Classification régularisée par la récompense pour l'Apprentissage par Imitation

Bilal Piot, Matthieu Geist, Olivier Pietquin

Classification régularisée par la récompense pour l'Apprentissage par Imitation

Bilal Piot[1,2], Matthieu Geist[1], Olivier Pietquin[1,2]

[1] MaLIS research group (SUPELEC)
2 rue Edouard Belin, Metz, 57070 FRANCE prenom.nom@supelec.fr

[2] UMI 2958 (GeorgiaTech-CNRS)

Résumé : Cet article traite le problème d'Apprentissage par Démonstrations (AD) dans lequel un agent appelé apprenti cherche à apprendre à partir des démonstrations d'un autre agent appelé expert. Pour aborder ce problème assez général, il est commun d'adopter le paradigme des Processus Décisionnels de Markov (PDM) qui est approprié pour les problèmes de prises de décisions séquentielles. Dans la littérature, il y a principalement deux façons de traiter ce problème (en faisant appel aux PDM) qui sont l'Apprentissage par Imitation (AI) où l'apprenti cherche directement à imiter la politique de l'expert et l'Apprentissage par Renforcement Inverse (ARI) où l'apprenti essaye d'apprendre une récompense qui pourrait expliquer la politique de l'expert. Ici, nous introduisons un paradigme inédit, appelé cadre de travail des politiques d'ensembles (*set-policy framework*), pour lequel il y a un lien naturel entre les méthodes d'AI et d'ARI. Ce paradigme permet de dériver des nouveaux algorithmes qui nécessitent uniquement la connaissance de couples état-action experts et d'exemples de transitions du PDM. Des expériences sont réalisés sur un problème qui fait référence (un simulateur de trafic routier) et sur une tâche plus générique (les Garnets) qui permet une comparaison plus générale des algorithmes.
Mots-clés : Apprentissage par Renforcement Inverse, Apprentissage par Imitation.

1 Introduction

Cet article traite le problème d'Apprentissage par Démonstrations (AD) dans lequel un agent, appelé apprenti, essaie d'apprendre à partir de démonstrations d'un agent appelé expert. Ce problème peut être traité comme un problème d'Apprentissage par Imitation (Abbeel & Ng, 2004). Dans ce paradigme, l'apprenti observe l'expert qui agit optimalement, vis à vis d'une fonction de récompense inconnue, dans un Processus Décisionnel de Markov (PDM). Le but de l'apprenti, à partir d'observations de la politique experte, est d'apprendre cette politique ou du moins une politique qui serait aussi bonne que celle de l'expert relativement à la récompense inconnue. La première idée qui vient à l'esprit afin de tenter de résoudre ce problème est de le modéliser comme un problème de pure classification (Pomerleau, 1989; Atkeson & Schaal, 1997; Langford & Zadrozny, 2005). Cependant, d'autres auteurs (Abbeel & Ng, 2004; Syed *et al.*, 2008) se sont aussi inspirés du paradigme d'Apprentissage par Renforcement Inverse (ARI) pour produire des algorithmes qui tentent de résoudre le problème d'AI. L'ARI (Russell, 1998; Ng *et al.*, 2000) est relié de très près à l'AI. La seule différence est que les algorithmes d'ARI ont pour sortie une récompense qui cherche à expliquer la politique de l'expert plutôt qu'une politique. L'idée centrale derrière le concept d'ARI est que la fonction de récompense est la représentation la plus compacte de la tâche. Ici, après avoir introduit la notion de PDM et diverses notations dans la Sec. 2, nous construisons, dans la Sec. 3, le cadre de travail des politiques d'ensembles dans lequel nous pourrons établir un lien formel entre les méthodes d'ARI et d'AI. Ce nouveau cadre de travail permet de dériver des algorithmes pour le problème d'AI. En effet, dans la Sec. 4, le problème d'AI sera modélisé comme un problème d'Apprentissage Supervisé (AS) pour lequel il ne sera pas nécessaire de résoudre itérativement des PDM ou de connaître entièrement la dynamique du PDM, ce qui est souvent nécessaire pour la plupart des algorithmes d'AI (voir Sec. 5). L' algorithme obtenu a besoin uniquement d'un ensemble de couples état-action experts et d'un ensemble d'exemples de transitions du PDM. De plus, une version idéalisée de l'algorithme peut être implémentée quand la dynamique du PDM est entièrement connue. Finalement, notre algorithme est comparé à une méthode de classification et à deux récents algorithmes d'ARI, SCIRL (Klein *et al.*, 2012) et Relative Entropy IRL (Boularias *et al.*, 2011). Les expériences sont réalisées sur une tâche générique (les Garnets) et sur un simulateur de trafic routier.

2 Contexte et notations

Tout d'abord, nous introduisons des notations à caractère général qui seront utiles tout au long du papier. Soient $(\mathbb{R}, |.|)$ l'espace des réels muni de sa norme canonique et X un ensemble fini, $\mathbb{R}^X$ est l'ensemble des fonctions de X vers $\mathbb{R}$. L'ensemble des parties de X est noté $\mathcal{P}(X)$. L'ensemble des mesures de probabilité sur X est noté Δ_X. Soit Y un ensemble fini, Δ_X^Y est l'ensemble de fonctions de Y vers Δ_X. Soit $\zeta \in \Delta_X^Y$ et $y \in Y$, $\zeta(y) \in \Delta_X$, peut être vue comme une probabilité conditionnelle sachant y, est aussi notée $\zeta(.|y)$ et $\forall x \in X, \zeta(x|y) = [\zeta(y)](x)$. Soient $\alpha, \beta \in \mathbb{R}^X : \alpha \leq \beta \Leftrightarrow \forall x \in X, \alpha(x) \leq \beta(x)$. Le support de α est noté $\mathrm{Supp}(\alpha)$. Soit $p \in \mathbb{N}^*$, nous définissons la $\mathbf{L}_p$-norme de α, notée $\|\alpha\|_p$, par : $\|\alpha\|_p = (\sum_{x \in X} |\alpha(x)|^p)^{\frac{1}{p}}$. Soit $x \in X$, $x \sim \nu$ signifie que x est tiré selon ν et $\mathbb{E}_\nu[\alpha] = \sum_{x \in X} \nu(x)\alpha(x)$ est l'espérance de α sous ν.

Maintenant, nous introduisons des notations relatives aux PDM qui vont nous permettre de construire le cadre de travail des politiques d'ensembles et de définir l'AI et l'ARI dans ce nouveau cadre (voir Sec. 3). Un PDM fini modélise les interactions d'un agent agissant dans un environnement et est représenté par un quintuplet $M = \{S, A, R, \gamma, P\}$. L'ensemble des états $S = \{s_i\}_{1 \leq i \leq N_S}$ représente les états de l'environnement, l'ensemble des actions $A = \{a_i\}_{1 \leq i \leq N_A}$ représente les actions que peut prendre l'agent, $R \in \mathbb{R}^{S \times A}$ est la fonction de récompense qui est la représentation locale du bénéfice de faire l'action a dans l'état s, $\gamma \in]0, 1[$ est un facteur d'actualisation et $P \in \Delta_S^{S \times A}$ est la dynamique markovienne du PDM. La dynamique P du PDM encode l'évolution de l'environnement : elle donne $P(s'|s, a)$ qui est la probabilité d'atteindre s' en choisissant l'action a dans l'état s. Une politique markovienne et stationnaire π est un élément de Δ_A^S et définit le comportement de l'agent. Pour un état s et une action a, la politique π indique la probabilité de choisir l'action a sachant l'état s. L'ensemble des politiques markoviennes et stationnaires est noté $\Pi_{MS} = \Delta_A^S$. Ici, nous choisissons le cadre des politiques markoviennes et stationnaires (à la place du cadre plus standard mais aussi plus restrictif qu'est celui des politiques déterministes) car l'expert à imiter peut être une personne et dès lors choisir différentes actions dans un même état.

