Différences temporelles de Kalman

Cas déterministe

Matthieu Geist[1,2,3] — Olivier Pietquin[1] — Gabriel Fricout[2]

[1] *Equipe IMS*
Supélec, Metz, France
{matthieu.geist,olivier.pietquin}@supelec.fr
[2] *Cluster MC*
ArcelorMittal Research, Maizières-lès-Metz, France
[3] *Equipe-projet CORIDA*
INRIA Nancy - Grand Est, France

RÉSUMÉ. *Un des thèmes importants de l'apprentissage par renforcement est l'approximation en ligne de la fonction de valeur. En plus de leur capacité à prendre en compte de grands espaces d'état, les algorithmes associés devraient présenter certaines caractéristiques comme un apprentissage rapide, la faculté de traquer la solution plutôt que de converger vers elle (particulièrement en raison de l'entrelacement entre contrôle et apprentissage) ou encore la gestion de l'incertitude relative aux estimations faites. Dans cette optique, nous introduisons un cadre de travail général inspiré du filtrage de Kalman que nous nommons* différences temporelles de Kalman. *Une forme d'apprentissage actif utilisant l'information d'incertitude est également introduite, et comparaison est faite à l'état de l'art sur des problèmes classiques.*

ABSTRACT. *A topic of importance in reinforcement learning is online value function approximation. Related algorithms should exhibit some features such as sample efficiency, tracking the solution rather than converging to it (especially because control and learning are interleaved) and maintaining an uncertainty information about approximated values. A Kalman-based Temporal Differences framework is introduced to deal with all these aspects at the same time. A form of active learning which uses the available uncertainty information is also introduced, and the proposed framework is compared to state-of-the-art algorithms on classic benchmarks.*

MOTS-CLÉS : *apprentissage par renforcement, filtrage de Kalman, approximation de la fonction de valeur, gestion de l'incertitude, traque.*

KEYWORDS: *reinforcement learning, Kalman filtering, value function approximation, uncertainty handling, tracking.*

1. Introduction

L'apprentissage par renforcement (AR) est un paradigme général dans lequel un agent apprend à contrôler de manière optimale un système dynamique à partir d'interactions avec ce dernier (Sigaud *et al.*, 2008). A chaque pas de temps, il choisit une action qui modifie l'état du système, ce qui lui apporte une récompense locale, relative à la transition effectuée. L'objectif de cet agent est de maximiser le cumul de récompenses sur le long terme, ce qui est généralement modélisé par une fonction dite de valeur. Un algorithme d'AR devrait présenter certaines caractéristiques importantes. La première est de pouvoir prendre en compte des espaces d'état cardinaux importants, voire continus. Ceci implique d'approcher la fonction de valeur, une représentation exacte n'étant plus possible généralement. Un autre aspect important est l'efficacité en termes d'échantillons : il faudrait que l'agent puisse apprendre un bon contrôle avec aussi peu d'interactions que possible. Un autre aspect important mais peu étudié dans la littérature est la prise en compte de la non-stationnarité. Le cas d'intérêt le plus évident est celui d'un système non stationnaire. Cependant, même si ce dernier est stationnaire, cette capacité est intéressante. En effet, beaucoup d'approches en AR consistent à alterner des phases où la politique est évaluée, c'est-à-dire où l'on évalue la fonction de valeur associée, et des phases d'amélioration de la politique, où l'agent agit de façon gloutonne par rapport à la fonction de valeur apprise. Les échelles de temps correspondantes peuvent beaucoup varier d'une approche à l'autre. En conséquence, l'algorithme en charge d'apprendre la fonction de valeur doit "traquer" la solution plutôt que de converger vers elle, dans la mesure où cette dernière change à chaque amélioration de la politique. D'autres raisons pour traquer une solution plutôt que de converger vers elle, dans le contexte de l'AR et pour un système stationnaire, sont données par Sutton *et al.* (2007). Un autre problème important est le dilemme entre exploration et exploitation. A chaque instant, l'agent doit choisir entre agir de façon supposée optimale, respectivement à sa connaissance imparfaite du monde (exploration), ou chercher à améliorer cette connaissance (exploration). Parmi les approches les plus efficaces, beaucoup utilisent une information d'incertitude, du comptage du nombre de passage par chaque état (Strehl *et al.*, 2004) jusqu'à la quantification de la valeur du gain d'information (Dearden *et al.*, 1998). Un algorithme d'AR devrait donc fournir une information d'incertitude relative à l'estimation des valeurs, ce qui est rarement proposé par les approches permettant l'approximation de la fonction de valeur. Tous ces aspects sont traités par le cadre de travail proposé (Geist *et al.*, 2009a).

2. Contexte et état de l'art

Nous nous plaçons dans le cadre de processus décisionnels de Markov (PDM) déterministes, définis par un tuple $\{S, A, T, R, \gamma\}$ où S est l'espace d'état, A est l'espace d'action, $T : S \times A \rightarrow S$ est une fonction de transition déterministe, $R : S \times A \times S \rightarrow \mathbb{R}$ une fonction de récompense bornée et γ un coefficient d'actualisation. Une politique π associe à chaque état une action : $\pi : S \rightarrow A$. La fonction

de valeur d'une politique donnée est définie par $V^\pi(s) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, \pi]$ où $r_i = R(s_i, a_i, s_{i+1})$ est la récompense associée à la transition (s_i, a_i, s_{i+1}) observée au temps i. La Q-fonction (ou fonction de qualité) est définie de façon semblable, avec cependant un degré de liberté supplémentaire sur la première action : $Q^\pi(s,a) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, a_0 = a, \pi]$. L'objectif de l'AR est de déterminer (à partir d'interactions) la politique π^* qui maximise la fonction de valeur pour chaque état : $\pi^* = \mathrm{argmax}_\pi(V^\pi)$. Deux schémas généraux (parmi d'autres) peuvent mener à la solution. Le premier, nommé *itération de la politique*, consiste à apprendre la fonction de valeur d'une politique donnée, puis à améliorer cette politique, la nouvelle étant gloutonne par rapport à la fonction de valeur apprise. Cela implique de résoudre l'*équation d'évaluation de Bellman*, donnée ici respectivement pour les fonctions de valeur et de qualité (rappelons le déterminisme des transitions) :

$$V^\pi(s) = R(s, \pi(s), s') + \gamma V^\pi(s'),\ s' = T(s, \pi(s)),\ \forall s \quad [1]$$

$$Q^\pi(s,a) = R(s, a, s') + \gamma Q^\pi(s', \pi(s')),\ s' = T(s,a),\ \forall s, a \quad [2]$$

Le second schéma, appelé *itération de la valeur*, permet de trouver directement la politique optimale (*via* la fonction de valeur optimale). Il implique de résoudre l'*équation d'optimalité de Bellman*, qui est donnée ici pour la Q-fonction :

$$Q^*(s,a) = R(s, a, s') + \gamma \max_{b \in A} Q^*(s', b),\ s' = T(s,a),\ \forall s, a \quad [3]$$

L'objectif de cette contribution est de trouver, en ligne et sans connaissance *a priori* du modèle, une solution approchée aux équations de Bellman, pour la fonction de valeur ou de qualité, lorsque l'espace d'état est trop grand pour les approches classiques, et ceci en respectant les contraintes posées section 1.

Pour cela, nous considérons les méthodes dites de différences temporelles (ou TD pour *Temporal Differences*). Elles forment une classe d'algorithmes qui consistent à corriger une représentation paramétrique de la fonction de valeur (ou de qualité) selon l'erreur de différence temporelle (l'erreur TD, définie ci-après) faite sur cette dernière. La plupart de ces approches peuvent s'écrire de la façon générique suivante :

$$\theta_i = \theta_{i-1} + K_i \delta_i \quad [4]$$

Dans cette expression, θ_{i-1} est l'ancienne estimation de la représentation paramétrique de la fonction de valeur, θ_i est sa mise à jour selon la dernière transition observée, δ_i est l'erreur de différence temporelle et K_i est un gain indiquant dans quelle direction la représentation de la fonction de valeur doit être corrigée. Si les espaces d'état S et d'action A sont finis de cardinaux suffisamment faibles, une représentation exacte de la fonction de valeur est possible, et θ est alors un vecteur avec autant de composantes qu'il y a d'états (ou de couples état-action pour la Q-fonction). C'est une représentation dite tabulaire. Si ces espaces sont trop larges, une approximation $\hat{V}_\theta(s)$ est nécessaire. Un choix classique en AR est la paramétrisation linéaire, pour laquelle la fonction de valeur est approchée par $\hat{V}_\theta(s) = \theta^T \phi(s)$ où $\phi(s) = (\phi_1(s), \ldots, \phi_p(s))^T$

