Online Knowledge Gradient Exploration in an Unknown Environment

Saba Q. Yahyaa and Bernard Manderick
Department of Computer Science, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium

Keywords: Online reinforcement learning, value function approximation, (kernel-based) least squares policy iteration, approximate linear dependency kernel sparsification, knowledge gradient exploration policy.

Abstract: We present online kernel-based LSPI (or least squares policy iteration) which is an extension of offline kernel-based LSPI. Online kernel-based LSPI combines characteristics of both online LSPI and offline kernel-based LSPI to improve the convergence rate as well as the optimal policy performances of the online LSPI. Online kernel-based LSPI uses knowledge gradient policy as an exploration policy and the approximate linear dependency based kernel sparsification method to select features automatically. We compare the optimal policy performance of online kernel-based LSPI and online LSPI on 5 discrete Markov decision problems, where online kernel-based LSPI outperforms online LSPI.

1 INTRODUCTION

A Reinforcement Learning (RL) agent has to learn to make optimal sequential decisions while interacting with its environment. At each time step, the agent takes an action and as a result the environment transits from the current state to the next one while the agent receives feedback signal from the environment in the form of a scalar reward.

The mapping from states to actions that specifies which actions to take in states is called a policy π and the goal of the agent is to find the optimal policy π^*, i.e. the one that maximises the total expected discounted reward, as soon as possible. The state-action value function $Q^\pi(s,a)$ is defined as the total expected discounted reward obtained when the agent starts in state s, takes action a, and follows policy π thereafter. The optimal policy maximises these $Q^\pi(s,a)$ values.

When the agent's environment can be modelled as a Markov Decision Process (MDP) then the Bellman equations for the state-action value functions, one per state-action pair, can be written down and can be solved by algorithms like policy iteration or value iteration (Sutton and Barto, 1998). We refer to Section 2.1 for more details.

When no such model is available, the Bellman equations cannot be written down. Instead, the agent has to rely only on information collected while interacting with its environment. At each time step, the information collected consists of the current state, the action taken in that state, the reward obtained and the next state of the environment. The agent can either learn *offline* when firstly a batch of past experience is collected and subsequently used and reused or *online* when it tries to improve its behaviour at each time step based on the current information.

Fortunately, the optimal Q-values can still be determined using Q-learning (Sutton and Barto, 1998) which represents the actions-value $Q^\pi(s,a)$ as a lookup table and uses the agent's experience to build the $Q^\pi(s,a)$. Unfortunately, when the state and/or the action spaces are large finite or continuous space, the agent faces a challenge called the curse of dimensionality, since the memory space needed to store all the Q-values grows exponentially in the number of states and actions. Computing all Q-values becomes infeasible. To handle this challenge, function approximation methods have been introduced to approximate the Q-values, e.g. (Lagoudakis and Parr, 2003) have proposed Least Squares Policy Iteration (LSPI) to find the optimal policy when no model of the environment is available. LSPI is an example of both approximate policy iteration and offline learning. LSPI approximates the Q-values using a linear combination of predefined basis functions. The used predefined basis functions have a large impact on the performance of LSPI in terms of the number of iterations that LSPI needs to converge to a policy, the probability that the converged policy is optimal, and the accuracy of the approximated Q-values.

Q. Yahyaa S. and Manderick ..
Online Knowledge Gradient Exploration in an Unknown Environment.
DOI: 10.5220/0004718700050013
In *Proceedings of the 6th International Conference on Agents and Artificial Intelligence* (ICAART-2014), pages 5-13
ISBN: 978-989-758-015-4

To improve the accuracy of the approximated Q-values and to find a (near) optimal policy, (X. Xu and Lu, 2007) have proposed Kernel-Based LSPI (KBLSPI), an example of offline approximated policy iteration that uses Mercer kernels to approximate Q-values (Vapnik, 1998). Moreover, kernel-based LSPI provides automatic feature selection by the kernel basis functions since it uses the approximate linear dependency sparsification method described in (Y. Engel and Meir, 2004).

(L. Buşoniu and Babuška, 2010) have adapted LSPI, which does offline learning, for online reinforcement learning and the result is called *online LSPI*. A good online learning algorithm must quickly produce acceptable performance rather than at the end of the learning process as is the case in offline learning. In order to obtain good performance, an online algorithm has to find a proper balance between exploitation, i.e. using the collected information in the best possible way, and exploration, i.e. testing out the available alternatives (Sutton and Barto, 1998). Several exploration policies are available for that purpose and one of the most popular ones is ε-greedy exploration that selects with probability $1-\varepsilon$ the action with the highest estimated Q-value and selects uniformly, randomly with probability ε one of the actions available in the current state. To get good performance, the parameter ε has to be tuned for each problem. To get rid of parameter tuning and to increase the performance of online LSPI, (Yahyaa and Manderick, 2013) have proposed using Knowledge Gradient (KG) policy (I.O. Ryzhov and Frazier, 2012) in the online-LSPI.

To improve the performance of online-LSPI and to get automatic feature selection, we propose online kernel-based LSPI and we use the knowledge gradient (KG) as an exploration policy. The rest of the paper is organised as follows: In Section 2 we present Markov decision processes, LSPI, the knowledge gradient policy for online learning, kernel-based LSPI and the approximate linear dependency test. While in Section 3, we present the knowledge gradient policy in online kernel-based LSPI. In Section 4 we give the domains used in our experiments and our results. We conclude in Section 5.

2 PRELIMINARIES

In this section, we discuss Markov decision processes, online LSPI, the knowledge gradient exploration policy (KG), offline kernel-based LSPI (KBLSPI) and approximate linear dependency (ALD).