Pour un PDM donné $M = \{S, A, R, \gamma, P\}$ et une politique donnée $\pi \in \Pi_{MS}$, la fonction de valeur $V_R^\pi \in \mathbb{R}^S$ est définie comme :

$$\forall s \in S, V_R^\pi(s) = \mathbb{E}_s^\pi[\sum_{t=0}^{+\infty} \gamma^t R(s_t, a_t)],$$

où $\mathbb{E}_s^\pi$ est l'espérance sur les trajectoires admissibles $(s_0, a_0, s_1, \dots)$ obtenues en exécutant la politique π et en partant de $s_0 = s$. La fonction de valeur V_R^π quantifie le comportement de l'agent représenté par la politique π relativement à la récompense R et cela sur le long terme (horizon infini). De plus, la fonction $V_R^* \in \mathbb{R}^S$, définie par :

$$V_R^* = \sup_{\pi \in \Pi_{MS}} V_R^\pi,$$

est appelée fonction de valeur optimale. Une politique $\pi \in \Pi_{MS}$ telle que $V_R^\pi = V_R^*$ est dite optimale et existe toujours pour un PDM fini (Puterman, 1994, Ch. 6). Le but de la programmation dynamique est de trouver une politique optimale pour un PDM M donné. Un outil souvent utile, pour une politique donnée $\pi \in \Pi_{MS}$, est la fonction de qualité $Q_R^\pi \in \mathbb{R}^{S \times A}$:

$$\forall s \in S, \forall a \in A, Q_R^\pi(s, a) = R(s, a) + \gamma \mathbb{E}_{P(.|s,a)}[V_R^\pi].$$

Elle représente la qualité du comportement de l'agent si ce dernier choisit l'action a dans l'état s puis applique la politique π ensuite. De plus, la fonction de qualité optimale $Q_R^* \in \mathbb{R}^{S \times A}$ est telle que :

$$Q_R^* = \sup_{\pi \in \Pi_{MS}} Q_R^\pi.$$

Le théorème suivant, utile pour la suite, caractérise les politiques optimales via la fonction de qualité (voir aussi Melo *et al.*, 2010).

Théorème 1 (Caractérisation des politiques optimales)
Pour un PDM donné $M = \{S, A, R, \gamma, P\}$, une politique $\pi \in \Pi_{MS}$ est dite optimale si et seulement si :

$$V_R^\pi = V_R^* \Leftrightarrow \forall s \in S, \mathrm{Supp}(\pi(.|s)) \subset \operatorname*{argmax}_{a \in A} Q_R^*(s, a).$$

Le Th. 1 est un corollaire des théorèmes 6.2.6 et 6.2.7 de Puterman (1994, Ch. 6). Dans le reste du papier, R peut varier, mais S, A, γ et P sont fixés. Le quadruplet $M \backslash R = \{S, A, \gamma, P\}$ est un PDM sans récompense et pour chaque fonction de récompense $R \in \mathbb{R}^{S \times A}$ le PDM $M_R = \{S, A, R, \gamma, P\}$ est associé.

3 Le cadre de travail des politiques d'ensembles

Cette section introduit le cadre de travail de politiques d'ensembles qui est utilisé pour établir un lien formel entre ARI et AI. Ce cadre de travail est aussi utile pour dériver de nouveaux algorithmes à la fois d'ARI et d'AI. Le Th. 1 montre qu'une politique stationnaire et markovienne est optimale si et seulement si, pour chaque état s, elle choisit une action dans l'ensemble $\text{argmax}_{a\in A}[Q_R^*(s,a)]$. Ainsi, pour caractériser l'optimalité d'une politique $\pi \in \Pi_{MS}$, l'information nécessaire et suffisante est le support $\text{Supp}(\pi(.|s)) \subset A$, qui est un ensemble d'actions fini et non vide. Ayant remarqué cela, il est assez naturel de considérer les fonctions, appelées politiques d'ensembles, qui associent les états aux ensembles d'actions non-vides.

Définition 1 (Politiques d'ensembles)
Une politique d'ensembles $\overline{\pi}$ est un élément de $(\mathcal{P}(A)\backslash\emptyset)^S$. L'ensemble des politique d'ensembles est aussi noté $\overline{\Pi}$. Il contient $(2^{N_A}-1)^{N_S}$ éléments.

Définition 2 (L'inclusion pour les politiques d'ensembles)
Soient $\overline{\pi}_1$ et $\overline{\pi}_2$ deux politiques d'ensembles, $\overline{\pi}_1 \subset \overline{\pi}_2$ si : $\forall s \in S, \overline{\pi}_1(s) \subset \overline{\pi}_2(s)$. De plus, $\overline{\pi}_1 = \overline{\pi}_2$ si $\overline{\pi}_1 \subset \overline{\pi}_2$ et $\overline{\pi}_2 \subset \overline{\pi}_1$.

Pour chaque politique $\pi \in \Pi_{MS}$, nous associons une politique d'ensembles $\overline{\pi} \in \overline{\Pi}$ appelé *politique d'ensembles associée à π* :

Définition 3 (Politique d'ensembles associée)
Soit $\pi \in \Pi_{MS}$, $\overline{\pi} \in \overline{\Pi}$ est appelée une politique d'ensembles associée à π et est telle que : $\forall s \in S, \text{Supp}(\pi(.|s)) = \overline{\pi}(s)$. La politique d'ensembles associée à $\pi \in \Pi_{MS}$ indique, pour chaque état s, l'ensemble d'actions qui peut être choisi par π.

De plus, pour chaque M_R, nous associons une politique d'ensembles appelée *la politique d'ensembles optimale générée par M_R* :

Définition 4 (Politique d'ensembles optimale générée)
Soit $R \in \mathbb{R}^{S\times A}$, $\overline{\pi}_R^* \in \overline{\Pi}$, appelée la politique d'ensembles optimale générée par M_R, est telle que : $\forall s \in S, \overline{\pi}_R^*(s) = \text{argmax}_{a\in A}[Q_R^*(s,a)]$.

La politique d'ensembles optimale générée par M_R indique, pour chaque état, l'ensemble d'actions optimales qu'il est possible de choisir pour optimiser la fonction de récompense R. Grace aux Defs. 3 et 4, le Th. 1 peut se réécrire comme : $V_R^\pi = V_R^* \Leftrightarrow \overline{\pi} \subset \overline{\pi}_R^*$. Désormais, nous disposons des outils pour définir plus précisément l'ARI et l'AI dans le cadre de travail des politiques d'ensembles.

Originellement, le problème de l'ARI (Russell, 1998) consiste à trouver une fonction de récompense qui puisse expliquer le comportement observé de l'expert. Dans le cadre de travail proposé, cela signifie qu'à partir de quelques observations de la politique experte π_E, une fonction de récompense R doit être apprise telle que *la politique d'ensembles optimale générée par M_R*, $\overline{\pi}_R^*$, puisse expliquer la politique π_E représentée par *la politique d'ensembles associée à π_E*, $\overline{\pi}_E$. Plus formellement :

Définition 5 (L'ARI dans le cadre des politiques d'ensembles)
Soit $M\backslash R = \{S, A, \gamma, P\}$ et des observations de la politique experte $\pi_E \in \Pi_{MS}$, trouver une fonction de récompense $R \in \mathbb{R}^{S\times A}$ telle que : $\overline{\pi}_R^* = \overline{\pi}_E$.

Il est intéressant de noter que trouver une fonction de récompense R telle que $\overline{\pi}_R^* = \overline{\pi}_E$ implique, via Th. 1, que $V_R^{\pi_E} = V_R^*$ et garantit que toutes les actions optimales sont effectivement prises par l'expert. De plus, pour une politique d'ensembles donnée $\overline{\pi} \in \overline{\Pi}$, considérons l'ensemble :

$$C_{\overline{\pi}} = \{R \in \mathbb{R}^{S\times A}, \overline{\pi} = \overline{\pi}_R^*\}.$$

Par définition, résoudre le problème d'ARI dans le cadre de travail des politiques d'ensembles consiste à trouver un élément de $C_{\overline{\pi}_E}$. La caractérisation de l'ensemble des solutions du problème d'ARI a déjà été faite pour les politiques déterministes par Ng *et al.* (2000) et pour les politiques stationnaires et markoviennes par Melo *et al.* (2010). Ces caractérisations correspondent bien au fait d'appartenir à $C_{\overline{\pi}_E}$.

L'AI essaie d'estimer la politique π_E avec quelques observations de cette politique. Cela peut être fait par imitation (par exemple Atkeson & Schaal, 1997) ou par une approche basée sur l'ARI (par exemple Abbeel

& Ng, 2004). Dans le cadre de travail des politiques d'ensembles, le problème d'AI consiste, via quelques exemples de la politique experte π_E, de trouver la politique d'ensembles $\overline{\pi}_E$, c'est-à-dire toutes les actions que l'expert peut prendre dans un état. Plus formellement :

Définition 6 (L'AI dans le cadre des politiques d'ensembles)
Soit $M \backslash R = \{S, A, \gamma, P\}$ et des observations de la politique experte $\pi_E \in \Pi_{MS}$, trouver la politique d'ensembles $\overline{\pi}_E$.

