Apprentissage off-policy appliqué à un système de dialogue basé sur les PDMPO

Lucie Daubigney, Matthieu Geist, Olivier Pietquin

Apprentissage *off-policy* appliqué à un système de dialogue basé sur les PDMPO

Lucie Daubigney[1,3], Matthieu Geist[1] et Olivier Pietquin[1,2]

[1]SUPELEC - IMS (Metz, France)
[2]UMI 2958 (GeorgiaTech-CNRS) (Metz, France)
[3]Project Team MAIA, Loria (Nancy, France)

prénom.nom@supelec.fr

Résumé

L'apprentissage par renforcement (AR) fait maintenant partie de l'état de l'art dans le domaine de l'optimisation de systèmes de dialogues vocaux. La plupart des méthodes appliquées aux systèmes de dialogue basées sur l'AR, comme par exemple celles qui utilisent des processus gaussiens, requièrent de tester des changements plus ou moins aléatoires dans la politique. Cette manière de procéder est appelée apprentissage « on-policy ». Néanmoins, celle-ci peut induire des comportements de la part du système incohérents aux yeux de l'utilisateur. Les algorithmes devraient idéalement trouver la politique optimale d'après l'observation d'interactions générées par une politique sous-optimale mais proposant un comportement cohérent à l'utilisateur : c'est l'apprentissage « off-policy ». Dans cette contribution, un algorithme efficace sur les échantillons permettant l'apprentissage off-policy et en ligne de la politique optimale est proposé. Cet algorithme, combiné à une représentation compacte, non-linéaire de la fonction de valeur (un réseau de neurones), permet de gérer des systèmes à grande échelle.

Mots Clef

Apprentissage par renforcement, Système de dialogue vocal

Abstract

Reinforcement learning is now part of the state of the art in the domain of spoken dialogue systems optimisation. Most performant RL methods applied to spoken dialogue systems, such as those based on Gaussian Processes, require to test more or less random changes in the policy on real or simulated users to assess them as improvements or degradations. This process is called on-policy *learning. Nevertheless, it can result in system behaviors that are not acceptable by users. Learning algorithms should ideally infer an optimal strategy by observing interactions generated by a non-optimal but acceptable strategy, that is learning* off-policy. *Such methods usually fail to scale up, converge slowly and are thus unapplicable to real-world systems. In this contribution, a* sample-efficient, online *and* off-policy *reinforcement learning algorithm is proposed to learn an optimal policy. This algorithm is combined with a* compact *non-linear value function representation enabling to handle large scale systems.*

Keywords

Reinforcement Learning, Spoken Dialogue System

1 Introduction

Les systèmes de dialogues vocaux (SDV) sont maintement fréquemment utilisés dans la vie quotidienne pour remplir des tâches variées : prise de rendez-vous, aide au diagnostic de pannes, information, tutorat, *etc*. La conception de ces systèmes est souvent laissée à des experts du traitement du langage naturel afin que le SDV soit efficace et interagisse de façon naturelle avec l'utilisateur. Ces deux qualités sont primordiales car le SDV interagit avec un humain et ce dernier peut très rapidement être lassé de communiquer avec une machine peu fiable. Le gestionnaire de dialogue (GD) du SDV, module responsable de la prise de décisions à propos de ce qui est dit et à quel moment, doit donc avoir un comportement prenant en compte les réactions de l'utilisateur. Cette tâche est compliquée par le fait que les modules de reconnaissance vocale et d'analyse sémantique, qui respectivement retranscrivent et reconnaissent le sens de ce qu'a dit l'utilisateur, commettent des erreurs. Cela rend la requête de l'utilisateur partiellement observable du point de vue du GD. Une solution parmi les plus courantes pour éviter les erreurs de reconnaissance est de demander à l'utilisateur de confirmer ce qu'il a dit. Mais cette solution ralentit les échanges et les rend moins naturels.

Il est donc important que le GD utilise une stratégie d'interaction efficace et naturelle, qui puisse faire aboutir la requête de l'utilisateur. Définir à la main cette stratégie devient rapidement impossible quand la tâche est réaliste. En effet, cela nécessite d'identifier toutes les situations qui peuvent être rencontrées au cours du dialogue. Des solutions ont donc été proposées pour automatiquement chercher la stratégie optimale. Parmi celles-ci, nous nous concentrons sur celles basées sur l'apprentissage par renforcement (AR) [1]. L'AR est une technique d'appren-

tissage automatique qui a déjà été appliquée avec succès à l'optimisation de SDV [2, 3, 4, 5]. L'idée fondamentale de ces algorithmes est d'apprendre une stratégie optimale à partir d'interactions entre un SDV et des utilisateurs de façon à optimiser une valeur numérique (appelée récompense) liée à la satisfaction de l'utilisateur. Cette récompense est généralement une combinaison linéaire de différentes mesures connues comme étant liées à la satisfaction de l'utilisateur (par exemple l'aboutissement de la requête, la longueur du dialogue, *etc*) [6]. La qualité d'une stratégie est mesurée comme étant la récompense cumulée espérée obtenue par le système alors qu'il suivait cette stratégie. Ceci mène à la définition d'une fonction, appelée *fonction de valeur* qui associe une récompense cumulée espérée à chaque contexte de dialogue. La stratégie optimale est celle qui associe la plus haute récompense cumulée espérée à chaque situation du dialogue.