2.1 Markov Decision Process

A finite Markov decision process (MDP) is a 5-tuple (S,A,P,R,γ), where the state space S contains a finite number of states s and the action space A contains a finite number of actions a, the transition probabilities $P(s,a,s')$ give the conditional probabilities $p(s'|s,a)$ that the environment transits to state s' when the agent takes action a in state s, the reward distributions $R(s,a,s')$ give the expected immediate reward when the environment transits to state s' after taking action a in state s, and $\gamma \in [0,1)$ is the discount factor that determines the present value of future rewards (Puterman, 1994; Sutton and Barto, 1998).

A deterministic policy $\pi : S \rightarrow A$ determines which action a the agent takes in each state s. For the MDPs considered, there is always a deterministic optimal policy and so we can restrict the search process to such policies (Puterman, 1994; Sutton and Barto, 1998). By definition, the state-action value function $Q^{\pi}(s,a)$ for a policy π gives the expected total discounted reward $\mathbb{E}_{\pi}(\sum_{i=t}^{\infty}\gamma^{i}r_t)$ when the agent starts in state s, takes action a and follows policy π thereafter. The goal of the agent is to find the optimal policy π^*, i.e. the one that maximizes Q^{π} for every state s and action a: $\pi^*(s) = argmax_{a\in A}Q^*(s,a)$ where $Q^*(s,a) = max_{\pi}Q^{\pi}(s,a)$ is the optimal state-action value function. For the MDPs considered, the Bellman equations for the state-action value function Q^{π} are given by

$$Q^{\pi}(s,a) = R(s,a,s') + \gamma\sum_{s'}P(s,a,s')Q^{\pi}(s',a') \quad (1)$$

In Equation 1, the sum is taken over all states s' that can be reached from state s when action a is taken, and the action a' taken in next state s' is determined by the policy π, i.e. $a' = \pi(s')$. If the MDP is completely known then algorithms such as value or policy iteration find the optimal policy π^*. Policy iteration starts with an initial policy π_0, e.g. randomly selected, and repeats the next two steps until no further improvement is found: 1) *policy evaluation* where the current policy π_i is evaluated using Bellman equations 1 to calculate the corresponding value function Q^{π_i}, and 2) *policy improvement* where this value function is used to find an improved new policy π_{i+1} that is greedy in the previous one, i.e. $\pi_{i+1} = argmax_{a\in A}Q^{\pi_i}(s,a)$ (Sutton and Barto, 1998).

For finite MDPs, the action-value functions Q^{π} for a policy π can be represented by a lookup table of size $|S| \times |A|$, one entry per state-action pair. However, when the state and/or action spaces are large, this approach becomes computationally infeasible due to the curse of dimensionality and one has to rely on function approximation instead. Moreover, the agent does

not know the transition probabilities $P(s,a,s')$ and the reward distributions $R(s,a,s')$. Therefore, it must rely on information collected while interacting with the environment to learn the optimal policy. The information collected is a trajectory of samples of the form (s_t,a_t,r_t,s_{t+1}) or $(s_t,a_t,r_t,s_{t+1},a_{t+1})$, where s_t, a_t, r_t, s_{t+1}, and a_{t+1}, are the state, the action in the state, the reward, the next state, and the next action in the next state, respectively. To overcome these problems, least squares policy iteration (LSPI) uses such samples to approximate the Q^π-values (Lagoudakis and Parr, 2003).

More recently, (L. Buşoniu and Babuška, 2010) have adapted LSPI so that it can work online and (Yahyaa and Manderick, 2013) have used the knowledge gradient (KG) policy in this online LSPI. Since we are interested in the most challenging RL problem: online learning in a stochastic environment of which no model is available. Therefore, we are going to compare the performance of online-LSPI with the proposed algorithm using KG policy.

2.2 Least Squares Policy Iteration

LSPI approximates the action-value Q^π for a policy π in a linear way (Lagoudakis and Parr, 2003):

$$\hat{Q}^\pi(s,a;w^\pi) = \sum_{i=1}^{n} \phi_i(s,a) w_i^\pi \tag{2}$$

where n, $n << |S \times A|$, is the number of basis functions, the weights $(w_i^\pi)_{i=1}^n$ are parameters to be learned for each policy π, and $\{\phi_i(s,a)\}_{i=1}^n$ is the set of predefined basis functions. Let Φ be the basis matrix of size $|S \times A| \times n$, where each row contains the values of all basis functions in one of the state-action pairs (s,a) and each column contains the values of one of the basis functions ϕ_i in all state-action pairs and let w^π be a column weight vector of length n.

Given a trajectory of length L of samples $(s_t,a_t,r_t,s_{t+1})_{t=1}^L$. Offline-LSPI is an example of approximated policy iteration and repeats the following two steps until no further improvement in the policy is obtained: 1) *Approximate policy evaluation* that approximates the state-action value function Q^π of the current policy π, and 2) *Approximate policy improvement* that derives from the current estimated state-action value functions $\hat{Q}^\pi$ a better policy π', i.e. $\pi' = argmax_{a \in A} \hat{Q}^\pi(s,a)$

