discutée dans plusieurs récents travaux (Berridge 2007 ; Schultz 2013 ; Bellot et al. 2012 ; Kishida et al. 2015) où l'AR est le modèle utilisé pour expliquer les comportements observés.

Dans Doya 1999, les auteurs associent les différents types d'apprentissage à des zones du cerveau à partir des informations anatomiques et physiologiques connues. Ainsi, l'apprentissage non-supervisé est supposé être associé au cortex cérébral, tandis que le cervelet supporterait un apprentissage supervisé. L'apprentissage par renforcement est associé aux ganglions de la base (dont le principal noyau d'entrée est le striatum). Ces derniers sont supposés être impliqués dans l'apprentissage de comportements séquentiels (Graybiel 1995). L'activité des neurones dopaminergiques présentée précédemment ainsi que la performance des modèles basés sur l'AR à reproduire les comportements observés appuient cette hypothèse (Dominey et al. 1995 ; Berns et Sejnowski 1998).

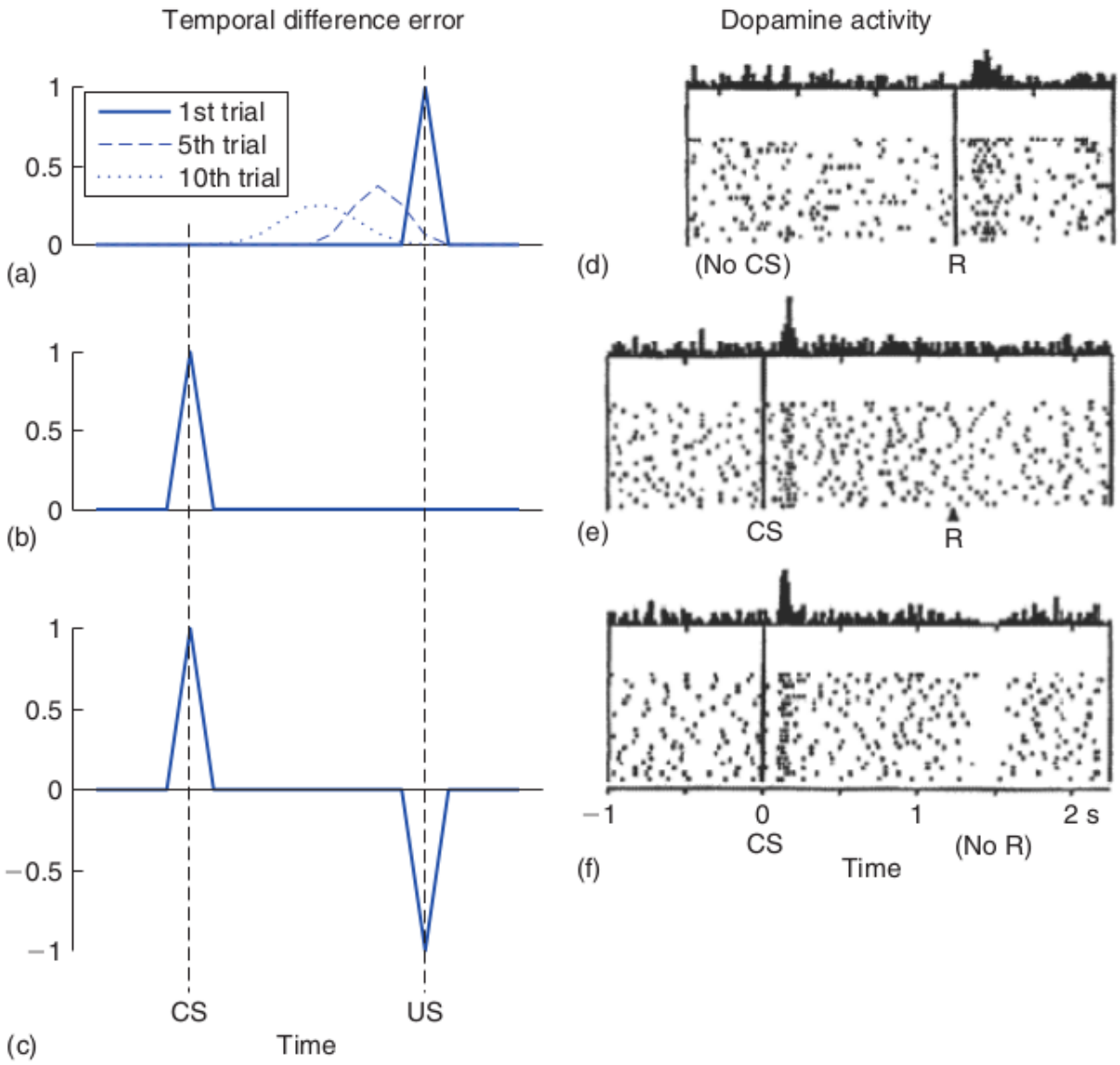

Figure 5.2 – *Equivalence entre apprentissage par renforcement et activité dopaminergique.* ***Droite :*** *Les valeurs de la Prédiction de l'erreur dans les algorithmes d'apprentissage par renforcement.* ***Gauche :*** *L'activité des neurones dopaminergiques tels que relevés dans (*Schultz *1998). La première ligne montre l'activité avant apprentissage ou la RPE et l'activité dopaminergique ont un pic au moment de la récompense. La ligne du milieu montre l'activité après apprentissage où le pic est transféré au moment de la présentation du stimulus. La dernière ligne montre l'activité après apprentissage et avec omission de la récompense au moment attendu.*

5.2 Modélisation du compromis exploration et exploitation en environnement non stationnaire

A partir de ce qui a été décrit plus haut, plusieurs travaux ont été menés afin d'expliquer les comportements d'animaux dans des environnements plus ou moins complexes en utilisant des algorithmes d'apprentissage par renforcement, qu'ils soient en apprentissage direct, indirect où en mettant en place des architectures qui combinent les deux approches (Daw et al. 2005, Chavarriaga et al. 2005, Dollé et al. 2010, Keramati et al. 2011, Caluwaerts et al. 2012, Lesaint et al. 2014, Viejo et al. 2015).

Ces tâches sont généralement des tâches stationnaires, ce qui veut dire que durant une suite d'essais, l'environnement récompense le sujet de la même manière, par exemple un seul levier donnant la récompense tout au long de la procédure d'apprentissage. Dans le cas d'un environnement non stationnaire la distribution des récompenses varie en fonction du temps. En reprenant l'exemple précédent, le levier distribuant la récompense change au cours du temps.

Dans le cas des travaux présentés dans cette section, il s'agira de modéliser les comportements d'animaux dans le cadre d'un environnement incertain et non stationnaire. Incertain car la distribution de la récompense se fera dépendamment d'une distribution de probabilités et non stationnaire car le levier dominant (ayant la plus grande probabilité de distribuer la récompense) change au cours du temps.

5.2.1 Protocole expérimental et résultats comportementaux

Les sujets étudiés sont des rats, évoluant dans une boîte de skinner avec trois leviers, chacun des leviers ayant une probabilité de distribuer une récompense (une boulette de nourriture). Le rat est d'abord pré-entraîné pour générer une motivation de la récompense de façon à ce qu'instinctivement il se dirige vers les leviers pour obtenir la récompense. On peut décrire le protocole expérimental comme suit (figure 5.3) :

- L'étude a été faite sur 24 rats qui doivent exécuter 24 sessions d'appui de levier.
- Chaque session est composée de 4 blocs et chaque bloc est composé de 24 à 48 essais dépendamment du type d'incertitude.
- Chaque bloc d'essai est caractérisé par un levier dominant et un type de risque
- Deux types d'incertitude sont mis en place :
  - Incertitude faible ou *low risk* (LR) donne une probabilité de 7/8 d'obtenir la récompense et 1/16 pour les deux autres
  - Incertitude forte ou *high risk* (HR) donne une probabilité de 3/4 d'obtenir la récompense et 1/8 pour les deux autres

Les expérimentateurs ont mesuré plusieurs critères de performances chez ces rats. Pour chaque type de risque, ils ont mesuré une performance globale qui est le nombre moyen de fois où le rat a appuyé sur le bon levier. Le second critère est considéré comme un critère

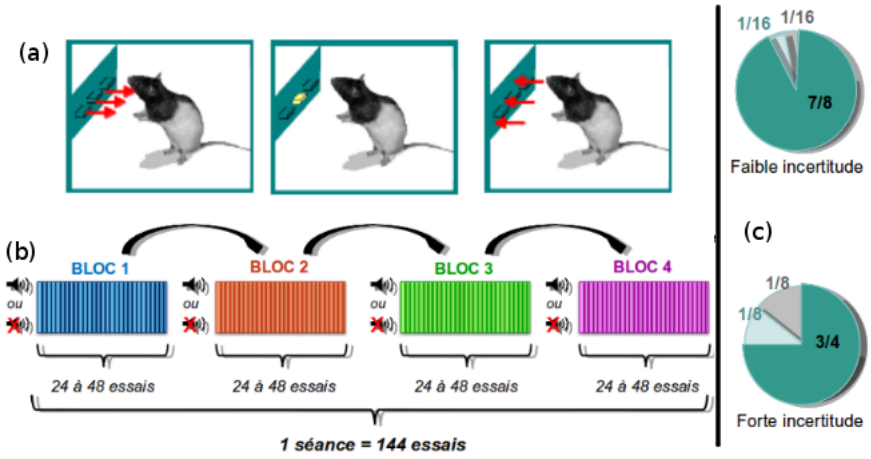

Figure 5.3 – *Protocole expérimental de tâche de conditionnement dans un environnement incertain et non stationnaire. a) Trois leviers sont présentés au rat, le rat choisit un levier puis le leviers sont rétractés. b) Agencement d'une séance : 4 blocs caractérisés par un type de risque et un levier dominant. c) Distribution de probabilité, en haut cas de faible incertitude (BR), en bas cas de forte incertitude (HR)*

d'exploration appelé *win-shift* qui mesure le nombre de fois où le rat a changé de levier malgré la réception d'une récompense. Le dernier critère et ce que l'on pourrait appeler un critère de correction dit *lose-shift* où le rat change de levier si aucune récompense n'est reçue (figure 5.4).

On observera que les performances globales (5.4) montrent que les rats apprennent finalement en moyenne à augmenter le nombre de fois où ils appuient sur le bon levier, en apprenant plus vite dans le cas où le risque est plus faible, ce qui confirme une intuition de départ. Le *win-shift* diminue ce qui montre que les rats en moyenne explorent l'environnement de moins en moins à mesure qu'ils améliorent leurs performances. Le *lose-shift* par contre ne montre aucune évolution qu'elle soit positive ou négative, les rats se corrigent en moyenne à 50/50.

5.2.2 Modélisation computationnelle

Nous tentons donc de reproduire ces résultats en utilisant des algorithmes de l'apprentissage par renforcement tels que le *Q-learning*. Le critère de performance considéré est la comparaison des résultats obtenus par les rats expérimentalement et celles retournés par les modèles grâces à des tests statistiques.

La tâche a été modélisée par un MDP simple à deux états (5.5) où l'agent commence à l'état *Stim* où le stimulus lui est présenté et passe à l'état *Rew* grâce à une actions choisie dans l'ensemble $A = \{leviergauche, leviermilieu, levierdroit\}$ où il obtient ou non une récompense conditionnellement à la distribution de probabilités associée à cet essai.

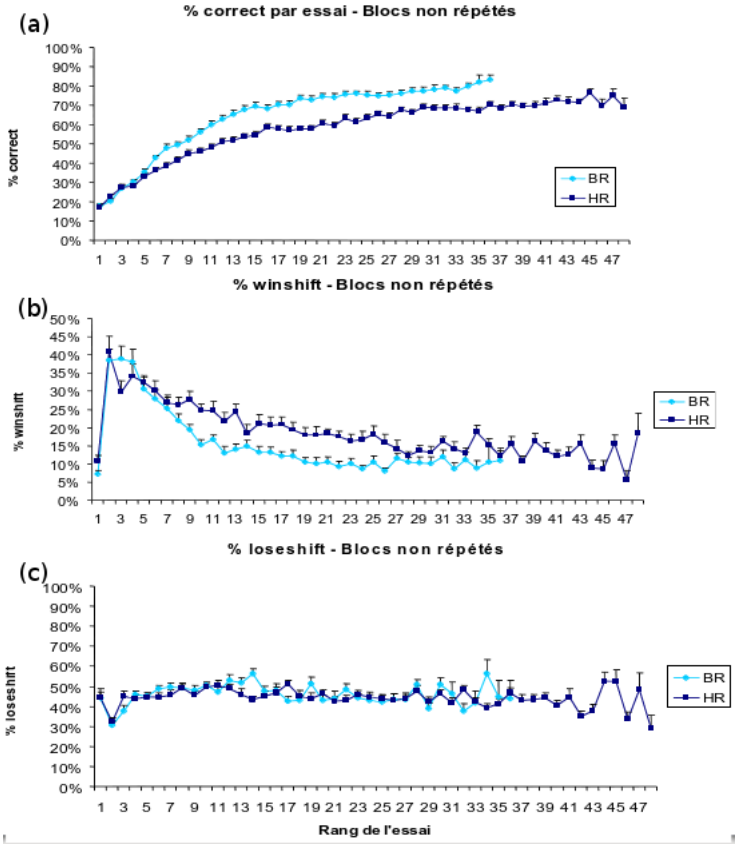

Figure 5.4 – *Résultats expérimentaux pour la tâche d'apprentissage instrumental non stationnaire : a) Performances du rat en pourcentage d'appui sur le levier dominant. b) Win-shift ou critère d'exploration. c) Lose-shift ou critère de correction.*

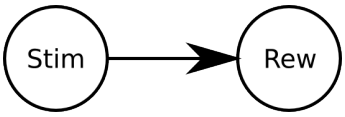

Figure 5.5 – *MDP à deux états : Stim étant l'état où les stimuli sont présentés et Rew l'état où l'agent obtient ou non sa récompense*