Les méthodes standard d'AR utilisées jusqu'alors présentent l'inconvénient de nécessiter beaucoup de données : la quantité est telle qu'il n'est pas possible d'en réunir assez pour apprendre la politique optimale. Durant la dernière décennie, les dialogues ont donc été simulés pour générer artificiellement suffisamment de données [7, 4, 8, 9]. Mais cette méthode introduit un biais de modélisation qui peut mener à une inadéquation entre la stratégie apprise et le comportement d'utilisateurs réels [10]. C'est pourquoi un autre type de solutions, basé sur des algorithmes d'AR utilisant une approximation de la fonction de valeur a été récemment introduit. Des méthodes hors-ligne, utilisant un petit jeu de données fixes, ont d'abord été proposées. Elles sont assez performantes [11, 12, 13]. Cependant, comme les stratégies apprises ne peuvent s'adapter en ligne ni à des changements de comportement de l'utilisateur, ni aux performances du système, des problèmes peuvent survenir si les données fournies ne sont pas représentatives de l'ensemble des situations possibles. Ces méthodes peuvent néanmoins donner une stratégie de départ correcte, à améliorer en ligne par la suite.

Des algorithmes mettant en place un apprentissage *en ligne* et *on-policy* de la stratégie ont récemment été proposés [14, 15]. Ces algorithmes effectuent un changement et un test permanents de la politique à apprendre (un problème connu sous le nom de « dilemme exploration/exploitation », comme traité dans [16]). Ces changements, puisqu'ils sont effectués durant l'apprentissage, pourraient mener à des comportements du GD qui gêneraient l'utilisateur (choix d'une action aléatoire). De plus, ces méthodes sont basées sur des approximations linéaires de la fonction de valeur rendant ainsi leur utilisation dans des systèmes à large échelle difficile. Pour passer à l'échelle, le contexte du dialogue doit être représenté de manière compacte [17] avec le risque que cela mène à des erreurs d'approximation.

Dans ce papier, nous proposons l'utilisation de l'algorithme KTD (*Kalman Temporal Differences* en anglais) [18] pour réaliser un apprentissage *en ligne*, efficace sur les échantillons et *off-policy* [19] sur un système à grande échelle. KTD consiste à adopter une représentation paramétrique de la fonction de valeur. Les paramètres associés sont modélisés comme un processus aléatoire caché, lié aux récompenses via l'équation de Bellman. Ces paramètres sont alors appris en utilisant un filtre de Kalman non-parfumé (voir [18] pour les détails). L'utilisation de cet algorithme sera combinée à celle d'une *représentation compacte* de la fonction de valeur de façon à minimiser l'impact des approximations. La solution proposée dans [14] servira de référence pour les résultats présentés ici. Dans [14], la fonction de valeur est estimée grâce à l'algorithme GPTD (*Gaussian Processes Temporal Differences* en anglais) [20] entraînant un apprentissage en ligne et une dépendance linéaire de la fonction de valeur aux paramètres. Dans cet algorithme, la fonction de valeur est modélisée comme étant un processus gaussien. Elle est liée à la récompense via un modèle génératif transcrivant l'équation de Bellman. Etant donné un a priori, la distribution a posteriori est alors déterminée par inférence Bayésienne.

Dans le cadre de KTD, deux manières de représenter la fonction de valeur vont être étudiées : une linéaire, basée sur des Fonctions à Bases Radiales (RBF) et une non-linéaire, basée sur des réseaux de neurones. La question de la représentation est importante car un compromis doit être trouvé entre la compacité de la représentation et l'information qu'elle peut représenter. Si la paramétrisation n'est pas assez compacte, elle ne sera pas utilisable.

L'adaptation au cadre de l'AR du problème d'optimisation de la stratégie pour le dialogue est présentée dans la Section 2. Ensuite (Section 3), la question de la représentation de la fonction de valeur sera posée. La Section 4 présentera les expériences et pour finir, dans la Section 5 quelques résultats seront donnés.

2 La gestion de dialogue vue comme un Processus Décisionnel de Markov continu

La gestion du dialogue (GD) est un problème de décisions séquentielles. A partir de données provenant de l'utilisateur, les *observations*, le gestionnaire de dialogue doit choisir et effectuer une *action*. Cette action doit être choisie dans le but d'interagir avec l'utilisateur de manière efficace et naturelle. La satisfaction est quantifiée par une *récompense* fournie à la fin d'un dialogue. Vue sous cet angle, la gestion du dialogue peut être envisagée comme un Processus Décisionnel de Markov Partiellement Observable (PDMPO) : des décisions doivent être prises en accord avec l'historique des observations et des actions du système. Une manière de représenter succinctement l'historique en un état unique, appelée *hidden information paradigm*, a été proposée par [5]. Ainsi la GD s'envisage ici comme un MDP continu.