Using the least square error of the projected Bellman's equation, Equation 1, the weight vector w^π can be approximated as follows (Lagoudakis and Parr, 2003):

$$\hat{A}\hat{w}^\pi = \hat{b} \tag{3}$$

where $\hat{A}$ is a matrix and $\hat{b}$ is a vector. Offline-LSPI updates the matrix $\hat{A}$ and the vector $\hat{b}$ from all available samples as follows:

$$\hat{A}_t = \hat{A}_{t-1} + \phi(s_t,a_t)[\phi(s_t,a_t) - \gamma\phi(s_{t+1},\pi(s_{t+1}))]^T$$
$$\hat{b}_t = \hat{b}_{t-1} + \phi(s_t,a_t) r_t \tag{4}$$

where T is the transpose and r_t is the immediate reward that is obtained at time step t. After iterating over all collected samples, $\hat{w}^\pi$ can be found. (L. Buşoniu and Babuška, 2010) have adapted offline-LSPI for online learning. The changes with respect to the offline algorithm are twofold: 1) online-LSPI updates the matrix $\hat{A}$ and the vector $\hat{b}$ after each time step t. Then, after every few samples K_θ obtained from the environment, online-LSPI estimates the weight vector $\hat{w}^\pi$ for the current policy π, computes the corresponding approximated $\hat{Q}$-function, and derives an improved new learned policy π', $\pi' = argmax_{a \in A} \hat{Q}^\pi(s,a)$. When $K_\theta = 1$, online-LSPI is called fully optimistic and when $K_\theta > 1$ is a small value, online-LSPI is called partially optimistic. 2) online-LSPI needs an exploration policy and (Yahyaa and Manderick, 2013) proposed using KG policy as an exploration policy instead of ε-greedy policy. (Yahyaa and Manderick, 2013) have shown that the performance of the online-LSPI is increased, e.g. the average frequency that the learned policy is converged to the optimal policy. Therefore, we are going to use KG policy in our algorithm and experiments.

2.3 KG Exploration Policy

Knowledge gradient KG (I.O. Ryzhov and Frazier, 2012) assumes that the rewards of each action a are drawn according to a probability distribution and it takes normal distributions $N(\mu_a,\sigma_a^2)$ with mean μ_a and standard deviation σ_a. The current estimates, based on the rewards obtained so far, are denoted by $\hat{\mu}_a$ and $\hat{\sigma}_a$. And, the root-mean-square error (RMSE) of the estimated mean reward $\hat{\mu}_a$ given n rewards resulting from action a is given by $\tilde{\sigma}_a = \hat{\sigma}_a/\sqrt{n}$. The KG is an index strategy that determines for each action a the index $V^{KG}(a)$ and selects the action with the 'highest' index. The index $V^{KG}(a)$ is calculated as follows:

$$V^{KG}(a) = \tilde{\sigma}_a f\left(-\left|\frac{\hat{\mu}_a - \max_{a' \neq a} \hat{\mu}_{a'}}{\tilde{\sigma}_a}\right|\right) \tag{5}$$

In this equation, $f(x) = \phi_{KG}(x) + x\Phi_{KG}(x)$ where $\phi_{KG}(x) = 1/\sqrt{2\pi}\ \exp(-x^2/2)$ is the density of the standard normal distribution and $\Phi_{KG}(x) = \int_{-\infty}^{x} \phi(x')dx'$ is its cumulative distribution. The parameter $\tilde{\sigma}_a$ is the RMSE of the estimated mean reward $\hat{\mu}_a$. Then KG selects the next action according to:

$$a_{KG} = \underset{a \in A}{\operatorname{argmax}} \left(\hat{\mu}_a + \frac{\gamma}{1-\gamma} V^{KG}(a) \right) \quad (6)$$

where the second term in the right hand side is the total discounted index of action a. KG prefers those actions about which comparatively little is known. These actions are the ones whose RMSE (or spread) $\tilde{\hat{\sigma}}_a$ around the estimated mean reward $\hat{\mu}_a$ is large. Thus, KG prefers an action a over its alternatives if its confidence in the estimated mean reward $\hat{\mu}_a$ is low.

For discrete MDPs, (Yahyaa and Manderick, 2013) estimated the Q-values $\hat{Q}(s_t, a_i)$ and the RMSE of the estimated Q-value $\tilde{\hat{\sigma}}_q^2$ to calculate the index $V^{KG}(a_i)$ for each available action $a_i, a_i \in A_{s_t}$ in the current state s_t, where A_{s_t} is the set of actions in state s_t. The pseudocode algorithm of the KG exploration policy is shown in Figure 1. KG is easy to implement and does not have parameters to be tuned like ε-greedy or $softmax$ action selection policies (Sutton and Barto, 1998). KG balances between exploration and exploitation by adding an exploration bonus to the estimated Q-values for each available action a_i in the current state s_t and this bonus depends on all estimated Q-values $\hat{Q}(s_t, a_i)$ and the RMSE of the estimated Q-value $\tilde{\hat{\sigma}}_q^2$ (steps: 2-8 in Figure 1). The RMSE $\tilde{\hat{\sigma}}_q^2$ are updated according to (Powell, 2007).

1. Input: current state s_t;discount factor γ; the current estimates $\hat{Q}(s_t, a_i)$;the current RMSEs $\tilde{\hat{\sigma}}_q^2(s_t, a_i)$ for all actions a_i in state s_t
2. For $a_i \in A_{s_t}$
3. $\acute{\hat{Q}}(s_t, a_i) \leftarrow \underset{a_j \in A_{s_t}, a_j \neq a_i}{\operatorname{argmax}} \hat{Q}(s_t, a_j)$
4. End for
5. For $a_i \in A_{s_t}$
6. $\zeta_{a_i} \leftarrow -\operatorname{abs}((\hat{Q}(s_t, a_i) - \acute{\hat{Q}}(s_t, a_i))/\tilde{\hat{\sigma}}_q(s_t, a_i))$; $f(\zeta_{a_i}) \leftarrow \zeta_{a_i} \Phi_{KG}(\zeta_{a_i}) + \phi_{KG}(\zeta_{a_i})$
7. $V^{KG}(a_i) \leftarrow \hat{Q}(s_t, a_i) + \frac{\gamma}{1-\gamma} \tilde{\hat{\sigma}}_q(s_t, a_i) f(\zeta_{a_i})$
8. End for
9. Output: $a_t \leftarrow \underset{a_i \in A}{\operatorname{argmax}} V^{KG}(a_i)$

Figure 1: Algorithm: (Knowledge Gradient).