Pour une politique donnée $\overline{\pi} \in \overline{\Pi}$, considérons :

$$H_{\overline{\pi}} = \{Q \in \mathbb{R}^{S \times A}, \forall s \in S, \operatorname*{argmax}_{a \in A} Q(s,a) = \overline{\pi}(s)\}.$$

Par définition, résoudre le problème d'AI dans le cadre de travail des politiques d'ensembles consiste à trouver un élément de $H_{\overline{\pi}_E}$.

Maintenant, il est possible d'établir un lien formel entre l'AI et l'ARI dans le cadre qui vient d'être mis en place. Un outil important est l'opérateur J^*, appelé opérateur de Bellman inverse (voir aussi Melo *et al.*, 2010), qui est une fonction de $\mathbb{R}^{S \times A}$ vers $\mathbb{R}^{S \times A}$:

$$\forall Q \in \mathbb{R}^{S \times A}, \forall (s,a) \in S \times A : [J^*Q](s,a) = Q(s,a) - \gamma \sum_{s' \in S} P(s'|s,a) \max_{b \in A} [Q(s',b)].$$

L'opérateur J^* représente la relation biunivoque existant entre les fonctions optimales de qualités et leurs récompenses. Voici les principales propriétés de cet opérateur :

Théorème 2 (Propriétés de J^*)
L'opérateur inverse de Bellman J^* est une bijection de $\mathbb{R}^{S \times A}$ vers $\mathbb{R}^{S \times A}$ et nous avons :

$$\forall Q \in \mathbb{R}^{S \times A}, Q = Q^*_R, \text{ avec } R = J^*Q.$$

Il est assez clair, via J^*, de voir que :

$$\begin{aligned} J^*(H_{\overline{\pi}}) &= \{R \in \mathbb{R}^{S \times A}, \forall s \in S, \operatorname*{argmax}_{a \in A} [Q^*_R(s,a)] = \overline{\pi}(s)\} \\ &= \{R \in \mathbb{R}^{S \times A}, \overline{\pi}^*_R = \overline{\pi}\} = C_{\overline{\pi}}. \end{aligned}$$

Donc, pour chaque politique d'ensembles $\overline{\pi}$, l'ensemble de solution du problème d'ARI, $C_{\overline{\pi}}$, est l'image de l'ensemble des solutions du problème d'AI, $H_{\overline{\pi}}$, par l'opérateur J^*. Ainsi, un lien formel a été établi entre les deux problèmes : si nous sommes capables d'avoir J^*, alors résoudre le problème d'AI est équivalent à résoudre le problème d'ARI. Contrairement à $C_{\overline{\pi}}$, il est facile de caractériser $H_{\overline{\pi}}$: $Q \in H_{\overline{\pi}}$ si et seulement si

$$\forall s \in S, \forall a \in \overline{\pi}(s), \begin{cases} \max_{b \in \overline{\pi}(s)} Q(s,b) = Q(s,a), \\ \max_{b \in \overline{\pi}(s)} Q(s,b) > \max_{b \notin \overline{\pi}(s)} Q(s,b). \end{cases} \quad (1)$$

Cette caractérisation ne dépend pas de P mais seulement de la politique d'ensembles $\overline{\pi}$. Dans le reste du papier, nous cherchons une solution Q du problème d'AI, gardant à l'esprit que trouver une solution R du problème d'ARI est possible en appliquant J^* à Q. Dans la section suivante, un algorithme sera proposé qui cherche à trouver un élément de $H_{\overline{\pi}_E}$ grâce aux observations du comportement de l'expert. En général, ces observations sont un ensemble de couples état-action experts $D_E = (s_i, a_i)_{\{1 \leq i \leq N\}}$ où $N \in \mathbb{N}^*$ et (s_i, a_i) est tel que $s_i \sim \nu_E \in \Delta_S$ et $a_i \sim \pi_E(.|s_i)$. On ne donne aucune interprétation spécifique à la mesure de probabilité ν_E mais elle peut être, si elle existe, la mesure de probabilité stationnaire induite par la politique experte π_E.

4 Un algorithme pour le problème d'AI

Dans cette section, le problème d'AI est modélisé comme un problème d'Apprentissage Supervisé (AS). Cela a déjà été fait dans le passé et même justifié par une analyse de Syed & Schapire (2010), où le problème d'AI était vu comme un problème de classification (toutefois cette analyse est dans le cas des PDM à horizon fini). Cependant, pour Syed & Schapire (2010), le choix de l'algorithme de classification n'est

pas central, alors qu'en pratique ce choix a une importance majeure pour obtenir de bons résultats (voir les expériences en Sec.6). Il y a trois éléments à définir pour un problème d'AS : l'ensemble de données à disposition, la fonction de risque et l'espace d'hypothèses. Ici, l'ensemble des données est l'ensemble D_E et il est fixé, cependant la fonction de risque et l'espace d'hypothèses restent à être choisis afin d'obtenir un algorithme pratique avec une bonne performance. Plus précisément, imposer certaines contraintes spécifiques sur la fonction de récompense nous amènera au choix d'un espace d'hypothèses et à un algorithme qui pourra être vu comme une classification à large-marge régularisée par un terme dépendant de la norme de la récompense. Ainsi, le concept d'ARI permet de trouver un espace d'hypothèses adapté afin de résoudre le problème d'AI vu comme un problème d'AS. De plus, notre algorithme n'a pas besoin de résoudre itérativement des PDM, contrairement à la plupart des algorithmes d'AI, et a uniquement besoin d'un ensemble de données expertes et d'un ensemble d'exemples de transitions du PDM (pour en quelque sorte estimer la dynamique du PDM).

4.1 L'AI vu comme un problème d'AS

Dans la section précédente, le problème d'AI se ramenait à trouver un élément de $H_{\overline{\pi}_E}$. Maintenant, nous allons voir que minimiser une certaine fonction de risque correspond à trouver un élément de $H_{\overline{\pi}_E}$. Tout d'abord, nous rappelons que la caractérisation de $H_{\overline{\pi}_E}$ est faite via l' Eq. (1). Si nous définissons l'ensemble suivant :

$$\mathfrak{L}_{\overline{\pi}_E} = \{l \in \mathbb{R}_+^{S\times A}, \forall s \in S, \forall a \in \overline{\pi}_E(s), \forall b \notin \overline{\pi}_E(s), l(s,b) > l(s,a) = 0\},$$

alors, via l'Eq. (1), $Q \in H_{\overline{\pi}_E}$ si et seulement si :

$$\exists l \in \mathfrak{L}_{\overline{\pi}_E}, \forall s \in S, \forall a \in \overline{\pi}_E(s), Q(s,a) = \max_{b\in A}[Q(s,b) + l(s,b)]. \tag{2}$$

L'ensemble $\mathfrak{L}_{\overline{\pi}_E}$ est un ensemble de fonctions de marge qui permettent de faire la différence entre actions optimales et non-optimales. De plus, si nous définissons l'ensemble suivant :

$$\mu_{\overline{\pi}_E} = \{\mu \in \Delta_{S\times A}, \forall s \in S, \forall a \in \overline{\pi}_E(s), \forall b \notin \overline{\pi}_E(s), \mu(s,b) = 0 \text{ et } \mu(s,a) > 0\},$$

alors , via l'Eq. (2), $Q \in H_{\overline{\pi}_E}$ si et seulement si :

$$\exists l \in \mathfrak{L}_{\overline{\pi}_E}, \exists \mu \in \mu_{\overline{\pi}_E}, \sum_{(s,a)\in S\times A} (\max_{b\in A}[Q(s,b) + l(s,b)] - Q(s,a))\mu(s,a) = 0. \tag{3}$$

L'ensemble $\mu_{\overline{\pi}_E}$ est l'ensemble des mesures de probabilités dont le support est celui des actions optimales (*i.e.*, expertes). Il y a donc un moyen numérique, donné par l'Eq. (3), de décrire le fait que Q appartienne à $H_{\overline{\pi}_E}$. Pour $\mu \in \Delta_{S\times A}$ et $l \in \mathbb{R}_+^{S\times A}$, soient $I_{\mu,l}$ et M_l tels que :

$$\forall Q \in \mathbb{R}^{S\times A}, \forall (s,a) \in S \times A, \begin{cases} M_l(s,a,Q) = \max_{b\in A}[Q(s,b) + l(s,b)] - Q(s,a), \\ I_{\mu,l}(Q) = \sum_{s\in S, a\in A} M_l(s,a,Q)\mu(s,a). \end{cases}$$

Alors, l'Eq. (3) dit que $Q \in H_{\overline{\pi}_E}$ si et seulement si : $\exists l \in \mathfrak{L}_{\overline{\pi}_E}, \exists \mu \in \mu_{\overline{\pi}_E}, I_{\mu,l}(Q) = 0$. Le terme $I_{\mu,l}$ peut être vu comme une fonction de risque et le terme M_l comme une fonction de perte (cf. Vapnik, 1998) et les seules fonctions telles que $I_{\mu,l}(Q) = 0$ sont des éléments de $H_{\overline{\pi}_E}$ (via l'Eq.(3)).