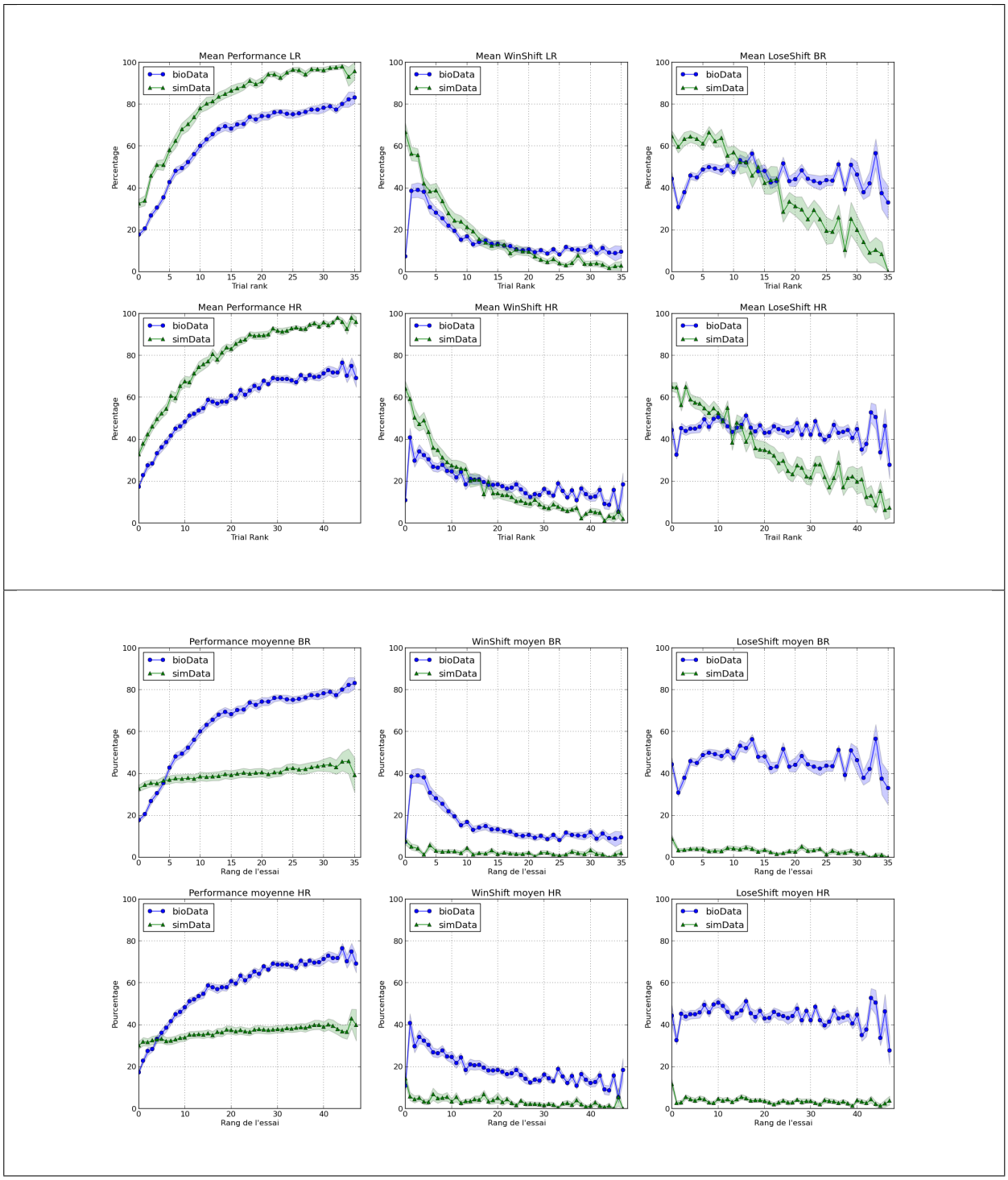

Figure 5.6 – *Résultats de l'AC avec (Bas) et sans (Haut) meta learning. Chaque ligne correspond à un type de risque (ligne du haut : Bas risque, ligne du bas : haut risque) et chaque colonne à un critère de performance (Gauche : Performance, centre : Win-shift, droite : lose-shift). Les lignes bleues représentent les performances des rats et les lignes en vert représentent les performances des agents*

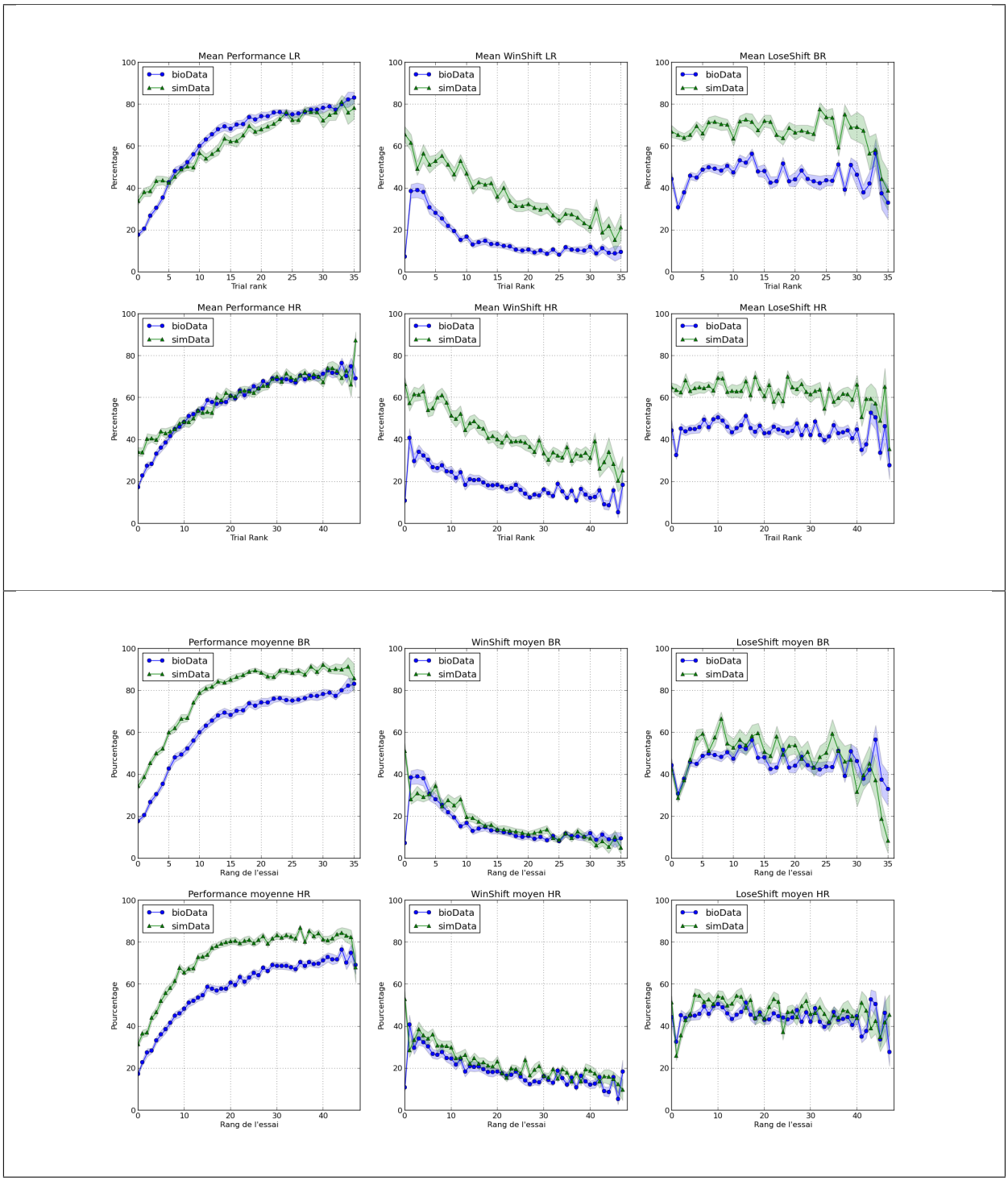

Figure 5.7 – *Résultats de l'QL. Mêmes conventions que dans la figure 5.6*

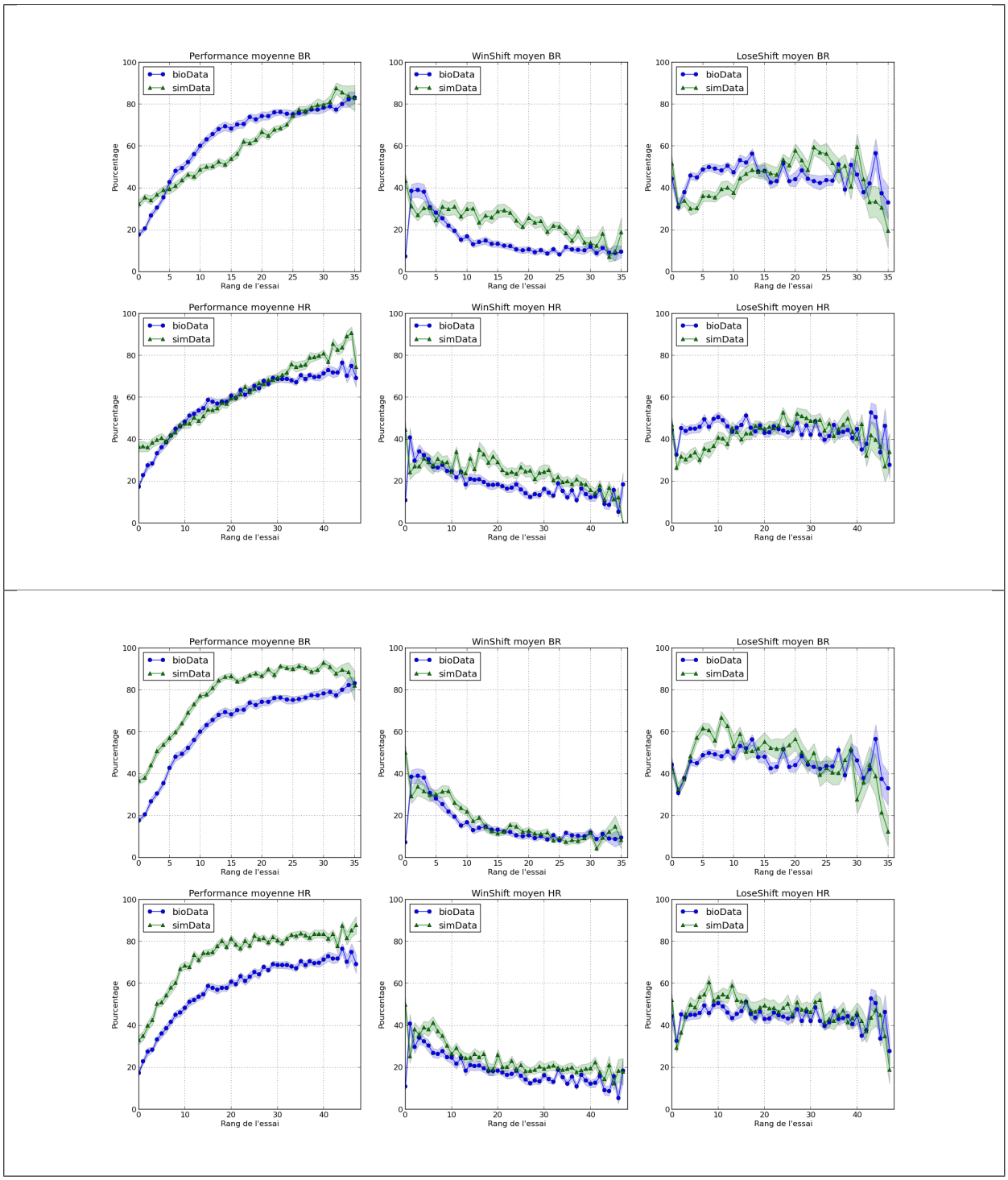

Figure 5.8 – *Résultats du SARSA. Mêmes conventions que dans la figure 5.6*

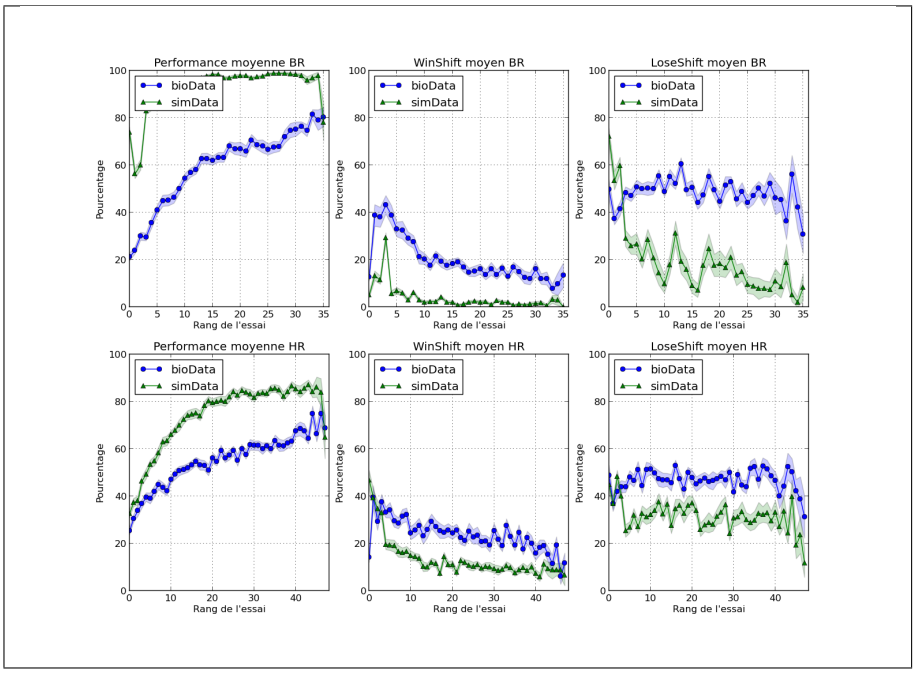

Figure 5.9 – *Résultats du D-UCB. Haut : Performances en Bas risque, Bas : Performances en haut risque. Colonne de gauche : Performance moyenne, colonne du milieu : Win shift, colonne de droite : Lose shift.*

Nous avons modélisé l'apprentissage en utilisant différents algorithmes. Nous avons comparé les performances obtenues des trois principaux algorithmes d'apprentissage par renforcement (*Actor-Critic, Q-learning et SARSA*). Nous avons aussi exploré une variante qui permet de gérer le compromis entre exploration et exploitation en paramétrant la fonction de prise de décision avec une estimation continue de la récompense. Nous avons testé deux fonctions de prise de décision : une fonction *SoftMax* (équation 2.18) et une fonction ϵ-*greedy* (équation 2.17) présentées dans le chapitre 2. Alternativement au fait d'utilité des fonctions de décision à taux d'exploration fixe, nous introduisons également un processus dit de *meta-learning* permettant de réguler dynamiquement des variables β et ϵ qui adapte le taux d'exploration dépendamment de l'intensité de la récompense à court terme et à long terme. Cette modification est formalisée comme tel :

$$\beta_t = \beta_{t-1} + \eta * (\hat{r_{x:t}} - \hat{r_{1:t}}) \tag{5.1}$$

$$\epsilon_t = \epsilon_{t-1} - \eta * (\hat{r_{x:t}} - \hat{r_{1:t}}) \tag{5.2}$$

où $\hat{r_{x:t}}$ est la moyenne de la récompense à court terme de fenêtre fixe (de taille 50 dans les modélisations qui suivent) et $\hat{r_{1:t}}$ est la moyenne de la récompense à long terme qui est accumulée depuis le début de la tâche. Les β et ϵ étant les variables gérant l'exploration de l'agent (plus le β est grand plus l'agent exploite la meilleure action dans le cas du *SoftMax* et inversement dans le cas de ϵ, plus la valeur est grande plus l'agent explore),

cette équation permet de gérer la balance entre l'exploration et l'exploitation.

En complément de cela, nous avons confronté les algorithmes UCB décrits dans le chapitre 2 aux données réelles pour observer la capacité d'adaptation de ces algorithmes à, d'une part, la non stationnarité de la tâche et d'autre part, à refléter des comportements observés chez des agents réels.