2.4 Kernel-based LSPI

Kernel-based LSPI (X. Xu and Lu, 2007) is a kernelized version of offline-LSPI. Kernel-based LSPI uses Mercer's kernels in the approximated policy evaluation and improvement (Vapnik, 1998). Given a finite set of points, i.e. $\{z_1, z_2, \cdots, z_t\}$, where z_i is the state-action pair, with the corresponding set of basis functions, i.e. $\phi(z) : z \rightarrow \mathcal{R}$. Mercer theorem states the kernel function K is a positive definite matrix, i.e. $K(z_i, z_j) = < \phi(z_i), \phi(z_j) >$.

Given a trajectory of length L of samples and an initial policy π_0. Offline kernel-based LSPI (KBLSPI) uses the approximate linear dependency based sparsification method to select a part of the data samples and consists a dictionary Dic elements set, i.e. $Dic = \{(s_i, a_i)\}_{i=1}^{|Dic|}$ with the corresponding kernel matrix K_{Dic} of size $|Dic \times Dic|$ (Y. Engel and Meir, 2004). Kernel-based LSPI repeats the following two steps: 1) *Approximate policy evaluation*, kernel-based LSPI approximates the weight vector $\hat{w}^\pi$ for policy π, Equation 3 from all available samples as follows:

$$\hat{A}_t = \hat{A}_{t-1} + k((s_t, a_t), j)[k((s_t, a_t), j) - \gamma k((s_{t+1}, \pi(s_{t+1})), j)]^T$$

$$\hat{b}_t = \hat{b}_{t-1} + k((s_t, a_t), j) r_t, j \in Dic, j = 1, \cdots, |Dic| \quad (7)$$

where $k(.,.)$ is a kernel function between two points (a state-action pair (s,a) and j, where j is the state-action pair z_j that is element in the dictionary Dic, i.e. $j \in \{z_1, z_2, \cdots, z_{|Dic|}\}$). The matrix $\hat{A}$ should be initialized to a small multiple of the identity matrix to calculate the inverse of $\hat{A}$ or using the pseudo inverse. After iterating for all the collected samples, $\hat{w}^\pi$ can be found and the approximated Q^π-values for policy π is the following linear combination:

$$\hat{Q}^\pi(s,a) = \hat{w}^\pi k((s,a), j),\ j \in Dic,\ j = 1, 2, \cdots, |Dic| \quad (8)$$

2) *Approximate policy improvement*, KBLSPI derives a new learned policy which is the greedy one, i.e. $\pi'(s) = \operatorname{argmax}_{a \in A} \hat{Q}^\pi(s,a)$. The above two steps are repeated until no change in the improved policy or a maximum number of iterations is reached.

2.5 Approximate Linear Dependency

Given a set of data samples D from a MDP, i.e. $D = \{z_1, \ldots, z_L\}$, where z_i is a state-action pair and the corresponding linear independent basis functions set Φ, i.e. $\Phi = \{\phi(z_1), \cdots, \phi(z_L)\}$. Approximate linear dependency ALD method (Y. Engel and Meir, 2004) over the data samples set D is to find a subset Dic, i.e. $Dic \subset D$ whose elements $\{z_i\}_{i=1}^{|Dic|}$ and the corresponding basis functions are stored in Φ_{Dic}, i.e. $\Phi_{Dic} \subset \Phi$.

The data dictionary Dic is initially empty, i.e. $Dic = \{\}$ and ALD is implemented by testing every basis function ϕ in Φ, one at time. If the basis function $\phi(z_t)$ can not be approximated, within a predefined accuracy v, by the linear combination of the basis functions of the elements that stored in Dic_t, then the basis function $\phi(z_t)$ will be added to Φ_{Dic_t} and z_t will be added to Dic_t, otherwise z_t will not be added to Dic_t and $\phi(z_t)$ will not be added to Φ_{Dic}. As a result, after the ALD test, the basis functions of Φ_{Dic} can approximate all the basis functions of Φ.

At time step t, let $Dic_t = \{z_j\}_{j=1}^{|Dic_t|}$ and the corresponding basis functions are stored in Φ_{Dic_t}, i.e. $\Phi_{Dic_t} = \{\phi(z_j)\}_{j=1}^{|Dic_t|}$ and z_t is a given state-action pair at time t. The ALD test on the basis function $\phi(z_t)$ supposes that the basis functions are linearly dependent and uses least squares error to approximate $\phi(z_t)$ by all the basis functions of the elements in Dic_t, for more detail we refer to (Engel and Meir, 2005). The least squares error is:

$$error = \min_c || \sum_{j=1}^{|Dic_t|} c_j \phi(z_j) - \phi(z_t)||^2 < \nu \quad (9)$$

$$error = k(z_t, z_t) - k_{Dic_t}^T(z_t) c_t, \text{ where} \quad (10)$$

$$c_t = K_{Dic_t}^{-1} k_{Dic_t}(z_t),$$

$$k_{Dic_t}^T = [k(1, z_t), \cdots, k(j, z_t), \cdots, k(|Dic_t|, z_t)]$$

If the error is larger than predefined accuracy ν, then z_t will be added to the dictionary elements, i.e. $Dic_{t+1} = Dic_t \cup \{z_t\}$, otherwise $Dic_{t+1} = Dic_t$. After testing all the elements in the data samples set D, the matrix K_{Dic}^{-1} can be computed, this is in the offline learning method. For online learning, the matrix K_{Dic}^{-1} can be updated at each time step (Y. Engel and Meir, 2005).