Les algorithmes visent à apprendre une stratégie, appelée

politique π qui associe une action $a \in A$ à chaque état $s \in S$. La qualité de la politique est quantifiée par la fonction Q qui donne la récompense cumulée espérée en partant d'une paire état-action donnée, (s, a), et suivant la politique π :

$$Q^\pi(s,a) = E[\sum_{i\geq 0} \gamma^i r_i | s_0 = s, a_0 = a, \pi].$$

Le facteur γ est le facteur d'actualisation, (s, a) la paire état-action et $(r_i)_{i\geq 0}$ le jeu de récompenses obtenues lors de l'interaction. La fonction Q optimale, notée Q^*, est telle que pour toute politique π et pour toute paire (s, a), $Q^* \geq Q^\pi$. La politique optimale est gloutonne par rapport à la fonction Q optimale : $\pi^*(s) = \arg\max_a Q^*(s, a)$. Il est à noter que la fonction Q permet de comparer non seulement des politiques entre elles mais aussi des actions en partant d'un même état et d'une politique fixée. Trouver la politique optimale se résume donc à trouver la fonction Q optimale. Dans la plupart des cas, l'espace formé par l'ensemble des états et des actions est trop grand pour autoriser un calcul exact de la fonction Q ce qui impose le calcul d'une approximation de cette fonction, $\hat{Q}$. La représentation choisie pour cette approximation est souvent paramétrique : $\hat{Q}_\theta$, θ étant le jeu de paramètres à apprendre.
De façon générale, la fonction Q optimale se calcule de deux manières. L'apprentissage *on-policy* améliore incrémentalement la politique courante par des phases répétées d'évaluation de la valeur de cette politique suivi de l'amélioration de cette dernière, amélioration qui se fait en prenant la politique gloutonne par rapport à la fonction de valeur nouvellement mise à jour. La phase d'évaluation peut se faire grâce à l'équation d'évaluation de Bellman :

$$Q^\pi(s,a) = E_{s'|s,a}[R(s,a,s') + \gamma Q^\pi(s', \pi(s'))].$$

L'état s' est l'état dans lequel se trouve le système après avoir effectué l'action a en partant de s.
La politique optimale est trouvée de cette façon avec l'algorithme GPTD utilisé dans [14]. Dans ce cas, l'équation de Bellman est linéaire, ce qui est requis pour l'utilisation des processus gaussiens. Ces derniers contraignent aussi la paramétrisation de la fonction Q à être linéaire.
L'autre façon de calculer la politique optimale est l'apprentissage *off-policy*. Cette méthode consiste à directement calculer la fonction Q optimale, $Q^*(s, a)$, en utilisant l'équation d'optimalité de Bellman (non-linéaire) :

$$Q^*(s,a) = E_{s'|s,a}[R(s,a,s') + \gamma \max_{b\in A} Q^*(s', b)].$$

Le cadre défini par KTD [18] permettant de gérer les non-linéarités, cette équation peut directement être résolue et un apprentissage *off-policy* est ainsi rendu possible [19]. Dans ce papier, nous profitons de cet avantage pour utiliser une paramétrisation non-linéaire, $Q^*_\theta(s, a)$, afin d'obtenir une représentation compacte de la fonction Q et d'envisager une représentation de l'espace d'état plus riche.

3 Approximation de la fonction Q

Ainsi qu'il a été précisé dans la partie précédente, dans certains cas, comme lorsque l'espace d'état S est continu, la fonction Q (optimale ou non) doit être approchée par $\hat{Q}_\theta(s, a)$ puisque $Q : S \times A \to \mathbb{R}$. La représentation pour la fonction Q peut être linéaire ou non.
Dans le cas linéaire, la fonction Q est représentée par $\hat{Q}_\theta(s, a) = \theta^T \Phi(s, a)$, où $\Phi(s, a)$ est une matrice formée de N fonctions de bases définies à l'avance, ϕ_i ($i \in [1..n]$), telle que $\Phi = [\phi_1(s, a)...\phi_N(s, a)]^T$ et θ est un vecteur contenant les poids associés. Ici, des Fonctions à Bases Radiales (RBF) ont été utilisées.
Si une représentation non-linéaire est choisie, la fonction Q peut être représentée par $\hat{Q}_\theta(s, a) = f_\theta(s, a)$, avec f_θ n'ayant pas de dépendance linéaire aux paramètres θ. Dans ce papier, nous avons utilisé un réseau de neurones dont une des propriétés est de pouvoir approximer une fonction quelconque s'il contient un nombre suffisant de neurones sur chaque couche cachée. Dans ce cas, le vecteur θ contient les poids synaptiques du réseau.
Nous définissons $\mathcal{H}$ comme étant l'espace d'hypothèses généré par $\Phi(s, a)$ ou par $f_\theta(s, a)$. L'espace $\mathcal{H}$ doit être assez riche pour contenir la vraie fonction Q mais le nombre de paramètres ne doit pas être trop grand à cause des risques de sur-apprentissage et des coûts de calcul. En effet, s'il y a trop de paramètres à trouver pour un nombre restreint de données d'entraînement, l'approximation de la fonction suivra les exemples et n'aura pas un grand pouvoir de généralisation. Un compromis est à faire entre l'information à représenter et la taille de la paramétrisation.

4 Expériences

4.1 Tâche

Les résultats présentés dans la section suivante ont été obtenus avec le système CamInfo [5], un SDV à grande échelle, développé afin de fournir des informations touristiques sur la ville de Cambridge. La requête de l'utilisateur peut contenir jusqu'à 12 attributs différents. Tous les résultats ont été obtenus avec des utilisateurs simulés [21] à cause de la difficulté à obtenir des données réelles et à cause de la grande variabilité inter-utilisateurs qui ne rend pas les expériences facilement comparables. Les erreurs de compréhension sont aussi simulées.