Pour plus de lisibilité nous résumons les modèles testés pour la modélisation de la tâche décrite :

— Actor Critic avec/sans *méta learning*
— Q-learning avec/sans *méta learning*
— SARSA avec/sans *méta learning*
— UCB-1
— D-UCB
— SW-UCB

5.2.3 Résultats de la modélisation

La comparaison entre les données comportementales des rats et les résultats obtenus par les agents est faite grâce à des tests statistiques : le test de Wilcoxon (Wilcoxon 1945) qui permettra de tester l'hypothèse que la distribution des performances obtenues par les rats et celles obtenues par les agents ont la même médiane, et le test de Kolmogorov-Smirnov qui testera si la distribution obtenue par les agents et celle obtenue par les rats s'éloignent significativement l'une de l'autre. Dans les graphiques qui suivent nous avons séparé les résultats dans le cas HR et dans le cas BR, nous comparons ensuite les critères cités plus haut (Performance, *win shift* et *lose shift*) moyennés par bloc sur les individus entre celles récupérées sur les rats et celles retournées pas les agents simulés.

Nous avons sélectionné les paramètres de chaque algorithme grâce à un *grid-search* qui seront résumés dans le tableau (5.1). Les intervalles explorés pour chaque paramètre sont :

— TD-RL : $\alpha \in [0.1, 2]$ ($pas = 0.05$)
— SoftMax : $\beta \in [1, 10]$ ($pas = 0.5$)
— ϵ-greedy : $\epsilon \in [0.1, 0.9]$ ($pas = 0.1$)
— Meta-learning : $\eta \in [0.01, 0.05$ ($pas = 0.01$)
— D-UCB : $\gamma, \epsilon \in [0.1, 1]$ ($pas = 0.05$)
— UCB-SW : ϵ même que D-UCB, $\tau \in [100, 250]$ ($pas = 50$)

On observe que les algorithmes de bandit manchot sont extrêmement performants dans ce type de tâche, ce qui est assez facilement expliqué par le fait qu'ils sont spécifiquement développé dans le cadre de l'apprentissage en environnement non stationnaire, mais ne reflètent clairement pas les tendances d'apprentissage observées chez les rats. En effet, ces algorithmes ont eu la capacité à trouver en moyenne les meilleures actions après très peu d'essais, ce qui n'est pas le cas des rats, qui sont en moyenne lents à trouver les bons leviers (5.9).

Algos	Parameters					
	α	β	γ	ϵ	τ	η
SoftMax AC	0.09	9.5	0.0	X	X	X
SoftMax QL	0.05	7.0	0.0	X	X	X
SoftMax Sarsa	0.05	8.0	0.0	X	X	X
ϵ-greedy AC	0.7	X	0.0	0.4	X	X
ϵ-greedy QL	0.9	X	0.0	0.2	X	X
ϵ-greedy Sarsa	0.7	X	0.0	0.2	X	X
Meta-learning AC	0.9	4.0	0.0	X	X	0.03
Meta-learning QL	0.3	4.0	0.0	X	X	0.03
Meta-learning Sarsa	0.3	4.0	0.0	X	X	0.03
D-UCB	X	X	0.1	0.5	X	X
SW-UCB	X	X	X	0.7	160	X
ϵ-greedy Meta-Learning AC	0.4	X	0.0	0.6	X	0.04
ϵ-greedy Meta-Learning QL	0.4	X	0.0	0.4	X	0.02
ϵ-greedy Meta-Learning Sarsa	0.3	X	0.0	0.4	X	0.02

Table 5.1 – *Liste des paramètres retenus pour chaque algorithme testé suite au grid-search.*

Algoritmes	Performances	*Win-shift*	*Lose-shift*	Score cumulé
SoftMax AC	98	107	29	234
SoftMax QL	86	86	104	276
SoftMax Sarsa	91	65	107	263
ϵ-greedy AC	12	75	65	152
ϵ-greedy QL	23	129	123	275
ϵ-greedy Sarsa	52	130	119	301
SoftMax Meta-learning AC	95	90	46	231
SoftMax Meta-learning QL	66	108	74	248
SoftMax Meta-learning Sarsa	5	131	**138**	274
ϵ-greedy Meta-Learning AC	9	24	122	155
ϵ-greedy Meta-learning QL	78	133	110	321
ϵ-greedy Meta-learning Sarsa	**143**	**147**	136	**426**
D-UCB	20	116	126	262
SW-UCB	34	7	2	43
Kalman QL	66	108	74	248

Table 5.2 – *Tableau des scores cumulés avec des tests statistiques de Wilcoxon et de Kolmogorov-Smirnov pour chaque algorithme testé. Chaque score cumulé est la somme des tests statistiques validés. Si tout les tests sont validés le score maximal est de 432.*

A chaque test statistique validé, nous incrémentons le score de 1. Le score cumulé est le score obtenu pour l'ensemble des critères de comparaison. Les résultats avec les scores de chaque algorithme sont résumés dans le tableau 5.2. Dans le cas des algorithmes d'apprentissage par renforcement, nous observons que le AC, QL et SARSA sans le processus de régulation de l'exploration par *méta learning* sont très peu fiable à reproduire les performances observées. Cette difficulté est due au fait que ces algorithmes ne captent pas la non stationnarité de la tâche, ce qui provoque des changements d'actions très lents, un changement que les agents ne sont capables de combler qu'après une longue séquence d'essai-erreur. En revanche, l'utilisation de la régulation dynamique de β et ϵ mène à une observation intéressante qui serait que l'exploration serait régulée par l'estimation d'une récompense moyenne tout au long de la tâche en plus d'une estimation plus locale qui concernerait que les derniers essais. Effectivement les performances observées grâce à la régulation dynamique de β ou d'ϵ sont bien supérieures à celles que l'on a eu sans cette régulation. Or l'analyse du modèle simulé avec les paramètres obtenus par optimisation (i.e. les paramètres qui permettent de maximiser le « cumulative score » (tableau 5.2) qui donne la proximité des courbes du modèle avec celles du rat) montre que le modèle optimisé à tendance à réduire β (donc à explorer) en début de bloc où la performance chute car le levier récompensé vient de changer, et à augmenter β (donc à exploiter) en fin de bloc où la performance augmente à mesure que le modèle choisit de plus en plus souvent le bon levier.

Ces analyses nous ont permis d'avoir une première confirmation que le comportement des rats dans cette tâche s'explique le mieux par un modèle qui régule dynamiquement l'exploration, de façon à explorer davantage quand l'environnement change (i.e. en début de bloc) et à exploiter davantage quand la tâche est stationnaire (en fin de bloc). De plus, ces analyses suggèrent une autre piste intéressante qui mériterait d'être creusée dans des travaux futurs : parmi les modèles avec *meta-learning*, ceux qui expliquent le mieux le comportement du rat sont ceux faisant une régulation dynamique d'ϵ plutôt que de β. Or la différence entre les deux est que pour les essais d'exploration (i.e. où le modèle ne choisit pas le levier qu'il pense être le meilleur), ϵ-*greedy* choisit parmi tous les leviers au hasard alors que le *SoftMax* a plus de chance de choisir le levier que le modèle considère comme le deuxième meilleur. Cela pourrait suggérer que les rats ont un comportement plus proche du premier que du deuxième. Or il s'agit d'un sujet sensible en neurosciences actuellement, car les travaux ont jusqu'à peu systématiquement étudié le cas particulier de décision entre 2 alternatives et ne se sont que récemment penchés sur ce que change le fait d'avoir au moins 3 alternatives (Louie et al. 2013). Enfin, on peut noter une limite au travail présenté ici : afin d'avoir une première idée de la régulation de l'exploration chez l'animal, nous avons présenté une comparaison des modèles à des courbes moyennées chez un ensemble de rats. Or il nous est apparu clairement qu'il y avait de fortes différences inter-individuelles, mais que ces différences étaient difficiles à évaluer dans la mesure où chaque rat prenait un nombre différent d'essais par bloc (entre 24 et 48) pour atteindre un critère de performance suffisant pour considérer qu'il a bien appris.

Dans le travail de Cinotti et al. 2017 un changement a été apporté au protocole ex-

périmental, où les essais sont de même durée (que ce soit en HR ou en BR) en plus d'une expérimentation pharmacologique dérégulant la capacité de prise de décision des animaux par l'injection d'un antagoniste de la dopamine, de façon à tester si cette manipulation altère principalement le taux d'apprentissage ou le taux d'exploration. Enfin, nous avons également proposé un changement du paradigme de la régulation de la variable d'exploration, en utilisant l'erreur de prédiction plutôt que la récompense pour moduler le taux d'exploration de la fonction *Softmax* décrite dans l'équation 2.18. Toutes ces modifications ne seront pas décrites plus avant ici, car ils ne sont pas centrales pour le sujet de cette thèse (un article concernant ces modifications est en préparation Cinotti et al. In Preparation). Néanmoins, les résultats présentés dans cette partie constituent une manière d'illustrer comment on pourrait s'inspirer de l'adaptation comportementale chez l'animal pour faire de l'apprentissage par renforcement en ligne et régulé dynamiquement.

5.3 Conclusion

Dans ce chapitre nous avons introduit l'état de l'art de l'étude du comportement chez les mammifères d'un point de vue neuroscience computationnelle. Nous avons aussi introduit le lien fort existant entre l'apprentissage par renforcement et les résultats observés dans les études comportementales et de pourquoi ce type de modèles est un des plus utilisés pour la modélisation comportementale.

Nous avons aussi introduit la problématique de la gestion du compromis entre exploration et exploitation grâce à une étude de cas réel dans un environnement incertain et non stationnaire où nous avons pu reproduire les comportements observés chez les rats avec des algorithmes d'AR en augmentant le processus de prise de décision par une modulation de la variable d'exploration. Cette analyse indique que les sujets ont une capacité d'adaptation à la non stationnarité de la tâche grâce à une estimation de la récompense avec des fenêtres temporelles.

Chapitre 6

Selection séquentielle d'actions pour la robotique

6.1 Introduction

Dans ce chapitre, nous proposons une architecture pour une classification supervisée séquentielle appliquée à une tâche de localisation en robotique. Cette architecture est pensée de façon à ce que l'agent ou le robot apprenne une politique opportune à l'exploration de l'environnement et en même temps à mener une tâche de localisation. L'architecture est donc composée de "modules" séquentiels où à chaque pas de temps le modèle choisit une action et prédit les effets de cette action.

Tout d'abord nous introduirons les traitements classiques en robotique pour la navigation et la localisation. Nous détaillerons ensuite l'architecture du modèle et les résultats obtenus dans la tâche de localisation dans un cas de simulation et un cas d'application sur données réelles. Le travail décrit dans ce chapitre a été publié dans l'article Aklil et al. 2017.

6.2 La localisation en robotique

La localisation dans l'espace est une des problématiques des plus difficiles en robotique. Le problème consiste principalement de prendre des décisions spatiale dans un environnement en utilisant les différentes ressources à disposition dans le but de se localiser. Les tâches de localisation sont complexes à cause de l'interdépendance entre les étapes de localisation et de création de carte (*mapping*) : dans le but de se localiser, les robots doivent reconnaître les différents stimuli caractérisant un endroit dans la carte et doivent construire une carte se basant sur ces stimuli en se localisant en fonction de ces stimuli (Filliat et al. 2012).

Généralement, on distinguera trois types de solutions pour résoudre des tâches de localisation et de navigation :

- La navigation sans carte (*mapless navigation*) : Dans ce cas le robot apprend en utilisant uniquement ses senseurs à se localiser ou à résoudre une tâche de navigation sans avoir d'information extérieure qui serait un oracle (voir par exempleChen et al. 2011)
- La navigation avec carte (*mapbased navigation*) : ici le robot utilise une connaissance explicite de son environnement qui est externe à sa perception (une carte ou une donnée GPS, par exemple) comme dans les travaux de (voir Filliat et Meyer 2003 pour une revue de ce type de solutions)
- Le SLAM (*Simultaneous Localization And Mapping*) qui est une procédure médiane, où le robot utilise ses senseurs pour construire une carte qui sera utilisée à posteriori dans d'autres tâches (Thrun et Leonard 2008, Moutarlier et Chatila 1990, Smith et Cheeseman 1986, Montemerlo et al. 2002)

Le point commun entre ces trois approches réside dans la possibilité d'une utilisation infinie des données et des ressources. En effet, ces algorithmes sont très précis et ont des performances extrêmement satisfaisantes quant à une localisation ou une navigation métrique, mais ne prennent en aucun cas compte le coût de l'utilisation d'une donnée plutôt qu'une autre ou bien le temps de prise de décision pour des robots. De plus, tout en utilisant les ressources physiques (senseurs), les algorithmes du SLAM ne prennent pas en compte la sélection de l'action et ne peuvent dire la manière dont l'acquisition des données peut entrer en ligne de compte dans une politique globale devant maximiser une fonction de coût. Cette approche peut avoir des inconvénients dans des cas d'applications réelles. Par exemple il serait préférable d'utiliser une seule et unique donnée dans certains cas précis plutôt que de faire appel à une fusion de tous les senseurs. Ou bien il serait intéressant d'effectuer plusieurs actions d'exploration avant d'avoir à utiliser des données pour pouvoir accomplir sa tâche. De cette manière le robot a la possibilité d'accomplir la tâche avec une plus grande rapidité. Enfin il serait effectivement avantageux de limiter le nombre d'actions allouées au robot, du fait qu'il sera plus rapide à prendre des décisions,

qui par exemple en cas de danger se révélera précieux.

Cette façon de modéliser la sélection de l'action est au cœur du domaine de l'*active sensing* (Mihaylova et al. 2003) qui prend en compte l'acquisition de données comme une action accessible par le robot, mais contrairement à ce que l'on propose, ces modèles proposent des manières d'apprendre les politiques une fois les représentations des tâches apprises. Ici nous nous sommes intéressés à faire simultanément l'apprentissage des représentations pour la tâche et des actions de façon à ce que ce travail puisse être plus directement applicable à la navigation robotique.

Dans la suite de ce chapitre, nous proposons un modèle d'apprentissage des actions capable de mener en parallèle l'apprentissage des représentations et de la politique basé sur le modèle développé dans l'article Denoyer et Gallinari 2014 introduit dans le chapitre 4 et qui prend en compte un budget en limitant le nombre d'actions autorisées au robot.

6.3 Modèle de sélection séquentielle d'actions

Le modèle proposé a pour but de résoudre une tâche de localisation, où à chaque pas de temps le modèle apprend séquentiellement à sélectionner une action parmi un ensemble prédéfini d'actions (par exemple, pouvoir choisir entre un mouvement ou une acquisition de donnée). Le modèle est restreint par un budget B limitant le nombre d'actions autorisées pour compléter sa tâche. Le but final est de faire en sorte que l'agent apprenne à alterner entre des actions d'exploration et des actions qui lui fournissent de l'information sur son environnement. Le modèle est basé sur le paradigme du *Deep Reinforcement Learning* décrit dans le chapitre 3. La différence première est que la politique ne sera pas dirigée par une récompense explicite mais par une erreur de classification notée Δ.