At each time step t, if the error that results from testing the basis functions of z_t is smaller than ν, then $Dic_{t+1} = Dic_t$ and $K_{Dic_{t+1}}^{-1} = K_{Dic_t}^{-1}$, otherwise $Dic_{t+1} = Dic_t \cup \{z_t\}$. The matrix $K_{Dic_{t+1}}^{-1}$ is updated as follows:

$$K_{Dic_{t+1}}^{-1} = \frac{1}{error_t} \begin{bmatrix} error_t K_{Dic_t}^{-1} & -c_t \\ -c_t^T & 1 \end{bmatrix} \quad (11)$$

3 ONLINE KERNEL-BASED LSPI

Online kernel-based LSPI (KBLSPI) is a kernelised version of online-LSPI and the pseudocode is given in Figure 2. Given the basis function set Φ, the initial learned policy π_0, the accuracy parameter ν and the initial state s_1. At each time step t, online-KBLSPI uses the KG exploration policy, the algorithm in Figure 1. to select the action a_t in the state s_t (step: 4) and observes the new state s_{t+1} and reward r_t. The action a_{t+1} in s_{t+1} is chosen by the learning policy π_t. The algorithm in Figure 2 performs the ALD test, Section 2.5 on the basis functions of z_t and z_{t+1} to provide feature selection (steps: 7-14), where z_t is the state-action pair (s_t, a_t) at time step t and z_{t+1} is the state-action pair (s_{t+1}, a_{t+1}) at time step $t+1$. If the basis functions of a given state-action pair, i.e. z_t and z_{t+1} can not approximated by the basis functions of the elements that stored in the dictionary Dic_t, then the given state-action pair will be added to the dictionary, the inverse kernel matrix K^{-1} will be updated, the number of columns and rows of the matrix $\hat{A}$ will be increased and the number of dimensions of the vector $\hat{b}$ will be increased (step: 11). Otherwise, the given state-action pair will not be added to the dictionary (step: 12). Then, online-KBLSPI updates the matrix $\hat{A}$ and the vector $\hat{b}$ (steps: 15-16). After few samples K_θ obtained from the environment, online-KBLSPI estimates the weight vector $\hat{w}^{\pi_t}$ under the current policy π_t (step: 18) and approximates the corresponding state-action value function $\hat{Q}^{\pi_t}$ (step: 19), i.e. *approximate policy evaluation*. Then, online-KBLSPI derives an improved new learned policy π_{t+1} which is a greedy one (step: 20), i.e. *approximated policy improvement*. This procedure is repeated until the end of playing L steps which is the horizon of an experiment.

4 EXPERIMENTS

In this section, we describe the test domain, the experimental setup and the experiments where we compare online-LSPI and online-KBLSPI using KG policy. All experiments are implemented in MATLAB.

4.1 Test Domain/Experimental Setup

The *test domain* consists of 5 MDPs as shown in Figure 3, each with discount factor $\gamma = 0.9$. The first three domains are the 4-, 20-, and 50- chain. The 4-, and 20-domain are also used in (Lagoudakis and Parr, 2003; X. Xu and Lu, 2007) and the 50-chain is used in (Lagoudakis and Parr, 2003). In general, the x-open chain which is originally studied in (Koller and Parr, 2000) consists of a sequence of x states, labeled from s_1 to s_x. In each state, the agent has 2 actions, either *GoRight* (R) or *GoLeft* (L). The actions succeed with probability 0.9 changing the state in the intended direction and fail with probability 0.1 changing the state in the opposite direction. The reward structure can vary such as the agent gets reward for visiting the middle states or the end states. For the 4-chain problem, the agent is rewarded 1 in the middle states, i.e. s_2 and s_3, and 0 at the edge states, i.e. s_1 and s_4. The optimal policy is R in states s_1 and s_2 and L in states s_3 and s_4. (Koller and Parr, 2000) used a policy iteration method to solve the 4-chain and showed that the resulting suboptimal policies oscillate between R R R R and L L L L. The reason is because of the limited approximation abilities of basis functions in policy evaluation. For the 20-chain, the agent is rewarded 1 in states s_1 and s_{20}, and 0 else-

1. Input: $|S|$;$|A|$;discount factor γ;accuracy v; set of basis functions $\Phi = \{\phi_1, \cdots, \phi_n\}$;initial learned policy π_0;length of trajectory L; policy improvement interval K_θ;reward $r \sim N(\mu_a, \sigma_a^2)$; initial state s_1.

2. Intialize: $\hat{A} \leftarrow 0; \hat{b} \leftarrow 0; s_t; Dic_t = \{\ \}$; $K_{|SA|\times|SA|} =< \Phi^T, \Phi >; K_{Dic_t}^{-1} = []; \hat{Q}_{|SA|} \leftarrow 0$

3. For $t = 1, \cdots, L$
4. $a_t \leftarrow KG$
5. $s_t,\ a_t$; observe: $s_{t+1}; r_t; a_{t+1} \leftarrow \pi_t(s_{t+1})$
6. $z_t \leftarrow (s_t) * |A| + a_t,\ z_{t+1} \leftarrow (s_{t+1}) * |A| + a_{t+1}$
7. For $z_i \in \{z_t,\ z_{t+1}\}$
8. $k^T(.,z_i) = [k(1,z_i), \cdots, k(j,z_i), \cdots, k(|Dic_t|,z_i)]$, $c(z_i) = K_{Dic_t}^{-1} * k(.,z_i)$
9. $error(z_i) = k(z_i,z_i) - k^T(.,z_i) * c(z_i)$
10. If $error(z_i) > v$
11. $Dic_{t+1} \leftarrow Dic_t \cup z_i$;