4.2 Paramètres du MDP

Durant la phase d'apprentissage, une récompense de $+20$ est donnée au système à la fin du dialogue s'il a réussi à répondre à la requête de l'utilisateur. Une pénalité de -1 est donnée à chaque fois que le système effectue une action. L'espace d'état utilisé est celui décrit dans [5]. Il est constitué d'un vecteur contenant deux variables continues (représentant les deux meilleurs scores de confiance associés aux deux meilleures hypothèses de ce qui a été reconnu par le système ; ces scores sont fournis par les modules de reconnaissance vocale et de l'analyse

sémantique). Le GD propose une action choisie parmi un jeu de 12 meta-actions qui sont retranscrites en 22 actions effectivement proposées à l'utilisateur (voir [5]). Cet espace d'état est ensuite amélioré afin de construire des stratégies plus robustes (voir Section 5.2).

4.3 Algorithmes

L'algorithme GPTD qui utilise une paramétrisation basée sur un dictionnaire construit durant l'apprentissage [20] est comparé à KTD qui utilise d'abord une paramétrisation linéaire puis une paramétrisation non-linéaire. Les resultats ont été obtenus en laissant le GD construire une estimation de la fonction Q, $\hat{Q}_{learnt}(s,a), \forall (s,a) \in S \times A$. Dans le cas où GPTD est utilisé, l'estimation de la fonction concerne la fonction de valeur de la politique suivie tandis que dans le cas où KTD est utilisé, l'estimation faite correspond à la fonction de valeur associée à la politique optimale. Ensuite, la fonction Q estimée est utilisée pour améliorer la stratégie. Cette stratégie, appelée π_{learnt}, est calculée de façon gloutonne par rapport à l'estimation : $\pi_{learnt} = \arg\max_a \hat{Q}_{learnt}(s,a)$. Avec un nombre suffisant de données, les deux algorithmes sont supposés apprendre la même stratégie optimale : π_{learnt} devrait être égale à π^*. Sur les graphes présentés, les récompenses cumulées moyennes obtenues en utilisant π_{learnt} sont tracées pour différentes tailles de jeux de dialogues d'entraînement.

4.4 Stratégie comportementale

L'apprentissage suppose d'utiliser une stratégie comportementale pour explorer l'espace d'état. Les algorithmes KTD et GPTD fournissent une estimation de la fonction Q et des informations d'incertitude quant à la qualité de cette estimation ($\hat{\sigma}_Q$) [22, 14]. Une approche dans laquelle l'agent apprenant fait un compromis sûr entre l'exploitation de l'information déjà sue et l'exploration de l'espace d'état, a été étudiée dans [16] pour le même SDV. Cette approche est appelée *bonus-gloutonne* (inspirée de [23]). Le choix de l'action suivante est fait ainsi :

$$a_{i+1} = \arg\max_a \left(\hat{Q}_i(s_{i+1},a) + \frac{\beta \hat{\sigma}_{Q_i}(s_{i+1},a)}{\beta_0 + \hat{\sigma}_{Q_i}(s_{i+1},a)} \right).$$

L'approche *bonus-gloutonne* donne les meilleurs résultats comparée à l'approche ϵ-gloutonne classique :

$$a_{i+1} = \begin{cases} \arg\max_a \hat{Q}_i(s_{i+1},a) \text{ w.p. } 1-\epsilon \\ \text{aléatoire (uniform.) w.p. } \epsilon. \end{cases}$$

Elle sera donc utilisée lors des expériences (Section 5).

5 Résultats

Dans cette partie, des résultats comparant l'efficacité des algorithmes KTD et GPTD dans différentes conditions de bruit sont donnés.

5.1 Paramétrisation linéaire

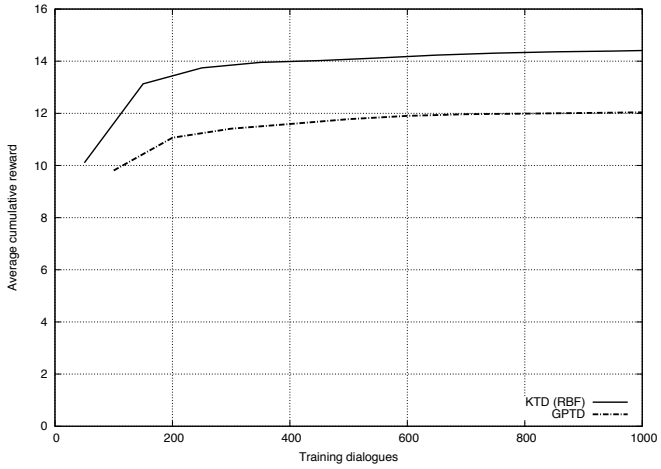
Tout d'abord, KTD utilisant une paramétrisation linéaire pour l'approximation de la fonction Q est comparé à GPTD dans une environnement presque sans bruit (taux d'erreurs de reconnaissance fixé à 10% : une fois sur dix, ce qui est dit par l'utilisateur est mal reconnu par la machine). Le nombre de paramètres est d'environ 300 pour l'approche avec GPTD et 144 pour celle avec KTD. Pour cette dernière, le vecteur de paramètres, défini pour tout $s \in S$ et pour tout $a \in A$ est : $\Phi(s,a) = [\delta_{a,a_1}\phi(s,a_1), ..., \delta_{a,a_{12}}\phi(s,a_{12})]^T$, avec $\delta_{a,a'} = 1$ si $a = a'$ sinon $\delta_{a,a'} = 0$ et $\phi(s,a)^T = [1, \varphi_1^1, \varphi_2^1, \varphi_3^1, \varphi_1^2, \varphi_2^2, \varphi_3^2, E_1, E_2](s,a)$. Les fonctions φ sont des gaussiennes. Trois gaussiennes par dimension sont utilisées sur les dimensions continues de l'espace d'état et deux entiers (E_1,E_2) sur les dimensions discrètes. Le GD choisit ses actions parmi 12 actions $(a_1, a_2, ..., a_{12})$.

