Classification structurée pour l'apprentissage par renforcement inverse

Edouard Klein[1,2], Bilal Piot[1,3], Matthieu Geist[1], Olivier Pietquin[1,3]

[1] Supélec, IMS Research group, France, prenom.nom@supelec.fr

[2] Equipe ABC, LORIA, France

[3] UMI 2958 GeorgiaTech-CNRS, France

Résumé : Cette contribution traite du problème de l'apprentissage par imitation par le biais de l'apprentissage par renforcement inverse (*ARI*). Dans ce contexte, un expert accomplit une tâche qu'un agent artificiel doit essayer de reproduire. L'ARI part du postulat que l'expert optimise avec succès une fonction de récompense ; le problème consiste à deviner cette fonction à partir de traces du comportement de l'expert. Les algorithmes d'ARI existants nécessitent une ou plusieurs des conditions suivantes pour fonctionner : trajectoires complètes de la part de l'expert, un modèle génératif pour les estimations de type Monte-Carlo, la connaissance des probabilités de transition, la capacité de résoudre le problème direct (celui de l'apprentissage par renforcement) de manière répétée ou l'accès à la strategie complète de l'expert. Notre contribution consiste en un nouvel algorithme d'ARI levant l'ensemble de ces contraintes. En utilisant une méthode supervisée dans laquelle nous introduisons implicitement la structure du processus décisionnel de Markov (*PDM*) sous-jacent, nous créons un algorithme basé sur une descente de sous-gradient, possèdant une faible complexité tant en échantillons que calculatoire et surtout ne nécessitant pas la résolution du problème direct. **Mots-clés** : Apprentissage par renforcement inverse, classification multi-classe, attribut vectoriel moyen

1. Introduction

Dans cette contribution, nous nous plaçons dans le contexte de l'apprentissage par démonstration. Dans ce cadre, un agent artificiel apprend à reproduire une tâche grâce à l'observation d'un expert réalisant cette tâche de manière optimale. Pour trouver une solution à ce problème, nous nous proposons d'utiliser le paradigme de l'apprentissage par renforcement inverse. On suppose alors que l'expert agit de manière à être récompensé pour son comportement

relativement à une fonction de récompense inconnue. Le but de l'agent est alors d'inférer cette fonction des démonstrations données par l'expert pour ensuite optimiser son comportement relativement à cette fonction lui aussi.

La littérature sur le sujet est assez récente et trouve sa genèse dans Russell (1998). Plusieurs directions ont depuis été explorées à partir de deux articles fondateurs Ng & Russell (2000) et Abbeel & Ng (2004).
Dans cette littérature (voir Neu & Szepesvári (2009) pour un bon survol), plusieurs verrous sont identifiés mais souvent ignorés pour proposer des algorithmes de résolution. Particulièrement, on suppose souvent connue la dynamique de l'environnement dans lequel évoluent l'expert et l'agent ou bien la mise à disposition de cet environnement (ou d'un simulateur) pour y tester les effets d'une politique quelconque. De plus, ces approches supposent de résoudre le problème direct (optimiser une politique pour une récompense donnée) un grand nombre de fois pour résoudre le problème inverse (trouver une récompense expliquant la politique). De plus, certaines de ces approches donnent en sortie une politique généralisant celle de l'expert plutôt qu'une récompense l'expliquant. Ces contraintes sont souvent incompatibles avec la mise en oeuvre des algorithmes proposés sur des cas réels.

Dans cette contribution, nous proposons une méthode de résolution du problème d'apprentissage par renforcement inverse s'affranchissant de toute connaissance additionnelle aux démonstrations fournies par l'expert. Nous proposons une preuve de convergence de cet algorithme et fournissons une expérimentation sur le problème du pendule inversé. Le choix de cette application se justifie par la difficulté qu'elle pose pour les méthodes de résolution de la littérature.

2. Contexte

2.1. Apprentissage par renforcement direct et inverse

Le cadre dans lequel nous plaçons notre étude est celui de la prise de décisions séquentielles. La configuration du système à contrôler est alors complètement décrite à chaque instant discret t par un état $s_t \in S$. Confronté à cet état, l'agent doit prendre une décision $a_t \in A$. Le système évolue alors vers l'état suivant s_{t+1} selon une certaine probabilité de transition markovienne $p(s_{t+1}|s_t, a_t)$. Une politique de contrôle déterministe $\pi : S \to A$ définit

le comportement d'un agent confronté à un tel problème de décisions séquentielles.