6.3.1 Description

Nous noterons χ l'ensemble des données possibles, dans ce cas là cela représente l'ensemble des positions de départ du robot dans l'environnement. Au début de chaque épisode à $t = 1$, l'agent est dans une position inconnue x (décrite par un triplet abscisse, ordonnée et orientation) puis à chaque pas de temps t choisit une action a_t parmi l'ensemble des actions $\mathcal{A}$ pour, à la fin du processus, prédire sa position $\hat{y}$. La qualité de la prédiction sera mesurée par une erreur de prédiction $\Delta(y, \hat{y}) \in \mathbb{R}^+$ (y étant la position réelle et $\hat{y}$ la position prédite par le modèle).

Certaines actions a_t donnent lieu à un retour que l'on appelle observation $o_t \in \mathbb{R}^{n_{a_t}}$, tel que n_{a_t} est la taille de l'observation correspondant à l'action a_t. Ici aussi réside la différence avec les modèles classiques de DRL : l'agent généralement n'exécute que des actions en réponse à une entrée du modèle qui contient des données qui sont distribuées

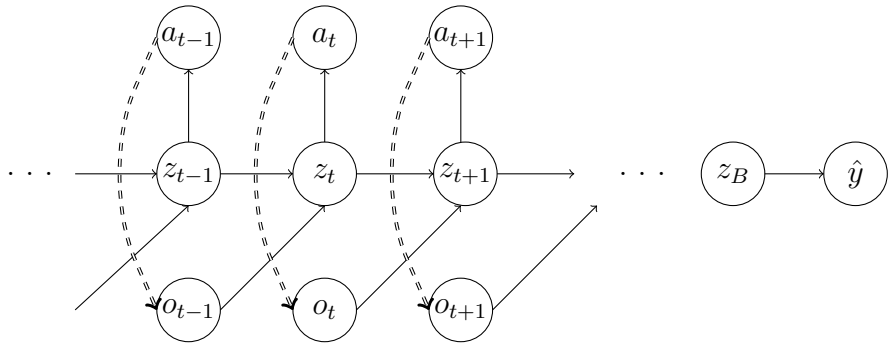

Figure 6.1 – *Architecture du modèle séquentiel. z_t est un état latent du modèle. Depuis z_t une action a_t est tirée. L'action choisie retourne une observation o_t utilisée pour le calcul de z_{t+1} qui sera une agrégation de z_t et o_t. A la fin du budget fixé B et après avoir calculé z_B le modèle prédit le $\hat{y}$ équivalent.*

automatiquement et de même dimension.

La valeur de o_t est décrite par la distribution inconnue $P(o_t|a_t, a_{t-1}, ..., a_1, x)$ qui dépend de l'environnement. Nous dénotons aussi $\pi(a_t|a_{t-1}, o_{t-1}, ..., a_1, o_1)$ la politique du robot, plus précisément la probabilité de choisir l'action a_t dépendamment de la suite d'observations $o_{t-1}, ...o_1$ et de la séquence d'actions précédentes $a_{t-1}, ..., a_1$. Le processus se termine par la prédiction d'un $\hat{y}$ tel que $\hat{y} = f(a_t, o_t, ..., a_1, o_1, x)$.

L'inférence peut être décrite dans l'algorithme ci-dessous et peut aussi être décrite selon l'architecture 6.1 :

Algorithm 3 Inference

1 : **for** $t \leftarrow 1, T$ **do**
2 : $\quad a_t \sim \pi(a|a_{t-1}, o_{t-1}, ..., a_1, o_1)$
3 : $\quad$ Appliquer a_t sur l'environnement
4 : $\quad$ Récupérer l'observation o_t
5 : **end for**
6 : $\hat{y} \leftarrow f(a_T, o_T, ..., a1, o_1, x)$ ▷ Prédiction

6.3.2 Apprentissage

Nous noterons $(x_1, ..., x_m)$ l'ensemble des positions d'apprentissage, les m positions qui seront utilisées durant l'apprentissage. Nous notons B comme étant le budget maximum autorisé au modèle. Dans cette application, on considère le budget fixe. Le but étant de trouver la politique π^* et la fonction de prédiction f^* qui minimisent l'erreur de prédiction :

$$\gamma^*, \theta^* = argmin_{\gamma,\theta} L(\gamma, \theta) \tag{6.1}$$

où γ sont les paramètres de la politique π et θ les paramètres de la fonction de prédiction f. L'erreur de prédiction L est définie comme :

$$L(\gamma, \theta) = \mathbb{E}_\pi \left[\Delta(f(a_B, o_B, ..., a_1, o_1, x), y)\right] \tag{6.2}$$

où les trajectoires $a_B, o_B, ..., a_1, o_1$ sont tirées selon la politique π. La minimisation de cet objectif sera faite en utilisant un algorithme de *policy gradient* qui a été décrit dans le chapitre 3.

Pour faciliter la compréhension des équations suivantes, nous noterons $T = a_B, o_B, ..., a_1, o_1$ pour la trajectoire englobant les actions et les observations obtenues. De cette façon nous pouvons réécrire le critère cité ci-dessus de cette manière :

$$L(\gamma, \theta) = \int (P(T|x)\Delta(f(x, T), y))P(x, y)dTdxdy \tag{6.3}$$

L'apprentissage se fait par descente de gradient, où en intégrant sur les trajectoires on obtient :

$$\nabla_{\gamma,\theta} L(\gamma, \theta) = \int \nabla_{\gamma,\theta}(P(T|x)\Delta(f(x, T), y))P(x, y)dTdxdy \tag{6.4}$$

que l'on pourra simplifier de cette façon :

$$\begin{aligned}
\nabla_{\gamma,\theta} L(\gamma, \theta) &= \int \nabla_{\gamma,\theta}(P(T|x))\Delta(f(x, T), y))P(x, y)dTdxdy && (6.5)\\
&+ \int P(T|x)\nabla_{\gamma,\theta}\Delta(f(x, T), y))P(x, y)dTdxdy && (6.6)\\
&= \int \frac{P(T|x)}{P(T|x)}\nabla_{\gamma,\theta}(P(T|x))\Delta(f(x, T), y))P(x, y)dTdxdy && (6.7)\\
&+ \int P(T|x)\nabla_{\gamma,\theta}\Delta(f(x, T), y))P(x, y)dTdxdy && (6.8)\\
&= \int P(T|x)\nabla_{\gamma,\theta}(logP(T|x))\Delta(f(x, T), y))P(x, y)dTdxdy && (6.9)\\
&+ \int P(T|x)\nabla_{\gamma,\theta}\Delta(f(x, T), y))P(x, y)dTdxdy && (6.10)
\end{aligned}$$

Enfin, en estimant ce gradient grâce aux méthodes de Monte Carlo sur l'ensemble des positions d'apprentissage en posant M comme le nombre total des trajectoires, on obtient :

$$\nabla_{\gamma,\theta} L(\gamma, \theta) \approx \frac{1}{m}\sum_{i=1}^{m}\left[\frac{1}{M}\sum_{k=1}^{M}\nabla_{\gamma,\theta}(\log P(T|x_i))\Delta(f(x_i, T), y_i) + \nabla_{\gamma,\theta}\Delta(f(x_i, T), y_i)\right] \tag{6.11}$$

Intuitivement on peut interpréter ce gradient comme étant la somme de deux gradients :

— Le premier terme est le gradient équivalent au *policy gradient* qui a pour but de pénaliser les trajectoires avec une grande erreur de prédiction - et donc à encourager les trajectoires avec de petites erreurs de prédictions. Par exemple, lorsque l'erreur est de 0 le gradient sera nul et donc la trajectoire sera exploitée de nouveau.
— Le second terme est le gradient calculé sur la prédiction résultante de la trajectoire utilisée et revient à faire une mise à jour en descente de gradient comme pour un réseau de neurones classique.

A noter que le terme $\nabla_{\gamma,\theta}(\log P(T|x_i))$ est :

$$\nabla_{\gamma,\theta} \log P(T|x_i) = \nabla_{\gamma,\theta} \sum_{t=1}^{B} \log P(a_t|a_{t-1}, o_{t-1}, ..., a_1, o_1) \tag{6.12}$$

6.3.3 Agrégation de l'information

Avec les notations définies ci dessous, il nous faut modéliser les deux fonctions $\pi(a_t|a_{t-1}, o_{t-1}, ..., a_1, o_1)$ et $f(a_t, o_t, ..., a_1, o_1, x)$. L'information étant obtenue séquentiellement il faut trouver une manière d'agréger cette information au long du processus et pouvoir garder en mémoire les informations données par une trajectoire T. Pour ce faire, nous modélisons des réseaux neuronaux qui feront en sorte d'agréger dans l'état latent noté z_t, la valeur de z_{t-1} et l'observation obtenue o_{t-1}. Plus précisément nous écrirons :

$$z_t = h_{a_{t-1}}(z_{t-1}, o_{t-1}) \tag{6.13}$$

La fonction h_a peut être modélisée de différentes manières, soit par des cellules indépendante, des réseaux récurrents ou des réseaux récurrents avec mémoire tels que les GRU et les LSTM (voir chapitre 3). Pour notre travail nous avons utilisé des réseaux neuronaux indépendants, des réseaux récurrents classiques et des réseaux récurrents avec mémoire GRU afin de pouvoir comparer leur efficacité dans notre tâche de localisation robotique.

Dans le cas d'un réseau récurrent classique la cellule d'agrégation est décrite par

$$h_a(z_t, o_t) = tanh(W_z z_t + W_a o_t) \tag{6.14}$$

La fonction h_a sera réécrite h_a^t dans le cas où le réseau n'est pas récurrent. Dans le cas récurrent on fait passer l'information dans des réseaux ayant des poids en commun, dans l'autre cas les réseaux sont totalement indépendants.

En ayant noté cela nous pouvons réécrire l'algorithme d'inférence comme montré dans l'algorithme 4.

6.4 Expérimentation

Le modèle a été testé sur des données simulées dites "jouet" (car leur simplicité permet de faire dans un premier temps une preuve de concept), puis sur des données réelles extraites d'un robot naviguant dans un environnement.

Algorithm 4 Inférence

for $t \leftarrow 1, B$ **do**
  $p = g(a/z_t)$ ▷ Calcul de la distribution de probabilité
  Appliquer action a_t
  Acquérir l'observation o_t
  $z_{t+1} \leftarrow h_{a_t}(z_t, o_t)$ ▷ Calcul du vecteur latent
end for
$\hat{y} \leftarrow f(z_B)$ ▷ Prédiction

6.4.1 Données simulées

Protocole expérimental

Un agent est simulé dans un environnement en forme de grille en 2D de taille fixe (posons de taille $h * w$ px). Chaque position est soit vide soit remplie avec une couleur symbolisant un obstacle. L'agent a la possibilité de faire un mouvement de rotation (tourner à gauche ou à droite d'un angle de 45^o) ou de relever une donnée (image ou laser). Les données seront donc un vecteur en 1D, dans le cas des images ce sera la description RGB de ce qui se trouve en face du robot, le laser sera le vecteur descripteur de la distance avec de potentiels obstacles. La tâche étant modélisée en un problème de classification, le labyrinthe est divisé en une grille de classes ou de cellules (qui seront considérées comme nos classes y). Le but de l'agent est d'arriver à retrouver dans quelle cellule il se trouve en enchaînant et en alternant les actions de mouvement et les actions d'acquisition de données. Nous donnons un exemple d'un tel labyrinthe dans la figure 6.2.

Nous avons utilisé certains paramètres fixes :
- Taille du labyrinthe : 50x50 px
- Nombre de classes : 16 (grille de 4x4 cellules)
- Angle de rotation du robot : $\pi/4$
- Taille de l'ensemble d'entraînement : 5000
- Taille de l'ensemble de test et de validation : 2500
- Les angles de départ à chaque cordonnée est posé : $\{0, \pi/2, \pi, 3\pi/2\}$

Pour les autres paramètres nous avons conduit un *grid search* dans des ensembles limités et comparé les résultats pour récupérer le meilleur set de paramètres. Les paramètres non fixés sont :
- Le *learning rate* pour la classification et la politique : $\{0.1, 0.01, 0.001, 0.0001\}$
- La taille de l'espace latent z_t : $\{10, 50, 100\}$
- La résolution (dénombrant le nombre de caractéristiques à récupérer dans le périmètre du senseur) des capteurs : $\{10, 30, 50\}$.

Nous comparons trois types d'acquisitions de données :
- La base de comparaison correspond à une tâche de classification simple où l'on récupère les images (selon le budget donné) en tournant l'agent sur lui même dans une seule direction. Nous donnons la concaténation des images comme donnée d'entrée au classifieur.

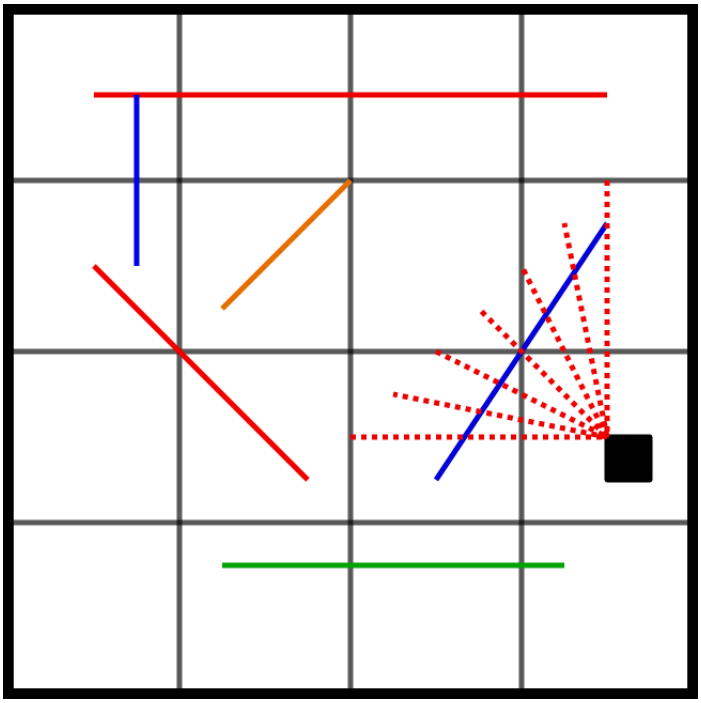

Figure 6.2 – *Exemple d'un labyrinthe simulé. Le carré noir représente l'agent ou le robot, les traits colorés sont des murs et obstacles, le faisceau rouge représente le spectre des capteurs. La grille représente la division de l'environnement en "cellules" pour la classification*

		Budget			
Data acquisition		1	3	5	7
Image classification		44.4	52.8	56.7	58.1
Forced Policy	Recurrent	58.8	70.2	75.2	76.1
	Non Recurrent	55.8	60.8	61.2	65.3
Free Policy	Recurrent	43.6	71.6	75.2	79.2
	Non Recurrent	46.8	69.4	70.4	73.2

Table 6.1 – *Résultats dans le cas de la tâche simulée. Les résultats sont les performances en pourcentage sur l'ensemble des donnée en test.*

— La politique forcée (*forced policy*) est une version où la politique π a été fixée manuellement et le modèle ne fait qu'apprendre la classification et la formation de la représentation z. La politique forcée a été choisie de façon à alterner les action d'exploration et les actions de récolte de données. Par exemple en prenant $B = 5$ la trajectoire résultante serait $(image, turn, image, turn, image)$
— La politique libre (*free policy*) consiste à laisser le modèle apprendre seul la politique et la représentation.