Les résultats présentés Fig. 1(a) montrent que la politique apprise avec l'algorithme KTD est meilleure que celle apprise avec GPTD : en moyenne, il faut 2 tours de moins au GD pour satisfaire la requête de l'utilisateur. Dans les deux cas, la solution trouvée est imparfaite du fait de la perte d'information sur l'état lors de sa compression, détaillée dans l'article [5]. En quelques mots, les modules de reconnaissance vocale et d'analyse sémantique fournissent une liste d'hypothèses sur ce qui a été reconnu. A chaque hypothèse est associé un score de confiance. De cette liste d'hypothèses est extrait un certain nombre d'informations (deux meilleures hypothèses, action de l'utilisateur associée à la meilleure et but de l'utilisateur pressenti). Comme toutes les hypothèses ne peuvent être prises en compte, de l'information est perdue.

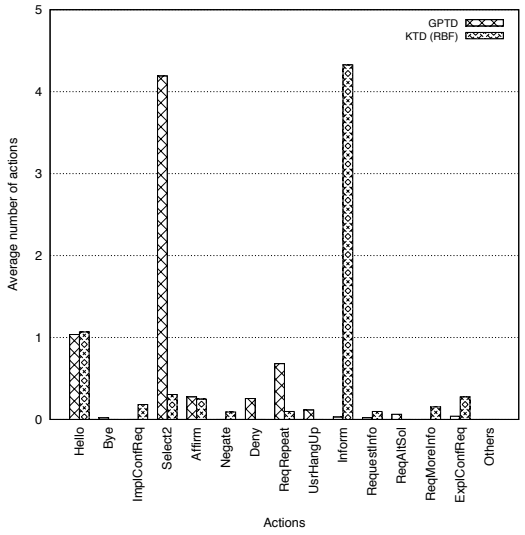
Sur la Fig. 1(b), les fréquences moyennes des actions possibles proposées par le gestionnaire de dialogue sont présentées. Elles ont été obtenues par le test de la politique apprise avec 1000 dialogues d'entraînement. En étudiant quelles sont les actions proposées, les différences apparaissant Fig. 1(a) peuvent être expliquées.

Les deux algorithmes ne proposent pas la même action principale : l'approche utilisant GPTD demande à l'utilisateur de faire un choix entre deux propositions (action ≪ Select2 ≫) tandis que celle utilisant KTD fournit directement à l'utilisateur une information (action ≪ Inform ≫). La politique apprise avec GPTD demande à l'utilisateur de répéter assez souvent ce qu'il a dit (action ≪ RedRepeat ≫) tandis que l'autre politique apprise (avec KTD) préfère demander une confirmation explicite (action ≪ ExplConfReq ≫). Il est plus facile de reconnaître une réponse à une question fermée (oui/non) qu'une phrase complète. La politique apprise avec KTD fait aussi usage d'une action qui en combine deux autres en fournissant une information tout en faisant une confirmation implicite (action ≪ ImplConfReq ≫) ce qui peut aussi contribuer à raccourcir le dialogue. L'approche qui utilise GPTD propose quelques fois une solution partiellement correcte (action ≪ Deny ≫) à laquelle est liée l'utilisation de l'action ≪ Select2 ≫. Ainsi, l'étude des

différentes actions proposées par les politiques permet de les comparer entre elles.



(a) Récompense cumulée moyenne



(b) Fréquence des actions

FIGURE 1 – Comparaison de GPTD et KTD utilisant une paramétrisation linéaire.

5.2 Paramétrisation non-linéaire

Une paramétrisation non-linéaire, basée sur un réseau de neurones, est maintenant introduite. Un réseau ayant une couche cachée qui comporte 8 neurones est utilisé ($N_{H_1} = 8$). Le nombre de neurones sur la couche d'entrée est déterminé par le fait que chaque paire $(s, a) \in (S \times A)$ doit être associée à une combinaison binaire étant donné qu'il n'y a pas de métrique définie sur l'espace d'action. Le nombre de neurones sur la couche d'entrée, N_I est : pour l'espace d'état, 2 neurones pour les composantes continues (qui correspondent aux deux meilleurs scores de confiance) et 22 pour l'action de l'utilisateur reconnue ; pour l'espace d'action, 12 neurones sont nécessaires. Donc $N_I = 2 + 6 + 22 + 12 = 42$. La couche de sortie du réseau contient un seul neurone dont la valeur correspond à l'estimation de la fonction Q. Le nombre de paramètres est ainsi de : $N_I \cdot N_{H_1} + (N_{H_1} + 1) \cdot 1 = 42 \cdot 8 + 9 = 345$.