Dans le cadre de l'apprentissage par renforcement,la résolution de ce problème est guidée par une fonction de récompense $R : S \to \mathbb{R}$ qui indique le degré de désirabilité de chaque état. Une des forces de ce cadre réside dans le fait que l'agent ne va pas apprendre à maximiser la récompense immédiate mais au contraire un critère prenant le futur en compte, ce qui permet de spécifier uniquement le but et non la façon de l'atteindre. On définit pour ce faire la fonction de valeur V^π :

$$V^\pi(s) = E[\sum_t \gamma^t R(s_t)|s_0 = s, \pi] = R(s) + \gamma E[V^\pi(s')|s, \pi]$$

où $\gamma \in [0, 1)$ est un facteur d'actualisation. Il est également possible de définir une fonction de qualité qui ajoute un degré de liberté sur le choix de la première action :

$$Q^\pi(s, a) = E[\sum_t \gamma^t R(s_t)|s_0 = s, a_0 = a, \pi] = R(s)+\gamma E[V^\pi(s')|s, a]. \quad (1)$$

Une politique optimale π^* est définie comme une politique dont la fonction de valeur (optimale) V^* vérifie $\forall\pi, \forall s, V^*(s) \geq V^\pi(s)$. La politique optimale se calcule simplement à partir de la fonction de qualité optimale Q^* via un mécanisme glouton :

$$\pi^*(s) \in \arg\max_a Q^*(s, a). \quad (2)$$

L'ensemble formé par l'espace d'état S, l'espace d'action A, les probabilités de transition p, le facteur γ et la fonction de récompense R forme un processus décisionnel de Markov (*PDM*).

Parfois, définir la fonction de récompense est une tâche ardue alors qu'il est possible à l'opérateur de convenablement contrôler le système de manière intuitive. Un exemple de ce type de tâche est la conduite d'une voiture. Nous accomplissons ce genre de chose au quotidien sans trop y penser mais nous serions bien en peine de préciser les poids précis que nous attribuons aux différents critères tels que la distance nous séparant de la voiture devant nous, la brutalité avec laquelle nous appuyons sur la pédale de frein lorsqu'un danger se présente et ainsi de suite. Dans de tels cas, il est utile d'inférer la récompense à partir d'un comportement démontré.

C'est la définition de l'apprentissage par renforcement inverse. Le problème est de retrouver la fonction de récompense optimisée par un expert. Usuellement l'expert est considéré comme un agent optimal dans un PDM et l'on a accès ou bien à la politique complète π_E de l'expert ou bien à quelques trajectoires tirées selon cette politique.

Ce problème est mal posé dans la mesure où il n'y a pas unicité de la récompense pour laquelle un comportement est optimal (Ng *et al.* (1999)). Particulièrement, tout comportement est optimal vis-à-vis de la récompense uniformément nulle. Il convient donc de contraindre l'espace des solutions.

2.2. Attribut vectoriel moyen

La récompense R est l'inconnue du problème. Nous supposons que l'utilisateur est en mesure de fournir p fonctions de base $\psi_{1\leq i\leq N}$ telles que

$$\exists\theta|R(s) = \theta^T\psi(s) = \sum_{i=1}^{p}\theta_i\psi_i(s). \tag{3}$$

Cette contrainte peut cependant être relâchée si nécessaire. Introduire cette expression dans la définition de la fonction de qualité (Eq. (1)) fait apparaître un terme intéressant et dont l'usage est assez répandu dans les algorithmes existants :

$$Q^\pi(s,a) = \theta^T\mu^\pi(s,a), \mu^\pi(s,a) = E[\sum_t \gamma^t\psi(s_t)|s_0 = s, a_0 = a, \pi] \tag{4}$$

Le terme μ^π est appelé l'attribut vectoriel moyen (*feature expectation*) d'une politique. On peut voir que la paramétrisation choisie pour R et la dynamique imposée par l'application de la politique π dans le PDM fixe la paramétrisation de Q^π.

On peut noter que si deux politiques ont le même attribut vectoriel moyen, alors ces deux politiques ont la même valeur vis-à-vis de la récompense, quel que soit le vecteur de poids θ définissant celle-ci. En effet

$$\mu^{\pi_1} = \mu^{\pi_2} \Rightarrow \theta^T\mu^{\pi_1} = \theta^T\mu^{\pi_2} \Rightarrow V^{\pi_1} = V^{\pi_2}.$$

L'attribut vectoriel moyen de l'expert, μ_E, peut être calculé sans forcément avoir accès à l'intégralité de la politique π_E. La plupart des algorithmes de la

littérature ont, comme nous allons le voir, pour objectif de minimiser la distance entre les attributs vectoriels moyens respectifs de l'agent et de l'expert. Cela permet d'obtenir un agent dont le comportement a la même fonction de valeur que celui de l'expert, quelle que soit la récompense (voir par exemple Neu & Szepesvári (2009)).

3. Classification structurée pour l'ARI

3.1. Principe

Comme nous l'avons vu Eq. (2), il existe un mécanisme glouton reliant la politique de l'expert, π_E et la fonction de qualité optimale Q_R^E relative à la fonction de récompense inconnue : $\pi_E(s) \in \arg\max_a Q_R^E(s,a)$.
Nous allons utiliser cette propriété pour considérer un risque dépendant d'une fonction de score q qui devra partager avec la fonction inconnue Q_R^E la propriété d'optimalité de l'expert que nous venons d'exprimer.
Notons $(s_i, a_i, s_{i+1})_{1\leq i\leq N}$ les échantillons représentant une ou plusieurs trajectoires de l'expert. Nous cherchons donc une fonction de score qui vérifie pour chacun de ces échantillons :