Ainsi, pour trouver un élément de $H_{\overline{\pi}_E}$, il suffit de choisir $l \in \mathfrak{L}_{\overline{\pi}_E}$, $\mu \in \mu_{\overline{\pi}_E}$, un espace d'hypothèses $\mathfrak{Q} \subset \mathbb{R}^{S\times A}$ où le risque minimal $\min_{\tilde{Q}} I_{\mu,l}(\tilde{Q})$ est nul et trouver un minimiseur de $I_{\mu,l}$. C'est donc un problème d'AS. Cependant, en pratique, nous ne disposons pas de $\mu \in \mu_{\overline{\pi}_E}$, mais seulement de l'ensemble de couples état-action experts $D_E = (s_i, a_i)_{\{1\leq i\leq N\}}$ où $s_i \sim \nu_E \in \Delta_S$ et $a_i \sim \pi_E(.|s_i)$. De plus, l'accès à un élément de $\mathfrak{L}_{\overline{\pi}_E}$ est aussi impossible, puisque nous ne connaissons pas $\overline{\pi}_E$ (comment régler ce problème sera expliqué par la suite). Pour le moment, nous nous concentrons sur le problème de distribution. Si nous définissons $\mu_E(s,a) = \nu_E(s)\pi_E(s|a)$, alors $\mu_E \in \mu_{\overline{\pi}_E}$. Ainsi, nous avons l'ensemble $D_E = (s_i, a_i)_{\{1\leq i\leq N\}}$ où $(s_i, a_i) \sim \mu_E$. L'idée naturelle, comme nous ne disposons pas directement de μ_E pour trouver le minimiseur de $I_{\mu_E,l}$, est de trouver un minimiseur du risque empirique $I^N_{\mu_E,l}$:

$$\forall Q \in \mathbb{R}^{S\times A}, I^N_{\mu_E,l}(Q) = \frac{1}{N} \sum_{1\leq i\leq N} M_l(s_i, a_i, Q) \geq 0,$$

sur un espace d'hypothèses $\mathfrak{Q}$ bien choisi. L'idée, pour trouver cet espace d'hypothèses, est d'utiliser le principe de Minimisation du Risque Structurel Étendu (MRSE) (Evgeniou *et al.*, 2000). Ce dernier consiste à choisir une famille d'espaces d'hypothèses imbriqués, comme $\mathfrak{F}_Q = (\mathfrak{Q}_\xi = \{Q \in \mathbb{R}^{S\times A}, \|Q\|_2 \leq \xi\})_{\{\xi\in\mathbb{R}_+\}}$, et de résoudre, pour chaque espace de la famille $\mathfrak{F}_Q$, le problème suivant :

$$\begin{aligned} &\text{minimiser } I^N_{\mu_E,l}(Q), \\ &\text{sujet à } \|Q\|_2 \leq \xi. \end{aligned}$$

Pour résoudre ce problème de minimisation, nous minimisons $I^N_{\mu_E,l}(Q) + \lambda_\xi(\|Q\|_2^2 - \xi^2)$ par rapport à Q et nous le maximisons par rapport au multiplicateur de Lagrange λ_ξ. Ainsi, nous obtenons l'ensemble de couples solutions $(\lambda^*_\xi, Q^*_\xi)_{\{\xi\in\mathbb{R}_+\}}$. Puis, le principe MRSE nous dit qu'il y a un λ^*_ξ particulier, noté λ^*_Q, qui réalise le meilleur compromis entre l'erreur empirique et la capacité de généralisation de l'espace $\mathfrak{Q}_\xi$, qui est meilleure si ξ est petit. Si nous avons λ^*_Q, nous pourrions simplement minimiser le critère suivant : $I^N_{\mu_E,l}(Q)+\lambda^*_Q\|Q\|_2^2$. En pratique, λ^*_Q peut être trouvé par une recherche quadrillée, par exemple. Cependant, pour notre algorithme, nous considérons une autre famille d'espaces d'hypothèses imbriqués :

$$\mathfrak{F}_R = (\tilde{\mathfrak{Q}}_\xi = \{Q \in \mathbb{R}^{S\times A}, \|J^*Q\|_2 = \|R_Q\|_2 \leq \xi\})_{\{\xi\in\mathbb{R}_+\}},$$

où R_Q est une fonction de récompense telle que $R_Q = J^*Q$. La famille $\mathfrak{F}_R$ contient des espaces d'hypothèses qui contraignent la norme de R_Q associée à Q. Comme la récompense est la représentation la plus succincte de la tâche à réaliser, nous pouvons nous attendre à ce que le principe MRSE appliquée à la famille $\mathfrak{F}_R$ trouve un espace d'états $\tilde{\mathfrak{Q}}_\xi$ avec un petit ξ et une bonne capacité de généralisation. Le paramètre réalisant le meilleur compromis entre erreur empirique et capacité de généralisation est notée λ^*_R pour la famille $\mathfrak{F}_R$. Pour ce paramètre particulier, notre problème d'AS devient la résolution du problème de minimisation sans contraintes suivant :

$$I^N_{\mu_E,l}(Q) + \lambda^*_R\|R_Q\|_2^2.$$

4.2 Les algorithmes en pratique

Par souci de clarté, une représentation tabulaire de Q est choisie (le cas non-tabulaire est traité dans la Sec. 4.3). Comme vu précédemment, l'ensemble de couples état-action experts D_E et une fonction de marge $l \in \mathfrak{L}_{\overline{\pi}_E}$ sont donnés, et nous voulons réaliser la minimisation suivante :

$$\min_{Q\in\mathbb{R}^{S\times A}} \frac{1}{N}\sum_{1\leq i\leq N} M_l(s_i,a_i,Q) + \frac{\lambda}{N_S N_A}\|R_Q\|_2^2, \tag{4}$$

où λ est un paramètre de compromis qui est choisie par une recherche quadrillée. La division par le facteur $N_S N_A$ est réalisée afin de rendre le paramètre λ moins dépendant de la taille du PDM. Il y a toujours un élément dans l'Eq. (4) dont nous ne disposons pas. En effet, l nécessite la connaissance de $\overline{\pi}_E$ pour être construite. Cependant, si nous considérons l telle que :

$$\forall s \in S, \forall a \in \overline{\pi}_E(s), \forall b \notin \overline{\pi}_E(s), l(s,b) = 1 > l(s,a) = 0,$$

alors il est possible de construire une fonction $\hat{l}$ similaire à l avec seulement l'ensemble de données D_E. En effet, définissons $\hat{l}$ telle que :

$$\forall (s,a) \in S\times A, \text{ si } \exists 1 \leq i \leq N, (s,a) = (s_i,a_i) \text{ alors } \hat{l}(s,a) = 0 \text{ sinon } \hat{l}(s,a) = 1.$$

La marge $\hat{l}$ est alors telle que : $\forall 1 \leq i \leq N, l(s_i,a_i) = \hat{l}(s_i,a_i)$. De plus, si l'expert est déterministe, il est clair que : $\forall 1 \leq i \leq N, \forall a \in A, l(s_i,a) = \hat{l}(s_i,a)$. Donc, dans le cas déterministe, nous sommes sûrs que :

$$\forall Q \in \mathbb{R}^{S\times A}, \forall 1 \leq i \leq N, M_l(s_i,a_i,Q) = M_{\hat{l}}(s_i,a_i,Q).$$

Si l'expert n'est pas déterministe, $\hat{l}$ peut être considérée comme une bonne approximation de l car $\hat{l}$ converge en probabilité vers l quand le nombre d'exemples augmente. Donc, en pratique, l'algorithme consiste à minimiser :

$$\mathfrak{L}_1(Q) = \frac{1}{N}\sum_{1\leq i\leq N} M_{\hat{l}}(s_i,a_i,Q) + \frac{\lambda}{N_S N_A}\|R_Q\|_2^2.$$