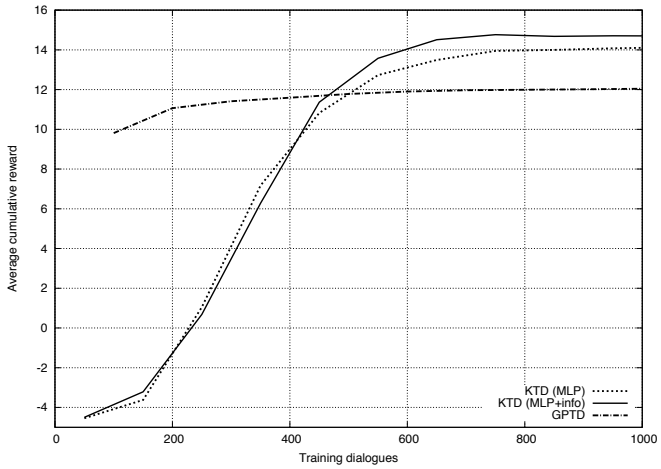
Une paramétrisation basée sur un réseau de neurones est intéressante car même si l'espace d'état est enrichi, le nombre de paramètres requis pour correctement représenter la fonction à estimer ne devient pas démesuré. Par exemple, si une valeur discrète pouvant prendre N valeurs différentes est ajoutée à l'espace d'état, le nombre de paramètres utilisés dans le cas linéaire est multiplié par N tandis qu'avec l'approche utilisant un réseau de neurones, seulement $N_{H_1} \cdot N$ paramètres sont ajoutés à la paramétrisation précédente (si le nombre de neurones sur la couche cachée est identique). Dans le cas d'une variable continue, seulement un neurone est ajouté à la couche d'entrée. Cette propriété peut être utilisée pour accroître l'espace d'état et enrichir la représentation d'un état. Ceci évite la perte d'information due à la compression de l'état (processus décrit Sec. 5.1).

Dans cette partie, l'espace d'état est enrichi par l'ajout d'une dimension continue qui correspond au troisème plus important score de confiance et par l'ajout d'une dimension discrète qui est l'action de l'utilisateur associée au deuxième meilleur score de confiance. Le nouvel espace d'état a donc six composantes, trois continues et trois discrètes. Le nombre de neurones sur la couche d'entrée est $N_I = 42 + 22 + 1 = 65$ donc le nombre de paramètres est $65 \cdot 8 + 9 = 529$. A titre de comparaison, si une approche de type RBF était utilisée, le nomre de paramètres serait de $(1+3^3) \cdot 6 \cdot 22 \cdot 22 \cdot 12 = 975\,744$, avec seulement 3 gaussiennes par dimension, ce qui n'est pas envisageable.

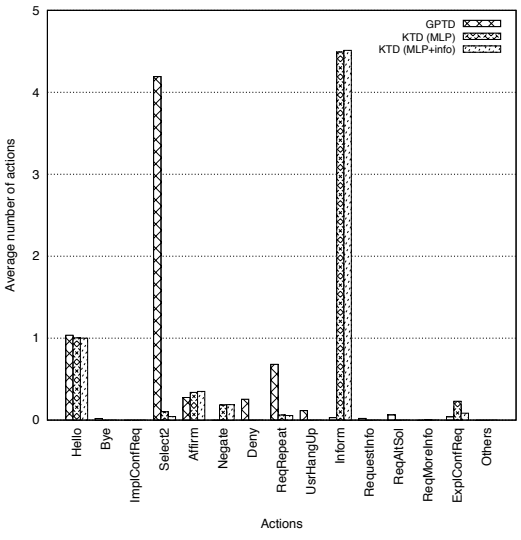
La performance de l'approche utilisant KTD dans un environnement peu bruité (taux d'erreurs de reconnaissance fixé à 10%) utilisant la paramétrisation non-linéaire décrite précédemment en utilisant l'espace d'état initial puis l'espace d'état enrichi est comparée à celle de GPTD sur la Fig. 2(a).

Les résultats obtenus avec KTD sont meilleurs que ceux obtenus avec GPTD si un nombre suffisant de données d'entraînement est fourni. En effet, les mauvaises performances de KTD avec un nombre de données d'entraînement faible sont dues au réseau de neurones qui soit fournit une mauvaise estimation de la fonction soit en fournit une correcte, sans intermédiaire. La largeur de la zone de transition entre les deux phases, c'est-à-dire ici entre 250 et 600 données, dépend des politiques apprises : pour certaines, la transition se fera tôt, vers 250 échantillons alors que pour d'autres, celle-ci se fera vers 600. La différence entre les deux courbes lorsque de l'information est ajoutée à l'espace d'état n'est pas significative. En effet, ajouter de l'information lorsque l'environnement est peu bruité n'est pas très intéressant puisque le module de reconnaissance vocale et l'analyseur sémantique sont sûrs de l'information qu'ils fournissent. La liste d'hypothèses qui est faite n'est pas très longue et toute l'information qu'elle contient est dans les toutes premières hy-

pothèses. En prendre en compte de nouvelles n'influe pas sur la décision.
Sur la Fig. 2(b) sont comparées les fréquences des actions proposées par le GD lors du test de la politique après apprentissage avec 1000 dialogues d'entraînement. Les histogrammes sont similaires à ceux obtenus Fig. 1(b).