$$a_i \in \arg\max_a q(s_i, a).$$

Nous souhaitons donc introduire une fonction de risque qui devrait pénaliser les cas où $\exists a, q(s_i,a) > q(s_i,a_i)$, car alors la fonction q ne permet pas de justifier (au sens de l'équation (2)) le choix a_i de l'expert. Nous introduisons de surcroit une fonction $l : S \times A \to \mathbb{R}_+$; cette fonction permet à l'opérateur d'introduire de la connaissance *a priori* dans le système en précisant l'écart qu'il souhaite obtenir entre la qualité de l'action choisie par l'expert et la qualité de la seconde meilleure action selon l'inégalité $\forall a, q(s_i,a)+l(s_i,a) \leq q(s_i,a_i)$. Ce type d'argument est assez commun dans la littérature (voir Sec. 5.). Nous introduisons ainsi la fonction de risque empirique suivante :

$$J_N(q) = \frac{1}{N}\sum_{i=1}^{N}\left(\max_a(q(s_i,a)+l(s_i,a)) - q(s_i,a_i)\right). \qquad (5)$$

Par défaut, définir $l(s_i,a) = 1$ si $a \neq a_i$ et $l(s_i,a_i) = 0$ fonctionne bien comme le montrent les expériences (Sec. 6.) et l'analyse (Sec. 4.).

Nous traitons le problème de l'imitation comme un problème de classification multi-classe. Notre propos n'est cependant pas d'utiliser une approche purement supervisée. Nous ne souhaitons pas apprendre le contrôle de l'expert mais la description de la tâche qu'il effectue (la récompense). Il faut pour cela prendre en compte la structure du PDM, notamment l'information sur les transitions entre états conditionnées par la politique de l'expert.

Comme nous l'avons vu Eq. (4), l'hypothèse faite d'une représentation linéaire selon certains attributs $\psi : S \to \mathbb{R}^p$ de la fonction de récompense impose d'exprimer la fonction de qualité d'une politique en fonction de l'attribut vectoriel moyen de cette politique. Cette paramétrisation prend en compte la dynamique imposée par π_E. L'expert ne déroge pas à cette règle et notant μ_E l'attribut vectoriel moyen de l'expert, nous pouvons donc faire l'hypothèse que la fonction q que nous recherchons peut s'exprimer sous la forme $q(s,a) = \theta^T \mu_E(s,a)$. L'estimation de ce terme μ_E est centrale à notre algorithme et nous proposons quelques solutions en Sec. 3.2. Nous pouvons donc maintenant exprimer notre risque, non plus comme une fonction de q, mais comme une fonction du vecteur de paramètres θ, vecteur qui définit la récompense (Eq. (3)) :

$$J_N(\theta) = \frac{1}{N} \sum_{i=1}^{N} \Big(\max_a (\theta^T \mu_E(s_i,a) + l(s_i,a)) - \theta^T \mu_E(s_i,a_i) \Big) \qquad (6)$$

Ce risque est non linéaire et n'est pas différentiable partout, en raison du $\max$. Nous utilisons une descente de sous-gradient (normalisé), une généralisation du gradient, pour le minimiser. Le sous-gradient d'une fonction convexe $f : \mathbb{R}^p \to \mathbb{R}$ en x est défini comme un vecteur g tel que $\forall x' \in \mathbb{R}^p, g^T(x' - x) \leq f(x') - f(x)$. Le sous-gradient n'est pas partout unique, cependant l'unique sous-gradient d'une fonction différentiable est son gradient. Il s'agit d'un opérateur linéaire et la fonction convexe, différentiable par morceaux $\max_y [f(x,y)]$ définie à l'aide de la fonction différentiable $f(\cdot, y)$ admet comme l'un de ses sous-gradients l'expression $\nabla_x f(x, y^*)$ avec $y^* = \arg\max_y f(x,y)$. En appliquant ces règles, on obtient le sous gradient et la règle de mise à jour

associée :

$$\begin{aligned}
\nabla_\theta J_N(\theta) &= \frac{1}{N}\sum_{i=1}^{N}(\mu_E(s_i,a_i^*)-\mu_E(s_i,a_i)) \\
a_i^* &= \arg\max_a(\theta^T\mu_E(s_i,a)+l(s_i,a)) \\
\theta_{t+1} &\leftarrow \theta_t - \alpha_t\frac{\nabla_{\theta_t}J_N(\theta_t)}{||\nabla_{\theta_t}J_N(\theta_t)||_2}.
\end{aligned}$$

3.2. Calcul de μ_E

Comme signalé dans la Sec. 2.2., beaucoup d'algorithmes de la littérature reposent sur un calcul de l'attribut vectoriel moyen. Notre algorithme a cependant ceci de particulier qu'il ne nécessite que le calcul de μ_E, l'attribut vectoriel moyen de l'expert et non le calcul répété de l'attribut vectoriel moyen d'une politique arbitrairement éloignée de celle de l'expert. Nous avons besoin de connaître $\mu_E(s_i,a), \forall i = 1..N, \forall a \in A$. Cela implique d'inférer l'attribut vectoriel moyen de l'expert sur des états où celui-ci est passé, mais pour des actions qu'il n'a pas forcément choisies.