Cet algorithme est seulement valable quand la dynamique P du PDM est disponible car $R_Q = J^*Q$. Si la dynamique P est souvent inconnue, il peut toutefois être possible de disposer d'exemples de transitions du PDM ou de pouvoir générer ces exemples. Donc, nous supposons l'accès à un ensemble de données supplémentaires, constitué d'exemples de transitions du PDM : $D_P = (s_j, a_j, s'_j)_{\{1 \leq j \leq N'\}}$ où $(s_j, a_j) \sim \mu_P \in \Delta_{S \times A}$ et $s'_j \sim P(.|s_j, a_j)$. Ici encore, il n'y a pas d'hypothèse faite a priori sur μ_P. Soit $\hat{\mathcal{R}}_Q$ un vecteur colonne de taille N' tel que $\forall Q \in \mathbb{R}^{S \times A}$:

$$\forall 1 \leq j \leq N', \hat{\mathcal{R}}_Q(j) = Q(s_j, a_j) - \gamma \max_{a \in A} Q(s'_j, a).$$

Une façon simple d'adapter l'algorithme est donc :

$$\mathfrak{L}_2(Q) = \frac{1}{N} \sum_{1 \leq i \leq N} M_{\hat{l}}(s_i, a_i, Q) + \frac{\lambda}{N'} \|\hat{\mathcal{R}}_Q\|_2^2.$$

Pour résumer, nous disposons de deux algorithmes : le premier algorithme suppose la connaissance de la dynamique P du PDM et l'accès à un ensemble de couples état-action experts D_E. Il réalise ensuite la minimisation suivante : $\min_{Q \in \mathbb{R}^{S \times A}} \mathfrak{L}_1(Q)$. Le second algorithme suppose l'accès à un ensemble d'exemples de transitions du PDM D_P et l'accès à l'ensemble D_E. Il réalise ensuite la minimisation suivante : $\min_{Q \in \mathbb{R}^{S \times A}} \mathfrak{L}_2(Q)$. Un algorithme de sous-gradient est choisi pour réaliser la minimisation (voir Shor *et al.*, 1985) : $Q^{i+1} = Q^i - \frac{\delta_i \partial_{Q^i} \mathfrak{L}_k}{\|\partial_{Q^i} \mathfrak{L}_k\|_2^2}$, où $\partial_{Q^i} \mathfrak{L}_k$ est le sous-gradient de $\mathfrak{L}_k$, avec $k = \{1, 2\}$, sur Q^i et δ_i est un taux d'apprentissage tel que : $\sum_{i \in \mathbb{N}} \delta_i = +\infty$ et $\sum_{i \in \mathbb{N}} \delta_i^2 < +\infty$.

Une interprétation de notre algorithme comme une méthode de classification régularisée est aussi possible. En effet, si nous considérons la famille $\mathfrak{F}_Q$ au lieu de $\mathfrak{F}_R$, par les mêmes arguments cela nous aurait mené à la minimisation du critère suivant :

$$\mathfrak{L}_0(Q) = \frac{1}{N} \sum_{1 \leq i \leq N} M_l(s_i, a_i, Q) + \frac{\lambda}{N_S N_A} \|Q\|_2^2.$$

La minimisation de $\mathfrak{L}_0$ peut être vu comme un algorithme de classification à large marge structurée (Taskar *et al.*, 2005). Cette approche est un algorithme de classification pure. La seule différence entre le critère $\mathfrak{L}_0$ et $\mathfrak{L}_2$ est le terme de régularisation $\frac{\lambda}{N'} \|\hat{\mathcal{R}}_Q\|_2^2$. C'est pourquoi, nous pouvons voir notre algorithme comme une classification régularisée, par un terme contenant d'une certaine manière la dynamique du PDM. Notre algorithme (minimisation de $\mathfrak{L}_2$ via une descente de sous-gradient) est nommée *Regularized Classification for Apprenticeship Learning* (RCAL). L'algorithme qui consiste à minimiser le critère $\mathfrak{L}_1$ (via une descente de sous-gradient) est appelé *Model-Based Regularized Classification for AL* (MBRCAL) car il nécessite la connaissance de la dynamique.

4.3 Le problème dans le cas non-tabulaire

Dans certaines applications, par exemple quand l'espace d'état est trop grand, une représentation tabulaire de Q est impossible. Une représentation assez générale de Q est de choisir $Q \in \mathfrak{H} \subset \mathbb{R}^{S \times A}$ où $\mathfrak{H}$ est un ensemble de fonctions. Dans ce cas, notre algorithme devient simplement :

$$\min_{Q \in \mathfrak{H}} \mathfrak{L}_2(Q).$$

La seule différence avec le cas tabulaire est que la minimisation est réalisée sur $\mathfrak{H}$ au lieu de $\mathbb{R}^{S \times A}$. Cependant, toutes les représentations de Q ne sont pas adaptées à notre algorithme puisqu'il y a une descente de sous-gradient à implémenter. Un choix naturel est une représentation linéaire de Q via q fonctions de base (ou attributs vectoriels) $(\phi_i)_{\{1 \leq i \leq q\}}$ où $q \in \mathbb{N}^*$ et $\phi_i \in \mathbb{R}^{S \times A}$. Choisir un ensemble d'attributs vectoriels, adaptés au problème considéré, est un problème important en apprentissage-machine, ici nous supposons que ce choix est fait par l'utilisateur. Puis, pour chaque vecteur colonne de paramètres $\theta = (\theta_i)_{\{1 \leq i \leq q\}} \in \mathbb{R}^q$, nous définissons $Q_\theta = \sum_{1 \leq i \leq q} \theta_i \phi_i$. Nous travaillons donc sur l'espace $\mathfrak{H} = \{Q_\theta, \theta \in \mathbb{R}^q\}$. De plus, pour chaque couple $(s, a) \in \mathbb{R}^{S \times A}$, il est possible de définir le vecteur colonne $\Phi(s, a) = (\Phi(s, a)_i = \phi_i(s, a))_{\{1 \leq i \leq q\}} \in \mathbb{R}^q$. Par conséquent, nous avons :

$$\forall (s, a) \in \mathbb{R}^{S \times A}, Q_\theta(s, a) = \theta^T \Phi(s, a),$$

où θ^T est le transposé de θ. Maintenant, nous donnons le pseudo-code de notre algorithme RCAL pour une représentation linéaire de Q. Comme nous souhaitons minimiser $\mathfrak{L}_2(Q_\theta)$ via une descente de sous-gradient, nous avons besoin de calculer $\partial_\theta \mathfrak{L}_2(Q_\theta)$:

$$\partial_\theta \mathfrak{L}_2(Q_\theta) = \frac{1}{N} \sum_{1 \leq i \leq N} \partial_\theta M_{\hat{l}}(s_i, a_i, Q_\theta) + \frac{\lambda}{N'} \sum_{1 \leq j \leq N'} \partial_\theta |\hat{R}_{Q_\theta}(j)|^2,$$

$$\begin{cases} \text{où } \partial_\theta M_{\hat{l}}(s_i, a_i, Q_\theta) & = \Phi(s_i, a_i^*) - \Phi(s_i, a_i), \\ \text{et } \partial_\theta |\hat{R}_{Q_\theta}(j)|^2 & = \partial_\theta |Q_\theta(s_j, a_j) - \gamma \max_{a \in A} Q_\theta(s'_j, a)|^2, \\ & = 2(Q_\theta(s_j, a_j) - \gamma \max_{a \in A} Q_\theta(s'_j, a))(\Phi(s_j, a_j) - \gamma \Phi(s'_j, a_j^*)). \end{cases}$$

Avec $a_i^* = \text{argmax}_{a \in A}[Q_\theta(s_i, a_i) + \hat{l}(s_i, a_i)]$ et $a_j^* = \text{argmax}_{a \in A} Q_\theta(s'_j, a)$. Ceci nous donne finalement :

Algorithm 1 RCAL

Require: D_E, D_P, ϵ critère d'arrêt, $C > \epsilon$, $\theta_0 \in \mathbb{R}^q$, $i = 0$ et $(\delta_j)_{\{j \in \mathbb{N}\}}$ une famille de taux d'apprentissage.