Ce qui nous permettra de comparer l'efficacité des différentes manières d'agréger de l'information (entre la concaténation et la transformation non linéaire due au RNN) et de voir si les performances de la politique libre atteignent les performances de la politique forcée, ce qui montrerait que l'apprentissage de la politique converge vers la politique forcée considérée comme étant la meilleure.

L'exploration des paramètres a été faite avec un algorithme de *grid search* qui nous a retourné les paramètres ayant les meilleurs performances dans ce protocole de simulation :

— *Learning rate* pour les modules de politique : 0.0001
— *Learning rate* pour les modules de transformation : 0.001
— Taille de l'espace latent : 50
— Résolution du capteur : 10

Résultats et analyses

Le tableau (6.1) représente les résultats obtenus avec le protocole expérimental décrit ci-dessus. Les pourcentages représentent la moyenne des performances pour chaque budget sur cinq simulations dans le même labyrinthe. De ces résultats nous pouvons relever trois observations :

— La performance augmente respectivement à l'augmentation du budget, ce qui confirme l'intuition première qui veut que l'augmentation des données augmente la performance de la prédiction du modèle.
— La performance est meilleure en agrégeant les données plutôt qu'en les concaténant simplement, ce qui implique que l'utilisation de réseaux neuronaux récurrents du modèle permet d'avoir des représentations plus précises des données. Ceci est dû au fait que les réseaux récurrents ont moins de paramètres à apprendre que les

Budget	Run	Policy	Performance
1	1	*Image*	46.4
	2	*Image*	45.3
	3	*Image*	43.3
	4	*Image*	42.1
	5	*Image*	41.3
3	1	*Image-Right Turn-Image*	75.2
	2	*Image-Right Turn-Image*	73.8
	3	*Image-Right Turn-Image*	70.5
	4	*Image-Left Turn-Image*	69.8
	5	*Image-Right Turn-Image*	68.8
5	1	*Image-Left Turn-Image-Left Turn-Image*	78.8
	2	*Image-Left Turn-Image-Left Turn-Image*	77.2
	3	*Left Turn-Image-Right Turn-Right Turn-Image*	74.8
	4	*Image-Right Turn-Image-Image-Image*	73.7
	5	*Image-Image-Image-Right Turn- Image*	71.6
7	1	*Right Turn-Image-Right Turn-Image-Right Turn-Image-Image*	80.7
	2	*Image-Image-Left Turn-Left Turn-Image-Left Turn-Image*	80.6
	3	*Image-Image-Right Turn-Image-Image-Right Turn-Image*	79.7
	4	*Image-Image-Left Turn-Image-Left Turn-Image-Image*	79.4
	5	*Image-Image-Left Turn-Image-Left Turn-Image-Image*	75.5

Table 6.2 – *Performances de la politique libre (en pourcentage). La table est triée en classant, pour chaque budget, depuis la meilleure politique apprise à la moins performante avec les performances en test équivalentes.*

réseaux non récurrents.

— La performance en politique libre est équivalente à la performance en politique forcée, ce qui implique que le modèle converge vers les politiques forcées optimales. De ce fait ce modèle apprend à atteindre les meilleures performances, que ce soit d'un point de vue classification que d'un point de vue de la politique.

Pour préciser ce dernier point et voir à quel point le modèle converge vers une politique qui alterne entre acquisition de donnée et actions de mouvement, on montre dans la table (6.2) les politiques apprises pour chaque simulation par budget.

Dans le tableau 6.2, nous avons 100% de convergence vers des politiques efficaces dans le cas des budget $B = 1$ et $B = 3$. En effet, dans le cas de $B = 1$, le modèle converge à chaque fois vers l'action qui demande une image, qui est dans ce cas la seule action qui apporte de l'information. Le modèle est aussi capable, dans le cas de $B = 3$, d'apprendre à alterner entre une exploitation d'une donnée informative et d'une action exploratoire. Dans le cas des budget au delà de $B = 5$ on voit que le modèle converge vers des politiques plus proches de $B = 3$, ce qui finalement revient à des performances équivalentes. Dans le cas de $B = 5$ le modèle converge deux fois sur cinq sur une politique efficace qui

alterne entre exploitation de donnée et mouvement d'exploration. En convergeant vers des politiques se rapprochant des politiques apprises avec $B = 3$, cela montre que ce type de fonctions d'approximation ont des minima locaux très importants ce qui conduit le modèle à converger vers des politiques qui minimisent l'erreur mais qui ne maximisent pas forcément les performances. Ceci est encore plus clair dans le cas $B = 7$ où le modèle converge vers des politiques équivalentes à celles apprises à $B = 5$.

Nous verrons dans le chapitre 7 comment étendre ce modèle pour trouver le budget qui donne le meilleur rapport coût/bénéfice de l'algorithme de classification, et comment transférer la politique apprise à des environnements similaires de façon à permettre au robot de n'avoir à apprendre que la tâche de classification dans ces nouveaux environnements. Avant ceci, la partie suivante montre un test de l'algorithme sur des données réelles que nous avons collectées avec le robot PR2.

6.4.2 Données Réelles

Figure 6.3 – *Robot PR2 utilisé pour l'acquisition des données nourrissant le modèle*

Figure 6.4 – *Données images récoltées par le PR2*

		Budget		
Type d'acquisition		1	3	5
Classification d'images		49	52.2	53
Politique forcée	Récurrent	42	43	44.2
Politique Libre	Récurrent	43	46	47

Table 6.3 – *Résultats dans le cas des données expérimentales. Les résultats sont en pourcentage.*

Protocole expérimental

Le protocole dans le cas des données réelles est à quelques détails près le même que dans le cas des données simulées. Nous avons utilisé le robot PR2 (6.3) pour l'acquisition des données en environnement réel.

Comme la tâche est modélisée comme une tâche de classification, nous avons fixé le nombre de cellules à 16, de surface de $4m^2$ (2mx2m), dans lesquels nous avons tiré au hasard 40 coordonnées avec des angles fixes. Les mouvements du robot sont donc limités pour chaque classe à la cellule correspondante. A chaque coordonnée nous avons fait un relevé 360^o avec un angle de $\pi/4$ pour les images et les lasers. Nous avons utilisé la caméra grand angle du robot pour les images et le laser de la base du robot dont le faisceau est de 270^o. De ce fait nous avons 8 images par coordonnées ce qui fait au total 5120 points. Le laser retourne un vecteur de taille 1000 avec les distances aux obstacles. Les images sont de taille 640*480 px (un exemple des images relevées est donnée dans la figure 6.4). Pour une facilité de traitement les images ont été pré-traitées avec un réseau de neurones convolutionnel (CNN) déjà entraîné sur des images appelé OverFeat (Sermanet et al. p.d.). Nous fixons l'ensemble d'apprentissage à 2560 points et l'ensemble de test et de validation à 1280 chacun. Nous comparons les trois différents types d'acquisitions mentionnés dans le cas des données simulées.

Nous avons relancé le même protocole de *grid search* pour trouver les meilleurs paramètres du modèle avec les mêmes ensembles que ceux définis dans le cas des données simulées.

Les résultats de ce protocole expérimental sont résumés dans le tableau 6.3. Les principales observations faites dans le cas de l'expérience en environnement simulé restent vraies ici, plus le budget est grand meilleures sont les performances. Cependant, dans ce cas réel les performances sont loin d'être aussi satisfaisantes que dans le cas simulé. Dans le tableau 6.3, les performances ne dépassent pas les 53% dans le cas d'une classification simple pour $B = 5$, une performance qui n'est pas atteinte par le modèle alors qu'en phase d'apprentissage certaines performances atteignent les 99%, ce qui est reflète un phénomène de sur apprentissage. Cependant, on notera que les performances obtenues dans le cas d'une politique libre sont du même ordre de grandeur que les performances en politique forcée, ce qui suggère que, malgré les performances faibles du modèle en prédiction, il apprend tout de même des politiques efficaces alternant entre acquisition de données et actions d'exploration. Il est à noter également que la performance obtenue ici, bien que non pleinement satisfaisante, reste bien supérieure au niveau de la chance qui est à 6.25% pour une telle tâche de classification à 16 classes. Plusieurs hypothèses quand à cette différence de performances peuvent être avancées : premièrement il est possible que l'ensemble des données ne soit pas assez fourni et que le modèle n'ait pas assez de données pour pouvoir généraliser. Deuxièmement l'utilisation du CNN OverFeat peut ne pas être adaptée à ces données précises. Pour la première hypothèse nous avons divisé les images en plusieurs fenêtres pour pouvoir augmenter l'ensemble des données. Pour la deuxième hypothèse nous avons testé un CNN qui apprend uniquement sur les données images tirées du robot mais les performances obtenues suite à ces modifications sont de même nature que celles décrites plus haut. Ce qui suggère que les données ne permettent pas de généraliser l'apprentissage sur cette problématique précise.

6.5 Conclusion

Nous avons présenté dans ce chapitre un nouveau modèle d'apprentissage séquentiel budgétisé appliqué à une tâche de localisation en robotique où l'agent peut décider de lui même d'acquérir des données ou d'explorer son environnement. Ce problème original où l'acquisition de l'information est limitée par un budget fixe est novateur que ce soit d'un point de vue de la méthode d'apprentissage où l'on propose que le modèle aie la possibilité d'avoir des actions qui ne retournent pas d'observations, ou d'un point de de l'expérimentation robotique où le robot se voit contraint d'optimiser sa politique en sélectionnant les senseurs et les actions de façon à minimiser un critère d'erreur de classification.

Les résultats présentés ont été concluant dans le cas d'une simulation avec un modèle apprenant en un nombre de pas de temps restreints des politiques qui alternent entre des actions d'acquisition de données et des actions d'exploration. Néanmoins, dans le cadre de l'apprentissage *offline* de données réelles, le modèle n'offre pas de résultats pleinement

satisfaisants, cela malgré plusieurs tests et modifications des données. Ceci suggère que le modèle n'est pas capable de généraliser efficacement sur ces données spécifiques ressemblantes entres les différentes classes. Ceci est une des limites de l'apprentissage statistique dans le cas de la robotique : les modèles ne sont pas capables d'apprendre sur de petits ensembles de données, ce qui est un problème redondant en robotique, mais c'est aussi un problème intrinsèque aux réseaux de neurones : dans le travail de Szegedy et al. 2013, les auteurs montrent que les réseaux de neurones sont très influencés par des changements, même sensibles, entre les données, en introduisant un faible bruit sur des images apprises (cette modification n'étant pas perçue à l'œil). Les auteurs observent une chute de la performance du réseau pour la reconnaissance de ces images et il est possible que la ressemblance entre les différentes images tirées du robot soit la raison principale de la difficulté d'apprentissage de ce modèle sur les données robotiques.

Chapitre 7

Transfer Learning et sélection de budget a posteriori

7.1 Introduction

Dans le chapitre précédent, nous avons développé une architecture de sélection séquentielle d'actions appliquée à des tâches de robotique et avons conclu que le modèle apprenait en simulation de manière satisfaisante la politique consistant à alterner entre les différentes actions d'explorations et d'exploitation. La conclusion majeure observée est que le modèle apprend de mieux en mieux à mesure que le budget augmente, ce qui n'est pas une surprise, sachant qu'à chaque fois que le modèle acquiert une donnée il affine sa prédiction. Malgré tout nous avons observé que plus le budget augmente plus la convergence vers une politique performante est lente et le gain en performance n'est pas aussi important que celui permis par de plus petit budgets. A partir de cette observation, la question qui se pose est de savoir s'il est possible de trouver un budget optimal pour chaque tâche définie, et si ce budget optimal est transférable à d'autres tâches dans des environnements similaires (e.g. taille et forme du labyrinthe, nombre d'obstacles), ce qui pourrait constituer une connaissance de plus haut niveau ré-exploitable par le robot pour ne pas devoir tout apprendre depuis le début dans chaque nouvel environnement rencontré.

Dans le but d'explorer cette idée, nous testons dans un premier temps la capacité de ce modèle à transférer les actions apprises dans des environnements d'entraînement vers de nouveaux environnements de même ou différente nature. Dans un second temps nous

	Budget			
Type Env	1	3	5	7
Train Env (A)	46.4	75.2	78.8	80.7
Train Env (B)	81.5	89.5	93.6	96.3
Train Env (A) $\rightarrow$ Test Env (A')	51.6	56	69.1	75
Train Env (B) $\rightarrow$ Test Env (A')	56.4	57.5	61.7	66.6
Train Env (B) $\rightarrow$ Test Env (B')	72.2	75.7	90.7	96.4
Train Env (A) $\rightarrow$ Test Env (B')	74.6	81.4	89.2	95.2

Table 7.1 – *Performances du protocole de transfer learning dans des environnements de nature similaires ou différentes (les performances sont en pourcentage)*

mettons en place une fonction de coût qui tente de trouver le meilleur budget selon les différents environnements et les façons de transférer la connaissance d'un environnement à l'autre. Ce travail reste préliminaire et peut être retrouvé dans Aklil et al. Under Review.

7.2 *Transfer learning*

Tout d'abord nous voulons tester la capacité du modèle à généraliser sur les politiques apprises, en transférant les politiques développées dans un environnement d'entraînement et ensuite appliquées dans un environnement de test. Pour cela nous mettons en place deux familles d'environnement : les labyrinthes A de taille 50×50 et des labyrinthes B, où l'environnement est plus petit 20×20. Pour plus de facilité nous appellerons les environnements d'apprentissage A et B et les environnements de tests A' et B' comme montré dans la figure 7.1.

Pour résumer ce protocole simple, en reprenant les notations développées dans le chapitre précédent, nous lançons des apprentissages en *free policy* dans les labyrinthes A et B et en *forced policy* dans les labyrinthes A' et B' avec les politiques π apprises dans A et B. Cela nous permettra de savoir jusqu'à quel point les politiques apprises par le modèle sont performantes dans les différents environnements.

Le tableau 7.1 montre les performances obtenues après transfert des politiques aux environnements de test. Ces performances sont au niveau des performances obtenues dans les environnements d'apprentissage ce qui confirme la capacité du modèle à apprendre des politiques performantes pour des environnements de même types. Par contre lorsqu'on croise les environnement (de A vers B' ou de B vers A') on voit que la performance reste haute dans le cas d'un transfert de A vers B' mais est basse dans le cas de B vers A'. Ceci s'explique par le fait que l'environnement A étant plus grand, le modèle tend à apprendre des politiques qui extraient un maximum de données alors que dans les environnements de type B, le modèle apprend des politiques souvent redondantes.