Cette inférence est facilitée par le fait que l'attribut vectoriel moyen obéit à l'équation de Bellman. En effet, lorsque l'on écrit sa définition pour une composante $1 \leq j \leq p$, on trouve :

$$\mu_E^j(s,a) = E[\sum_{t=0}^{\infty}\gamma^t\psi^j(s)|s_0=s,a_0=a,\pi_E],$$

expression qui lorsqu'on la rapproche de l'équation (1) permet d'affirmer que μ_E^j est en fait la fonction de qualité de la politique π_E vis-à-vis de la récompense définie par la j-ème composante du vecteur d'attributs ψ. Selon les informations disponibles, plusieurs méthodes peuvent donc être utilisées pour calculer μ_E.

Dans le cas idéal où le modèle est connu, le calcul exact de μ_E peut être effectué grâce aux algorithmes de programmation dynamique. Dans le cas plus réaliste où les probabilités de transitions sont inconnues, on peut, si l'on dispose d'un simulateur et qu'il est possible d'interroger l'expert en un grand nombre d'états arbitraires, utiliser une simple approximation de Monte-Carlo

où pour approximer $\mu_E(s,a)$ (définition Eq. (4)) l'on utilise

$$\hat{\mu}_E(s,a) = \frac{1}{N}\sum_{i=1}^{N}\{\sum_t \gamma^t \psi(s), s_0 = s, a_0 = a, s_{t+1} \sim p(\cdot|s_t, \pi_E(s_t))\}.$$

Ensuite, afin d'entrer dans le cadre où les seules données qui nous sont accessibles sont des trajectoires fournies par l'expert, il est possible d'utiliser l'algorithme LSTDμ (Klein *et al.*, 2011) dans sa version *on-policy*. Cet algorithme adapte l'algorithme LSTD (Bradtke & Barto (1996)) dont il conserve les caractéritiques, notamment la capacité à effectuer le calcul en mode *batch*. C'est la méthode que nous illustrons dans la section 6. Le principe est d'estimer chacun des composantes de μ_E en utilisant LSTD. Certains calculs sont factorisables, donnant ainsi à LSTDμ un coût du même ordre de grandeur que celui de LSTD.

D'autres méthodes d'approximation sont envisageables, comme par exemple une approximation de Monte-Carlo moins coûteuse et moins contraignante que celle que nous venons de proposer, se basant uniquement sur les données fournies par l'expert et impliquant l'utilisation d'une heuristique pour les actions inconnues. Cela est reporté pour des travaux futurs.

4. Analyse

4.1. Hypothèses

Ici, nous présentons le cadre sous lequel notre algorithme de sous gradient converge vers un minimiseur de J_N qui rende la politique de l'expert optimale. On supposera que la récompense R est de la forme $R(s) = \theta_E^T \psi(s)$ où $\theta_E \in \mathbb{R}^p$, que notre base d'exemples est telle que $S \subset \{s_i\}_{1\leq i\leq N}$ et que μ_E est estimée sans erreur sur $\{(s_i,a_i)\}_{1\leq i\leq N}$. Nous supposons également l'unicité de l'action optimale (cette hypothèse pouvant être relâchée au prix de notations plus complexes) :

$$\forall 1 \leq i \leq N, \theta_E^T \mu_E(s_i,a_i) > \max_{a\in A\backslash a_i} \theta_E^T \mu_E(s_i,a), \tag{10}$$

où la notation $A\backslash a_i$ désigne l'ensemble A privé de l'action a_i. Ainsi tout vecteur θ vérifiant l'équation (10) rendra la politique de l'expert optimale. Si la base d'exemples est tel que $\{s_i\}_{1\leq i\leq N} \subset S$, alors un θ vérifiant l'équation (10) rendra la politique de l'expert optimale au moins sur $\{s_i\}_{1\leq i\leq N}$

car l'équation (10) impliquera dans ce cas : $\forall s \in \{s_i\}_{1\leq i\leq N}, V^{\pi_E}(s) \geq \max_{a\in A} Q^{\pi_E}(s,a)$. Dans ce cadre, nous allons montrer qu'il existe bien un θ vérifiant l'équation (10) qui minimise J_N, que les minimiseurs de J_N vérifient bien l'équation (10) et que l'algorithme de sous-gradient converge vers un minimiseur.

