Managing Uncertainty within Value Function Approximation in Reinforcement Learning

Matthieu Geist Olivier Pietquin
IMS Research Group
Supélec, Metz, France
{matthieu.geist,olivier.pietquin}@supelec.fr

Abstract

The dilemma between exploration and exploitation is an important topic in reinforcement learning (RL). Most successful approaches in addressing this problem tend to use some uncertainty information about values estimated during learning. On another hand, scalability is known as being a lack of RL algorithms and value function approximation has become a major topic of research. Both problems arise in real-world applications, however few approaches allow approximating the value function while maintaining uncertainty information about estimates. Even fewer use this information in the purpose of addressing the exploration/exploitation dilemma. In this paper, we show how such an uncertainty information can be derived from a Kalman-based Temporal Differences (KTD) framework. An active learning scheme for a second-order value-iteration-like algorithm (named KTD-Q) is proposed. We also suggest adaptations of several existing exploration/exploitation dilemma schemes. This is a first step towards global handling of continuous state and action spaces and exploration/exploitation dilemma.

1 Introduction

Reinforcement learning (RL) (Sutton and Barto, 1996) is the machine learning answer to the well-known problem of optimal control of dynamic systems. In this paradigm, an agent learns to control its *environment* (*i.e.* the dynamic system) through examples of actual interactions. To each of these interactions is associated an immediate reward which is a local hint about the quality of the current control policy. More formally, at each (discrete) time step i the dynamic system to be controlled is in a state s_i. The agent chooses an action a_i, and the dynamic system is then driven in a new state, say s_{i+1}, following its own dynamics. The agent receives a reward r_i associated to the transition (s_i, a_i, s_{i+1}). The agent's objective is to maximize the expected cumulative rewards, which it internally models as a so-called value or Q-function (see later). In the most challenging cases, learning has to be done online and the agent has to control the system while trying to learn the optimal policy. A major issue is then the choice of the behavior policy and the associated dilemma between exploration and exploitation. Indeed at each time step, the agent can choose an optimal action according to its (maybe) imperfect knowledge of the environment (exploitation) or an action considered to be suboptimal so as to improve its knowledge (exploration) and subsequently its policy. The ϵ-greedy action selection is a popular choice which consists in selecting the greedy action with probability $1-\epsilon$, and an equally distributed random action with probability ϵ. Another popular scheme is the *softmax* action selection (Sutton and Barto, 1996) drawing the behavior action from a Gibbs distribution. Most successful approaches tend to use an uncertainty information to choose between exploration and exploitation but also to drive exploration. Dearden et al. (1998) maintain a distribution for each Q-value. They propose two schemes. The first one consists in sampling the action according to the Q-value distribution. The second one uses a myopic value of imperfect information which approximates the utility of an information-gathering action in terms of the expected improvement of the decision quality. Strehl and Littman (2006) maintain a confidence interval for each Q-value and the policy is greedy respectively to the upper bound of

Preliminary work. Under review for the workshop on Active Learning and Experimental Design, collocated with AISTATS 2010. Do not distribute.

this interval. This approach allows deriving probably-approximately-correct (PAC) bounds. Sakaguchi and Takano (2004) use a Gibbs policy. However a reliability index (actually a form of uncertainty) is used instead of the more classic temperature parameter. Most of these approaches are designed for problems where an exact (tabular) representation of the value function is possible. Nevertheless, approximating the value in the case of large state spaces is another topic of importance in RL. Unfortunately quite few value function approximator allow deriving an uncertainty information about estimated values. Engel (2005) proposes such an algorithm, but the actual use of value uncertainty is left as a perspective. In this paper, we show how some uncertainty information about estimated values can be derived from the Kalman Temporal Differences (KTD) framework of Geist et al. (2009a,b). We also introduce a form of active learning which uses this uncertainty information in order to speed up learning, as well as some adaptations of existing schemes designed to handle the exploration/exploitation dilemma. Each contribution is illustrated and experimented.

2 Background

2.1 Reinforcement Learning

This paper is placed in the framework of Markov decision process (MDP). An MDP is a tuple $\{S, A, P, R, \gamma\}$, where S is the state space, A the action space, $P : s, a \in S \times A \to p(.|s, a) \in P(S)$ a family of transition probabilities, $R : S \times A \times S \to \mathbb{R}$ the bounded reward function, and γ the discount factor. A policy π associates to each state a probability over actions, $\pi : s \in S \to \pi(.|s) \in P(A)$. The value function of a given policy is defined as $V^\pi(s) = E[\sum_{i=0}^\infty \gamma^i r_i | s_0 = s, \pi]$ where r_i is the immediate reward observed at time step i, and the expectation is done over all possible trajectories starting in s given the system dynamics and the followed policy. The Q-function allows a supplementary degree of freedom for the first action and is defined as $Q^\pi(s, a) = E[\sum_{i=0}^\infty \gamma^i r_i | s_0 = s, a_0 = a, \pi]$. RL aims at finding (through interactions) the policy π^* which maximises the value function for every state: $\pi^* = \text{argmax}_\pi(V^\pi)$. Two schemes among others can lead to the optimal policy. First, *policy iteration* involves learning the value function of a given policy and then improving the policy, the new one being greedy respectively to the learnt value function. It requires solving the *Bellman evaluation equation*, which is given here for the value and Q-functions:

$$V^\pi(s) = E_{s',a|\pi,s}[R(s, a, s') + \gamma V^\pi(s')] \quad (1)$$

$$Q^\pi(s, a) = E_{s',a'|\pi,s,a}[R(s, a, s') + \gamma Q^\pi(s', a')] \quad (2)$$