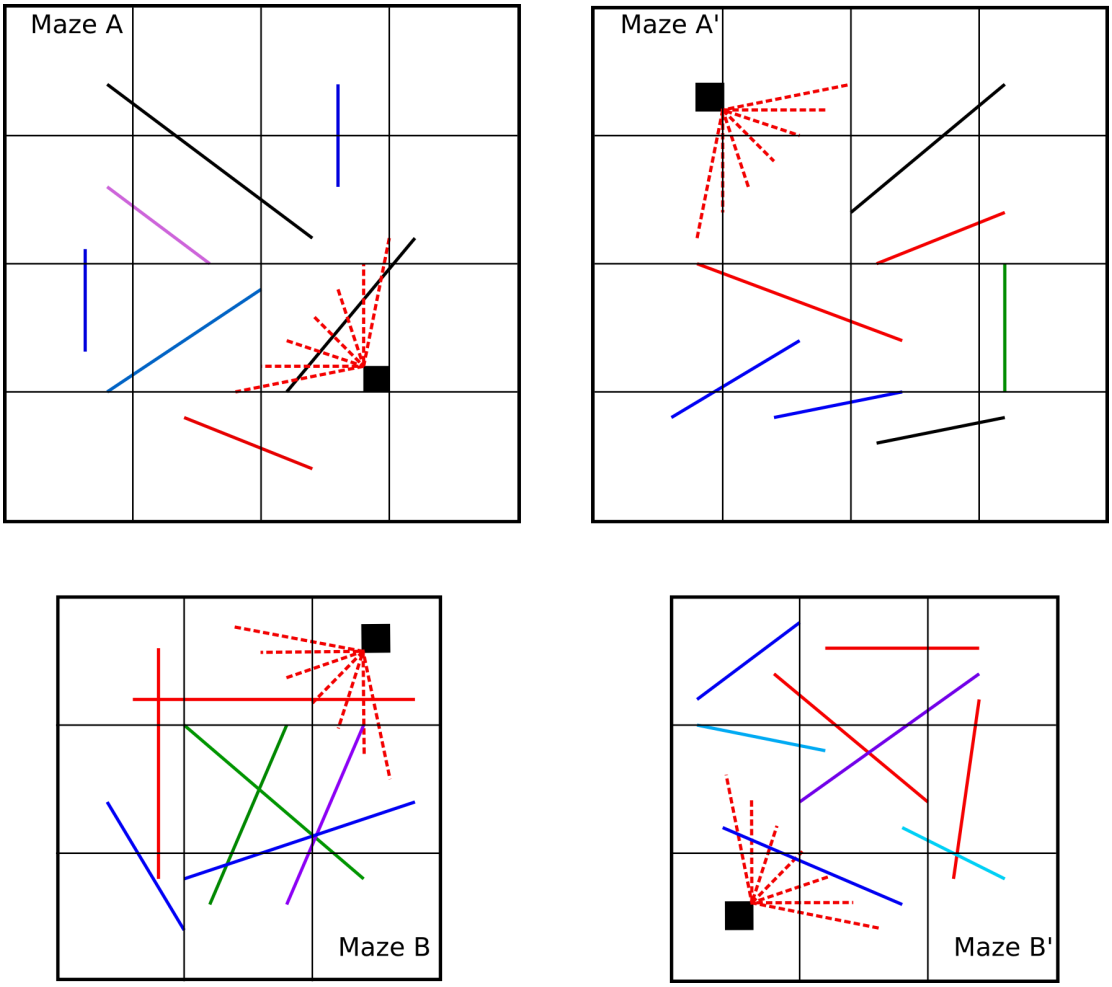

Figure 7.1 – *Types de labyrinthes utilisés dans le protocole de Transfer learning*

De ces observations nous pouvons d'une part tirer la conclusion que la politique apprise est bien transférable entre les environnements testés de même nature et dans certains cas entre environnements testés de nature différente. D'autre part, nous pouvons émettre l'hypothèse que pour chaque type d'environnement il existe ce que l'on appellera un "budget optimal" qui balancerait efficacement entre une bonne performance et une rapidité de traitement reflétée par un budget faible.

7.3 Fonctions coût et budgets optimaux

Dans le but de quantifier l'apport de l'augmentation des budgets à la performance en classification, nous mettons en place un critère qui calcule le gain relatif entre un budget et un autre budget supérieur. Ce critère est défini comme :

$$RG(i) = \frac{(Perf(B = i) - Perf(B = i - 2))}{(1 - Perf(B = i - 2))} \tag{7.1}$$

Ce critère nous donnera une indication de l'utilité d'une augmentation du budget selon les performances retournées pour chacun de ces budgets. Comme nous l'avons vu dans le chapitre 6, il apparaît qu'entre $B = 1$ et $B = 3$ la performance augmente en moyenne de 12% à 23% alors qu'entre $B = 3$ et $B = 5$ cette performance n'augmente que de 5% à 6% ce qui interroge l'utilité d'une augmentation du budget pour une augmentation en performance faible.

De plus, dans le but de définir un budget optimal pour chaque famille d'environnement donnée, nous définissons une simple fonction servant de critère coût-bénéfice calculant un compromis entre la performance et le budget de cette façon :

$$CB(i) = \frac{Perf(B = i)}{\sqrt[n]{(B = i)}} \tag{7.2}$$

où n est un paramètre appartenant à $\mathbb{N}^+$ qui détermine l'importance de l'effet du budget. Plus n est grand, plus ce critère aura tendance à accepter des budgets plus grands pour de petits gains de performances.

La figure 7.2 montre les résultats obtenus de ces deux critères sur les performances obtenues dans les environnements simulés A et B, pour $n = 3$. La figure montre que le meilleur ratio coût-bénéfice ainsi que le gain relatif correspondent à une politique limitée avec un budget de $B = 3$ dans l'environnement complexe A et de $B = 1$ dans l'environnement simple B.

Ceci peut être utilisé comme un critère de sélection a priori du budget le plus adéquat dans des environnements de même nature sans devoir réapprendre la tâche depuis le début.

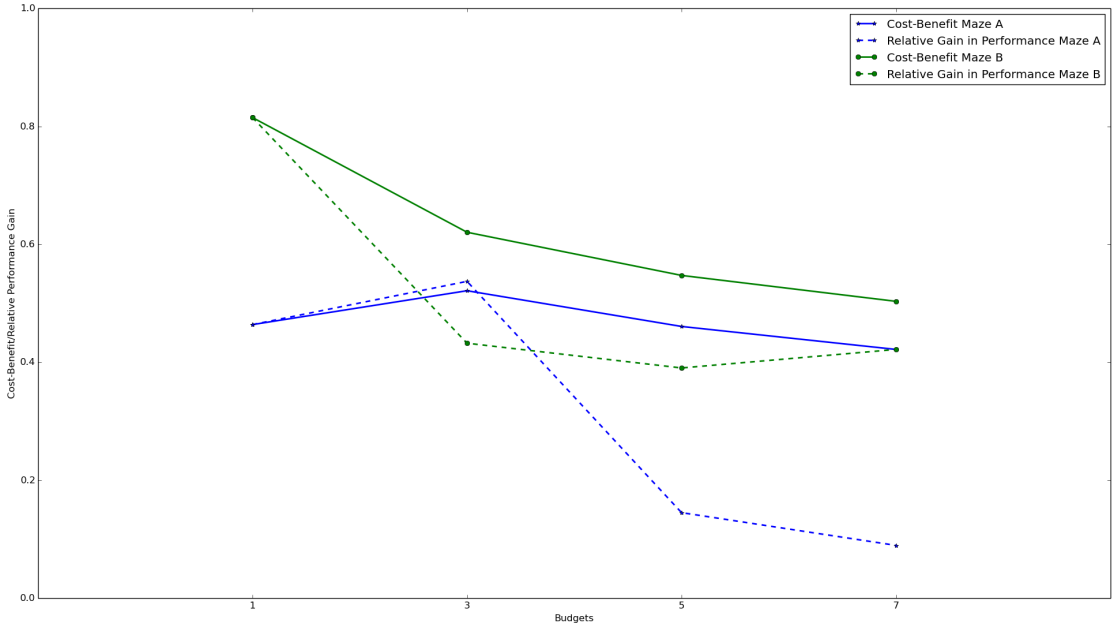

Figure 7.2 – *Courbe de Coût-Bénéfice/Gain en performance relatif : Les lignes bleues le critère de coût-bénéfice/gain relatif en performance dans le labyrinthe A, en vert dans le labyrinthe B. Les lignes pleines montrent le critère Coût-bénéfice (7.2) et les pointillés le critère de gain en performance relatif (7.1).*

7.4 Conclusion

Dans ce chapitre nous avons montré que le modèle décrit dans le chapitre 6 peut apprendre à généraliser les politiques qui peuvent être apprises sur différents environnements qui sont de différentes ou de mêmes nature. Nous avons aussi montré que des politiques apprises dans des environnements plus grands peuvent être transférées vers des environnements plus petits mais que l'inverse n'est pas utile. Enfin nous avons introduit une manière simple d'implémenter une fonction coût-bénéfice à maximiser en sélectionnant différents budgets optimaux pour chaque famille d'environnements. Ces modèles peuvent être utilisés comme base de construction de robots navigant dans des environnements en utilisant ces critères sans devoir réapprendre de base les différentes politiques une fois qu'il est dans un environnement considéré comme de même famille qu'un des environnements visités précédemment. Malgré tout, le critère coût-bénéfice proposé ici reste préliminaire, il serait plus efficace de trouver une formule CB plus ancrée dans la réalité robotique (liée aux coûts réels du robot) ainsi qu'augmenter le nombre de simulations. Cette amélioration est actuellement en cours, ainsi que l'augmentation des types d'environnements testés dans le cadre du transfert.

Chapitre 8

Conclusion

8.1 Résumé des contributions

Dans le travail présenté dans ce manuscrit nous nous sommes intéressés aux processus de décision d'un point de vue neurosciences et d'un point de vue robotique. Du côté des neurosciences, nous avons testé la capacité des modèles d'apprentissage par renforcement à réguler dynamiquement le taux d'exploration d'agents apprenants. Nous avons proposé une régulation basée sur une estimation de la récompense de la part de l'agent à long et court terme et avons observé que cette régulation est importante pour la capacité des modèles d'AR à expliquer les données obtenues chez les rats. Ceci suggère que les rats s'emploient en effet à réguler leur exploration de manière à maximiser les retours obtenus en se basant sur une estimation continue de la récompense.

D'un point de vue robotique, nous avons proposé une architecture de sélection séquentielle d'actions budgétisée prenant en compte un budget explicite limitant le nombre d'actions autorisées au robot pouvant implicitement représenter un certain coût de l'action comme celui lié aux limitations matérielles du robot. Nous avons appliqué ce modèle à une tâche de navigation robotique en simulation et dans un cas réel. Nous avons pu voir que ce type de modèle était avantageux pour minimiser les données récoltées par des agents en environnement simulé, en réussissant à apprendre à partir de rien des séquences d'actions lui permettant de se localiser efficacement dans son environnement. Nous avons aussi pu montrer les limites de ce type de modèle lors de l'application sur des tâches en environnement réel. Malgré la capacité du modèle à apprendre des politiques efficaces alternant entre des actions d'acquisition de donnée et des actions de mouvement, il reste

malgré tout peu satisfaisant quant à sa capacité à généraliser sur les images tirées du robot.

Dans le cas d'une application simulée nous avons observé que la performance en moyenne obtenue n'augmentait que de peu en augmentant le budget, ce qui nous a fait supposer que le modèle atteignait un maximum de performance pour un certain budget dans un environnement donné. Dans cette optique, nous avons testé des fonctions de calcul de gain relatif en performance pour différents budgets et de déterminer le rapport coût/bénéfice de façon à pouvoir trouver un budget "optimal" (i.e. qui maximise ce rapport) dans un environnement donné. Nous avons ensuite proposé un protocole expérimental pour tester la capacité du modèle à transférer des politiques apprises dans des environnements d'une certaine nature vers des environnements de même ou différente nature et de tester par la même occasion la possibilité de l'existence d'un budget optimal pour chaque type d'environnement en utilisant une fonction de gain relatif et une fonction coût qui ont montré qu'en effet à chaque type d'environnement correspondait un budget optimal, une connaissance qui pourrait être utilisée à priori lors de modélisations de même nature. Ce travail nous permet d'ouvrir des perspectives sur les connaissances apprises par notre modèle qui pourraient être généralisées à d'autres environnements, de façon à éviter au robot de ré-apprendre les tâches depuis le début à chaque fois.

8.2 Limites et perspectives

L'inspiration biologique pour l'implémentation de modèles de prise de décision robotique n'est pas nouvelle (voir par exemple Brooks 1986 ; Guillot et Meyer 2000 ; Meyer et Guillot 2008). Récemment, (Renaudo 2016, Viejo 2016, Dollé et al. 2010, Quoy et al. 2001) ont modélisé la prise de décision animale et robotique en mettant en compétition (ou collaboration) des modèles directs et indirects de l'apprentissage par renforcement. Or, ces modèles ont des temps de prise de décision différents. En effet, un modèle direct prends des décisions rapidement selon les estimations qu'il a des valeurs des états qui lui sont fournies alors qu'un modèle indirect prends des décisions lentes, le temps de déployer sa prédiction d'une séquence d'action (Giovanni et al. 2013, Viejo et al. 2015). Ces deux systèmes sont gérés par un arbitre, qui donnera la main à l'un ou l'autre des modèles (e.g. Renaudo et al. 2015). Il serait de fait intéressant d'introduire un arbitre budgétisé qui puisse prendre en compte un compromis entre la performance attendue et le temps de prise de décision. De cette façon les modèles robotiques neuro-inspirés pourraient être applicables efficacement à des situations réelles, en associant, par exemple, un coût en terme d'énergie et de temps de calcul à chaque action, qui permettrait ainsi de trouver un comportement robotique maximisant un certain rapport coût/bénéfice.

Par ailleurs, dans le but de mettre au point des robots capables de s'adapter à des situations de la vie réelle, il est important de s'inspirer des études comportementales des neurosciences computationnelles (une revue de l'histoire des liens entre l'intelligence artificielle et la robotique peut être trouvée dans cet article récent Hassabis et al. 2017). C'est pour cela que l'étude de la capacité des mammifères à gérer des compromis entre

exploration et exploitation est importante à la mise en place de processus décisionnels flexibles. Cependant, il est à noter que l'approche par MDP et apprentissage par renforcement montre certaines limites. Dans la majorité des études menées dans le cadre des neurosciences computationnelles, on considère une connaissance parfaite de l'environnement en modélisant les états du MDP tels qu'observés par l'expérimentateur qui mettra en place un modèle de l'animal dans le cadre de ce problème uniquement. Or, il est plus difficile dans le cas d'un robot de mettre en place un état représentatif de son environnement. Nous avons modestement tenté de mettre en place une manière simple de créer ces représentations à partir des entrées des capteurs en utilisant des RNN comme agrégateurs de données. Si les résultats obtenus en simulation sont prometteurs, les résultats sur données réelles ne sont pas encore pleinement satisfaisants. Ceci suggère donc qu'il est important de pouvoir trouver un moyen plus efficace de représenter les états internes au modèle de façon à augmenter la performance.