1: Tant que $C > \epsilon$ faire
2: Calculer $\partial_{\theta_i} \mathfrak{L}_2(Q_{\theta_i})$.
3: $\theta_{i+1} = \theta_i - \delta_i \frac{\partial_{\theta_i} \mathfrak{L}_2(Q_{\theta_i})}{\|\partial_{\theta_i} \mathfrak{L}_2(Q_{\theta_i})\|_2^2}$
4: $C = \|\theta_{i+1} - \theta_i\|_2$, $i = i + 1$
5: fin Tant que, renvoyer Q_{θ_i}

5 Travaux connexes

Cette section donne un bref aperçu de la littérature en AI et ARI. L'idée d'ARI a tout d'abord été introduite par Russell (1998) et plus précisément formalisée par Ng *et al.* (2000). La notion d'ARI a été utilisée pour créer des algorithmes qui tentent de résoudre le problème d'AI. L'idée, initiée par Abbeel & Ng (2004), consiste à trouver une politique (via l'intermédiaire d'une fonction de récompense) telle qu'une mesure de la distribution des trajectoires induites par cette politique corresponde à celle de la politique experte. Neu & Szepesvári (2009) proposent une revue de ce type d'approches. Cependant, la plupart de ces méthodes (Syed & Schapire, 2008; Ramachandran, 2007) ont besoin de résoudre itérativement des PDM, qui est en soit un problème difficile. Récemment, deux algorithmes d'ARI se sont affranchis de ce désavantage important. Le premier est l'algorithme Relative Entropy (RE) (Boularias *et al.*, 2011), où la résolution itérative des PDM est évitée grâce à une méthode d'échantillonnage préférentiel. Cette méthode requiert, comme notre algorithme, un ensemble d'exemples de transitions du PDM. Le deuxième algorithme est SCIRL proposé par Klein *et al.* (2012), où la notion d'attribut vectoriel moyen pour une politique π est centrale : pour une récompense linéairement paramètrée par un vecteur d'attributs vectoriels, c'est l'espérance décompté du vecteur d'attributs vectoriels cumulés en partant d'un état donné et en suivant la politique π. Ensuite, l'algorithme SCIRL prend une estimée de l'attribut vectoriel moyen de l'expert comme paramétrisation d'une fonction de score d'un classifieur. Grâce à l'utilisation d'une heuristique, l'algorithme SCIRL peut seulement avoir besoin des couples état-actions experts.

Une autre façon de résoudre le problème d'AI est de le voir comme un problème de classification. Cela a été fait par Atkeson & Schaal (1997) et analysé par Syed & Schapire (2010) pour le cas à horizon fini. Cependant, une approche par pure classification n'utilise pas la structure du PDM. C'est pourquoi, Melo & Lopes (2010) proposent de construire un classifieur à noyaux qui utilise la métrique du PDM. Cependant, le calcul de cette métrique est assez complexe, prend du temps et nécessite la connaissance de la dynamique P. Donc, notre algorithme semble être une bonne alternative pour tenter de résoudre le problème d'AI car il réussit à introduire la structure du PDM dans le problème avec seulement des exemples de transitions du PDM représentés par l'ensemble D_P.

6 Expériences

Cette section présente différents résultats empiriques qui corroborent le fait que notre algorithme (RCAL) a une bonne performance. Dans nos expériences, nous comparons nos algorithmes (RCAL et MBRCAL) avec deux algorithmes d'ARI récents (SCIRL et RE) qui utilisent uniquement des exemples de transitions expertes et non expertes, ainsi qu'avec un algorithme de classification pure. L'algorithme de classification pure est réalisé via la minimisation de $\mathfrak{L}_0$ par une descente de sous-gradient. Le paramètre de compromis λ est fixé à 0.1, le taux d'apprentissage est $\delta_i = \frac{1}{i+1}, i \in \mathbb{N}$ et le facteur d'actualisation est $\gamma = 0.99$ dans toutes nos expériences.

6.1 Le cadre de travail des Garnets

La première expérience se concentre sur les Garnets qui sont une classe de PDM construits de manière aléatoire, qui sont totalement abstraits bien que représentatifs du genre de PDM finis que nous pouvons rencontrer en pratique (voir Archibald *et al.*, 1995). La routine pour créer un Garnet stationnaire est caractérisée par 3 paramètres et s'écrit $Garnet(N_S, N_A, N_B)$. Les paramètres N_S et N_A sont le nombre d'états et d'actions respectivement, et N_B est un facteur de branchement qui spécifie le nombre d'états suivants pour chaque couple état-action. Les états suivants sont choisis aléatoirement dans l'espace d'état sans remplacement pour chaque couple état-action. La probabilité de transiter vers chaque état suivant est générée en partitionnant l'intervalle unité avec $N_B - 1$ points de coupure choisis aléatoirement. La valeur de la récompense récompense $R(s, a)$ est tirée selon une variable gaussienne centrée-réduite pour chaque couple état-action. Pour chaque Garnet, il est possible de calculer la politique experte π_E via l'algorithme d'itération de la politique.

Tous les algorithmes utilisés ont pour entrée des ensembles du type : $D_E = (s_i, a_i)_{\{1 \leq i \leq N\}}$ où $a_i \sim \pi_E(.|s_i)$. Plus précisément, $D_E = (\omega_j)_{\{1 \leq i \leq K_E\}}$ où $\omega_j = (s_{i,j}, a_{i,j})_{\{1 \leq i \leq H_E\}}$ est une trajectoire obtenue en commençant par un état choisi aléatoirement $s_{1,j}$ (distribution uniforme) et en suivant la politique π_E H_E fois ($s_{i+1,j} \sim P(.|s_{i,j}, a_{i,j})$). Donc, D_E est composé par K_E trajectoires de la politique π_E de taille H_E et nous avons $K_E H_E = N$. On procure aussi aux algorithmes RE et RCAL un ensemble de transitions $D_P = (s_i, a_i, s'_i)_{\{1 \leq i \leq N'\}}$ où $a_i \sim \pi_R(.|s_i)$ avec π_R la politique aléatoire (distribution uniforme sur les actions pour chaque état) et où $s'_i \sim P(.|s_i, a_i)$. En fait, D_P a la forme particulière suivante $D_P = (\tau_j)_{\{1 \leq j \leq K_P\}}$ où $\tau_j = (s_{i,j}, a_{i,j}, s'_{i,j})_{\{1 \leq i \leq H_P\}}$ est une trajectoire obtenue en commençant par un état choisi aléatoirement $s_{1,j}$ (distribution uniforme) et en suivant la politique π_R H_P fois ($s'_{i,j} = s_{i+1,j} \sim P(.|s_{i,j}, a_{i,j})$). Donc, D_P est composé par K_P trajectoires de la politique π_R de taille H_P et nous avons $K_P H_P = N'$. Ainsi, si pour un Garnet donné on dispose de π_E et π_R, alors l'ensemble de paramètres (K_E, H_E, K_P, H_P) est suffisant pour instantier les ensembles de type D_E et D_P.

Tout d'abord, nous voulons montrer que pour (K_P, H_P) fixé, si nous augmentons le paramètre H_E (longueur des trajectoires de la politique experte π_E) ou K_E (nombre de trajectoires de la politique experte π_E), alors la performance de tous les algorithmes est améliorée. De plus, nous montrerons aussi que pour (K_E, H_E) fixé, si nous augmentons K_P (nombre de trajectoires de le politique aléatoire π_R) ou H_P (longueur des trajectoires de la politique aléatoire π_R), nous améliorons la performance de l'algorithme RCAL.

Notre première expérience montre l'amélioration de la performance des algorithmes quand H_E augmente. Cela consiste à générer 100 Garnets du type $Garnet(100, 5, 2)$ qui nous donne l'ensemble de Garnets $\mathfrak{G} = (G_p)_{\{1 \leq p \leq 100\}}$. Sur chaque G_p dans l'ensemble $\mathfrak{G}$, nous calculons π_E^p et π_R^p. Le paramètre H_E prend ses valeurs dans l'ensemble $(H_E^k)_{\{1 \leq k \leq 15\}} = (10, 20, 30, .., 150)$, $K_E = 1$, $H_P = 10$, $K_P = 50$. Nous choisissons $H_P K_P = N_S N_A$ pour cette expérience. Puis, pour chaque ensemble de paramètre (K_E, H_E^k, K_P, H_P) et chaque G_p, nous générons 100 ensembles de données experts $(D_E^{i,p,k})_{\{1 \leq i \leq 100\}}$ et 100 ensembles de données aléatoires $(D_P^{i,p,k})_{\{1 \leq i \leq 100\}}$. Notre critère de performance pour chaque couple $(D_E^{i,p,k}, D_P^{i,p,k})$ est le suivant :

$$T^{i,p,k} = \frac{\mathbb{E}_\rho[V_R^{\pi_E^p} - V_R^{\pi_A^{i,p,k}}]}{\mathbb{E}_\rho[V_R^{\pi_E^p}]},$$

où π_E^p est la politique experte, $\pi_A^{i,p,k}$ est la politique induite par l'algorithme qui a pour entrée le couple $(D_E^{i,p,k}, D_P^{i,p,k})$ et où ρ est la mesure de probabilité uniforme sur l'espace S. Pour la classification pure, nous avons $\pi_A^{i,p,k}(s) \in \operatorname{argmax}_{a \in A} \hat{Q}^*(s, a)$ où $\hat{Q}^*$ est le minimiseur du critère $\mathfrak{L}_0$ via la descente de sous-gradient. Pour les algorithmes SCIRL et RE, $\pi_A^{i,p,k}$ est la politique obtenue en optimisant la récompense