The second scheme, *value iteration*, aims directly at finding the optimal policy. It requires solving the *Bellman optimality equation*:

$$Q^*(s, a) = E_{s'|s,a}[R(s, a, s') + \gamma \max_{b \in A} Q^*(s', b)] \quad (3)$$

For large state and action spaces, exact solutions are tricky to obtain and value or $Q-$function approximation is required.

2.2 Kalman Temporal Differences - KTD

Originally, the Kalman (1960) filter paradigm is a statistical method aiming at online tracking the hidden state of a non-stationary dynamic system through indirect observations of this state. The idea behind KTD is to cast value function approximation into such a filtering paradigm: considering a function approximator based on a familly of parameterized functions, the parameters are then the hidden state to be tracked, the observation being the reward linked to the parameters through one of the classical Bellman equations. Thereby value function approximation can benefit from the advantages of Kalman filtering and particularly uncertainty management because of statistical modelling.

The following notations are adopted, given that the aim is the value function evaluation, the Q-function evaluation or the Q-function direct optimization:

$$t_i = \begin{cases} (s_i, s_{i+1}) \\ (s_i, a_i, s_{i+1}, a_{i+1}) \\ (s_i, a_i, s_{i+1}) \end{cases} \quad (4)$$

$$g_{t_i}(\theta_i) = \begin{cases} \hat{V}_{\theta_i}(s_i) - \gamma \hat{V}_{\theta_i}(s_{i+1}) \\ \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \hat{Q}_{\theta_i}(s_{i+1}, a_{i+1}) \\ \hat{Q}_{\theta_i}(s_i, a_i) - \gamma \max_b \hat{Q}_{\theta_i}(s_{i+1}, b) \end{cases} \quad (5)$$

where $\hat{V}_\theta$ (resp. $\hat{Q}_\theta$) is a parametric representation of the value (resp. Q-) function, θ being the parameter vector. A statistical point of view is adopted and the parameter vector is considered as a random variable. The problem at sight is stated in a so-called *state-space formulation*:

$$\begin{cases} \theta_i = \theta_{i-1} + v_i \\ r_i = g_{t_i}(\theta_i) + n_i \end{cases} \quad (6)$$

Using the vocabulary of Kalman filtering, the first equation is the evolution equation. It specifies that the searched parameter vector follows a random walk which expectation corresponds to the optimal estimation of the value function at time step i. The evolution noise v_i is centered, white, independent and of variance matrix P_{v_i}. The second equation is the observation equation, it links the observed transitions and rewards to the value (or Q-) function through one of the Bellman equations. The observation noise n_i is supposed centered, white, independent and of variance P_{n_i}.

KTD is a second order algorithm: it updates the mean parameter vector, but also the associated covariance matrix after each interaction. It breaks down into three steps. First, *predictions* of the parameters first- and second-order moments are obtained according to the evolution equation and using previous estimates. Then some *statistics of interest* are computed. The third step applies a *correction* to predicted moments of the parameters vector according to the so-called Kalman gain K_i (computed thanks to the statistics obtained in second step), the predicted reward $\hat{r}_{i|i-1}$ and the observed reward r_i (their difference being a form of temporal difference error).

Statistics of interest are generaly not analytically computable, except in the linear case. This does not hold for nonlinear parameterizations such as neural networks and for the Bellman optimality equation (because of the max operator). Nevertheless, a derivative-free approximation scheme, the unscented transform (UT) of Julier and Uhlmann (2004), allows estimating first and second order moments of a nonlinearly mapped random vector. Let X be a random vector (typically the parameter vector) and $Y = f(X)$ its nonlinear mapping (typically the g_{t_i} function). Let n be the dimension of the random vector X. A set of $2n + 1$ so-called sigma-points and associated weights are computed as follows:

$$\begin{cases} x^{(0)} = \bar{X} & j = 0 \\ x^{(j)} = \bar{X} + (\sqrt{(n+\kappa)P_X})_j & 1 \leq j \leq n \\ x^{(j)} = \bar{X} - (\sqrt{(n+\kappa)P_X})_{n-j} & n+1 \leq j \leq 2n \end{cases} \tag{7}$$

$$\text{and } \begin{cases} w_0 = \frac{\kappa}{n+\kappa} & j = 0 \\ w_j = \frac{1}{2(n+\kappa)} & 1 \leq j \leq 2n \end{cases} \tag{8}$$