4.2. Résultat

On va montrer, dans un premier temps, qu'il existe bien un $\hat{\theta}$ qui vérifie l'équation (10) et qui minimise J_N. On note tout d'abord $M^* = \{\theta \in \mathbb{R}^p : J_N(\theta) = \inf_{\theta\in\mathbb{R}^p} J_N(\theta)\}$. Désormais, on imposera que notre fonction de perte vérifie : $\forall\ 1 \leq i \leq N,\ \forall\ a \in A \setminus a_i,\ l(s_i,a) > l(s_i,a_i) \geq 0$. Il est facile de borner inférieurement le risque J_N, pour tout $\theta \in \mathbb{R}^p$ on a : $J_N(\theta) \geq \frac{1}{N}\sum_{i=1}^N l(s_i,a_i)$. Soit $\hat{\theta} \in \mathbb{R}^p$ vérifiant l'équation (10), l'existence de ce dernier est assuré par l'hypothèse sur θ_E. Notons $\hat{\theta}_x = x\frac{\hat{\theta}}{\|\hat{\theta}\|_2} \in \mathbb{R}^p$ avec $x \geq 0$, $\hat{M} = \{\hat{\theta} \in \mathbb{R}^p : \hat{\theta}$ vérifie l'équation (10) $\}$, et posons $\forall\ 1 \leq i \leq N,\ \beta_i = \hat{\theta}_1^T\ \mu_E(s_i,a_i) - \max_{a\in A\setminus a_i} \hat{\theta}_1^T\ \mu_E(s_i,a) > 0$. Montrons qu'il existe un $x \geq 0$ tel que $J_N(\hat{\theta}_x) = \frac{1}{N}\sum_{i=1}^N l(s_i,a_i)$, et cela nous assurera que $\hat{\theta}_x \in M^*$ d'après la minoration $J_N(\theta) \geq \frac{1}{N}\sum_{i=1}^N l(s_i,a_i),\ \forall\ \theta \in \mathbb{R}^p$. Avec les notations précédentes, on vérifie $\forall 1 \leq i \leq N$:

$$\begin{aligned}
&\hat{\theta}_x^T\mu_E(s_i,a_i) - \max_{a\in A\setminus a_i} \hat{\theta}_x^T\mu_E(s_i,a) + l(s_i,a_i) - \max_{a\in A\setminus a_i} l(s_i,a) \geq 0,\\
\Leftrightarrow &x \geq \frac{\max_{a\in A\setminus a_i} l(s_i,a) - l(s_i,a_i)}{\beta_i}.
\end{aligned}$$

Or comme la somme des $\max$ est supérieure au $\max$ d'une somme, il est clair que $\forall 1 \leq i \leq N$:

$$\begin{aligned}
&x \geq \frac{\max_{a\in A\setminus a_i} l(s_i,a) - l(s_i,a_i)}{\beta_i},\\
\Leftrightarrow &\hat{\theta}_x^T\mu_E(s_i,a_i) - \max_{a\in A\setminus a_i} \hat{\theta}_x^T\mu_E(s_i,a) + l(s_i,a_i) - \max_{a\in A\setminus a_i} l(s_i,a) \geq 0,\\
\Rightarrow &\hat{\theta}_x^T\mu_E(s_i,a_i) + l(s_i,a_i) - \max_{a\in A\setminus a_i} (\hat{\theta}_x^T\mu_E(s_i,a) + l(s_i,a)) \geq 0,\\
\Leftrightarrow &J_N(\hat{\theta}_x) = \frac{1}{N}\sum_{i=1}^N l(s_i,a_i).
\end{aligned}$$

C'est bien le résultat recherché. Montrons à présent que les minimiseurs de J_N vérifient bien l'équation (10). Soit $\theta^* \in M^*$. Pour $\hat{\theta} \in \hat{M}$, on a vu qu'il existait un $x > 0$ tel que $J_N(\hat{\theta}_x) = \frac{1}{N}\sum_{i=1}^{N} l(s_i, a_i)$, ainsi d'après la minoration $\forall \theta \in \mathbb{R}^p, J_N(\theta) \geq \frac{1}{N}\sum_{i=1}^{N} l(s_i, a_i)$, $J_N(\theta^*) = \frac{1}{N}\sum_{i=1}^{N} l(s_i, a_i)$. Or on a $\forall 1 \leq i \leq N$:

$$
\begin{aligned}
&J_N(\theta^*) = \frac{1}{N}\sum_{i=1}^{N} l(s_i, a_i), \\
\Leftrightarrow &\forall a \in A \backslash a_i, (\theta^*)^T \mu_E(s_i, a_i) - (\theta^*)^T \mu_E(s_i, a) \geq l(s_i, a) - l(s_i, a_i) > 0, \\
\Rightarrow &(\theta^*)^T \mu_E(s_i, a_i) - \max_{a \in A \backslash a_i} (\theta^*)^T \mu_E(s_i, a) > 0.
\end{aligned}
$$

Donc $\theta^* \in \hat{M}$. C'est bien ce que l'on recherchait. Si on récapitule, on a montré dans un premier temps que $M^* \neq \emptyset$ et dans un second temps que $M^* \subset \hat{M}$. Finalement, montrons que l'algorithme de sous-gradient converge. Tout d'abord, il est clair que J_N est une fonction convexe, fermée, propre, à valeurs finies et continue en la variable $\theta \in \mathbb{R}^p$. De plus on sait que $M^* \neq \emptyset$, on peut donc appliquer le théorème présenté en section 5 de l'article de Correa & Lemaréchal (1993) qui nous assure que la suite :

$$\theta_0 \in \mathbb{R}^p, \forall t \in \mathbb{N}, \theta_{t+1} = \theta_t - \alpha_t \frac{\nabla_{\theta_t} J_N(\theta_t)}{||\nabla_{\theta_t} J_N(\theta_t)||_2},$$

est convergente vers un minimiseur θ^* de J_N, où α_t est un taux d'apprentissage satisfaisant les conditions usuelles ($\sum_{t>0} \alpha_t$ diverge et $\sum_{t>0} \alpha_t^2$ converge). Ce minimiseur vérifiera l'équation (10) car $M^* \subset \hat{M}$ et la politique de l'expert sera optimale pour la récompense relative à θ^* car l'équation (10) implique l'équation d'optimalité de Bellman.