(a) Récompense cumulée moyenne



(b) Fréquence des actions

FIGURE 2 – Comparaison de GPTD et KTD (avec une paramétrisation non-linéaire)(taux d'erreur de reconnaisance à 10%).

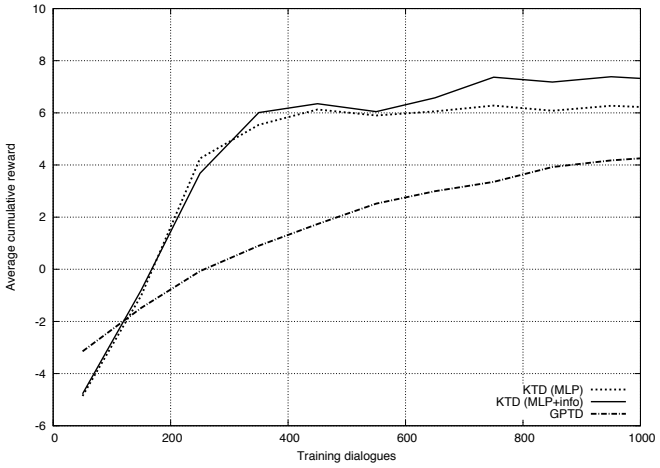
Enrichir l'espace d'état devrait être plus intéressant dans des environnements bruités car la liste d'hypothèses est plus longue et les scores de confiance associés distribués plus uniformément sur les hypothèses. L'approche utilisant KTD a donc été comparée à celle utilisant GPTD dans une situation très bruitée (taux d'erreurs de reconnaissance fixé à 50% : une fois sur deux, ce qui est dit par l'utilisateur est mal reconnu par la machine). Le nombre moyen de paramètres dans l'approche utilisant GPTD est d'environ 700. Il est supérieur par rapport au cas où il y a peu de bruit car il est directement lié à la dispersion des états rencontrés dans l'espace d'état et lorsqu'il y a beaucoup de bruit, l'espace d'état est plus vastement utilisé. Les resultats sont présentés Fig. 3. Alors qu'il n'y avait pas de différence notable entre les cas où l'espace d'état initial et celui enrichi étaient utilisés dans une situation peu bruitée, une différence apparaît dans ce cas-là. Sur la Fig. 3(a), l'approche utilisant KTD et un réseau de neurones est toujours plus performante que celle utilisant GPTD. Mais maintenant que de l'information est ajoutée dans l'espace d'état en situation très bruitée, la politique apprise sur un espace d'état plus riche est meilleure : la longueur du dialogue est raccourcie de 2 tours en moyenne.
Sur la Fig. 3(b), les fréquences des actions proposées durant le test de la politque apprise avec 1000 dialogues d'entraînement sont présentés. La tendance entre les politiques apprises par GPTD et KTD sont les mêmes que celles obtenues dans un environnement peu bruité. Néanmoins, les histogrammes montrent qu'une demande explicite d'information (action « ExplConfReq ») est plus fréquente quand moins d'information est présente dans l'espace d'état.

5.3 Discussion

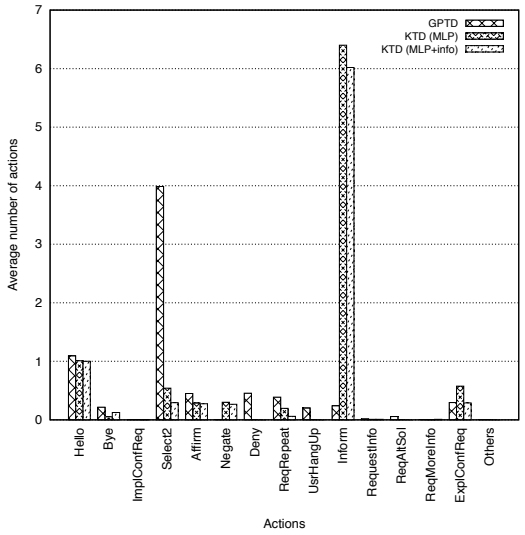
Dans cette partie, après avoir testé une paramétrisation linéaire pour représenter la fonction Q estimée par l'approche utilisant KTD, une paramétrisation non-linéaire a été employée. Avec une approche utilisant un réseau de neurones, un bon compromis entre la taille de la paramétrisation et la capacité d'approximation de la fonction Q a été trouvé. Plus d'information peut être représentée avec un nombre raisonnable de paramètres. Ainsi, comme plus d'information est prise en compte pour l'estimation, moins de biais de modélisation est introduit. Dans les deux cas, linéaires et non-linéaires, lorsque KTD est utilisé, la paramétrisation, dès lors que l'espace d'état est constuit, utilise toute l'information rencontrée sans faire d'approximation, contrairement à l'approche qui utilise GPTD, ce qui peut expliquer les meilleurs résultats obtenus de KTD par rapport à GPTD.