where $\bar{X}$ is the mean of X, P_X is its variance matrix, κ is a scaling factor which controls the accuracy, and $(\sqrt{P_X})_j$ is the j^{th} column of the Cholesky decomposition of P_X. Then the image of each sigma-point through the mapping f is computed:

$$y^{(j)} = f(x^{(j)}), \quad 0 \leq j \leq 2n \tag{9}$$

The set of sigma-points and their images can then be used to compute the following approximations:

$$\begin{cases} \bar{Y} \approx \bar{y} = \sum_{j=0}^{2n} w_j y^{(j)} \\ P_Y \approx \sum_{j=0}^{2n} w_j (y^{(j)} - \bar{y})(y^{(j)} - \bar{y})^T \\ P_{XY} \approx \sum_{j=0}^{2n} w_j (x^{(j)} - \bar{X})(y^{(j)} - \bar{y})^T \end{cases} \tag{10}$$

Thanks to the UT, practical algorithms can be derived. At time-step i, a set of sigma-points is computed from predicted random parameters characterized by mean $\hat{\theta}_{i|i-1}$ and variance $P_{i|i-1}$. Predicted rewards are then computed as images of these sigma-points using one of the observation functions (5). Then sigma-points and their images are used to compute statistics of interest. This gives rise to a generic algorithm valid for any of the three Bellman equations and any parametric representation of V or Q summarized in Alg. 1, p being the number of parameters.

Algorithm 1: KTD

```
Initialization: priors $\hat{\theta}_{0|0}$ and $P_{0|0}$;
for $i \leftarrow 1, 2, \ldots$ do
    Observe transition $t_i$ and reward $r_i$;
    Prediction Step;
    $\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;
    $P_{i|i-1} = P_{i-1|i-1} + P_{v_i}$;
    Sigma-points computation;
    $\Theta_{i|i-1} = \{\hat{\theta}^{(j)}_{i|i-1}, 0 \leq j \leq 2p\}$;
    /* from $\hat{\theta}_{i|i-1}$ and $P_{i|i-1}$ */
    $\mathcal{W} = \{w_j, 0 \leq j \leq 2p\}$ ;
    $\mathcal{R}_{i|i-1} = \{\hat{r}^{(j)}_{i|i-1} = g_{t_i}(\hat{\theta}^{(j)}_{i|i-1}), 0 \leq j \leq 2p\}$;
    /* see Eq. (5) */
    Compute statistics of interest;
    $\hat{r}_{i|i-1} = \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1}$;
    $P_{\theta r_i} = \sum_{j=0}^{2p} w_j (\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$;
    $P_{r_i} = \sum_{j=0}^{2p} w_j (\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})^2 + P_{n_i}$;
    Correction step;
    $K_i = P_{\theta r_i} P_{r_i}^{-1}$ ;
    $\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i (r_i - \hat{r}_{i|i-1})$ ;
    $P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$ ;
```

3 Computing Uncertainty over Values

3.1 Principle

The parameters being modeled as random variables, the parameterized value for any given state is a random variable. This model allows computing the mean and associated uncertainty. Let $\hat{V}_\theta$ be the approximated value function parameterized by the random vector θ of mean $\bar{\theta}$ and variance matrix P_θ. Let $\bar{V}_\theta(s)$ and $\hat{\sigma}^2_{V_\theta}(s)$ be the associated mean and variance for a given state s. To propagate the uncertainty from the parameters to the approximated value function a first step is to compute the sigma-points associated to the parameter vector, that is $\Theta = \{\theta^{(j)}, 0 \leq j \leq 2p\}$, as well as corresponding weights, from $\bar{\theta}$ and P_θ as described before. Then the images of these sigma-points are computed using the parameterized value function: $V_\theta(s) = \{\hat{V}^{(j)}_\theta(s) = \hat{V}_{\theta^{(j)}}(s), \quad 0 \leq j \leq 2p\}$. Knowing these images and corresponding weights, the statistics of interest are computed:

$$\begin{cases} \bar{V}_\theta(s) = \sum_{j=0}^{2p} w_j \hat{V}_\theta^{(j)}(s) \\ \hat{\sigma}^2_{V_\theta}(s) = \sum_{j=0}^{2p} w_j (\hat{V}_\theta^{(j)}(s) - \bar{V}_\theta(s))^2 \end{cases} \qquad (11)$$

This is illustrated on Fig. 1. Extension to Q-function is straightforward. So, as at each time-step uncertainty information can be computed in the KTD framework.