est un vecteur composé des fonctions de base, qui doivent être définies à l'avance, et les paramètres sont les poids associés : $\theta = [w_1, \ldots, w_p]^T$. Beaucoup d'algorithmes d'approximation de la fonction de valeur nécessitent une telle représentation pour assurer la convergence (Schoknecht, 2002), ou même pour être applicable (Bradtke *et al.*, 1996). D'autres représentations sont possibles, par exemple un réseau de neurones pour lequel θ est composé des poids des connexions synaptiques associées. Dans l'équation [4], le terme δ_i est l'erreur TD permettant de distinguer trois types d'algorithmes de différences temporelles. Supposons qu'au temps i la transition $(s_i, a_i, r_i, s_{i+1}, a_{i+1})$ soit observée. Pour les algorithmes d'AR de type TD, c'est-à-dire les algorithmes qui visent l'évaluation de la fonction de valeur pour une politique donnée π, l'erreur TD est $\delta_i = r_i + \gamma \hat{V}_{\theta_{i-1}}(s_{i+1}) - \hat{V}_{\theta_{i-1}}(s_i)$. Pour les algorithmes de type SARSA, c'est-à-dire les algorithmes qui ont pour but d'évaluer la fonction de qualité d'une politique donnée π, et étant donné l'approximation $\hat{Q}_{\theta_{i-1}}$ de la Q-fonction, l'erreur TD est $\delta_i = r_i + \gamma \hat{Q}_{\theta_{i-1}}(s_{i+1}, a_{i+1}) - \hat{Q}_{\theta_{i-1}}(s_i, a_i)$. Enfin, pour les algorithmes de type Q-learning, c'est-à-dire les algorithmes dont l'objectif est d'estimer la Q-fonction optimale, l'erreur TD est de la forme $\delta_i = r_i + \gamma \max_{b \in A} \hat{Q}_{\theta_{i-1}}(s_{i+1}, b) - \hat{Q}_{\theta_{i-1}}(s_i, a_i)$.

Le type de différence temporelle utilisé est directement lié au type d'équation de Bellman à résoudre, et donc si l'algorithme associé appartient à la famille de l'itération de la politique ou de la valeur. Le terme K_i est un gain spécifique à chaque méthode. Nous passons en revue certains des plus communs. Pour TD, SARSA et Q-learning (dans leur forme tabulaire), le gain s'écrit $K_i = \alpha_i e_i$ où α_i est un taux d'apprentissage classique dans le domaine de l'approximation stochastique, qui devrait vérifier $\sum_{i=0}^{\infty} \alpha_i = \infty$ et $\sum_{i=0}^{\infty} \alpha_i^2 < \infty$, et e_i est un vecteur unitaire, nul partout sauf en la composante correspondant à l'état s_i (ou à la paire état-action (s_i, a_i)) où il vaut un (fonction de Kronecker). Ces algorithmes ont été étendus au concept de traces d'éligibilité, et le gain s'écrit alors $K_i = \alpha_i \sum_{j=1}^{i} \lambda^{i-j} e_j$ où λ est le coefficient dit d'éligibilité. Ces algorithmes ont également été étendus à la prise en compte d'une représentation approchée de la fonction de valeur. Nous suivons Baird (1995) et les appelons algorithmes directs. Ils visent à minimiser un coût du type $\|V - \hat{V}\|^2$ avec une approche de type *bootstraping*. Sans traces d'éligibilité, le gain s'écrit $K_i = \alpha_i \nabla_{\theta_{i-1}} \hat{V}_{\theta_{i-1}}(s_i)$ où $\nabla_{\theta_{i-1}} \hat{V}_{\theta_{i-1}}(s_i)$ est le gradient selon le vecteur de paramètres de la fonction de valeur paramétrée évaluée en l'état courant. La fonction de valeur peut être remplacée simplement par la fonction de qualité. Les algorithmes directs peuvent également prendre en compte les traces d'éligibilité : $K_i = \alpha_i \sum_{j=1}^{i} \lambda^{i-j} \nabla_{\theta_{i-1}} \hat{V}_{\theta_{i-1}}(s_j)$.

Une autre approche classique est celle des algorithmes résiduels (Baird, 1995), pour lesquels le gain est obtenu par la minimisation de la norme L_2 du résidu de Bellman : $K_i = \alpha_i \nabla_{\theta_{i-1}} \left(\hat{V}_{\theta_{i-1}}(s_i) - \gamma \hat{V}_{\theta_{i-1}}(s_{i+1}) \right)$. La dernière approche que nous passons en revue est l'algorithme LSTD (*Least Squares Temporal Differences*) de Bradtke *et al.* (1996), qui n'est défini que pour une paramétrisation linéaire, et pour lequel le gain est défini récursivement par $K_i = \frac{C_{i-1}\phi(s_i)}{1+(\phi(s_i)-\gamma\phi(s_{i+1}))^T C_{i-1}\phi(s_i)}$ avec $C_i = C_{i-1} - K_i(\phi(s_i) - \gamma\phi(s_{i+1}))^T C_{i-1}$. Cet algorithme vise à minimiser le résidu quadratique de Bellman à l'aide d'une approche par moindres carrés et en utilisant des

variables instrumentales. Il a également été étendu aux traces d'éligibilité, voir Boyan (1999) pour les détails.

Toutes ces approches tentent de converger vers une solution du problème. Nous proposons une approche statistique visant à "traquer" la solution plutôt qu'à converger vers elle. Le filtrage de Kalman (1960) est une réponse standard au problème de la "traque" d'une solution.

3. Différences temporelles de Kalman

A l'origine, le paradigme du filtrage de Kalman vise à traquer en ligne l'état caché (modélisé comme un vecteur aléatoire) d'un système dynamique non stationnaire à partir d'observations indirectes de cet état. L'idée derrière la contribution que nous proposons est d'exprimer l'approximation de la fonction de valeur comme un problème de filtrage : les paramètres deviennent l'état caché à traquer, et les observations sont les récompenses, liées aux paramètres via la transition courante et l'une des équations de Bellman. Ainsi l'approximation de la valeur pourrait profiter des avantages propres au filtrage de Kalman, à savoir apprentissage en ligne et du second ordre, prise en compte de la non-stationnarité ou encore gestion de l'incertitude.

Dans cette section, un point de vue très général est adopté, les algorithmes spécifiques étant dérivés par la suite. Pour l'instant, une transition est notée :

$$t_i = \begin{cases} (s_i, s_{i+1}) \\ (s_i, a_i, s_{i+1}, a_{i+1}) \\ (s_i, a_i, s_{i+1}) \end{cases} \quad [5]$$

selon que l'objectif soit l'évaluation de la fonction de valeur, de qualité, ou l'optimisation de Q. De façon similaire, nous définissons la fonction g_{t_i} comme :

$$g_{t_i}(\theta_i) = \begin{cases} \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) \\ \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \hat{Q}_{\theta_i}(s_{i+1}, a_{i+1}) \\ \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \max_{b \in A} \hat{Q}_{\theta_i}(s_{i+1}, b) \end{cases} \quad [6]$$

Ainsi, tous les schémas de différences temporelles (TD, SARSA et Q-learning) peuvent génériquement s'écrire $\delta_i = r_i - g_{t_i}(\theta_i)$. Avec ces notations, $g_{t_i}(\theta_i)$ peut être vu comme la prédiction de la récompense au temps i en accord avec la représentation θ_i, et δ_i quantifie l'information gagnée en observant la nouvelle récompense r_i. Un point de vue statistique est adopté. Le vecteur de paramètres θ est modélisé comme étant une variable aléatoire suivant une marche aléatoire. Le problème peut s'exprimer sous une forme dite *espace-d'état* :

$$\begin{cases} \theta_i = \theta_{i-1} + v_i & \text{(équation d'évolution)} \\ r_i = g_{t_i}(\theta_i) + n_i & \text{(équation d'observation)} \end{cases} \quad [7]$$

La dénomination espace-d'état (*state-space* en anglais) vient du filtrage de Kalman et de l'automatique, elle n'a pas de lien avec la notion d'état d'un PDM. Cette formulation est fondamentale pour le paradigme proposé. La première équation est l'équation dite d'évolution, elle spécifie que le vecteur de paramètres suit une marche aléatoire dont la moyenne correspond à l'estimation optimale. Le bruit d'observation v_i est centré, blanc, indépendant et de variance P_{v_i} (choisie par le praticien). C'est cette équation qui spécifie l'aspect de traque de la solution en modélisant l'évolution temporelle du vecteur de paramètres. La seconde équation est l'équation dite d'observation, elle lie la transition observée ainsi que la récompense associée à la fonction de valeur (ou de qualité), et donc aux paramètres, à l'aide d'une des équations de Bellman. Le bruit d'observation n_i est supposé blanc, centré, indépendant et de variance P_{n_i} (également choisie par le praticien). Notons que cette hypothèse n'est pas vérifiée pour un PDM stochastique, ce qui est la motivation principale de l'hypothèse de transitions déterministes. Ce point est discuté plus avant dans la section 3.4. Pour un PDM déterministe, ce modèle du bruit d'observation modélise un biais inductif : la solution de l'équation de Bellman considérée n'appartient pas nécessairement à l'espace fonctionnel engendré par l'ensemble des paramètres (la structure d'approximation étant fixée *a priori*). Cette formulation espace-d'état spécifie que les récompenses sont générées en accord avec l'équation d'observation, cette dernière étant conduite par les paramètres aléatoires cachés qui définissent une famille de fonctions dont l'espérance est l'approximation optimale de la fonction de valeur. Si le problème est stationnaire, il n'y a pas de bruit d'évolution, cependant le modèle de marche aléatoire permet d'une part de prendre en compte les non-stationnarités éventuelles et, d'autre part, peut permettre d'éviter les *optima* locaux, son influence étant similaire à une forme de recuit simulé.