Plusieurs pistes nous semblent intéressantes parmi lesquels l'intégration d'un budget dans des architectures de décision multi-systèmes ainsi que de meilleures fonctions de représentations des états internes du modèle. A coté de cela, le budget appliqué dans ce cas rentre dans la catégorie *hard budget*, où le budget est donné explicitement et limite clairement le temps de traitement utilisé dans un problème donné. Or, il serait interessant de pouvoir mettre un place un modèle qui apprenne de façon autonome à s'arrêter au bon moment (e.g. une fois que le nombre de données acquises est considéré suffisant). Dans le travail de D. Benbouzid 2012, les auteurs proposent une telle architecture, limitée cependant par le fait que le traitement se fait *offline*. Nous pouvons en effet intégrer une telle contrainte au modèle d'apprentissage en ajoutant, en plus des actions de déplacements et d'acquisition de données, une action lui permettant de s'arrêter et d'accéder directement à la classification en utilisant le même algorithme d'apprentissage.

Il est à noter aussi que l'approche choisie dans notre travail est basée sur de l'apprentissage supervisé (i.e. on apprend à approcher une donnée connue par apprentissage), or dans le cas d'applications réelles, il est plus avantageux au robot de pouvoir apprendre à partir de rien (Droniou 2015). En effet, un robot ne peut avoir une connaissance de l'environnement qui l'entoure et l'interaction avec son environnement est ce qui lui permet de se créer une représentation sans devoir se référer à un "oracle", représenté ici par la supervision de l'apprentissage.

8.3 Conclusion

Dans ce travail, nous n'avons fait qu'effleurer le domaine de la prise de décision. Nous avons tenté de dresser un parallèle entre la capacité d'animaux à gérer un compromis entre exploration et exploitation et l'apprentissage budgétisé qui gère un compromis entre acquisition de données et actions de déplacements. Nous avons dressé un ensemble de perspectives qui se veulent riches en questionnements tels que la capacité de créer un

modèle d'apprentissage capable de considérer un environnement réel et l'importance de s'inspirer des études en neurosciences pour mettre en place des architectures robotiques qui s'adaptent aux changements de l'environnement.

Publications

Articles

Aklil, N., B. Girard, L. Denoyer et M. Khamassi (Under Review). « Sequential Action Selection and Active Sensing for Budgeted Localization in Robot Navigation ». In : *International Journal of Semantic Computing*.

Aklil, N., B. Girard, M. Khamassi et L. Denoyer (2017). « Sequential Action Selection for Budgeted Localization in Robots ». In : *IEEE Robotic Computing 2017*. Taipei, Taiwan, p. 97–100.

Aklil, N., A. Marchand, V. Fresno, E. Coutureau, L. Denoyer, B. Girard et M. Khamassi (2014). « Modelling rat learning behavior under uncertainty in a non-stationary multi-armed bandit task ». In : *Fourth Symposium on Biology of Decision Making (SBDM 2014)*. Poster. Paris.

Cinotti, F., V. Fresno, N. Aklil, E. Coutureau, B. Girard, A. Marchand et M. Khamassi (2016). « Dopamine blockade affects exploration but not learning rate in a non-stationary 3-armed bandit task ». In : *Sixth Symposium on Biology of Decision-Making (SBDM 2016)*. Paris, France.

— (2017). « Dopamine enables dynamic regulations of exploration ». In : *Third Multidisciplinary Conference on Reinforcement Learning and Decision Making*. Ann Arbor, Michigan.

— (In Preparation). « Dopamine enables dynamic regulation of exploration ». In : *Journal paper*.

Bibliographie

Adams, Christopher D et Anthony Dickinson (1981). « Instrumental responding following reinforcer devaluation ». In : *The Quarterly Journal of Experimental Psychology Section B : Comparative and Physiological Psychology* 33.2, p. 109–121. issn : 02724995.

Aklil, N., B. Girard, L. Denoyer et M. Khamassi (Under Review). « Sequential Action Selection and Active Sensing for Budgeted Localization in Robot Navigation ». In : *International Journal of Semantic Computing*.

Aklil, N., B. Girard, M. Khamassi et L. Denoyer (2017). « Sequential Action Selection for Budgeted Localization in Robots ». In : *IEEE Robotic Computing 2017*. Taipei, Taiwan, p. 97–100.

Aklil, N., A. Marchand, V. Fresno, E. Coutureau, L. Denoyer, B. Girard et M. Khamassi (2014). « Modelling rat learning behavior under uncertainty in a non-stationary multi-armed bandit task ». In : *Fourth Symposium on Biology of Decision Making (SBDM 2014)*. Poster. Paris.

Balleine, Bernard W et Anthony Dickinson (1998). « Goal-directed instrumental action : contingency and incentive learning and their cortical substrates ». In : *Neuropharmacology* 37.4, p. 407–419.

Barto, A. G., R. S. Sutton et C. W. Anderson (1983). « Neuronlike adaptive elements that can solve difficult learning control problems ». In : *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5, p. 834–846. issn : 0018-9472.

Bellman, Richard (1957). *Dynamic Programming*. 1re éd. Princeton, NJ, USA : Princeton University Press.

Bellot, Jean, Olivier Sigaud et Mehdi Khamassi (2012). « Which temporal difference learning algorithm best reproduces dopamine activity in a multi-choice task ? » In : *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7426 LNAI, p. 289–298. issn : 03029743.

Berns, G. S. et T. J. Sejnowski (1998). « A computational model of how basal ganglia produce sequences ». In : *Journal of Cognitive Neuroscience* 1.10.

Berridge, Kent C. (2007). « The debate over dopamine's role in reward : The case for incentive salience ». In : *Psychopharmacology* 191.3, p. 391–431. issn : 00333158.

Blodgett, H.C. (1929). « The Effect of the Introduction of Reward Upon the Maze Performance of Rats ». In : *University of California publications in psychology. Kraus Reprint Company*.

Bornstein, Aaron M. et Nathaniel D. Daw (2013). « Cortical and Hippocampal Correlates of Deliberation during Model-Based Decisions for Rewards in Humans ». In : *PLoS Computational Biology* 9.12. Sous la dir. de Tim Behrens, e1003387. issn : 1553-7358.

Brafman, Ronen I. et Moshe Tennenholtz (2001). « R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning ». In : *IJCAI International Joint Conference on Artificial Intelligence* 3, p. 953–958. issn : 10450823.

Brooks, R. (1986). « A robust layered control system for a mobile robot ». In : *IEEE journal on robotics and automation* 2.1, p. 14–23.

Burkitt, a N (2006). « A review of the integrate-and-fire neuron model ». In : *Biological Cybernetics* 95.1, p. 1–19, 97–112. issn : 0340-1200.

Caluwaerts, K, M Staffa, S N'Guyen, C Grand, L Dollé, a Favre-Félix, B Girard et M Khamassi (2012). « A biologically inspired meta-control navigation system for the Psikharpax rat robot. » In : *Bioinspiration {&} biomimetics* 7.2, p. 25009. issn : 1748-3190.

Chavarriaga, Ricardo, Denis Sheynikhovich et Wulfram Gerstner (2005). « A computational Model of parallel navigation systems in rodents ». In : *Neuroinformatics* 3.3, p. 223–241.

Chen, Minmin, Kilian Q Weinberger, Saint Louis, Olivier Chapelle, Santa Clara et Dor Kedem (2012). « Classifier Cascade for Minimizing Feature Evaluation Cost ». In : *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, PMLR* 22.

Chen, Shengyong, Youfu Li et Ngai Ming Kwok (2011). « Active vision in robotic systems : A survey of recent developments ». In : *The International Journal of Robotics Research* 30.11, p. 1–35.

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk et Yoshua Bengio (2014). « Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation ». In : *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 1724–1734. issn : 09205691. arXiv : 1406.1078.

Chung, Junyoung, Caglar Gulcehre, Kyunghyun Cho et Yoshua Bengio (2014). « Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling ». In : p. 1–9. arXiv : arXiv:1412.3555v1.

Cinotti, F., V. Fresno, N. Aklil, E. Coutureau, B. Girard, A. Marchand et M. Khamassi (2017). « Dopamine enables dynamic regulations of exploration ». In : *Third Multidisciplinary Conference on Reinforcement Learning and Decision Making*. Ann Arbor, Michigan.

— (In Preparation). « Dopamine enables dynamic regulation of exploration ». In : *Journal paper*.

Contardo, Gabriella, Ludovic Denoyer et Thierry Artières (2016). « Recurrent Neural Networks for Adaptive Feature Acquisition ». In : *Neural Information Processing : 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part III*. Sous la dir. d'Akira Hirose, Seiichi Ozawa, Kenji Doya, Ka-

zushi Ikeda, Minho Lee et Derong Liu. Cham : Springer International Publishing, p. 591–599.

D. Benbouzid R. Busa-Fekete, B. Kégl (2012). « MDDAG : learning deep decision DAGs in a Markov decision process setup ». In : *25th Annual Conference on Neural Information Processing Systems (NIPS 2011)*, p. 1–9.

Daw, Nathaniel D, Yael Niv et Peter Dayan (2005). « Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. » In : *Nature neuroscience* 8.12, p. 1704–11. issn : 1097-6256.

Daw, Nathaniel D. et John P. O'Doherty (2013). « Multiple Systems for Value Learning ». In : *Neuroeconomics : Decision Making and the Brain : Second Edition*. Elsevier Inc., p. 393–410.

Denoyer, Ludovic et Patrick Gallinari (2014). « Deep Sequential Neural Network ». In : *CoRR* abs/1410.0510.

Dollé, Laurent, Denis Sheynikhovich, Benoit Girard, Ricardo Chavarriaga et Agnès Guillot (2010). « Path planning versus cue responding : a bio-inspired model of switching between navigation strategies. » In : *Biological cybernetics* 103.4, p. 299–317. issn : 1432-0770.

Dominey, Peter, Jean Decety, Emmanuel Broussolle, Guy Chazot et Marc Jeannerod (1995). « Motor imagery of a lateralized sequential task is asymmetrically slowed in hemi Parkinson's patients ». In :

Doya, K. (1999). « What are the computations of the cerebellum, the basal ganglia and the cerebral cortex ? » In : *Neural Networks* 12.7, p. 961–974.

Dredze, Mark et Ari Elias-bachrach (2007). « Learning Fast Classifiers for Image Spam ». In : *Corpus* C, p. 2005–2008.

Droniou, Alain (2015). « Apprentissage de représentations et robotique développementale : quelques apports de l'apprentissage profond pour la robotique autonome ». Thèse de doct. Université Pierre et Marie Curie.

Dulac-Arnold, Gabriel, Ludovic Denoyer et Patrick Gallinari (2011). « Text Classification : A Sequential Reading Approach ». In : *Advances in Information Retrieval* 6611.2, p. 411–423. arXiv : 1107.1322.

Dulac-Arnold, Gabriel, Ludovic Denoyer, Nicolas Thome, Matthieu Cord et Patrick Gallinari (2014). « Sequentially Generated Instance-Dependent Image Representations for Classification ». In : *Proceedings of the International Conference on Learning Representations (ICLR2014)*. arXiv : arXiv:1312.6594v3.

Dutech, Alain, Etienne Coutureau et Alain R. Marchand (2011). « A reinforcement learning approach to instrumental contingency degradation in rats ». In : *Journal of Physiology* 105.1. Computational Neuroscience : Neurocomp 2010, p. 36–44.

Eggert, J. et J. L. van Hemmen (2001). « Modeling Neuronal Assemblies : Theory and Implementation ». In : *Neural Computation* 13.9, p. 1923–1974.

Filliat, D., E. Battesti, S. Bazeille, G. Duceux, A. Gepperth, L. Harrath, I. Jebari, R. Pereira, A. Tapus, C. Meyer, S. H. Ieng, R. Benosman, E. Cizeron, J. C. Mamanna et B. Pothier (2012). « RGBD object recognition and visual texture classification for indoor semantic mapping ». In : *2012 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, p. 127–132.

Filliat, David et Jean-Arcady Meyer (2003). « Map-based navigation in mobile robots : a survey ». In : *Cognitive Systems Research* 4.4, p. 243–282.

Gao, Tianshi et Daphne Koller (2011). « Active Classification based on Value of Classifier ». In : *Advances in Neural Information Processing Systems 24*. Sous la dir. de J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira et K. Q. Weinberger. Curran Associates, Inc., p. 1062–1070.

Giovanni, Pezzulo, Rigoli F et Chersi F. (2013). « The Mixed Instrumental Controller : Using Value of Information to Combine Habitual Choice and Mental Simulation ». In : *Frontiers in Psychology* 4.

Girard, Benoit et Mehdi Khamassi (2016). « Cooperation of multiple reinforcement learning algorithms ». In : *Techniques de l'Ingenieur*, S7793.

Glascher, J., N. Daw, P. Dayan et J. P. O'Doherty (2010). « States versus Rewards : Dissociable neural prediction error signals underlying model-based and model free reinforcement learning ». In : *Neuron* 66.4, p. 585–595.

Glorot, Xavier, Antoine Bordes et Yoshua Bengio (2011). « Domain Adaptation for Large-Scale Sentiment Classification : A Deep Learning Approach ». In : *Proceedings of the 28th International Conference on Machine Learning* 1, p. 513–520.

Goodfellow, Ian J, David Warde-Farley, Mehdi Mirza, Aaron Courville et Yoshua Bengio (2013). « Maxout Networks ». In : *Proceedings of the 30th International Conference on Machine Learning (ICML)* 28, p. 1319–1327. arXiv : 1302.4389.

Graves, Alex (2016). « Adaptive Computation Time for Recurrent Neural Networks ». In : *CoRR* abs/1603.08983.

Graybiel, Ann M (1995). « Building action repertoires : memory and learning functions of the basal ganglia ». In : *Current Opinion in Neurobiology* 5.6, p. 733–741.

Greiner, Russell, Adam J. Grove et Dan Roth (2002). « Learning cost-sensitive active classifiers ». In : *Artificial Intelligence* 139.2, p. 137–174. issn : 0004-3702.

Guillot, A. et J. A. Meyer (2000). « From SAB94 to SAB2000 : what's new, animat ». In : *From animals to animats* 6, p. 3–12.

Hampton, Alan N, Peter Bossaerts et John P O'Doherty (2006). « The role of the ventromedial prefrontal cortex in abstract state-based inference during decision making in humans. » In : *The Journal of neuroscience : the official journal of the Society for Neuroscience* 26.32, p. 8360–7. issn : 1529-2401.

Hassabis, Demis, D. Kumaran, C. Summerfield et M. Botvinick (2017). « Neuroscience Inspired Artificial Intelligence ». In : *Neuron* 95.2, p. 245–258.

Hebb, Donald O. (1949). *The organization of behavior : A neuropsychological theory*. New York : Wiley.

Hochreiter, Sepp et J Urgen Schmidhuber (1997). « Long Short-Term Memory ». In : *Neural Computation* 9.8, p. 1735–1780. issn : 0899-7667. arXiv : 1206.2944.

Hodgkin, A. L. et A. F. Huxley (1953). « A quantitative description of membrane current and its application to conduction and excitation in nerve ». In : *Bulletin of Mathematical Biology* 52.1-2, p. 25–71. issn : 00928240. arXiv : NIHMS150003.

Hollerman, J R et W Schultz (1998). « Dopamine neurons report an error in the temporal prediction of reward during learning. » In : *Nature neuroscience* 1.4, p. 304–9. issn : 1097-6256. arXiv : NIHMS150003.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. Cambridge, MA : MIT Press.