6 Conclusion

Dans ce papier, nous avons proposé l'utilisation de l'algorithme KTD [18] pour apprendre de façon *off-policy*, efficacement, une stratégie de dialogue pour un système à grande échelle. A cause de sa capacité à gérer des paramétrisations non-linéaires, l'approche KTD peut être associée à une approximateur de fonction compact, sous la forme d'un réseau de neurones. Grâce à la représentation compacte de l'espace d'état, ce dernier peut-être enrichi avec des informations sur la distribution des scores de confiance sur la liste d'hypothèses. Cette nouvelle représentation fournit une meilleure stratégie et plus robuste que celles déjà exisantes, comme celle fournie par GPTD.
Dans le futur, une représentation d'états encore plus riche sera testée. Un nombre plus important de paramètres n'est



(a) Récompense cumulée moyenne



(b) Fréquence des actions

FIGURE 3 – Comparaison de GPTD and KTD (avec une paramétrisation non-linéaire) (taux d'erreurs de reconnaissance fixé à 50%).

en effet pas un frein pour les approches qui utilisent un réseau de neurone comme approximateur.

Remerciements

Les auteurs tiennent à remercier le groupe dialogue de l'Université de Cambridge, tout particulièrement Steve Young et Miliça Gašić, pour la mise à disposition et leur aide à l'utilisation du système CamInfo. Ce travail est financé par INTERREG IVa dans le cadre du projet ALLEGRO et par la région Lorraine (France).

Références

[1] R. Sutton and A. Barto, *Reinforcement learning : An introduction*. The MIT press, 1998.

[2] S. Singh, M. Kearns, D. Litman, and M. Walker, "Reinforcement learning for spoken dialogue systems," in *Proc. of NIPS'99*, 1999.

[3] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human-machine interaction for learning dialog strategies," *IEEE Transactions on Speech and Audio Processing*, 2000.

[4] O. Pietquin and T. Dutoit, "A probabilistic framework for dialog simulation and optimal strategy learning," *IEEE Transactions on Speech and Audio Processing*, 2006.

[5] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, "The hidden information state model : A practical framework for POMDP-based spoken dialogue management," *Computer Speech & Language*, 2010.

[6] M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella, "PARADISE : A framework for evaluating spoken dialogue agents," in *Proc. of ACL'97*, 1997.

[7] W. Eckert, E. Levin, and R. Pieraccini, "User modeling for spoken dialogue system evaluation," in *Proc. of ASRU'97*, December 1997.

[8] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young, "A survey of statistical user simulation techniques for rl of dialogue management strategies," *The Knowledge Engineering Review*, 2006.

[9] O. Pietquin, "Consistent Goal-Directed User Model for Realistic Man-Machine Task-Oriented Spoken Dialogue Simulation," in *Proc. of ICME'06*, 2006, pp. 425–428.

[10] J. Schatzmann, M. N. Stuttle, K. Weilhammer, and S. Young, "Effects of the user model on simulation-based learning of dialogue strategies," in *Proc. of ASRU'05*, 2005.

[11] J. Henderson, O. Lemon, and K. Georgila, "Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets," *Computational Linguistics*, 2008.

[12] L. Li, S. Balakrishnan, and J. Williams, "Reinforcement Learning for Dialog Management using Least-Squares Policy Iteration and Fast Feature Selection," in *Proc. of InterSpeech'09*, 2009.

[13] O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet, "Sample-Efficient Batch Reinforcement Learning for Dialogue Management Optimization," *ACM Transactions on Speech and Language Processing*, 2011.

[14] M. Gašić, F. Jurčíček, S. Keizer, F. Mairesse, B. Thomson, K. Yu, and S. Young, "Gaussian processes for fast policy optimisation of POMDP-based dialogue managers," in *Proc. of SIGDIAL'11*, 2010.

[15] F. Jurčíček, B. Thomson, S. Keizer, M. Gašić, F. Mairesse, K. Yu, and S. Young, "Natural Belief-Critic : a

reinforcement algorithm for parameter estimation in statistical spoken dialogue systems," in *Proc. of InterSpeech'10*, 2010.

[16] L. Daubigney, M. Gašić, S. Chandramohan, M. Geist, O. Pietquin, and S. Young, "Uncertainty management for on-line optimisation of a POMDP-based large-scale spoken dialogue system," in *Proc. of InterSpeech'11*, 2011.

[17] J. Williams and S. Young, "Scaling up POMDPs for dialogue management : the summary POMDP method," in *Proc. of ASRU'05*, 2005.

[18] M. Geist and O. Pietquin, "Kalman Temporal Differences," *Journal of Artificial Intelligence Research*, 2010.

[19] O. Pietquin, M. Geist, and S. Chandramohan, "Sample Efficient On-line Learning of Optimal Dialogue Policies with Kalman Temporal Differences," in *Proc. of IJCAI'11*, 2011.

[20] Y. Engel, S. Mannor, and R. Meir, "Reinforcement Learning with Gaussian Processes," in *Proc. of ICML'05*, 2005.

[21] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, "Agenda-based user simulation for bootstrapping a pomdp dialogue system." in *Proc. of HLT/NAACL'07*, 2007.

[22] M. Geist and O. Pietquin, "Managing Uncertainty within the KTD Framework," in *Proc. of the AL&E workshop*, ser. JMLR C&WP, 2011.

[23] J. Z. Kolter and A. Y. Ng, "Near-Bayesian Exploration in Polynomial Time," in *Proc. of ICML'09*, 2009.