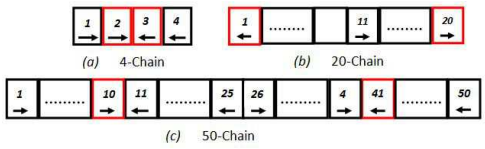
$$K_{Dic_{t+1}}^{-1} \leftarrow \frac{1}{error(z_i)} \begin{pmatrix} error(z_i)K_{Dic_t}^{-1} & -c(z_i) \\ -c(z_i)^T & 1 \end{pmatrix};$$

$$\hat{A}_t \leftarrow \begin{pmatrix} \hat{A}_t & 0 \\ 0 & 0 \end{pmatrix}; \hat{b}_t \leftarrow \begin{pmatrix} \hat{b}_t \\ 0 \end{pmatrix}$$

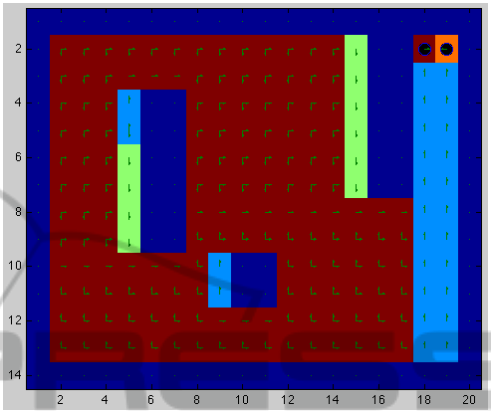
12. Else $Dic_{t+1} \leftarrow Dic_t; K_{Dic_{t+1}}^{-1} \leftarrow K_{Dic_t}^{-1}$
13. End if
14. End for
15. $\hat{A}_{t+1} \leftarrow \hat{A}_t + k(.,z_t)[k(.,z_t) - \gamma\ k(.,z_{t+1})]^T$
16. $\hat{b}_{t+1} \leftarrow \hat{b}_t + k(.,z_t)r_t$, $k(.,z_t) = [k(1,z_t), \cdots, k(j,z_t), \cdots, k(|Dic_{t+1}|,z_t)]^T$
17. If $t = (l+1)K_\theta$ then
18. $\hat{w}_l^{\pi_t} \leftarrow \hat{A}_{t+1}^{-1}\hat{b}_{t+1}$
19. for $z = z_1,\ z_2,\ \cdots,\ z_{|SA|}$
$k(.,z) = [k(1,z), \cdots, k(j,z), \cdots, k(|Dic_{t+1}|,z)]^T$
$\hat{Q}_l^{\pi_t}(z) = \hat{w}_l^{\pi_t,T} * k(.,z)$
end
20. $\pi_{t+1} \leftarrow \text{argmax}_a \hat{Q}_l^{\pi_t}(s,a) \forall_{s \in S}; \pi_t \leftarrow \pi_{t+1}; l \leftarrow l+1$
21. End if
22. $s_t \leftarrow s_{t+1}$
23. End for

24. Output: At each time step t, note down: the reward r_t and the learned policy π_t

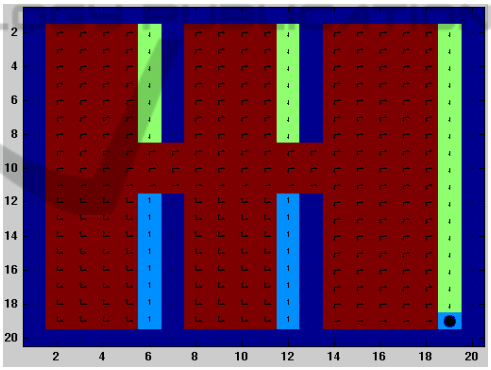
Figure 2: Algorithm: (Online-KBLSPI).



(*a*) The chain domains



(*b*) The grid$_1$ domain



(*c*) The grid$_2$ domain

Figure 3: Subfigure (*a*) is the chain domains, in the red cells, the agent gets rewards. Subfigure (*b*) is the grid$_1$ with 280 states and 188 accessible states. Subfigure (*c*) is the grid$_2$ with 400 states and 294 accessible states. The arrows show the optimal actions in each state.

where. The optimal policy is L from states s_1 through s_{10} and R from states s_{11} through s_{20}. And, for the 50-chain, The agent gets reward 1 in states s_{10} and s_{41} and 0 elsewhere. The optimal policy is R from state s_1 through state s_{10} and from state s_{26} through state s_{40}, and L from state s_{11} through state s_{25} and from state s_{41} through state s_{50} (Lagoudakis and Parr, 2003). The fourth and fifth MDPs, the grid$_1$ and grid$_2$ worlds, are used in (Sutton and Barto, 1998). The agent has 4 actions *Go Up, Down, Left* and *Right* and for each of them it transits to the intended state with probability 0.7 and fails with probability 0.1 changing the state to the one of other directions. The agent gets reward 1 if it reaches the goal state, -1 if it hits the wall, and 0 elsewhere.