Ji, Shihao et Lawrence Carin (2007). « Cost-sensitive feature acquisition and classification ». In : *Pattern Recognition* 40.5, p. 1474–1485. issn : 00313203.

Kaelbling, Leslie Pack, Michael L Littman et Anthony R Cassandra ' (1998). « Artificial Intelligence Planning and acting in partially observable stochastic domains ». In : *Artificial IntelligenceArtificial Intelligence* 101.101, p. 99–134. issn : 00043702.

Kapoor, Aloak et Russell Greiner (2005). « Learning and classifying under hard budgets ». In : *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 3720 LNAI, p. 170–181. issn : 03029743. arXiv : arXiv:1011.1669v3.

Kearns, Michael et Satinder Singh (2002). « Near-optimal reinforcement learning in polynomial time. Machine Learning ». In : *Machine Learning* 49, p. 209–232. issn : 0885-6125.

Keramati, Mehdi, Amir Dezfouli et Payam Piray (2011). « Speed/accuracy trade-off between the habitual and the goal-directed processes. » In : *PLoS computational biology* 7.5, e1002055. issn : 1553-7358.

Khamassi, Mehdi, Benoît Girard, Aurélie Clodic, Devin Sandra, Erwan Renaudo, Elisabeth Pacherie, Rachid Alami et Raja Chatila (2016). « Integration of Action, Joint Action and Learning in Robot Cognitive Architectures ». In : *Intellectica - La revue de l'Association pour la Recherche sur les sciences de la Cognition (ARCo)* 2016/1.65, p. 169–203.

Khamassi, Mehdi, Stéphane Lallée, Pierre Enel, Emmanuel Procyk et Peter F. Dominey (2011). « Robot cognitive control with a neurophysiologically inspired reinforcement learning model ». In : *Frontiers in Neurorobotics* 5.1 JULY, p. 1–14. issn : 16625218.

Kishida, Kenneth T, Ignacio Saez, Terry Lohrenz, Mark R Witcher, Adrian W Laxton, Stephen B Tatter, Jason P White, Thomas L Ellis, Paul E M Phillips et P Read Montague (2015). « Subsecond dopamine fluctuations in human striatum encode superposed error signals about actual and counterfactual reward. » In : *Proceedings of the National Academy of Sciences of the United States of America*, p. 1513619112–. issn : 1091-6490.

Kocsis, Levente et Csaba Szepesvári (2006). « Bandit based monte-carlo planning ». In : *Proceedings of ECML*, p. 282–203. issn : 03029743.

Konda, Vijay R et John N Tsitsiklis (2000). « Actor-Critic Algorithms ». In : *2nd Annual conference NIPS*.

Le Cun, Yann (1986). « Learning Process in an Asymmetric Threshold Network ». In : *Disordered Systems and Biological Organization*. Sous la dir. d'E. Bienenstock, F. Fogelman Soulié et G. Weisbuch. Springer Berlin Heidelberg, p. 233–240.

Lesaint, Florian, Olivier Sigaud, Shelly B. Flagel, Terry E. Robinson et Mehdi Khamassi (2014). « Modelling Individual Differences in the Form of Pavlovian Conditioned Approach Responses : A Dual Learning Systems Approach with Factored Representations ». In : *PLoS Computational Biology* 10.2. issn : 15537358.

Lienard, Jean (2013). « Étude de l'anatomie fonctionnelle et de la Pathophysiologie à l'aide d'Algorithmes Évolutionnistes Multi-Objectifs ». Thèse de doct.

Ljungberg T., P. Apicella et W. Schultz (1992). « Responses of monkey dopamine neurons during learning of behavioral reactions ». In : *Journal of Neurophysiology*.

Louie, Kenway, Mel W. Khaw et Paul W. Glimcher (2013). « Normalization is a general neural mechanism for context-dependent decision making ». In : *Proceedings of the National Academy of Sciences* 110.15, p. 6139–6144.

Maass, Wolfgang, Thomas Natschläger et Henry Markram (2002). « Real-Time Computing Without Stable States : A New Framework for Neural Computation Based on Perturbations ». In : *Neural Computation* 14.11, p. 2531–2560.

McCulloch, Warren S. et Walter Pitts (1943). « A logical calculus of the ideas immanent in nervous activity ». In : *The bulletin of mathematical biophysics*.

Meyer, J. A. et A. Guillot (2008). « Biologically inspired robots ». In : *Springer Handbook of Robotics*, p. 1395–1422.

Mihaylova, Lyudmila, Tine Lefebvre, Herman Bruyninckx, Klaas Gadeyne et Joris De Schutter (2003). « A Comparison of Decision Making Criteria and Optimization Methods for Active Robotic Sensing ». In : *Numerical Methods and Applications : 5th International Conference, NMA 2002 Borovets, Bulgaria, August 20–24, 2002 Revised Papers*. Sous la dir. d'Ivan Dimov, Ivan Lirkov, Svetozar Margenov et Zahari Zlatev. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 316–324.

Mnih, Volodymyr, Nicolas Heess, Alex Graves et Koray Kavukcuoglu (2014). « Recurrent Models of Visual Attention ». In : *CoRR* abs/1406.6247.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg et Demis Hassabis (2015). « Human-level control through deep reinforcement learning. » In : *Nature* 518.7540, p. 529–533.

Montague, P Read, Steven E Hyman et Jonathan D Cohen (2004). « Computational roles for dopamine in behavioural control. » In : *Nature* 431.7010, p. 760–767. issn : 1476-4687.

Montemerlo, Michael, Sebastian Thrun, Daphne Koller et Ben Wegbreit (2002). « FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem ». In : *Eighteenth National Conference on Artificial Intelligence*. Menlo Park, CA, USA : American Association for Artificial Intelligence, p. 593–598.

Moulines, Eric (2008). « On Upper-Confidence Bound Policies for Non-Stationary Bandit Problems ». In : 1985, p. 1–24. arXiv : arXiv:0805.3415v1.

Moutarlier, Philippe et Raja Chatila (1990). « An experimental system for incremental environment modelling by an autonomous mobile robot ». In : *Experimental Robotics I : The First International Symposium Montreal, June 19–21, 1989*. Sous la dir. de Vincent Hayward et Oussama Khatib. Berlin, Heidelberg : Springer Berlin Heidelberg, p. 327–346. isbn : 978-3-540-46917-9.

Nair, Vinod et Geoffrey E Hinton (2010). « Rectified Linear Units Improve Restricted Boltzmann Machines ». In : *Proceedings of the 27th International Conference on Machine Learning* 3, p. 807–814. issn : 1935-8237. arXiv : 1111.6189v1.

Neurone Formel (2006). https://fr.wikipedia.org/wiki/Réseau_de_neurones_artificiels.

Packard, Mark G. et James L. McGaugh (1996). « Inactivation of Hippocampus or Caudate Nucleus with Lidocaine Differentially Affects Expression of Place and Response Learning ». In : *Neurobiology of Learning and Memory* 65.1, p. 65–72. issn : 1074-7427.

Parker, D.B., Massachusetts Institute of Technology et Sloan School of Management (1985). *Learning Logic : Casting the Cortex of the Human Brain in Silicon*. Technical report : Center for Computational Research in Economics and Management Science. Massachusetts Institute of Technology, Center for Computational Research in Economics et Management Science.

Pavlov, Ivan P. (1927). « Conditional Reflex, An Investigation of The Psychological Activity of the Cerebral Cortex ». In : *New York, Oxford University press*.

Q-learning (2016). https://lopespm.github.io/machine_learning/2016/10/06/deep-reinforcement-learning-racing-game.html.

Quoy, Mathias, Jean-Paul Banquet et Emmanuel Daucé (2001). « Learning and control with chaos : From biology to robotics ». In : *Behavioral and Brain Sciences* 24.5, p. 824–825.

Renaudo, E, B. Girard, R Chatila et M. Khamassi (2015). « Respective advantages and disadvantages of model-based and model-free reinforcement learning in a robotics neuro-inspired cognitive architecture ». In : *Biologically Inspired Cognitive Architectures BICA 2015*. Lyon, France, TBA.

Renaudo, Erwan (2016). « Des comportements flexibles aux comportements habituels : Meta-apprentissage neuro-inspiré pour la robotique autonome ». Thèse de doct. Université Pierre et Marie Curie.

Renaudo, Erwan, Benoît Girard, Raja Chatila et Mehdi Khamassi (2014). « Design of a control architecture for habit learning in robots ». In : *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8608 LNAI, p. 249–260. issn : 16113349.

Rosenblatt, Frank (1958). « The perceptron : a probabilistic model for information storage and organization in the brain ». In : *Psychological review*.

Rummery, G a et M Niranjan (1994). « On-line Q-learning using connectionist systems ». In : September, Technical Report CUED/F/INFENG/TR 166.

Schultz, W, P Dayan et P R Montague (1997). « A neural substrate of prediction and reward. » In : *Science* 275.June 1994, p. 1593–1599. issn : 0036-8075. arXiv : NIHMS150003.

Schultz, Wolfram (1998). « Predictive reward signal of dopamine neurons. » In : *Journal of neurophysiology* 80.1, p. 1–27. issn : 0022-3077. arXiv : 1407.4831.

— (2013). « Updating dopamine reward signals ». In : *Current Opinion in Neurobiology* 23.2, p. 229–238. issn : 09594388.

Schweighofer, Nicolas et Kenji Doya (2003). « Meta-learning in Reinforcement Learning ». In : *Neural Networks* 16.1, p. 5–9.

Sermanet, Pierre, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus et Yann Lecun. « OverFeat : Integrated Recognition, Localization and Detection using Convolutional Networks ». In : arXiv : arXiv:1312.6229v4.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel et Demis Hassabis (2016). « Mastering the Game of Go with Deep Neural Networks and Tree Search ». In : *Nature* 529.7587, p. 484–489.

Skinner, B.F. (1938). *The Behavior of Organisms : An Experimental Analysis*. The Century Psychology Series. Appleton-Century-Crofts.

Smith, Randall C. et Peter Cheeseman (1986). « On the Representation and Estimation of Spatial Uncertainty ». In : *The International Journal of Robotics Research* 5.4, p. 56–68.

Sutton, R. S., A. G. Barto et R. J. Williams (1992). « Reinforcement learning is direct adaptive optimal control ». In : *IEEE Control Systems* 12.2, p. 19–22.

Sutton, R et Barto A (1998). *Reinforcement Learning : An introduction*. MIT Press.

Sutton, Richard S et Andrew G Barto (1998). « Reinforcement Learning : An Introduction ». In :

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow et Rob Fergus (2013). « Intriguing properties of neural networks ». In : *CoRR* abs/1312.6199.

Thorn, Catherine, Hisham Atallah, Mark Howe et Ann Graybiel (2010). « Differential Dynamics of Activity Changes in Dorsolateral and Dorsomedial Striatal Loops During Learning ». In : *Neuron* 66.5, p. 781–795.

Thorndike, Edward Lee (1911). *Animal intelligence : Experimental studies*. Macmillan.

Thrun, Sebastian et John J Leonard (2008). « Simultaneou 37. Simultaneous Localization and Mapping ». In : *Springer handbook of robotics*, p. 871–889. issn : 1070-9932. arXiv : thereisnot.

Tokic, M (2010). « Adaptive varepsilon-Greedy Exploration in Reinforcement Learning Based on Value Differences ». In : *KI 2010 : Advances in Artificial Intelligence*, p. 203–210.

Tolman, Edward C. « Cognitive maps in rats and men ». In : *The Psychological Review*. 55, p. 189–208.

Tricomi, E M, Bernard W Balleine et John P O'Doherty (2009). « A specific role for posterior dorsolateral striatum in human habit learning ». In : *Eur J Neurosci* 29.11, p. 2225–2232.

Turney, Peter (1995). « Cost-Sensitive Classification : Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm ». In : *Journal of Artificial Intelligence Research* 2, p. 369–409.

Valentin, V. V., A. Dickinson et J. P. O'Doherty (2007). « Determining the neural substrates of goal-directed learning in the human brain. » In : *The Journal of neuroscience : the official journal of the Society for Neuroscience* 27.15, p. 4019–26. issn : 1529-2401.

Viejo, G., M. Khamassi, A. Brovelli et B. Girard (2015). « Modelling choice and reaction time during arbitrary visuomotor learning through the coordination of adaptive working memory and reinforcement learning ». In : *Frontiers in Behavioral Neuroscience* 9, p. 225.

Viejo, Guillaume (2016). « Coordination de systèmes de mémoire : modèles théoriques du comportement animal et humain ». Thèse de doct. Université Pierre et Marie Curie.

Viola, Paul et Michael Jones (2001). « Robust Real-time Object Detection ». In : *International Journal of Computer Vision*.

Watkins, Christopher et Peter Dayan (1992). « Q-Learning ». In : *Machine Learning* 8, p. 179–292.

Werbos, P.J. (1975). *Beyond Regression : New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University.

Wilcoxon, Frank (1945). « Individual Comparisons by Ranking Methods ». In : *Biometrics Bulletin* 1.6, p. 80–83. issn : 00994987.

Wise, Roy A. (2004). « Dopamine, learning and motivation ». In : *Nature Reviews Neuroscience* 5.6, p. 483–494. issn : 1471-003X.

Witten, Ian H. (1977). « An adaptive optimal controller for discrete-time Markov environments ». In : *Information and Control*, p. 286–295.

Wunderlich, K., P. Smittenaar et R. J. Dolan (2012). « Dopamine Enhances Model-Based over Model Free Choice Behavior ». In : 75.3, p. 418–424.

Xu, Zhixiang, Matt Kusner, Minmin Chen et Kilian Q. Weinberger (2013). « Cost-Sensitive Tree of Classifiers ». In : *Journal of Machine Learning Research* 28, p. 133–141. issn : 1938-7228. arXiv : arXiv:1210.2771v3.

Yin, Henry H et Barbara J Knowlton (2006). « The role of the basal ganglia in habit formation ». In : *Nature Reviews* 7.6, p. 464–476. issn : 1471-003X.

Yin, Henry H, Shweta Prasad Mulcare, Monica R F Hilário, Emily Clouse, Terrell Holloway, Margaret I Davis, Anita C Hansson, David M Lovinger et Rui M Costa (2009). « Dynamic reorganization of striatal circuits during the acquisition and consolidation of a skill. » In : *Nature neuroscience* 12.3, p. 333–341. issn : 1097-6256. arXiv : NIHMS150003.

Glossaire

A	AC	*Actor-Critic*
	AR	Apprentissage par Renforcement
C	CNN	*Convolutional Neural Network*
D	DAG	*Direct Acyclic Graph*
	DL	*Deep Learning*, apprentissage profond
	DSNN	*Deep Sequential Neural Network*
G	GRU	*Gated Recurrent Unit*
L	LSTM	*Long Short Term Memory*
M	MDP	*Markov Decision Process*
P	POMDP	*Partially Observed Markov Decision Process*
Q	QL	*Q-learning*
R	RNN	*Recurrent Neural Networks*, réseaux de neurones récurrents
	RPE	*Reward Prediction Error*
S	SARSA	*State Action Reward State Action*
	SLAM	*Simultaneous Localization And Mapping*
U	UCB	*Upper Confidence Bound*

Index