3.1. *Coût minimisé*

L'objectif pourrait être d'estimer le vecteur de paramètres qui minimise l'espérance de l'erreur quadratique conditionnée aux récompenses observées depuis l'origine des temps. Cette estimation s'écrit :

$$\hat{\theta}_{i|i} = \underset{\theta}{\operatorname{argmin}}(J_i(\theta)) \text{ avec } J_i(\theta) = E[\|\theta_i - \theta\|^2 | r_{1:i}] \text{ et } r_{1:i} = r_1, \ldots, r_i \quad [8]$$

De façon générale, l'estimateur minimisant l'erreur quadratique moyenne est l'espérance conditionnelle : $\hat{\theta}_{i|i} = E[\theta_i | r_{1:i}]$. Cependant, à part pour des cas spécifiques (notamment le cas linéaire gaussien), cet estimateur ne peut pas être calculé analytiquement. A la place, l'objectif est ici de trouver le meilleur estimateur *linéaire*. Il peut être écrit sous une forme similaire à l'équation [4] : $\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i \tilde{r}_i$. Dans cette équation, $\hat{\theta}_{i|i}$ est l'estimation au temps i, $\hat{\theta}_{i|i-1} = E[\theta_i | r_{1:i-1}]$ est la prédiction de cette estimation en accord avec les récompenses observées dans le passé $r_{1:i-1}$. Pour le modèle de marche aléatoire adopté, le bruit v_i étant blanc et centré, nous avons $\hat{\theta}_{i|i-1} = E[\theta_i | r_{1:i-1}] = E[\theta_{i-1} + v_i | r_{1:i-1}] = E[\theta_{i-1} | r_{1:i-1}] = \hat{\theta}_{i-1|i-1}$. L'innovation $\tilde{r}_i = r_i - \hat{r}_{i|i-1}$ est la différence entre la récompense observée r_i et sa

prédiction $\hat{r}_{i|i-1}$ basée sur la précédente estimation du vecteur de paramètres, donnée par (le bruit d'observation étant aussi blanc et centré) : $\hat{r}_{i|i-1} = E\left[r_i|r_{1:i-1}\right] = E\left[g_{t_i}(\theta_i) + n_i|r_{1:i-1}\right] = E\left[g_{t_i}(\theta_i)|r_{1:i-1}\right]$. Notons que cette innovation $\tilde{r}_i$ n'est pas exactement l'erreur de différence temporelle $\delta_i = r_i - g_{t_i}(\theta_i)$, qui est une variable aléatoire en conséquence de sa dépendance au vecteur aléatoire θ_i : c'est son espérance conditionnée aux données précédemment observées. Etant donnée la mise à jour linéaire postulée, il s'agit de déterminer le gain qui permette la minimisation de [8].

3.2. *Gain optimal*

En utilisant des égalités classiques, la fonction de coût peut se réécrire de la façon suivante (l'opérateur trace associant à une matrice carrée la somme de ses éléments diagonaux) : $J_i(\theta) = E[\|\theta_i - \theta\|^2|r_{1:i}] = E[(\theta_i - \theta)^T(\theta_i - \theta)|r_{1:i}] = \mathrm{trace}(E[(\theta_i - \theta)(\theta_i - \theta)^T|r_{1:i}]) = \mathrm{trace}(\mathrm{cov}(\theta_i - \theta|r_{1:i}))$. Une première étape pour calculer le gain optimal est d'exprimer le terme $\mathrm{cov}(\theta_i - \theta|r_{1:i})$ comme une fonction du gain K_i. Pour ce faire, quelques notations supplémentaires sont introduites :

$$\begin{cases} \tilde{\theta}_{i|i} = \theta_i - \hat{\theta}_{i|i} & \text{et} \quad \tilde{\theta}_{i|i-1} = \theta_i - \hat{\theta}_{i|i-1} \\ P_{i|i} = \mathrm{cov}\left(\tilde{\theta}_{i|i}|r_{1:i}\right) & \text{et} \quad P_{i|i-1} = \mathrm{cov}\left(\tilde{\theta}_{i|i-1}|r_{1:i-1}\right) \\ P_{r_i} = \mathrm{cov}\left(\tilde{r}_i|r_{i|i-1}\right) & \text{et} \quad P_{\theta r_i} = E\left[\tilde{\theta}_{i|i-1}\tilde{r}_i|r_{1|i-1}\right] \end{cases} \quad [9]$$

En utilisant la mise à jour linéaire postulée, et les différents estimateurs étant non biaisés, la covariance peut être développée : $P_{i|i} = \mathrm{cov}(\theta_i - \hat{\theta}_{i|i}|r_{1:i}) = \mathrm{cov}(\theta_i - (\hat{\theta}_{i|i-1} + K_i\tilde{r}_i)|r_{1:i-1}) = \mathrm{cov}(\tilde{\theta}_{i|i-1} - K_i\tilde{r}_i|r_{1:i-1}) = P_{i|i-1} - P_{\theta r_i}K_i^T - K_iP_{\theta r_i}^T + K_iP_{r_i}K_i^T$. Le gain optimal peut ainsi être obtenu en annulant le gradient de la trace de cette matrice. Notons tout d'abord que le gradient étant linéaire, pour trois matrices de dimensions *ad hoc* A, B et C, B étant symétrique, nous avons les identités algébriques $\nabla_A(\mathrm{trace}(ABA^T)) = 2AB$ et $\nabla_A(\mathrm{trace}(AC^T)) = \nabla_A(\mathrm{trace}(CA^T))$, et donc en utilisant l'expression de $P_{i|i}$ qui vient d'être déterminée et les identités précédentes nous avons $\nabla_{K_i}(\mathrm{trace}(P_{i|i})) = 0 \Leftrightarrow K_i = P_{\theta r_i}P_{r_i}^{-1}$. En injectant ce gain optimal K_i dans l'expression de $P_{i|i}$, la matrice de covariance de l'erreur conditionnée aux récompenses observées, nous en obtenons une expressions simplifiée : $P_{i|i} = P_{i|i-1} - K_iP_{r_i}K_i^T$. Il est à noter qu'aucune hypothèse gaussienne n'a été faite pour obtenir ces résultats. Remarquons tout de même que dans le cas linéaire et gaussien, la mise à jour que nous contraignons à la linéarité est optimale.

3.3. *Algorithme général*

L'algorithme le plus général de différences temporelles de Kalman, qui se subdivise en trois parties, peut maintenant être obtenu. La première étape consiste à calculer les prédictions $\hat{\theta}_{i|i-1}$ et $P_{i|i-1}$. Rappelons que pour un modèle de marche aléatoire la prédiction du vecteur de paramètres est égale à son estimation précédente :

$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$. La covariance prédite peut également être calculée analytiquement, $P_{i|i-1} = \text{cov}(\tilde{\theta}_{i|i-1}|r_{1:i-1}) = \text{cov}(\tilde{\theta}_{i-1|i-1} + v_{i-1}|r_{1:i-1}) = P_{i-1|i-1} + P_{v_{i-1}}$. La seconde étape consiste à calculer quelques statistiques d'intérêt. C'est principalement cette partie qui sera spécialisée dans la section suivante. La première statistique à calculer est la prédiction de la récompense $\hat{r}_{i|i-1}$. La seconde est la covariance entre l'erreur sur les paramètres et l'innovation $P_{\theta r_i}$. Etant donné la forme de l'équation d'observation et l'indépendance du bruit d'observation qui est centré, cette statistique peut se réécrire $P_{\theta r_i} = E[(\theta_i - \hat{\theta}_{i|i-1})(g_{t_i}(\theta_i) - \hat{r}_{i|i-1})|r_{1:i-1}]$. Enfin, la dernière statistique à calculer est la variance de l'innovation, qui peut être écrite en utilisant les mêmes arguments $P_{r_i} = E[(g_{t_i}(\theta_i) - \hat{r}_{i|i-1})^2|r_{1:i-1}] + P_{n_i}$. La dernière étape de l'algorithme est la phase de correction qui consiste à calculer le gain K_i et à mettre à jour le vecteur de paramètres $\hat{\theta}_{i|i-1}$ en fonction du gain et de l'innovation, ainsi que la matrice de covariance associée $P_{i|i-1}$. Notons que la méthode proposée étant en ligne et se basant sur la mise à jour d'une représentation existante, elle se doit d'être initialisée avec un *a priori* sur l'espérance $\hat{\theta}_{0|0}$ et la covariance $P_{0|0}$ des paramètres. L'approche générale proposée est résumée dans l'algorithme 1. Le calcul de la variance $P_{i|i}$ peut sembler à première vue inutile, mais cette dernière est utilisée pour calculer les statistiques d'intérêt, comme explicité dans la section 4.