3.2 Illustration

This first experiment illustrates the available uncertainty information on a simple maze problem. The 2D continuous state space is the unit square: (x, y) $[0, 1]^2$. Actions are moves in the 4 compass directions, the magnitude being of 0.05 in each case. The reward is $+1$ if the agent leaves the maze in $\{x$ $[-\frac{3}{8}, \frac{3}{8}], y = 1\}$, -1 if the agent leaves the maze in $\{x$ $[-1, \frac{3}{8}[$ $]\frac{3}{8}, 1], y = 1\}$, and 0 otherwise. The algorithm is KTD-V (the Bellman value function evaluation equation is considered). The parameterized function famility is a linear combination of 9 equispaced Gaussian kernels (centered in $\{0, 0.5, 1\} \times \{0, 0.5, 1\}$ and with a standard deviation of 0.5). The parameters are the height of each kernel. The forgetting factor γ is set to 0.9. The agent starts in a random position (x_0, y_0) with x_0 sampled from a Gaussian distribution, x_0 $N(\frac{1}{2}, \frac{1}{8})$, and y_0 sampled from a uniform distribution, y_0 $U_{[0,0.05]}$. The behavior policy for which the value function is learnt is going up with probability 0.9, and going in one of the three other directions with probability $\frac{0.1}{3}$. The initial parameter vector is set to zero, the prior to $P_{0|0} = 10I$, and the noise covariances to $P_{n_i} = 1$ and $P_{v_i} = 0I$. The learning is

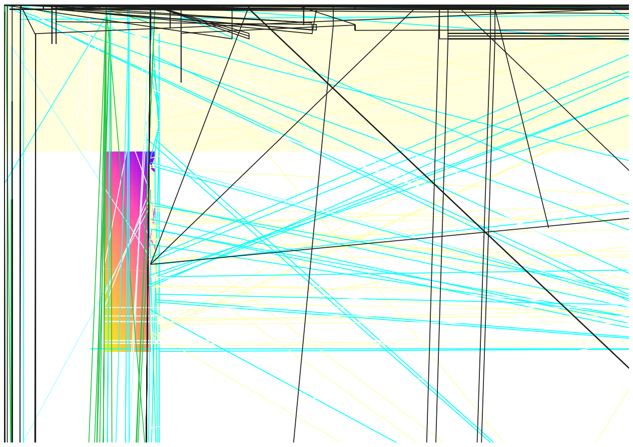


Figure 2: Uncertainty illustration.

done over 30 episodes, and results are given in Fig. 2, which shows the standard deviation of the approximated value function over the state space. Considering the x-axis, the uncertainty is lower in the middle than in the border, because learning trajectories occur more frequently in the center of the domain. Considering the y-axis, the uncertainty is lower near the upper bound ($y = 1$) than near the lower bound ($y = 0$), because retro-propagated values are less certain.

4 A Form of Active Learning

4.1 Principle

It is shown here how this available uncertainty information can be used in a form of active learning. The KTD algorithm derived from the Bellman optimality equation, that is Alg. 1 with third equation of Eq. (5), is named KTD-Q. It is an off-policy algorithm: it learns the optimal policy π^* while following a different behaviorial policy b. A natural question is: what behaviorial policy to choose so as to speed up learning? Let i be the current temporal index. The system is in a state s_i, and the agent has to choose an action a_i. The predictions $\hat{\theta}_{i|i-1}$ and $P_{i|i-1}$ are available and can be used to approximate the uncertainty of the Q-function parameterized by $\theta_{i|i-1}$ in the state s_i and for any action a. Let $\hat{\sigma}^2_{Q_{i|i-1}}(s_i, a)$ be the corresponding variance. The action a_i is chosen according to:

$$b(.|s_i) = \frac{\hat{\sigma}_{Q_{i|i-1}}(s_i, .)}{\sum_{a \in A} \hat{\sigma}_{Q_{i|i-1}}(s_i, a)} \qquad (12)$$

This totally explorative policy favours uncertain actions. The corresponding algorithm which we call active KTD-Q is summarized in Alg. 2.

4.2 Experiment

The second experiment is the inverted pendulum benchmark. This task requires maintaining a pendulum of unknown length and mass at the upright position by applying forces to the cart it is attached to. It is fully described by Lagoudakis and Parr (2003) and we use the same parameterization (a mixture of Gaussian kernels). The goal is here to compare two value-iteration-like algorithms, namely KTD-Q and Q-learning, which aim at learning directly the optimal policy. As far as we know, KTD-Q is the first second-order algorithm for Q-function approximation in a value iteration scheme, the difficulty being to handle the max operator (Yu and Bertsekas (2007) propose also such an algorithm, however for a restrictive class of MDP). That is why we compare it to a first-order algorithm. The active learning scheme is also experimented: it uses the uncertainty computed by KTD to speed up convergence.

For Q-learning, the learning rate is set to $\alpha_i = \alpha_0 \frac{n_0+1}{n_0+i}$ with $\alpha_0 = 0.5$ and $n_0 = 200$, according to Lagoudakis and Parr (2003). For KTD-Q, the parameters are set

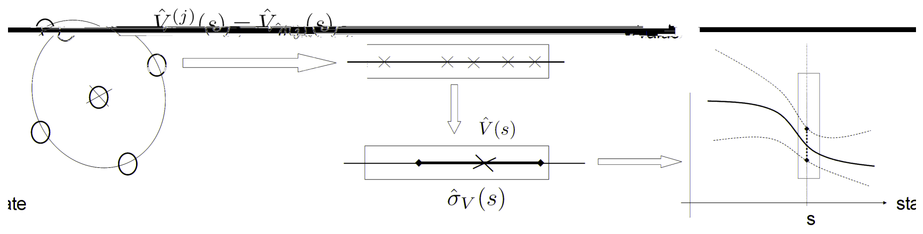


Figure 1: Uncertainty computation.