R, qui est renvoyée par l'algorithme, via l'algorithme d'itération de la politique. Pour nos algorithmes, $\pi_A^{i,p,k}(s) \in \text{argmax}_{a \in A} Q^*(s,a)$ où Q^* est la sortie de RCAL ou MBRCAL. Comme les performance de SCIRL et de RE sont obtenues à partir de l'algorithme d'itération de la politique, qui a besoin de la connaissance de la dynamique P du PDM, il est donc plus juste de les comparer à la performance de MBRCAL. Notre critère moyen de performance pour chaque ensemble de paramètres (K_E, H_E^k, K_P, H_P) est :

$$T^k = \frac{1}{10000} \sum_{1 \leq p \leq 100, 1 \leq i \leq 100} T^{i,p,k}.$$

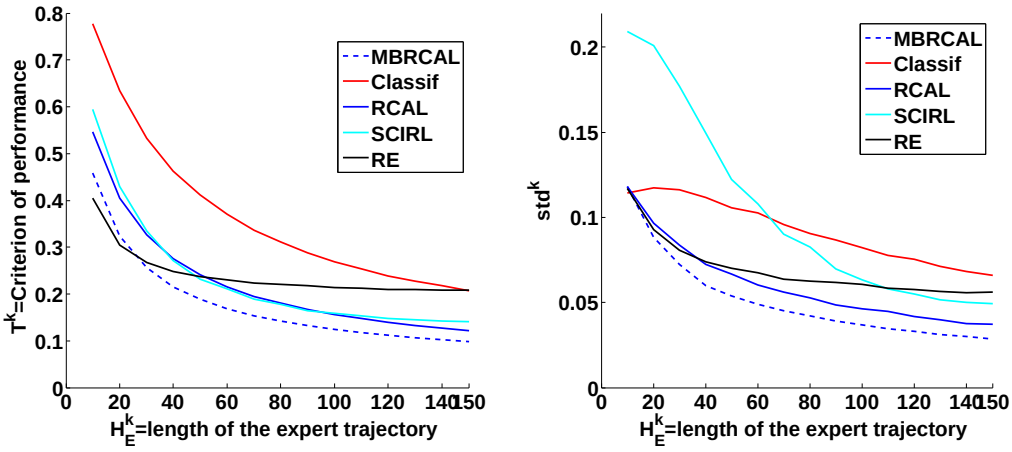
Pour chaque algorithme, nous traçons $(H_E^k, T^k)_{\{1 \leq k \leq 15\}}$. Un autre critère intéressant est l'écart-type des performances des algorithmes :

$$std^{p,k} = \left\{ \frac{1}{100} \sum_{1 \leq i \leq 100} [T^{i,p,k} - \sum_{1 \leq j \leq 100} T^{j,p,k}]^2 \right\}^{\frac{1}{2}}.$$

Et nous pouvons alors calculer l'écart-type moyen sur les 100 Garnets pour chaque ensemble de paramètre (K_E, H_E^k, K_P, H_P) :

$$std^k = \frac{1}{100} \sum_{1 \leq p \leq 100} std^{p,k}.$$

Pour chaque algorithme, nous traçons $(H_E^k, std^k)_{\{1 \leq k \leq 15\}}$. Les résultats sont reportés sur la Fig. 1(a). Sur cette figure, RCAL a pratiquement les mêmes performances que SCIRL mais est meilleur que RE.

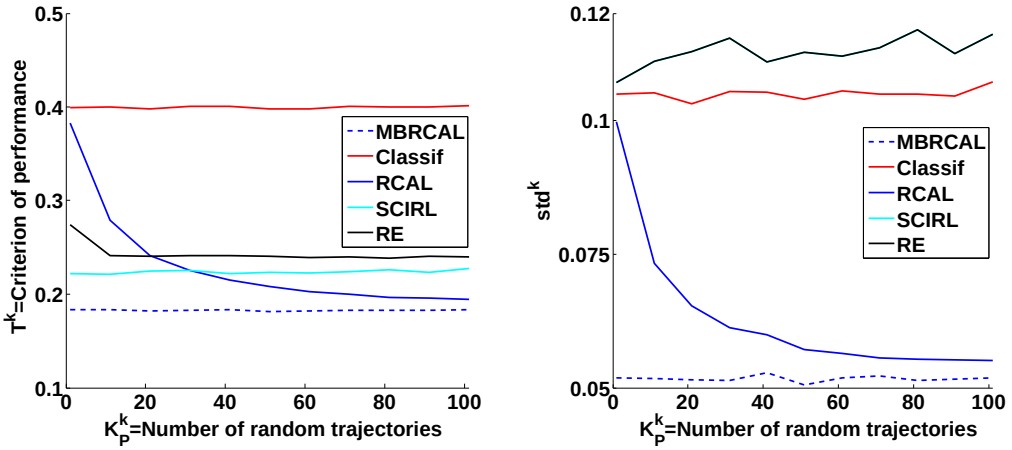


(a) Performance (b) Écart-type moyen

FIGURE 1 – Expérience sur les Garnets avec H_E pour paramètre d'évolution.

Cependant, il est à rappeler que SCIRL à une récompense en sortie qui doit être optimisée. Cela nécessite donc la résolution d'un PDM pour avoir une politique. On n'a pas cette contrainte avec RCAL puisqu'il admet comme sortie une fonction de qualité. Quand la dynamique P du PDM est donnée, MBRCAL donne des résultats meilleurs que RCAL ce qui était attendu mais aussi meilleurs que SCIRL. De plus, RCAL à un plus faible écart-type que SCIRL et RE ce qui garantit ne pas être trop loin de la performance moyenne pour un ensemble de données. Enfin, nous observons que RCAL a une bien meilleure performance qu'un pure classifieur même si la seule différence est le terme de régularisation.

Ensuite, nous montrons que si la taille de l'ensemble de données D_P augmente, cela améliore la performance de RCAL. Nous reproduisons exactement la même expérience que précédemment excepté que le paramètre d'évolution est K_P et non H_E. Le paramètre K_P prend ses valeurs dans l'ensemble $(K_P^k)_{\{1 \leq k \leq 11\}} = (1, 11, 21, .., 101)$, $K_E = 1$, $H_E = 50$, $H_P = 10$. Nous prenons le même critère de performance et nous traçons $(K_P^k, T^k)_{\{1 \leq k \leq 11\}}$ et $(K_P^k, std^k)_{\{1 \leq k \leq 11\}}$. Les résultats sont reportés sur la Fig. 2. Quand la taille des données D_P augmente, nous voyons que la performance de RCAL converge vers la performance de MBRCAL et est meilleur que SCIRL. En pratique, il y a certaines applications où l'ensemble de données



(a) Performance

(b) Écart-type moyen

FIGURE 2 – Expérience sur les Garnets avec K_P pour paramètre d'évolution.

D_P est facile à collecter. Dans ces applications, les données coûteuses sont celles de l'expert. C'est pourquoi RCAL s'avère intéressant : sa performance dépend de données qui peuvent être faciles à collecter. De plus, contrairement à SCIRL, RCAL n'a pas besoin d'heuristique pour fonctionner. Cependant, il a besoin de plus de données que SCIRL qui utilisent uniquement des données expertes (bien qu'il faille ensuite optimiser la récompense en sortie, ce qui nécessite des données supplémentaires). Donc, si pour une application donnée, la collecte de données via une politique aléatoire n'est pas coûteuse alors RCAL est un algorithme compétitif.

6.2 Le simulateur de trafic routier

Ce genre de problèmes a déjà été utilisé comme expériences dans la littérature d'ARI (Abbeel & Ng, 2004; Syed *et al.*, 2008; Klein *et al.*, 2012). Nous utilisons le même simulateur de trafic routier que Klein *et al.* (2012). Dans ce problème, le but est de conduire une voiture sur une autoroute fréquentée à trois voies dont le trafic est généré aléatoirement. La voiture peut bouger de droite à gauche, accélérer, décélérer ou garder une vitesse constante (5 attributs vectoriels d'actions). L'expert optimise ensuite une récompense créée à la main qui favorise la vitesse, punit la circulation hors des voies, punit davantage les collisions et est neutre sinon. Nous avons en tout 729 attributs vectoriels d'états correspondant à : 9 positions horizontales pour l'utilisateur de la voiture, 3 positions horizontales et 9 positions verticales pour la voiture la plus proche dans le trafic et 3 vitesses possibles.