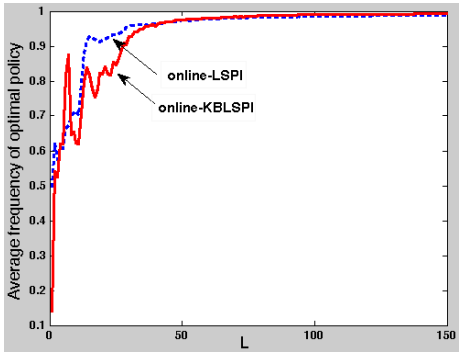
The *experimental setup* is as follows: For each of the 5 MDPs, we compared online-LSPI and online-KBLSPI using knowledge gradient KG policy as an exploration policy. For number of experiments $EXPs$ equals 1000 for the chain domains, 100 for the grid$_1$ domain and 50 for the grid$_2$ domain, each one with length L. The performance measures are: 1) the average frequency at each time step, i.e. at each time step t for each experiment, we computed the probability that the learned policy (step: 19) in Algorithm 2 reached to the optimal policy, then we took the average of $EXPs$ experiments to give us the average frequency at each time step. 2) the average cumulative

frequency at each time step, i.e. the cumulative average frequency at each time step t. (Mahadevan, 2008) used the 50-chain domain with length of trajectories L equals 5000, therefore, we used the same horizon. For other MDP domains we adapted the length of trajectories L according to the number of states, i.e. as the number of states is increased, L will be increased. For instance, L is set to 18800 for the grid world.

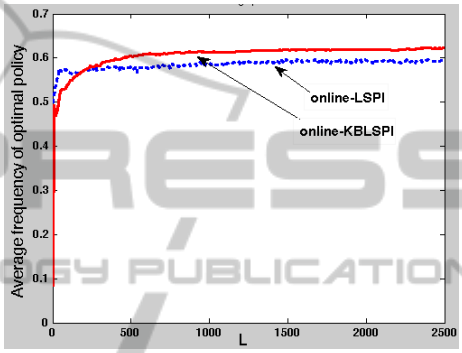
KG policy, needs estimated standard deviation and estimated mean for each state-action pair. Therefore, we assume that the reward has a normal distribution. For example, for the 50-chain problem, the agent is rewarded 1 if it goes to state 10, therefore, we set the reward in s_{10} to $N(\mu_1, \sigma_a^2)$, where $\mu_1 = 1$. And, the agent is rewarded 0 if it goes to s_1, therefore, we set the reward to $N(\mu_2, \sigma_a^2)$, where $\mu_2 = 0$. σ_a is the standard deviation of the reward which is set fixed and equal for each action, i.e. $\sigma_a = 0.01, 0.1, 1$. Moreover, KG exploration policy is a full optimistic policy, therefore, we set the policy improvement interval K_θ to 1. For each run, the initial state s_1 was selected uniformly, randomly from the state space S. We used the pseudo-inverse when the matrix $\hat{A}$ is non-invertible (Mahadevan, 2008).

For online KBLSPI, we define a kernel function K on state-action pairs, i.e. $K : |SA| \times |SA| \rightarrow \mathcal{R}$, we composed K into a state kernel K_s, i.e. $K_s : |S| \times |S| \rightarrow \mathcal{R}$ and an action kernel K_a, i.e. $K_a : |A| \times |A| \rightarrow \mathcal{R}$ as (Y. Engel and Meir, 2005). Therefore, the kernel function K is $K = K_s \otimes K_a$ where $\otimes$ is the Kronecker product. K is a kernel matrix because K_s and K_a are kernel matrices, we refer to (Scholkopf and Smola, 2002) for more details. The kernel state K_s is a Gaussian kernel, i.e. $k(s, s') = \exp^{-||s-s'||^2/(2\sigma_{ks}^2)}$ where σ_{ks} is the standard deviation of the kernel state function, s is the state at time t and s' is the state at time $t+1$. And, the action kernel is a Gaussian kernel, i.e. $k(a, a') = \exp^{-||a-a'||^2/(2\sigma_{ka}^2)}$ where σ_{ka} is the standard deviation of the kernel action function, a is the action at time t and a' is the action at time $t+1$. s and s', and a and a' are normalized as (X. Xu and Lu, 2007), e.g. for 50-chain with number of states $|S| = 50$ and number of actions $|A| = 2$, $s, s' \in \{1/|S|, \cdots, 50/|S|\}$ and $a, a' \in \{0.5, 1\}$. σ_{ks} and σ_{ka} are tuned empirically and set to 0.55 for the chain domains and 2.25 for the grid world domains (grid_1 and grid_2) We set the accuracy v in the approximated kernel basis to 0.0001.

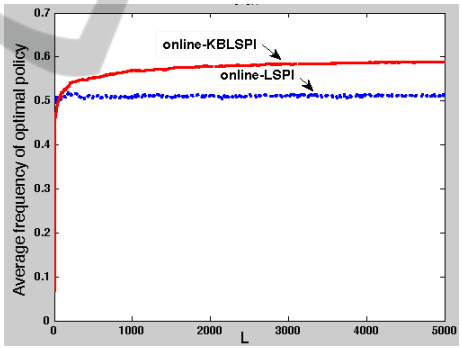
For online-LSPI, we used Gaussian basis functions $\phi_s = \exp^{-||s-c_i||^2/(2\sigma_\Phi^2)}$ where ϕ_s is the basis functions for state s with center nodes $(c_i)_{i=1}^n$ which are set with equal distance between each other, and σ_Φ is the standard deviation of the basis functions which is set to 0.55. The number of basis functions n equals 3 for 4-chain, 5 for 20-chain, and 10 for 50-chain as (Lagoudakis and Parr, 2003) and 40 for the grid_1 and grid_2 domains as (M. Sugiyama and Vijayakumar, 2008).