Algorithm 2: Active KTD-Q

```
Initialization: priors $\hat{\theta}_{0|0}$ and $P_{0|0}$, state $s_1$;
for $i \leftarrow 1, 2, \dots$ do
    Prediction Step;
    $\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;
    $P_{i|i-1} = P_{i-1|i-1} + P_{v_i}$;
    Sigma-points computation and sampling;
    $\Theta_{i|i-1} = \{\hat{\theta}^{(j)}_{i|i-1}, \quad 0 \le j \le 2p\}$ ;
    /* from $\hat{\theta}_{i|i-1}$ and $P_{i|i-1}$ */
    $\mathcal{W} = \{w_j, \quad 0 \le j \le 2p \quad \}$ ;
    for $a \in A$ do
        $\mathcal{Q}_{i|i-1}(s_i, a) = \{\hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a), 0 \le j \le 2p\}$;
        $\bar{Q}_{i|i-1}(s_i, a) = \sum_{j=0}^{2p} w_j \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a)$;
        $\hat{\sigma}^2_{Q_{i|i-1}}(s_i, a) =$
        $\sum_{j=0}^{2p} w_j (\hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a) - \bar{Q}_{i|i-1}(s_i, a))^2$;
    Sample $a_i$ according to $b(.|s_i)$, see Eq. (12);
    Observe $r_i$ and $s_{i+1}$ ;
    $\mathcal{R}_{i|i-1} = \{\hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i)$
        $- \gamma \max_{a \in A} \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, a), 0 \le j \le 2p\}$;
    Compute statistics of interest;
    $\hat{r}_{i|i-1} = \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1}$;
    $P_{\theta r_i} = \sum_{j=0}^{2p} w_j (\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$;
    $P_{r_i} = \sum_{j=0}^{2p} w_j (\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})^2 + P_{n_i}$;
    Correction step;
    $K_i = P_{\theta r_i} P_{r_i}^{-1}$ ;
    $\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i (r_i - \hat{r}_{i|i-1})$ ;
    $P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$ ;
```

to $P_{0|0} = 10I$, $P_{n_i} = 1$ and $P_{v_i} = 0I$. For all algorithms the initial parameter vector is set to zero. Training samples are first collected online with a random behavior policy. The agent starts in a randomly perturbed state close to the equilibrium. Performance is measured as the average number of steps in an test episode (a maximum of 3000 steps is allowed). Results are averaged over 100 trials. Fig. 3 compares KTD-Q and Q-learning. Fig. 4 adds active KTD-Q for which actions are sampled according to (12). Average length of episodes with totally random policy is 10, whereas it is 11 for policy (12). Consequently the increase in length can only slightly help to improve speed of convergence (at most 10%, much less than the real improvement which is about 100%, at least at the beginning).

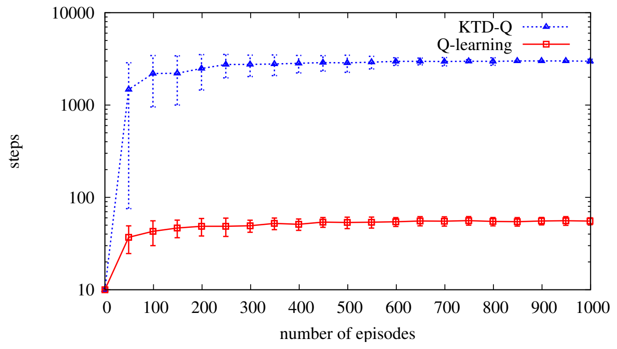


Figure 3: Optimal policy learning.

According to Fig. 3, KTD-Q learns an optimal policy (that is balancing the pole for the maximum number of steps) asymptotically and near-optimal policies are learned after only a few tens of episodes. With the same number of learning episodes, Q-learning with the same linear parameterization fails to learn a policy which balances the pole for more than a few tens of time steps. Similar results for Q-learning are obtained by Lagoudakis and Parr (2003). According to Fig. 4, it is clear that sampling actions according to uncertainty speeds up convergence. It is almost doubled in the first 100 episodes. Notice that this active learning scheme could not have been used for Q-learning with value function approximation, as this algorithm cannot provide uncertainty information.

5 Exploration/Exploitation Dilemma

In this section, we present several approaches designed to handle the dilemma between exploration and exploitation. The first one is the well known ϵ-greedy policy, and it serves as a baseline. Other approaches are inspired from the literature and use the available uncertainty information (see Sec. 3 for its computation). Combination of KTD-SARSA (Alg. 1 with second equation of Eq. (5)) with control is depicted in

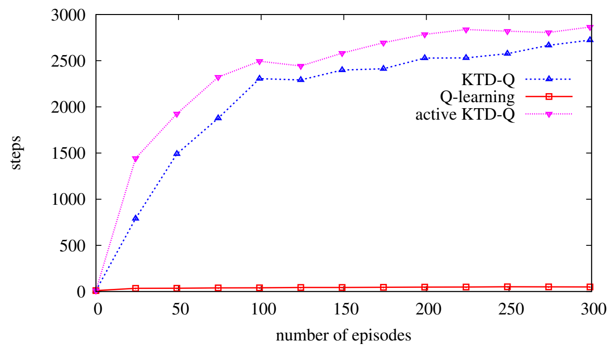


Figure 4: Random and active learning.