Algorithme 1 : Algorithme KTD général

Initialisation: *a priori* $\hat{\theta}_{0|0}$ et $P_{0|0}$;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Observer la transition t_i ainsi que la récompense associée r_i ;

Phase de prédiction;
$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;
$P_{i|i-1} = P_{i-1|i-1} + P_{v_{i-1}}$;

Calcul des statistiques d'intérêt;
$\hat{r}_{i|i-1} = E[g_{t_i}(\theta_i)|r_{1:i-1}]$;
$P_{\theta r_i} = E[(\theta_i - \hat{\theta}_{i|i-1})(g_{t_i}(\theta_i) - \hat{r}_{i|i-1})|r_{1:i-1}]$;
$P_{r_i} = E[(g_{t_i}(\theta_i) - \hat{r}_{i|i-1})^2|r_{1:i-1}] + P_{n_i}$;

Phase de correction;
$K_i = P_{\theta r_i} P_{r_i}^{-1}$;
$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i(r_i - \hat{r}_{i|i-1})$;
$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

3.4. *Transitions stochastiques*

Le cadre de travail introduit suppose un bruit d'observation blanc. Dans le cas de PDM déterministes, ce bruit modélise le biais inductif causé par l'approximation de la fonction de valeur. Cependant, pour les PDM stochastiques, il inclut également

la stochasticité des transitions et ne peut plus être considéré comme blanc. De façon similaire aux approches minimisant le résidu quadratique de Bellman, comme les algorithmes résiduels de Baird (1995), ce coût est biaisé. Il est possible de montrer que ce biais vaut $\|K_i\|^2 E[\mathrm{cov}_{s'|s_i}(r_i - g_{t_i}(\theta))|r_{1:i-1}]$, où K_i est le gain de Kalman, $\|.\|$ est la norme euclidienne usuelle, la covariance dépend des probabilités de transition et l'espérance est sur les paramètres conditionnés aux observations passées. La preuve, bien que non triviale, n'est pas donnée pour une raison de place. Ce biais, qui est nul pour les transitions déterministes, favorise les fonctions de valeur régulières (Antos *et al.*, 2008), mais cet effet de régularisation ne peut pas être contrôlé. Nous avons une solution à ce problème, basée sur l'introduction d'un bruit coloré (Geist *et al.*, 2009b). Elle n'est pas présentée ici, l'objectif de cet article étant de poser les bases de KTD.

4. Spécialisations

La principale difficulté de KTD est le calcul des statistiques d'intérêt $\hat{r}_{i|i-1}$, $P_{\theta r_i}$ et P_{r_i} (pour lesquelles les prédictions $\hat{\theta}_{i|i-1}$ et $P_{i|i-1}$ sont nécessaires). Dans le cas d'une équation d'observation linéaire, elles peuvent être déterminées analytiquement. Cependant nous nous intéressons au cas plus général non linéaire, voire non dérivable (opérateur $\max$ de l'équation d'optimalité de Bellman). Une linéarisation locale n'est donc pas suffisante. Ce problème de calcul des statistiques d'intérêt peut se formuler comme le calcul des moments d'ordre un et deux de la transformation non linéaire d'une variable aléatoire. Un schéma d'approximation, la transformation non parfumée de Julier *et al.* (2004) (UT pour *unscented transform*), peut être utilisé dans ce but. Comme cette approximation est exacte dans le cas linéaire, et que de plus sa complexité est quadratique, elle fournit le même résultat dans le cas linéaire que la solution analytique, et au même coût, nous ne présenterons donc pas cette dernière. Nous utiliserons cette transformation pour dériver trois algorithmes pratiques.

4.1. *Transformation non parfumée*

Laissons pour l'instant de côté l'AR et le filtrage de Kalman. Soit X un vecteur aléatoire, et $Y = f(X)$ une fonction déterministe potentiellement non linéaire et non dérivable de X. Le problème posé est de calculer la moyenne et la variance de Y en connaissant celles de X. L'idée de base de la transformation non parfumée est qu'il est plus judicieux d'approcher la distribution d'un vecteur aléatoire arbitraire qu'une fonction non linéaire arbitraire. Son principe est d'échantillonner de façon *déterministe* un ensemble de *sigma-points* à partir de la moyenne et de la covariance de X. Les images de ces sigma-points par la fonctionnelle f sont ensuite calculées, et elles sont utilisées pour calculer les statistiques d'intérêt. Ce schéma d'approximation ressemble aux méthodes de Monte-Carlo, cependant ici l'échantillonnage est déterministe et nécessite la génération de moins d'échantillons, en permettant cependant une précision donnée (Julier *et al.*, 2004). Nous décrivons à présent la transformation non parfumée originale. D'autres variantes ont été introduites depuis, mais le principe de

base est le même. Soit n la dimension de X. Un ensemble de $2n+1$ sigma-points et poids associés est calculé comme suit :

$$\begin{cases} x^{(0)} = \bar{X} & w_0 = \frac{\kappa}{n+\kappa}, \quad j=0 \\ x^{(j)} = \bar{X} + \left(\sqrt{(n+\kappa)P_X}\right)_j & w_j = \frac{1}{2(n+\kappa)}, \quad 1 \leq j \leq n \\ x^{(j)} = \bar{X} - \left(\sqrt{(n+\kappa)P_X}\right)_{n-j} & w_j = \frac{1}{2(n+\kappa)}, \quad n+1 \leq j \leq 2n \end{cases} \quad [10]$$

où $\bar{X}$ est la moyenne de X, P_X est sa matrice de variance, κ est un coefficient d'échelle permettant de contrôler la précision de l'UT, et $(\sqrt{(n+\kappa)P_X})_j$ est la $j^{\text{ème}}$ colonne de la décomposition de Cholesky de la matrice $(n+\kappa)P_X$. L'image de chacun de ces points par f est ensuite calculée : $y^{(j)} = f(x^{(j)})$, $0 \leq j \leq 2n$. L'ensemble des sigma-points et de leurs images peut alors être utilisé pour approcher $\bar{Y} \approx \bar{y} = \sum_{j=0}^{2n} w_j y^{(j)}$ et $P_Y \approx \sum_{j=0}^{2n} w_j (y^{(j)} - \bar{y})(y^{(j)} - \bar{y})^T$, les moments d'ordre un et deux de Y, et même $P_{XY} \approx \sum_{j=0}^{2n} w_j (x^{(j)} - \bar{X})(y^{(j)} - \bar{y})^T$, la covariance entre X et Y.

4.2. *KTD-V, KTD-SARSA et KTD-Q*

L'UT ayant été présentée, nous l'utilisons pour spécialiser le cadre de travail général de KTD pour l'évaluation de la fonction de valeur (KTD-V), l'évaluation de la Q-fonction (KTD-SARSA[1]) et l'optimisation directe de la fonction de qualité (KTD-Q). Rappelons que le problème principal de KTD est de calculer les statistiques d'intérêt $\hat{r}_{i|i-1}$, $P_{\theta r_i}$ et P_{r_i}. L'UT est utilisée pour approcher ces quantités. Les trois algorithmes effectuent le même calcul des sigma-points à partir des statistiques connues $\hat{\theta}_{i|i-1}$ et $P_{i|i-1}$, ainsi que les poids associés :

$$\Theta_{i|i-1} = \left\{ \hat{\theta}^{(j)}_{i|i-1},\ 0 \leq j \leq 2p \right\} \quad \text{et} \quad \mathcal{W} = \{w_j,\ 0 \leq j \leq 2p\}$$

Les images des sigma-points sont ensuite calculées, en utilisant l'une des équations d'observation [6]. Cette étape est propre à chaque algorithme :

$$\mathcal{R}_{i|i-1} = \begin{cases} \{\hat{r}^{(j)}_{i|i-1} = \hat{V}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i) - \gamma \hat{V}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}), 0 \leq j \leq 2p\} \\ \{\hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) - \gamma \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, a_{i+1}), 0 \leq j \leq 2p\} \\ \{\hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) - \gamma \max_{b \in A} \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, b), 0 \leq j \leq 2p\} \end{cases}$$

1. Notons que si l'algorithme SARSA original est souvent présenté avec une composante de contrôle, comme une politique ϵ-gloutonne par exemple, ici l'objectif est l'évaluation de la fonction de qualité d'une politique donnée, le contrôle étant considéré à part.