5. Travaux Connexes

L'approche historique de Ng & Russell (2000), ainsi que par exemple Abbeel & Ng (2004); Syed *et al.* (2008); Syed & Schapire (2008); Ziebart *et al.* (2008), approches résumées dans Neu & Szepesvári (2009) souffrent toutes de la même contrainte : la nécessité de résoudre le problème direct de manière répétée, c'est à dire le calcul de la politique optimale pour n'importe quelle récompense. Ces approches ont pour point commun de chercher à minimiser une métrique de la proximité entre l'attribut vectoriel moyen

de l'expert et celui d'une politique donnée. Par mises à jour successives du vecteur de paramètres θ, la politique définie selon un critère d'optimalité dépendant de θ (c'est là qu'intervient la résolution du problème direct car θ spécifie la récompense) va voir son attribut vectoriel moyen se rapprocher de celui de l'expert.

Cette résolution est très problématique. En effet trouver une politique optimisant une récompense est un sujet occupant un pan entier de la littérature. Particulièrement, l'information nécessaire à cet objectif est bien plus étendue que celle nécessaire au calcul d'uniquement l'attribut moyen de l'expert comme cela est nécessaire à notre algorithme. Comme cela est illustré dans Klein *et al.* (2011), l'algorithme d'Abbeel & Ng (2004) ne permet pas de résoudre le problème du pendule inversé à l'aide des seuls échantillons de l'expert car ceux-ci ne couvrent qu'une partie trop petite de l'espace d'état, ne permettant pas la résolution du problème direct en se basant uniquement sur les données de l'expert.

Un argument revenant parfois est celui qui consiste à voir en l'expert non pas un acteur optimal mais un acteur optimal par une certaine marge. C'est à dire que l'action choisie par l'expert doit avoir une qualité strictement supérieure à n'importe quelle autre action. Les approches de planification structurée comme Ratliff *et al.* (2006, 2007a,b); Kolter *et al.* (2008) permettent d'influer sur cette marge.

Particulièrement, notre travail présente des similarités avec celui de Ratliff *et al.* (2006). Les auteurs présentent une méthode d'ARI en posant le problème sous forme d'un *maximum margin classification problem*, notre fonction de risque étant similaire à la leur. La méthode utilisée pour résoudre ce problème peut être ramenée à un algorithme itératif dans la veine de Abbeel & Ng (2004). Vu à un plus haut niveau d'abstraction, la classification a lieu dans l'espace des trajectoires : à un PDM (entrée à classifier) est associée une politique (classe). Le cadre applicatif est déterministe, l'algorithme employé pour la résolution des PDM étant A^*. Il semble difficile de sortir de ce cadre en pratique (bien que l'algorithme soit introduit dans un cadre plus général).

Finalement, (Ratliff *et al.*, 2007a) traitent le problème de l'imitation comme un problème de classification multi-classe. Le principe de leur approche est exactement de résoudre le risque (Eq. (5)). Toutefois, ils ne prennent pas en compte la structure du MDP, ce à quoi nous remédions avec notre contribu-

tion (voir Melo & Lopes (2010) sur l'importance de prendre en compte cette structure). Finalement, dans (Ratliff *et al.*, 2007a) les auteurs utilisent un système de classification multi-classe qui met en jeu une fonction de score. Cette fonction de score $s : X \times Y \rightarrow \mathbb{R}$ est utilisé à chaque état pour prendre une décision de classification i.e. choisir le label (correspondant à une action) $y \in Y$ à appliquer à l'entrée (correspondant à un état) $x \in X$. Leur approche souffre de ne pas profiter de l'introduction de la structure dont profite la nôtre (Eq. (6)).

Notre approche se place en quelque sorte à mi-chemin entre ces deux contributions en proposant d'introduire la structure du PDM (ce que fait la première) au niveau du choix de l'action (niveau d'abstraction de la seconde). Il existe d'autres approches que nous ne passons pas en revue de manière exhaustive, mais à notre connaissance elles nécessitent toutes de résoudre le problème direct.

6. Illustration expérimentale

6.1. Description de la tâche

Le problème du pendule inversé consiste à maintenir en équilibre vertical un pendule à un degré de liberté. Il s'agit d'un problème classique en apprentissage par renforcement. Pour jouer le rôle de notre expert, nous utilisons d'ailleurs la politique trouvée par LSPI, un algorithme de RL décrit par Lagoudakis & Parr (2003). Les différents réglages utilisés, de la valeur de la masse du pendule codée dans notre simulateur jusqu'au choix des fonctions de base sont conformes à ce qui est donné dans cette publication. Notamment les fonctions de bases forment un réseau de gaussienne couvrant l'espace d'état à deux dimensions, la première étant la position angulaire du pendule (la verticale correspondant à un angle de valeur nulle) la seconde étant sa vitesse angulaire.

La politique optimisant la fonction de récompense donnée Fig. 1 (a) parvient systématiquement à maintenir le pendule en équilibre durant 3000 pas de temps. Ces derniers ayant une valeur d'un dixième de seconde, cela signifie que nous interrompons l'expert après 5 minutes de contrôle sans chute du pendule.