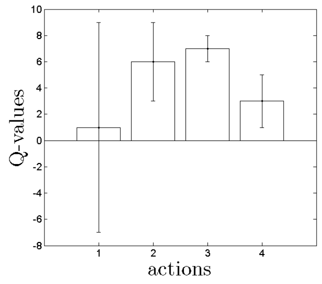


Figure 5: Q-values and associated uncertainty.

Alg. 3.

Algorithm 3: KTD and control

```
Initialization: priors $\hat{\theta}_{0|0}$ and $P_{0|0}$, state $s_1$, action $a_1$;
for $i \leftarrow 1, 2, \dots$ do
    Apply $a_i$ in $s_i$ ;
    Observe $r_i$ and $s_{i+1}$ ;
    Prediction Step;
    $\hat{\theta}_{i|i-1} = \hat{\theta}_{i-1|i-1}$;
    $P_{i|i-1} = P_{i-1|i-1} + P_{v_i}$;
    Sigma-points computation and sampling;
    $\Theta_{i|i-1} = \{\hat{\theta}^{(j)}_{i|i-1}, \quad 0 \leq j \leq 2p\}$ ;
    /* from $\hat{\theta}_{i|i-1}$ and $P_{i|i-1}$ */
    $\mathcal{W} = \{w_j, \quad 0 \leq j \leq 2p \}$ ;
    for $a \in A$ do
        $\mathcal{Q}_{i|i-1}(s_{i+1}, a) = \{\hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, a), 0 \leq j \leq 2p\}$;
        $\bar{Q}_{i|i-1}(s_{i+1}, a) = \sum_{j=0}^{2p} w_j \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, a)$;
        $\hat{\sigma}^2_{Q_{i|i-1}}(s_{i+1}, a) =$
        $\sum_{j=0}^{2p} w_j (\hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, a) - \bar{Q}_{i|i-1}(s_{i+1}, a))^2$;
    Sample $a_{i+1}$ according to $\pi(.|s_{i+1})$, see Eq. (13-16);
    $\mathcal{R}_{i|i-1} = \{\hat{r}^{(j)}_{i|i-1} = \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_i, a_i)$
        $- \gamma \hat{Q}_{\hat{\theta}^{(j)}_{i|i-1}}(s_{i+1}, a_{i+1}), 0 \leq j \leq 2p\}$;
    Compute statistics of interest;
    $\hat{r}_{i|i-1} = \sum_{j=0}^{2p} w_j \hat{r}^{(j)}_{i|i-1}$;
    $P_{\theta r_i} = \sum_{j=0}^{2p} w_j (\hat{\theta}^{(j)}_{i|i-1} - \hat{\theta}_{i|i-1})(\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})$;
    $P_{r_i} = \sum_{j=0}^{2p} w_j (\hat{r}^{(j)}_{i|i-1} - \hat{r}_{i|i-1})^2 + P_{n_i}$;
    Correction step;
    $K_i = P_{\theta r_i} P_{r_i}^{-1}$ ;
    $\hat{\theta}_{i|i} = \hat{\theta}_{i|i-1} + K_i(r_i - \hat{r}_{i|i-1})$ ;
    $P_{i|i} = P_{i|i-1} - K_i P_{r_i} K_i^T$ ;
```

5.1 ϵ-greedy Policy

With an ϵ-greedy policy (Sutton and Barto, 1996), the agent chooses a greedy action respectively to the currently estimated Q-function with a probability $1 - \epsilon$, and a random action with a probability ϵ (δ is the Kronecker symbol):

$$\pi(a_{i+1}|s_{i+1}) = (1 - \epsilon)\delta(a_{i+1} = \operatorname*{argmax}_{b \in A} \bar{Q}_{i|i-1}(s_{i+1}, b)) + \epsilon\delta(a_{i+1} = \operatorname*{argmax}_{b \in A} \bar{Q}_{i|i-1}(s_{i+1}, b)) \tag{13}$$

This policy is perhaps the most basic one, and it does not use any uncertainty information. An arbitrary Q-function for a given state and 4 different actions is illustrated on Fig. 5. For each action, it gives the estimated Q-value as well as the associated uncertainty (that is ± estimated standard deviation). For example, action 3 has the highest value and the lowest uncertainty, and action 1 the lowest value but the highest uncertainty. The probability distribution associated to the ϵ-greedy policy is illustrated on Fig. 6.a. The highest probability is associated to action 3, and other actions have the same (low) probability, despite their different estimated values and standard deviations.