Ensuite les sigma-points et leurs images sont utilisés pour calculer les statistiques d'intérêt. Les équations correspondantes sont les mêmes pour les trois algorithmes :

$$\begin{cases} \hat{r}_{i|i-1} & = \sum_{j=0}^{2p} w_j \hat{r}_{i|i-1}^{(j)} \\ P_{r_i} & = \sum_{j=0}^{2p} w_j (\hat{r}_{i|i-1}^{(j)} - \hat{r}_{i|i-1})^2 + P_{n_i} \\ P_{\theta r_i} & = \sum_{j=0}^{2p} w_j (\hat{\theta}_{i|i-1}^{(j)} - \hat{\theta}_{i|i-1})(\hat{r}_{i|i-1}^{(j)} - \hat{r}_{i|i-1}) \end{cases}$$

Cela donne une implémentation pratique pour KTD pour le cas de l'évaluation de la fonction de valeur, de l'évaluation de la fonction de qualité et de l'optimisation directe de la Q-fonction. L'approche est résumée dans l'algorithme 2. Autant que nous le sachions, KTD-Q est l'une des premières méthodes du second ordre permettant l'approximation de la fonction de qualité dans un schéma d'itération de la valeur (la difficulté étant principalement l'opérateur $\max$).

4.3. *Coût computationnel*

Soit p le nombre de paramètres. L'UT implique de calculer une décomposition de Cholesky qui a une complexité de $O(p^3)$. Cependant, pour les algorithmes considérés, la structure de mise à jour particulière de la matrice de covariance $P_{i|i-1}$ permet de considérer un algorithme spécifique de mise à jour de la décomposition de Cholesky dont la complexité est en $O(p^2)$. Les détails de cette approche "racine carrée", qui utilise une mise à jour incrémentale de la décomposition de Cholesky, sont donnés par van der Merwe *et al.* (2003). Les différents algorithmes impliquent d'évaluer $2p+1$ fois la fonction g_{t_i} à chaque pas de temps. Pour KTD-V et KTD-SARSA et une paramétrisation générale, chaque évaluation est en $O(p)$. Pour KTD-Q, le maximum selon les actions doit être calculé. Notons $\mathcal{A}$ le nombre d'actions si l'espace correspondant est fini, et la complexité de l'algorithme utilisé pour trouver ce maximum sinon (par exemple le nombre d'échantillons tirés pour Monte-Carlo). Ainsi chaque évaluation est bornée par $O(p\mathcal{A})$. Le reste des opérations est de l'algèbre linéaire basique, de complexité au plus $O(p^2)$. Ainsi, la complexité computationnelle (par itération) de KTD-V et KTD-SARSA est $O(p^2)$, et celle de KTD-Q est $O(\mathcal{A}p^2)$. Chaque algorithme requiert de stocker le vecteur de paramètres ainsi que la matrice de covariance associée, leur complexité en mémoire est donc $O(p^2)$.

5. Une forme d'apprentissage actif

Les paramètres étant modélisés par un vecteur aléatoire, et la fonction de valeur paramétrée étant pour un état donné fonction de ces paramètres, elle est elle-même une variable aléatoire. Supposons la fonction de valeur $\hat{V}_\theta$ paramétrée par le vecteur aléatoire θ de moyenne $\bar{\theta}$ et de variance P_θ. Il est possible de calculer $\bar{V}_\theta(s)$ son espérance et $\hat{\sigma}^2_{\hat{V}_\theta}(s)$ sa variance. Pour cela, l'ensemble des sigma-points $\Theta = \{\theta^{(j)},\, 0 \leq j \leq 2p\}$ et les poids associés $\mathcal{W} = \{w_j,\, 0 \leq j \leq 2p\}$ sont calculés à

Algorithme 2 : KTD-V, KTD-SARSA et KTD-Q

Initialisation: *a priori* $\hat{\theta}_{0|0}$ et $P_{0|0}$;

pour $i \leftarrow 1, 2, \ldots$ **faire**

Observer r_i et $t_i = \begin{cases} (s_i, s_{i+1}) \text{ (KTD-V)} \\ (s_i, a_i, s_{i+1}, a_{i+1}) \text{ (KTD-SARSA)} \\ (s_i, a_i, s_{i+1}) \text{ (KTD-Q)} \end{cases}$;

Phase de prédiction;

$\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;

$P_{i|i-1} = P_{i-1|i-1} + P_{v_{i-1}}$;

Calcul des sigma-points ;

$\Theta_{i|i-1} = \left\{ \hat{\theta}^{(j)}_{i|i-1}, \quad 0 \leq j \leq 2p \right\}$ (en utilisant $\hat{\theta}_{i|i-1}$ et $P_{i|i-1}$);

$\mathcal{W} = \{ w_j, \quad 0 \leq j \leq 2p \quad \}$;

$\mathcal{R}_{i|i-1} = \begin{cases} \{\hat{r}^{(j)}_{i|i-1} = \hat{V}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i) - \gamma \hat{V}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}), 0 \leq j \leq 2p\} \\ \{\hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) - \gamma \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, a_{i+1}), 0 \leq j \leq 2p\} \\ \{\hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i) - \gamma \max_{b \in A} \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, b), 0 \leq j \leq 2p\} \end{cases}$;

Calcul des statistiques d'intérêt;

$\hat{r}_{i|i-1} = \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1}$;

$P_{\theta r_i} = \sum_{j=0}^{2p} w_j (\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$;

$P_{r_i} = \sum_{j=0}^{2p} w_j \left(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1} \right)^2 + P_{n_i}$;

Phase de correction;

$K_i = P_{\theta r_i} P_{r_i}^{-1}$;

$\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i \left(r_i - \hat{r}_{i|i-1} \right)$;

$P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$;

partir de $\bar{\theta}$ et P_θ. Leurs images par la fonction de valeur pour un état donné sont obtenues : $\mathcal{V}(s) = \{\hat{V}^{(j)}(s) = \hat{V}_{\theta^{(j)}}(s), \quad 0 \leq j \leq 2p\}$. Enfin, connaissant ces images et les poids associés, il est possible d'approcher les moments $\bar{V}_\theta(s) = \sum_{j=0}^{2p} w_j \hat{V}^{(j)}(s)$ et $\hat{\sigma}^2_{\hat{V}_\theta}(s) = \sum_{j=0}^{2p} w_j (\hat{V}^{(j)}(s) - \hat{V}(s))^2$. La complexité est là encore quadratique. Ainsi, comme à chaque pas de temps une estimation $\hat{\theta}_{i|i}$ et la variance associée $P_{i|i}$ sont disponibles, l'incertitude sur les paramètres peut être propagée à la fonction de valeur (ou de qualité), si nécessaire.

Un exemple d'utilisation effective de cette information d'incertitude est proposé. L'algorithme KTD-Q est dit *off-policy*, car la politique apprise π (la politique optimale π^* dans ce cas) est différente de la politique suivie ou comportementale (que nous noterons b). Une question naturelle est de savoir quelle politique comportementale

permet l'apprentissage le plus rapide de la politique optimale. Un élément de réponse utilisant l'information d'incertitude est proposé ici. Soit i l'index temporel courant. Le système à contrôler est dans un état s_i, et l'agent doit choisir une action a_i. Les estimations $\theta_{i-1|i-1}$ et $P_{i-1|i-1}$ sont disponibles. Elles peuvent être utilisées pour approcher l'incertitude de la Q-fonction paramétrée par θ_{i-1} en l'état s_i pour chaque action a. La variance associée est notée $\sigma^2_{Q_{\theta_{i-1}}}(s_i, a)$. L'action a_i est ensuite choisie selon la politique comportementale aléatoire b définie par :

$$b(a_i|s_i) = \frac{\sigma_{Q_{\theta_{i-1}}}(s_i, a_i)}{\sum_{a \in A} \sigma_{Q_{\theta_{i-1}}}(s_i, a)} \quad [11]$$

Une politique totalement exploratrice privilégiant les actions pour lesquelles l'incertitude est la plus grande est ainsi obtenue. Ce n'est qu'une façon parmi d'autres d'utiliser cette information d'incertitude, mais elle permet de montrer que la variance sur les valeurs obtenues dans le contexte de KTD a du sens (voir section 6.2).

6. Expérimentations

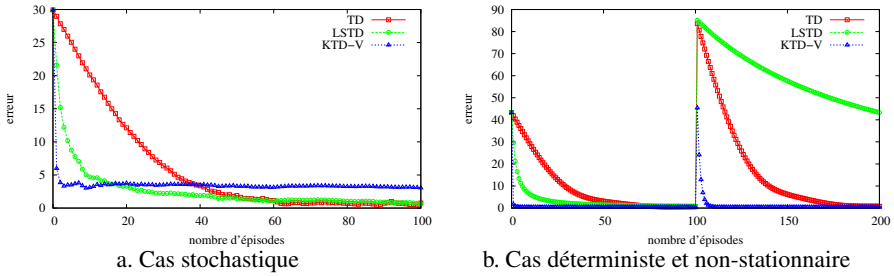
Dans cette section est proposé un ensemble de tests de référence classiques en AR de façon à comparer KTD à différents algorithmes de l'état de l'art et à illustrer les différents aspects de cette approche, à savoir la robustesse à la non-stationnarité (chaîne de Boyan, *mountain car*), l'incertitude de la valeur utilisée pour une forme d'apprentissage actif (pendule inversé), ainsi que l'efficacité en termes d'échantillons (toutes les expériences). Nous illustrons également le biais causé par les transitions stochastiques (chaîne de Boyan). Les autres algorithmes considérés sont TD, SARSA, Q-learning et LSTD. Nous ne considérons pas leurs extensions aux traces d'éligibilité, dans la mesure où LSTD produit de meilleurs résultats que TD(λ) et où varier la valeur du facteur d'éligibilité a peu d'incidence sur les performance de LSTD(λ), comme noté par Boyan (1999).