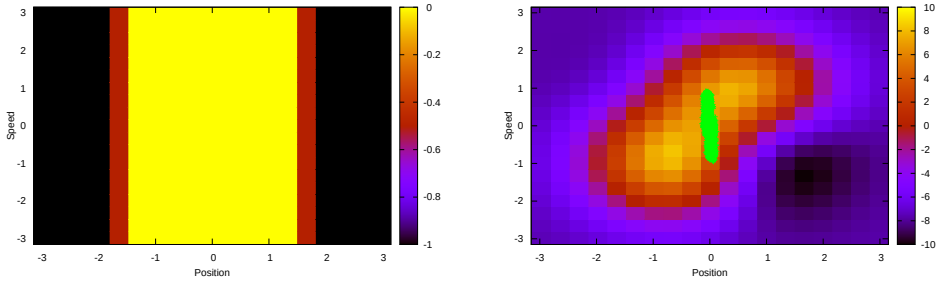
Le but de cette expérience est de montrer que notre algorithme est capable, à partir des seules transitions fournies par l'expert et de la définition des fonctions de bases, de retrouver une récompense telle qu'un agent entraîné sur cette récompense puisse lui aussi faire balancer le pendule durant 5 minutes sans échec. Cela n'est pas trivial étant donné que l'espace d'état est continu et que les données fournies par l'expert n'en couvrent qu'une petite partie (correspondant à la position verticale et son voisinage proche).

Ce dernier point est d'ailleur la raison pour laquelle l'algorithme d'Abbeel & Ng (2004) ne parvient pas à résoudre le problème à l'aide des seules transitions de l'expert. En effet, comme cela a été montré dans (Klein *et al.*, 2011), à moins de fournir un certain nombre de transitions aléatoires couvrant tout l'espace d'état il est impossible d'utiliser les algorithmes itératifs du type de celui d'Abbeel & Ng (2004). Ces algorithmes procèdent par itérations à l'élaboration d'une politique π dont l'attribut vectoriel moyen est utilisée en association à celle de l'expert. Cette politique est obtenue par optimisation d'une récompense sur le PDM dans lequel évolue l'expert ; cette résolution du problème direct à chaque itération ne peut se faire si l'on ne dispose que de données ne couvrant qu'une petite partie de l'espace d'état. Comme notre algorithme ne nécessite pas le calcul d'autre attribut vectoriel moyen que celui de l'expert, la résolution du problème direct est inutile et nous ne rencontrons pas ce problème.

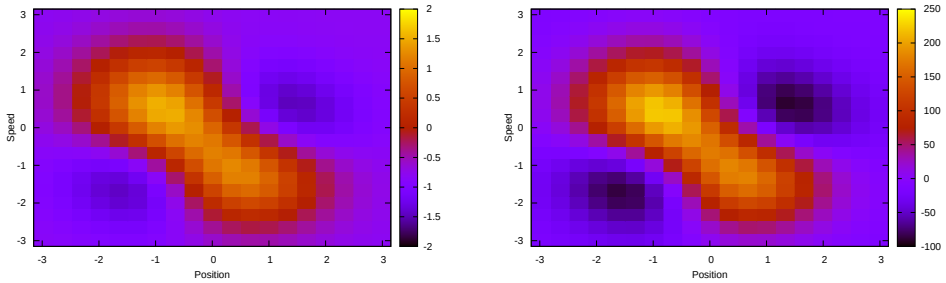
6.2. Illustration des résultats

Afin d'étudier le comportement de notre algorithme, nous avons fait générer à l'expert une base de données D_E de transitions contenant 10 trajectoires de 300 transitions chacune. La fonction l est définie sur les états présents dans D_E telle que $l(s,a) = 0$ si $a = \pi_E(s)$, 1 sinon. Le pas de temps est constant à $\alpha_t = 0.1$ et le nombre d'itérations est de $T = 20$. Le vecteur initial θ_0 est fixé à $[-1... -1]^T$, la récompense étant paramétrée à l'aide du vecteur d'attributs $\psi : S \to \mathbb{R}^p$ correspondant au mélange de gaussiennes défini dans Lagoudakis & Parr (2003). Pour l'approximation de l'attribut vectoriel moyen de l'expert, c'est l'algorithme LSTDμ qui est employé.

Si la fonction de récompense trouvée par notre algorithme (Fig. 1 (b)) diffère de celle fournie à l'expert (Fig. 1 (a)) on constate en revanche que les fonctions de valeurs sont très similaires (Fig. 1 (c) et 1 (d)). Cela est une illus-



(a) Récompense de l'expert (b) Récompense trouvée



(c) Fonction de valeur de l'expert (d) Fonction de valeur de l'agent

FIGURE 1: Résultats fournis par notre algorithme.

tration du fait que le problème de l'ARI est mal posé en ceci qu'il n'y a pas qu'une seule récompense pouvant expliquer une politique.

Avec le nombre d'échantillons fournis par l'expert (3000 en 10 trajectoires de 300 transitions chacune), l'agent parvient systématiquement à maintenir le pendule en équilibre durant 5 minutes, ce qui est notre critère de réussite.