5.2 Confident-greedy Policy

The second approach we propose consists in acting greedily according to the upper bound of an estimated confidence interval. The approach is not novel (Kaelbling, 1993), however some PAC (probably approximately correct) guarantees have been given recently by Strehl and Littman (2006) for a tabular representation (for which the confidence interval is proportional to the inverse of the square root of the number of visits to the considered state-action pair). In our case, we postulate that the confidence interval width is proportional to the estimated standard deviation (which is true if the parameters distribution is assumed to be Gaussian). Let α be a free positive parameter, we define the confident-greedy policy as:

$$\pi(a_{i+1}|s_{i+1}) = \delta\Big(a_{i+1} = \operatorname*{argmax}_{b \in A} \big(\bar{Q}_{i|i-1}(s_{i+1}, b) + \alpha\hat{\sigma}_{Q_{i|i-1}}(s_{i+1}, b)\big)\Big) \tag{14}$$

The same arbitrary Q-values are considered (see Fig. 5), and the confident-greedy policy is illustrated on Fig. 6.b which represents the upper bound of the confidence interval. Action 1 is chosen because it has the highest score (despite the fact that it has the lowest estimated value). Notice that action 3, which is greedy respectively to the estimated Q-function, has only the third score.

5.3 Bonus-greedy Policy

The third approach we propose is inspired from the method of Kolter and Ng (2009). The policy they use is greedy respectively to the estimated Q-function plus a bonus, this bonus being proportional to the inverse of the number of visits to the state-action pair of interest (which can be interpreted as a variance, instead of the square-root of this quantity for interval estimation-based approaches which can be interpreted as a standard deviation). The bonus-greedy policy we propose uses the variance rather than the standard deviation, and is defined as (β_0 and β being two free parameters):

$$\pi(a_{i+1}|s_{i+1}) = \delta\Big(a_{i+1} = \operatorname*{argmax}_{b\in A}\big(\bar{Q}_{i|i-1}(s_{i+1}, b) + \beta\frac{\hat{\sigma}^2_{Q_{i|i-1}}(s_{i+1}, b)}{\beta_0 + \hat{\sigma}^2_{Q_{i|i-1}}(s_{i+1}, b)}\big)\Big) \quad (15)$$

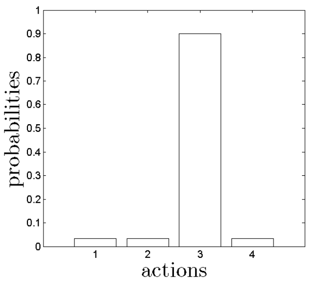
The bonus-greedy policy is illustrated on Fig. 6.c, still using the arbitrary Q-values and associated standard deviations of Fig. 5. Action 2 has the highest score, it is thus chosen. Notice that the three other actions have approximately the same score, despite the fact that they have quite different Q-values.

5.4 Thompson Policy

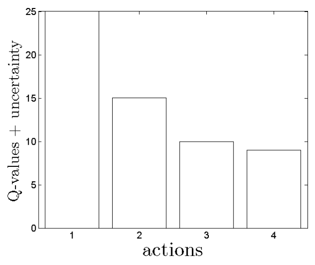
Recall that the KTD algorithm maintains the parameters mean vector and variance matrix. Assuming that the parameters distribution is Gaussian, we propose to sample a set of parameters from this distribution, and then to act greedily according to the resulting sampled Q-function. This type of scheme was first proposed by Thompson (1933) for a bandit problem, and it has been recently introduced into the reinforcement learning community in the tabular case (Dearden et al., 1998; Strens, 2000). Let the Thompson be:

$$\pi(a_{i+1}|s_{i+1}) = \operatorname*{argmax}_{b\in A} \hat{Q}_\xi(s_{i+1}, b) \text{ with } \xi \sim \mathcal{N}(\hat{\theta}_{i|i-1}, P_{i|i-1}) \quad (16)$$

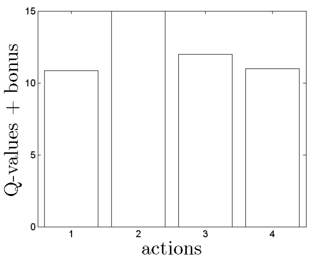
We illustrate the Thompson policy on Fig. 6.d by showing the distribution of the greedy action (recall that parameters are random, and thus the greedy action too). The highest probability is associated to action 3. However, notice that a highest probability is associated to action 1 than to action 4: the first one has a lower estimated Q-value, but it is less certain.



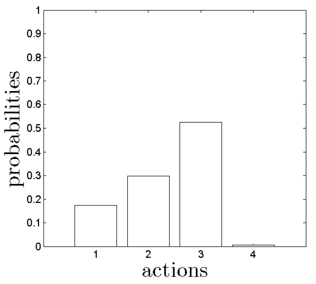
a. ϵ-greedy.



b. Confident-greedy.



c. Bonus-greedy.



d. Thompson.

Figure 6: Policies.