6.1. *Chaîne de Boyan*

La première expérimentation est la chaîne de Boyan (1999). Le but est d'une part de montrer l'efficacité de KTD en termes d'échantillons et sa capacité à prendre en compte un environnement non stationnaire sur une version déterministe du problème et, d'autre part, d'illustrer le problème posé par les transitions stochastiques.

La chaîne de Boyan est une chaîne de Markov valuée à 13 états dont l'état s^0 est absorbant, s^1 transite vers s^0 avec une probabilité de 1 et une récompense de -2, et s^i transite vers s^{i-1} ou s^{i-2}, $2 \leq i \leq 12$, pour chacun avec une probabilité de 0.5 et une récompense de -3. Pour cette expérience, KTD-V est comparé à TD ainsi qu'à LSTD. La paramétrisation est linéaire, et les vecteurs de base $\phi(s)$ pour les états s^{12}, s^8, s^4 et s^0 sont respectivement $[1,0,0,0]^T$, $[0,1,0,0]^T$, $[0,0,1,0]^T$ et $[0,0,0,1]^T$. Pour les

autres états, ils sont obtenus par interpolation linéaire. L'approximation de la fonction de valeur est donc $\hat{V}_\theta(s) = \theta^T \phi(s)$. La fonction de valeur optimale est linéaire en ces bases, et le vecteur de paramètres optimal correspondant est $\theta^* = [-24, -16, -8, 0]^T$. La performance est mesurée avec la distance euclidienne $\|\theta - \theta^*\|$ entre le vecteur de paramètres courant et le vecteur optimal. Le coefficient d'actualisation γ est fixé à 1 pour cette tâche épisodique. Pour TD, le taux d'apprentissage est fixé à $\alpha_i = 0.1$. Pour LSTD et KTD-V l'*a priori* est fixé à $P_{0|0} = I$, où I est la matrice identité. Etant donné le lien qu'il existe entre KTD et LSTD (voir section 7), nous pensons équitable d'utiliser le même *a priori*. Pour KTD-V, le bruit d'observation est fixé à $P_{n_i} = 10^{-3}$ et le bruit d'évolution à $P_{v_i} = 0I$. Choisir ces paramètres requiert un peu de pratique, mais pas forcément plus que le choix d'un taux d'apprentissage pour d'autres algorithmes. Pour toutes les méthodes considérées, le vecteur de paramètres est initialisé à zéro. Les résultats sont présentés figure 1a. LSTD converge plus rapidement que TD, comme attendu, et KTD-V converge encore plus vite (rappelons que l'*a priori* est le même pour KTD et LSTD). Cependant, ce dernier algorithme ne converge pas vers le vecteur de paramètres optimal, ce qui s'explique par le fait que la fonction de coût minimisée est biaisée. Cette expérience a été menée pour montrer le problème causé par les transitions stochastiques, et nous nous concentrons à présent sur des PDM déterministes.



a. Cas stochastique b. Cas déterministe et non-stationnaire

Figure 1. *Chaîne de Boyan*

La chaîne de Boyan est rendue déterministe en posant la probabilité de transiter de s^i à s^{i-1} à 1. KTD-V est à nouveau comparé à LSTD et TD. De plus, pour simuler un changement dans le PDM, et donc la non-stationnarité, le signe de la récompense est inversé à partir du $100^{\text{ème}}$ épisode. La fonction de valeur optimale est toujours linéaire en les fonctions de base, avec $\theta^*_{(-)} = [-35, -23, -11, 0]^T$ avant le changement, et $\theta^*_{(+)} = -\theta^*_{(-)}$ après. Les paramètres des différents algorithmes sont les mêmes, sauf le bruit d'évolution qui est maintenant fixé à $P_{v_i} = 10^{-3} I$. Les résultats sont présentés sur la figure 1b. A nouveau KTD-V converge plus rapidement que LSTD et TD, cependant maintenant vers le vecteur de paramètres optimal, l'environnement étant déterministe. Après le changement de récompense, LSTD est très lent à converger, à cause de la non-stationnarité induite. TD s'adapte plus rapidement, le taux d'apprentissage étant constant. KTD-V s'adapte quant à lui très rapidement. Ainsi, si KTD-V est biaisé dans le cas stochastique, il converge plus vite et s'adapte plus rapidement

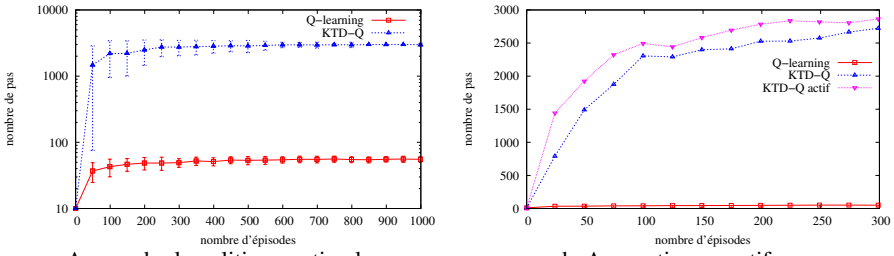
que TD et LSTD dans le cas déterministe. Cette capacité à prendre en compte les non-stationnarités est importante pour suivre la dynamique de la fonction de valeur. Même si l'environnement est stationnaire, cela peut être utile dans un contexte de contrôle, comme illustré dans la section 6.3.

6.2. *Pendule inversé*

La seconde expérience est le pendule inversé tel que décrit par Lagoudakis *et al.* (2003). Le but est ici de comparer deux algorithmes de type itération de la valeur, à savoir KTD-Q et Q-learning avec approximation de la Q-fonction, qui ont tous deux pour objectif d'estimer directement la fonction de qualité optimale. Autant que nous le sachions, KTD-Q est le seul algorithme d'ordre deux qui suive un schéma d'itération de la valeur, c'est pourquoi nous ne considérons que le Q-learning en comparaison. Cette tâche nécessite de balancer un pendule de longueur et masse inconnues de façon à ce qu'il reste vertical en appliquant des forces au chariot sur lequel il est fixé. Trois actions sont possibles : pousser à gauche (-1), pousser à droite ($+1$), ou ne rien faire (0). L'état du système est donné par la position angulaire ω et la vitesse angulaire $\dot{\omega}$. Les transitions déterministes sont calculées grâce à la dynamique du système physique, $\ddot{\omega} = \frac{g\sin(\omega) - \beta ml\dot{\omega}^2\sin(2\omega)/2 - 50\beta\cos(\omega)a}{4l/3 - \beta ml\cos^2(\omega)}$, où g est la constante de gravitation, m et l sont la masse et la longueur du pendule, M la masse du chariot et $\beta = \frac{1}{m+M}$. Une récompense nulle est donnée tant que la position angulaire appartient à l'intervalle $[-\frac{\pi}{2}, \frac{\pi}{2}]$. Sinon, l'épisode se termine et une récompense de -1 est donnée. La paramétrisation est donnée par un terme constant et un ensemble de 9 noyaux gaussiens équi-répartis (centrés en $\{-\frac{\pi}{4}, 0, \frac{\pi}{4}\} \times \{-1, 0, 1\}$ et d'écart type 1), cela pour chaque action. Il y a donc 30 fonctions de base. Le facteur d'actualisation γ est fixé à $0,95$.

Nous comparons la capacité des deux algorithmes à apprendre la politique optimale. Pour Q-learning, le taux d'apprentissage est fixé à $\alpha_i = \alpha_0 \frac{n_0+1}{n_0+i}$ où $\alpha_0 = 0,5$ et $n_0 = 200$, en accord avec Lagoudakis *et al.* (2003). Pour KTD-Q, les paramètres sont $P_{0|0} = 10I$, $P_{n_i} = 1$ et $P_{v_i} = 0I$. Les vecteurs de paramètres sont initialisés à zéro. La politique suivie par l'agent est aléatoire (équi-probabilité des actions), et les deux algorithmes apprennent à partir des mêmes trajectoires. Le pendule est initialisé dans un état aléatoire proche de l'équilibre $(0, 0)$. La longueur moyenne d'un tel épisode aléatoire est d'environ 10 pas. Les résultats sont présentés figure 2a. Pour chaque essai, l'apprentissage est fait sur 1 000 épisodes. Tous les 50 épisodes, l'apprentissage est gelé et la politique courante est testée. Pour cela, l'agent est initialisé dans un état aléatoire proche de l'équilibre et la politique gloutonne est suivie. Chaque test est répété 100 fois et moyenné (l'évaluation de la performance de la politique gloutonne courante étant initialisée dans un état aléatoire). La mesure de performance est le nombre de pas d'un épisode. Le nombre maximum de pas autorisé est de 3 000, ce qui correspond à maintenir le pendule pendant 5 minutes. Les résultats de la figure 2a sont moyennés sur 100 essais (chaque essai correspondant à 1 000 épisodes, la politique étant testée tous les 50 épisodes) et présentés en échelle semi-logarithmique. Asymptotiquement, KTD-Q apprend la politique optimale (c'est-à-dire maintenir le pendule

pendant le nombre maximum de pas autorisés) et de bonnes politiques sont apprises après seulement quelques dizaines d'épisodes. Avec le même nombre d'épisodes et la même paramétrisation, Q-learning échoue à apprendre une politique qui permette de maintenir le pendule pendant plus de quelques secondes, ce qui est en accord avec les résultats présentés par Lagoudakis *et al.* (2003).