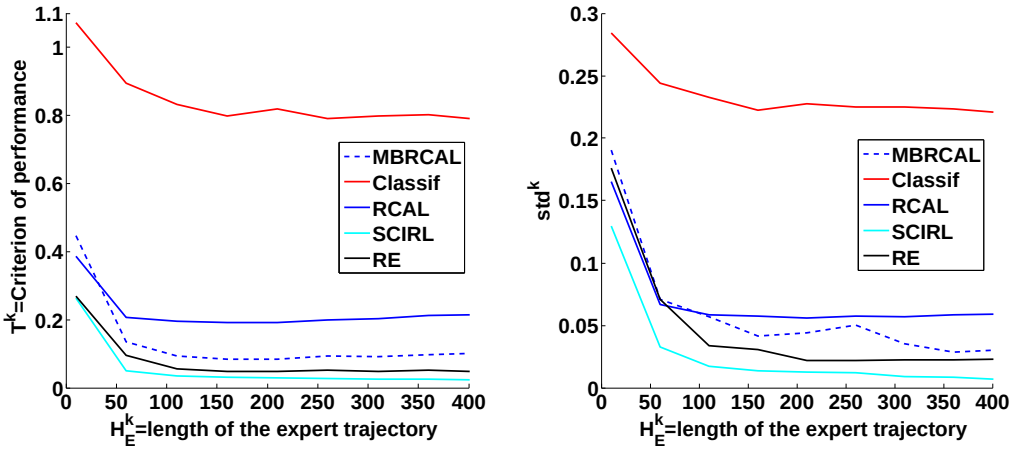
Nous calculons π_E via l'algorithme d'itération de la politique comme la dynamique P du PDM du simulateur de trafic est connue. Pour ce problème particulier, nous considérons que le paramètre prend ses valeurs dans l'ensemble $(H_E^k)_{\{1\leq k\leq 9\}} = (10, 60, 110, .., 410)$ et que les autres paramètres sont fixés $K_E = 1$, $H_P = 50$, $K_P = 50$. Ensuite, pour chaque ensemble de paramètres (K_E, H_E^k, K_P, H_P), nous calculons 100 ensembles de données $(D_E^{i,k})_{\{1\leq i\leq 100\}}$ et 100 ensembles de données aléatoires $(D_P^{i,k})_{\{1\leq i\leq 100\}}$. Notre critère de performance pour chaque couple $(D_E^{i,k}, D_P^{i,k})$ est le suivant :

$$T^{i,k} = \frac{\mathbb{E}_\rho[V_R^{\pi_E} - V_R^{\pi_A^{i,k}}]}{\mathbb{E}_\rho[V_R^{\pi_E}]},$$

où $\pi_A^{i,k}$ est la politique induite par l'algorithme ayant pour entrée $(D_E^{i,p,k}, D_P^{i,p,k})$ et où ρ est la distribution uniforme sur l'espace d'état. Notre critère de performance moyen pour l'ensemble de paramètre (K_E, H_E^k, K_P, H_P) est :

$$T^k = \frac{1}{100} \sum_{1\leq i\leq 100} T^{i,k}.$$

Pour chaque algorithme, nous traçons $(H_E^k, T^k)_{\{1\leq k\leq 9\}}$. Les résultats sont reportés sur la Fig. 3. Ici nous



(a) Performance (b) Écart-type

FIGURE 3 – Expérience sur le simulateur de trafic routier.

voyons que SCIRL et RE ont une meilleure performance que RCAL et MBRCAL. La raison est peut-être liée au fait que la récompense créée à la main est assez parcimonieuse ce qui peut favoriser les algorithmes d'ARI (sachant que la récompense est ensuite optimisée). Cependant, nous observons aussi que RCAL fait beaucoup mieux que la classification pure. Toutefois, afin de pouvoir avoir une comparaison plus générale entre les algorithmes, il est intéressant d'utiliser une tâche plus générique comme les Garnets.

7 Conclusion et perspectives

Dans cet article, le cadre de travail des politiques d'ensembles est introduit et nous permet d'établir un lien formel et clair entre l'ARI et l'AI. A partir de ce cadre, nous dérivons un algorithme appelé RCAL pour le problème d'AI qui est capable de prendre en compte la structure du PDM pour obtenir une meilleure performance qu'un algorithme de classification pure. Cet algorithme a uniquement besoin d'un ensemble de couples état-action experts et d'un ensemble d'exemples de transitions du PDM. De plus, il ne nécessite pas la résolution itérée de PDM ou de connaître entièrement la dynamique du problème. Toutefois, si la dynamique est connue, il est aussi possible d'utiliser l'algorithme appelé MBRCAL qui prend en entrée un ensemble de couples état-action experts. D'autre part, nous avons illustré notre approche via une tâche générique que sont les Garnets et sur un problème plus complexe qu'est le simulateur de trafic routier. Nous avons montré que notre algorithme peut avoir de meilleures performance que des algorithmes d'ARI existants si l'ensemble des exemples de transitions du PDM est assez grand. Finalement, il y a plusieurs perspectives intéressantes que nous souhaitons étudier dans un futur proche. La première est le choix automatique des attributs vectoriels permettant la représentation de Q pour un problème à partir d'un ensemble de données. Cela pourrait être réalisé avec une méthode de boosting. La seconde perspective serait de proposer un algorithme pratique d'ARI en trouvant une approximation $J^*\hat{Q}^*$ de J^*Q^* via une ensemble d'exemples de transitions du PDM, où Q^* est la sortie de RCAL. La troisième perspective serait de fournir une étude théorique où une expression explicite de l'erreur de classification de notre algorithme serait donnée et la comparer à l'erreur de classification de l'algorithme de classification pure.

Références

ABBEEL P. & NG A. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*.

ARCHIBALD T., MCKINNON K. & THOMAS L. (1995). On the generation of markov decision processes. *Journal of the Operational Research Society*.

ATKESON C. & SCHAAL S. (1997). Robot learning from demonstration. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*.

BOULARIAS A., KOBER J. & PETERS J. (2011). Relative entropy inverse reinforcement learning. In *JMLR Workshop and Conference Proceedings Volume 15 : AISTATS 2011*.

EVGENIOU T., PONTIL M. & POGGIO T. (2000). Regularization networks and support vector machines. *Advances in Computational Mathematics*, **13**(1), 1–50.

KLEIN E., GEIST M., PIOT B. & PIETQUIN O. (2012). Inverse reinforcement learning through structured classification. In *Advances in Neural Information Processing Systems 25 (NIPS)*.

LANGFORD J. & ZADROZNY B. (2005). Relating reinforcement learning performance to classification performance. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*.

MELO F. & LOPES M. (2010). Learning from demonstration using mdp induced metrics. In *Proceedings of the European Conference on Machine Learning (ECML)*.

MELO F., LOPES M. & FERREIRA R. (2010). Analysis of inverse reinforcement learning with perturbed demonstrations. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI)*.

NEU G. & SZEPESVÁRI C. (2009). Training parsers by inverse reinforcement learning. *Machine learning*, **77**(2).

NG A., RUSSELL S. *et al.* (2000). Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*.

POMERLEAU D. (1989). *Alvinn : An autonomous land vehicle in a neural network*. Rapport interne, DTIC Document.

PUTERMAN M. (1994). *Markov decision processes : Discrete stochastic dynamic programming*. John Wiley & Sons.

RAMACHANDRAN D. (2007). Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference of Artificial Intelligence (IJCAI)*.

RUSSELL S. (1998). Learning agents for uncertain environments. In *Proceedings of the 11th annual conference on Computational Learning Theory (COLT)*.

SHOR N., KIWIEL K. & RUSZCAYNSKI A. (1985). *Minimization methods for non-differentiable functions*. Springer-Verlag.

SYED U., BOWLING M. & SCHAPIRE R. (2008). Apprenticeship learning using linear programming. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*.

SYED U. & SCHAPIRE R. (2008). A game-theoretic approach to apprenticeship learning. In *Advances in Neural Information Processing Systems 21 (NIPS)*.

SYED U. & SCHAPIRE R. (2010). A reduction from apprenticeship learning to classification. In *Advances in Neural Information Processing Systems 23 (NIPS)*.

TASKAR B., CHATALBASHEV V., KOLLER D. & GUESTRIN C. (2005). Learning structured prediction models : A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*.

VAPNIK V. (1998). *Statistical learning theory*. Wiley.