5.5 Experiment

The bandit problem is an MDP with one state and N actions. Each action a implies a reward of 1 with probability p_a, and a reward of 0 with probability $1 - p_a$. For an action a^* (randomly chosen at the beginning of each experiment), the probability is set to $p_{a^*} = 0.6$. For all other actions, the associated probability is uniformly and randomly sampled between 0 and 0.5: p_a $U_{[0,0.5]}$, $a = a^*$. Presented results are averaged over 1000 experiments. The performance of a method is measured as the percentage of time the optimal action has been chosen, given the number of interactions between the agent and the bandit. A tabular representation is adopted for KTD-SARSA, and the following parameters are used: $N = 10$, $P_{0|0} = 0.1I$, $\theta_{0|0} = I$, $P_{n_i} = 1$, $\epsilon = 0.1$, $\alpha = 0.3$, $\beta_0 = 1$ and $\beta = 10$. As the considered bandit has $N = 10$ arms, a random policy has a performance of 0.1. Notice also that a purely greedy policy would choose systematically the first action for which the agent has observed a reward.

Results presented in Fig. 7 compare the four schemes. The ϵ-greedy policy serves as a baseline, and all proposed schemes using the available uncertainty performs better. Thompson policy and confident-greedy policy perform approximately equally well, and the best results are obtained by the bonus-greedy policy. Of course, these quite preliminary results do not allow to conclude about guarantees of convergence of

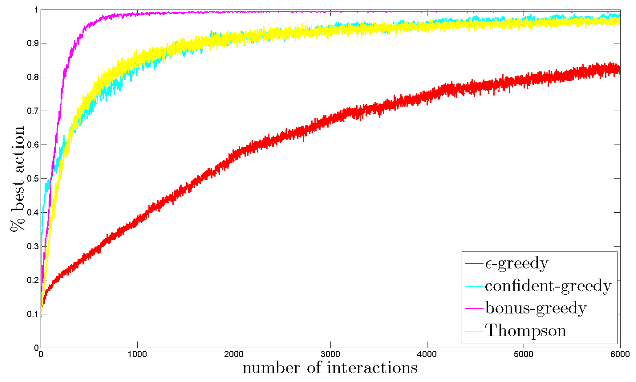


Figure 7: Bandit results.

the proposed schemes. However, they tend to show that the computed uncertainty information is meaningful and that it can provide useful for the dilemma between exploration and exploitation.

6 Conclusion

In this paper, we have shown how an uncertainty information about estimated values can be derived from KTD. We have also introduced an active learning scheme aiming at improving speed of convergence by sampling actions according to their relative uncertainty, as well as some adaptations of existing schemes for exploration/exploitation. Three experiments have been proposed. The first one illustrated on a simple continuous maze problem that the available uncertainty information makes sense. The second one shown that KTD-Q, a second-order value-iteration-like algorithm, is sample efficient. The improvement gained by using the proposed active learning scheme was also demonstrated. The proposed schemes for exploration/exploitation were also successfully experimented on a bandit problem. This is a first step toward combining the dilemma between exploration and exploitation with value function approximation. The next step is to adapt more existing approaches dealing with the exploration/exploitation dilemma designed for tabular representation of the value function to the KTD framework. New schemes can also be envisioned. More specifically, being involved in industrial applications of reinforcement learning, we are interested in safe exploration.

References

R. Dearden, N. Friedman, and S. J. Russell. Bayesian Q-Learning. In *AAAI/IAAI*, pages 761–768, 1998.

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

M. Geist, O. Pietquin, and G. Fricout. Kalman Temporal Differences: the deterministic case. In *Proceedings of the IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2009)*, Nashville, TN, USA, April 2009a.

M. Geist, O. Pietquin, and G. Fricout. Tracking in reinforcement learning. In *Proceedings of the 16th International Conference on Neural Information Processing (ICONIP 2009)*, Bangkok (Thailand), December 2009b. Springer.

S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

L. P. Kaelbling. *Learning in embedded systems*. MIT Press, 1993.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D): 35–45, 1960.

J. Z. Kolter and A. Y. Ng. Near-Bayesian Exploration in Polynomial Time. In *Proceedings of the 26th international conference on Machine learning (ICML 09)*, New York, NY, USA, 2009. ACM.

M. G. Lagoudakis and R. Parr. Least-Squares Policy Iteration. *Journal of Machine Learning Research*, 4: 1107–1149, 2003.

Y. Sakaguchi and M. Takano. Reliability of internal prediction/estimation and its application: I. adaptive action selection reflecting reliability of value function. *Neural Networks*, 17(7):935–952, 2004.

A. L. Strehl and M. L. Littman. An Analysis of Model-Based Interval Estimation for Markov Decision Processes. *Journal of Computer and System Sciences*, 2006.

M. Strens. A Bayesian Framework for Reinforcement Learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 943–950. Morgan Kaufmann, San Francisco, CA, 2000.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1996.

W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of two samples. *Biometrika*, (25):285–294, 1933.

H. Yu and D. P. Bertsekas. Q-Learning Algorithms for Optimal Stopping Based on Least Squares. In *Proceedings of European Control Conference*, Kos, Greece, 2007.