a. Apprendre la politique optimale b. Apprentissage actif

Figure 2. *Pendule inversé*

Nous expérimentons la forme d'apprentissage actif introduite section 5, l'algorithme correspondant est appelé KTD-Q actif. La politique suivie n'est plus uniformément aléatoire, mais pondérée par l'incertitude associée aux actions. La longueur moyenne d'un tel épisode est de 11 pas, ce qui change peu par rapport à une politique totalement aléatoire. La durée d'un épisode ne peut donc être que faiblement responsable de l'amélioration des résultats. Pour chaque essai, l'apprentissage est fait sur 300 épisodes. Moins d'épisodes sont considérés pour montrer l'accélération de la convergence (notons qu'asymptotiquement les deux variations de KTD se comportent aussi bien). Tous les 25 épisodes l'apprentissage est gelé et les politiques sont évaluées comme précédemment, avec la même mesure de performance. Les résultats de la figure 2b sont moyennés sur 100 essais. Notons que l'échelle n'est plus logarithmique. Cette expérience compare le KTD-Q actif à KTD-Q et Q-learning. En comparant les deux variantes de KTD, il est clair que choisir les actions en fonction de l'incertitude accélère la convergence. Cette dernière est quasiment doublée sur les 100 premiers épisodes : par exemple, une performance moyenne de 1500 pas est obtenue après seulement 25 épisodes avec le KTD-Q actif, alors qu'elle n'est atteinte qu'après environ 50 épisodes par KTD-Q.

6.3. *Mountain car*

La dernière expérience est le *mountain car* telle que décrite par Sutton *et al.* (1998). L'objectif ici est d'illustrer le comportement des algorithmes dans le cadre d'un schéma d'itération de la politique optimiste : apprendre en contrôlant induit des dynamiques de la valeur non stationnaires, et différents algorithmes d'évaluation de la politique sont comparés. Cette tâche consiste à conduire un véhicule en haut d'une route de montagne, le véhicule n'étant pas assez puissant pour le faire sans élan (c'est ce qui rend cette tâche intéressante, il faut s'éloigner du but pour pouvoir l'atteindre).

L'état est donné par la position et la vitesse $(x, \dot{x}) \in [-1.2, 0.5] \times [-0.07, 0.07]$. Les trois actions possibles sont aller à gauche (-1), à droite $(+1)$ ou ne rien faire (0). La dynamique du système est donnée par $\dot{x}_{i+1} = \text{bound}[\dot{x}_i + 10^{-3}(a_i - 2.5\cos(3x_i))]$ et $x_{i+1} = \text{bound}[x_i + \dot{x}_{i+1}]$, où l'opérateur bound force les bornes de la position et de la vitesse. Quand la position atteint la borne inférieure, la vitesse est mise à zéro. Quand elle atteint la borne supérieure, l'épisode se termine avec une récompense nulle. La récompense est de -1 le reste du temps. Le facteur d'actualisation est fixé à $0,1$. L'état est normalisé, et la paramétrisation est composée d'un terme constant et d'un ensemble de 9 noyaux gaussiens équi-répartis (centrés en $\{0, 0.5, 1\} \times \{0, 0.5, 1\}$ et d'écart type $0,1$), cela pour chaque action. Il y a donc 30 fonctions de base. Cette expérience compare SARSA avec approximation de la fonction de qualité, LSTD et KTD-SARSA, dans un contexte d'itération optimiste de la politique. La politique suivie est ϵ-gloutonne, avec $\epsilon = 0.1$. Pour SARSA, le taux d'apprentissage est fixé à $\alpha_i = 0.1$. Pour LSTD l'*a priori* est fixé à $10I$. Pour KTD-SARSA, le même *a priori* est utilisé, et les variances de bruit sont fixées à $P_{n_i} = 1$ et $P_{v_i} = 0.05I$. Pour tous les algorithmes le vecteur de paramètres est initialisé à zéro. Chaque épisode commence dans un état aléatoire (tirage uniforme sur le domaine). Un maximum de 1 500 pas est autorisé. Pour chaque essai, l'apprentissage se fait sur 200 épisodes, et la figure 3 montre la longueur de chaque épisode moyennée sur 300 essais. KTD-SARSA converge plus rapidement et vers une meilleure politique que LSTD, qui se comporte mieux que SARSA avec approximation de la Q-fonction. De meilleurs résultats ont peut-être été rapportés dans la littérature pour une paramétrisation de type *tile-coding*, cependant la paramétrisation choisie ici est plus brute et implique bien moins de paramètres. De plus, il est rapporté par Sutton *et al.* (1998) que même avec une paramétrisation de type *tile-coding* et un taux d'apprentissage judicieusement choisi, il faut une centaine d'épisodes pour atteindre une politique quasi optimale, ce qui est sensiblement plus que pour notre approche. Le fait de suivre une politique ϵ-gloutonne implique la non-stationnarité de la Q-fonction apprise, ce qui explique que LSTD échoue à trouver une politique quasi optimale (des résultats similaires sont obtenus par Phua *et al.* (2007)). KTD-SARSA obtient des politiques optimales très rapidement, après seulement quelques dizaines d'épisodes. L'apprentissage est également plus stable.

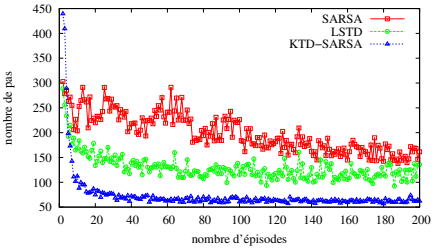


Figure 3. *Mountain car*

7. Discussion et travaux similaires

Des approches apparentées ont été proposées précédemment dans la littérature. Engel (2005) modélise le problème de l'approximation de fonction de valeur en apprentissage par renforcement à l'aide d'un modèle génératif basé sur des processus Gaussiens. Le point de vue adopté est donc sensiblement différent de celui de cette contribution. Les différences principales sont que GPTD permet une représentation non paramétrique, mais nécessite la linéarité des modèles considérés et ne peut pas prendre en compte de bruit d'évolution, alors que KTD nécessite une représentation paramétrique (et donc un choix de d'architecture d'approximation, par exemple les fonctions de base dans le cas d'une paramétrisation linéaire), mais peut prendre en compte des non-linéarités ainsi qu'un bruit d'évolution (ce qui implique de pouvoir traquer la solution plutôt que de converger vers elle, avec toutes les implications que nous avons présentées). De plus, le principe de représentation non paramétrique de GPTD peut être adapté à KTD, comme esquissé par Geist *et al.* (2008), à condition de connaître à l'avance l'espace d'état et d'action. Ce cadre de travail permet également de dériver une information d'incertitude, qui est illustrée par Engel (2005) mais non utilisée dans le cadre de l'apprentissage. Comme le filtrage de Kalman est fortement lié à la minimisation par moindres carrés (c'en est en fait une généralisation), notre approche partage des similitudes avec LSTD (Bradtke *et al.*, 1996), cependant sans prendre en compte le concept de variable instrumentale, introduit pour pouvoir traiter les transitions stochastiques. Choi *et al.* (2006) proposent une forme de filtres de Kalman conçu pour approcher le point fixe de l'opérateur de Bellman (ou de façon équivalente approcher la solution de l'équation de Bellman) dans le cas d'une paramétrisation linéaire de la valeur. Cette approche peut approximativement être vue comme une version "bootstrapée" de KTD-V. A la place de l'équation d'observation que nous considérons, la suivante est utilisée : $r_i + \gamma\phi(s_{i+1})^T\hat{\theta}_{i-1|i-1} = \phi(s_i)^T\theta_i + n_i$. Autrement dit, la récompense n'est pas considérée comme l'observation, mais une approximation de la fonction de valeur est utilisée pour calculer une pseudo-observation (ou observation "bootstrapée"), et la mise à jour du filtre est faite de façon à corriger l'erreur entre la fonction de valeur de l'état courant et la pseudo-observation. Phua *et al.* (2007) utilisent un banc de filtres de Kalman pour apprendre les paramètres d'une représentation linéaire par morceaux de la fonction de valeur (un filtre de Kalman par morceau linéaire). Cette méthode peut grossièrement être vue comme un cas particulier de KTD, cependant des différences existent : un banc de filtres est utilisé plutôt qu'un seul, et la paramétrisation est linéaire, fait exploité pour développer des spécificités de l'algorithme (notamment concernant la mise à jour des paramètres).