Deux éléments méritent d'être notés. Tout d'abord la faible couverture de l'espace d'état offerte par les échantillons de l'expert. Ceux-ci sont portés en vert sur la Fig. 1 (b) et l'on constate qu'ils n'occupent qu'une toute petite

partie de l'espace : celle dans laquelle le pendule est proche de la verticale. En l'absence de données dans le reste de l'espace d'état il n'est pas possible de pouvoir y inférer avec certitude la récompense. Le second point à noter est que la "vraie" récompense ne se trouve pas dans l'espace d'hypothèse que notre algorithme explore, en effet les attributs vectoriels utilisés ne peuvent qu'approximer la fonction de récompense utilisée sans l'atteindre. Malgré ces deux difficultés, notre algorithme parvient, comme nous l'avons dit, à extraire une récompense qui permet à un agent d'obtenir des performances similaires à celles de l'expert. Le lecteur attentif aura noté la différence d'amplitude des fonctions de valeur Fig. 1 Cela n'a aucune importance, une dilatation laissant la politique optimale invariante.

7. Conclusion

L'algorithme présenté dans cette contribution lève les problèmes les plus contraignants de l'apprentissage par renforcement inverse. En rendant la résolution du problème inverse indépendante de celle du problème direct, nous sommes en mesure d'inférer une récompense cohérente avec le comportement d'un expert en mode purement *batch*.

Comme nous l'avons illustré, les échantillons nécessaires à la réussite de notre algorithmes sont faciles à recueillir. Une simple trace de l'expert, même si elle ne couvre qu'une petite partie de l'espace d'état, suffit à inférer une récompense. Cette récompense, description compacte de la tâche effectuée par l'expert, peut alors être optimisée par un agent dont les capacités diffèrent de celles de l'expert. On rentre dans le cadre du *transfer learning*.

L'estimation de μ_E est centrale dans notre approche. Nous disposons d'un algorithme efficace avec LSTDμ, mais qui présente cependant les mêmes inconvénients que les algorithmes d'estimation d'une fonction de valeur. L'axe d'étude que nous envisageons d'explorer serait de ne plus passer par l'attribut vectoriel moyen en trouvant le moyen d'introduire autrement la structure du PDM dans la démarche de classification que nous avons adoptée. Des tests empiriques sur des problèmes plus complexes sont également envisagés.

Références

ABBEEL P. & NG A. (2004). Apprenticeship learning via inverse reinforce-

ment learning. In *Proc. ICML*.

BRADTKE S. & BARTO A. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning*, **22**(1), 33–57.

CORREA R. & LEMARÉCHAL C. (1993). Convergence of some algorithms for convex minimization. *Mathematical Programming*, **62**(1), 261–275.

KLEIN E., GEIST M. & PIETQUIN O. (2011). Batch, Off-policy and Model-Free Apprenticeship Learning. In *Proc. EWRL 2011)*, Lecture Notes in Computer Science (LNCS) : Springer Verlag - Heidelberg Berlin.

KOLTER J., ABBEEL P. & NG A. (2008). Hierarchical apprenticeship learning with application to quadruped locomotion. In *Proc. NIPS*, volume 20.

LAGOUDAKIS M. & PARR R. (2003). Least-squares policy iteration. *The Journal of Machine Learning Research*, **4**, 1107–1149.

MELO F. & LOPES M. (2010). Learning from demonstration using mdp induced metrics. *Machine Learning and Knowledge Discovery in Databases*, p. 385–401.

NEU G. & SZEPESVÁRI C. (2009). Training parsers by inverse reinforcement learning. *Machine learning*, **77**(2), 303–337.

NG A., HARADA D. & RUSSELL S. (1999). Policy invariance under reward transformations : Theory and application to reward shaping. In *Proc. ICML*, p. 278–287.

NG A. & RUSSELL S. (2000). Algorithms for inverse reinforcement learning. In *Proc. ICML*, p. 663–670 : Morgan Kaufmann Publishers Inc.

RATLIFF N., BAGNELL J. & SRINIVASA S. (2007a). Imitation learning for locomotion and manipulation. In *International Conference on Humanoid Robots*, p. 392–397 : IEEE.

RATLIFF N., BAGNELL J. & ZINKEVICH M. (2006). Maximum margin planning. In *Proc. ICML*, p. 736 : ACM.

RATLIFF N., BRADLEY D., BAGNELL J. & CHESTNUTT J. (2007b). Boosting structured prediction for imitation learning. *Proc. NIPS*, **19**, 1153.

RUSSELL S. (1998). Learning agents for uncertain environments (extended abstract). In *Annual Conference on Computational Learning Theory*, p. 103 : ACM.

SYED U., BOWLING M. & SCHAPIRE R. (2008). Apprenticeship learning using linear programming. In *Proc. ICML*, p. 1032–1039 : ACM.

SYED U. & SCHAPIRE R. (2008). A game-theoretic approach to apprenticeship learning. *Proc. NIPS*, **20**, 1449–1456.

ZIEBART B., MAAS A., BAGNELL J. & DEY A. (2008). Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, p. 1433–1438.