Le cadre de travail proposé présente certains aspects intéressants. Premièrement, il ne suppose pas la stationnarité. Une application immédiate est la prise en compte d'environnements non stationnaires. Un aspect peut-être encore plus intéressant est le cas du contrôle. Par exemple, l'algorithme LSTD est connu pour mal se comporter lorsqu'il est combiné avec un schéma d'itération de la politique optimiste (politique ϵ-gloutonne par exemple), à cause des non-stationnarités induites par ce schéma spécifique d'apprentissage et de contrôle. Dans la même idée, Bhatnagar *et al.* (2008)

préfèrent TD à LSTD comme acteur dans l'approche acteur-critique qu'il proposent, pour le même problème, alors que TD est moins efficace en termes d'échantillons. Le filtrage de Kalman et donc les algorithmes proposés sont robustes aux problèmes de non-stationnarité. Deuxièmement, comme le vecteur de paramètres est modélisé par une variable aléatoire, il est possible de calculer une information d'incertitude sur la valeur des états, comme explicité dans la section 5. Nous l'utilisons dans une forme d'apprentissage actif, mais cette incertitude pourrait être utile pour traiter du problème plus général du dilemme entre exploration et exploitation, en s'inspirant par exemple des travaux de Dearden *et al.* (1998) ou encore Strehl *et al.* (2006). Enfin, le cadre de KTD devrait assez naturellement pouvoir être étendu au problème de l'observabilité partielle. En effet, le problème d'inférer l'état d'un système à partir d'observations est un problème pouvant profiter du filtrage bayésien dont le formalisme est proche de celui proposé. Il est connu qu'un PDM partiellement observable (PDMPO) peut être exprimé comme un PDM dont les états sont en fait des distributions sur les états du PDMPO. Si ces distributions peuvent être estimées (par exemple en utilisant une méthode de filtrage), elles peuvent être naturellement prises en compte par KTD. La fonction de valeur dépend déjà de la distribution sur les paramètres, l'extension à une distribution sur les états peut se faire de la même façon. Une difficulté éventuelle de KTD peut être le choix des différents paramètres. Cependant, d'une part la nécessité de choisir certains paramètres n'est pas propre à notre approche (taux d'apprentissage pour TD, *a priori* pour LSTD, *et cetera*), et d'autre part KTD peut grandement bénéficier de la vaste littérature sur le filtrage adaptatif, domaine étudié depuis plusieurs décennies.

8. Conclusion et perspectives

Dans cette contribution nous avons proposé un nouveau paradigme pour l'approximation de la fonction de valeur en apprentissage par renforcement basé sur un cadre de travail statistique inspiré du filtrage de Kalman. Le paradigme général des différences temporelles de Kalman a été présenté section 3. Il a ensuite été spécialisé en un certain nombre d'algorithmes, à savoir KTD-V, KTD-SARSA et KTD-Q, à l'aide de la transformation non parfumée. D'autres points importants ont été abordés : complexité des algorithmes, problème posé par les transitions stochastiques, calcul de l'incertitude de la valeur et son utilisation dans une forme d'apprentissage actif. Les avantages de ce cadre de travail ont été discutés, notamment l'efficacité en termes d'échantillons (*id est* un apprentissage plus rapide pour les algorithmes proposés que pour l'état de l'art), la capacité à prendre en compte le cas non stationnaire et l'information d'incertitude inhérente au modèle, et ils ont été illustrés section 6, avec des comparaisons à certains algorithmes de l'état de l'art. De plus, les algorithmes proposés sont en ligne et permettent de faire du contrôle. Autant que nous le sachions, nous avons également proposé le premier algorithme de la littérature du type itération de la valeur étant du second ordre (KTD-Q). Si les résultats obtenus sont satisfaisants, il y a encore des perspectives intéressantes à développer. Premièrement, le cas des PDMs stochastiques est important, et nous développons une méthode basée sur un bruit d'observation co-

loré pour le prendre en compte. Un choix plus automatique des différents paramètres pourrait certainement bénéficier de la vaste littérature sur le filtrage adaptatif, ce point est à creuser. Il pourrait également être intéressant d'étendre le paradigme proposé au cas des PDMPO, ce qui pourrait se faire *a priori* assez naturellement pour les raisons exposées section 7. L'information d'incertitude inhérente au cadre de travail introduit pourrait être utile pour traiter du dilemme entre exploration et exploitation. Enfin, nous souhaitons utiliser KTD dans le cadre d'une architecture acteur-critique. Plus particulièrement, le modèle acteur-critique incrémental basé sur un gradient naturel proposé par Bhatnagar *et al.* (2008) pourrait bénéficier d'un algorithme d'ordre deux pour le critique, cependant TD y est préféré à LSTD en raison de l'aspect non stationnaire. Nous pensons KTD être une alternative intéressante pour le critique.

Remerciements

O. Pietquin souhaite remercier la région Lorraine ainsi que la communauté européenne (projet CLASSiC, FP7/2007-2013, subvention 216594) pour leur support financier.

9. Bibliographie

Antos A., Szepesvári C., Munos R., « Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path », *Machine Learning*, vol. 71, n° 1, p. 89-129, April, 2008.

Baird L. C., « Residual Algorithms: Reinforcement Learning with Function Approximation », *Proc. of the International Conference on Machine Learning (ICML 95)*, p. 30-37, 1995.

Bhatnagar S., Sutton R. S., Ghavamzadeh M., Lee M., « Incremental Natural Actor-Critic Algorithms », *Proceedings of the Twenty-First Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada, 2008.

Boyan J. A., « Technical Update: Least-Squares Temporal Difference Learning », *Machine Learning*, vol. 49, n° 2-3, p. 233-246, 1999.

Bradtke S. J., Barto A. G., « Linear Least-Squares Algorithms for Temporal Difference Learning », *Machine Learning*, vol. 22, n° 1-3, p. 33-57, 1996.

Choi D., Roy B. V., « A Generalized Kalman Filter for Fixed Point Approximation and Efficient Temporal-Difference Learning », *Discrete Event Dynamic Systems*, vol. 16, p. 207-239, 2006.

Dearden R., Friedman N., Russell S. J., « Bayesian Q-Learning », *AAAI/IAAI*, p. 761-768, 1998.

Engel Y., Algorithms and Representations for Reinforcement Learning, PhD thesis, Hebrew University, April, 2005.

Geist M., Pietquin O., Fricout G., « Filtrage bayésien de la récompense », *Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA 2008)*, Metz, France, 2008.

Geist M., Pietquin O., Fricout G., « Différences Temporelles de Kalman », *Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA 2009)*, Paris, France, June, 2009a.

Geist M., Pietquin O., Fricout G., « Différences Temporelles de Kalman : le cas stochastique », *Journées Francophones de Planification, Décision et Apprentissage pour la conduite de systèmes (JFPDA 2009)*, Paris, France, 2009b.

Julier S. J., Uhlmann J. K., « Unscented filtering and nonlinear estimation », *Proceedings of the IEEE*, vol. 92, n° 3, p. 401-422, 2004.

Kalman R. E., « A New Approach to Linear Filtering and Prediction Problems », *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, n° Series D, p. 35-45, 1960.

Lagoudakis M. G., Parr R., « Least-Squares Policy Iteration », *Journal of Machine Learning Research*, vol. 4, p. 1107-1149, 2003.

Phua C. W., Fitch R., « Tracking Value Function Dynamics to Improve Reinforcement Learning with Piecewise Linear Function Approximation », *International Conference on Machine Learning (ICML 07)*, 2007.

Schoknecht R., « Optimality of Reinforcement Learning Algorithms with Linear Function Approximation », *Conference on Neural Information Processing Systems (NIPS 15)*, 2002.

Sigaud O., Buffet O., *Processus décisionnels de Markov en intelligence artificielle*, Hermes Science Publications / Lavoisier, 2008.

Strehl A. L., Li L., Wiewiora E., Langford J., Littman M. L., « PAC Model-Free Reinforcement Learning », *23rd International Conference on Machine Learning (ICML 2006)*, Pittsburgh, PA, USA, p. 881-888, 2006.

Strehl A. L., Littman M. L., « An Empirical Evaluation of Interval Estimation for Markov Decision Processes », *16th IEEE International on Tools with Artificial Intelligence Conference (ICTAI 2004)*, Boca Raton, FL, USA, p. 128-135, 2004.

Sutton R. S., Barto A. G., *Reinforcement Learning: An Introduction*, 3rd edn, The MIT Press, March, 1998.

Sutton R. S., Koop A., Silver D., « On the role of tracking in stationary environments », *ICML '07: Proceedings of the 24th international conference on Machine learning*, ACM, New York, NY, USA, p. 871-878, 2007.

van der Merwe R., Wan E., « Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models », *Proceedings of the Workshop on Advances in Machine Learning*, Montreal, Canada, June, 2003.