(*a*) Performance on 4-chain domain



(*b*) Performance on 20-chain domain

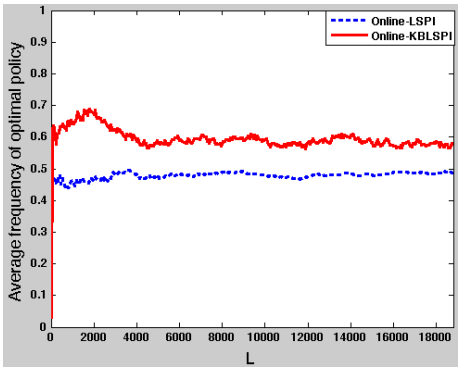


(*c*) Performance on 50-chain domain

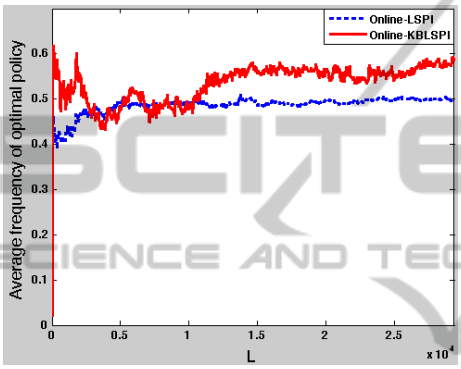
Figure 4: Performance of the average frequency by the KG policy in online-LSPI in blue and KG in online-KBLSPI in red. Subfigure (*a*) shows the performance on the 4-chain using standard deviation of reward $\sigma_a = 0.01$. Subfigure (*b*) shows the performance on the 20-chain using standard deviation of reward $\sigma_a = 1$. Subfigure (*c*) shows the performance on the 50-chain using $\sigma_a = 0.1$.

4.2 Experimental Results

The experimental results on the chain domains, i.e. 4-, 20-, and 50-chain show that the online-KBLSPI



(*a*) Performance on grid_1 domain



(*b*) Performance on grid_2 domain

Figure 5: Performance of the average frequency by the KG policy in online-LSPI in blue and KG in online-KBLSPI in red. Subfigure (*a*) shows the performance on the grid_1 domain using standard deviation of reward $\sigma_a = 0.01$. Subfigure (*b*) shows the performance on the grid_2 domain using standard deviation of reward $\sigma_a = 1$.

outperforms the online-LSPI according to the average frequency and cumulative average frequency of optimal policy performances for all values of the standard deviation of reward σ_a i.e. $\sigma_a = 0.01, 0.1$ and 1. Figure 4 shows how the performance of the learned policy is increased by using online-KBLSPI on the 4-chain, 20-chain and 50-chain.

The experimental results on the grid_1 domain show that the online-KBLSPI outperforms the online-LSPI according to the average frequency and cumulative average frequency of optimal policy performances for all values of the standard deviation of reward σ_a i.e. $\sigma_a = 0.01, 0.1$ and 1. And, the experimental results on the grid_2 domain show that the online-KBLSPI performs better than the online-LSPI for standard deviation of reward equals 1. Figure 5 shows how the performance of the learned policy is increased by using online-KBLSPI on the grid_1 and grid_2 domains.

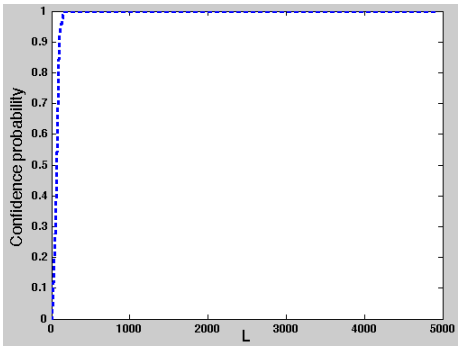
The results clearly show that online-KBLSPI usually converges faster than online-LSPI to the (near) optimal policies, i.e. the performance of the online KBLSPI is increased. Although, the performance of the online LSPI is better in the beginning and this is because the online LSPI uses its all basis functions, while online KBLSPI incrementally constructs its basis functions by the kernel sparsification method.

4.3 Statistical Methodology

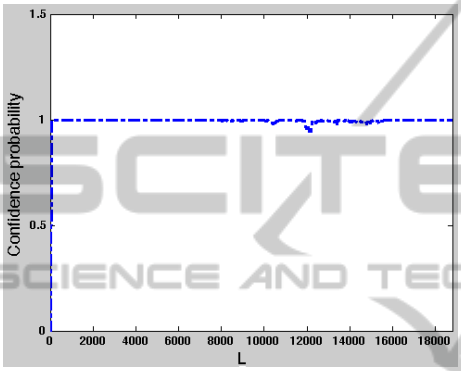
We used a statistical hypothesis test, i.e. students t-test with significance level $\alpha_{st} = 0.05$ to compare the performance of the average frequency of optimal policy that results from the online-LSPI and the online-KBLSPI at each time step t. The null hypothesis H_0 is the online-KBLSPI average frequency performance (AF_{KBLSPI}) larger than the online-LSPI average frequency performance (AF_{LSPI}) and the alternative hypothesis H_a is AF_{KBLSPI} less or equal AF_{LSPI}. We wanted to calculate the confidence in the null hypothesis, therefore, we computed the confidence probability *p*-value at each time step *t*. The *p*-value is the probability that the null hypothesis is correct. The confidence probability converges to 1 for all standard deviation of reward, i.e. $\sigma_a = 0.01, 0.1$, and 1 and for all domains, i.e. the 4-, 20-, and 50-chain domains and the grid world domains. Figure 6 shows how the *p*-value converges to 1 using the 50-chain, and the grid_1 domain with standard deviation of reward $\sigma_a = 0.1$. The *x*-axis gives the time steps (the length of trajectories). The *y*-axis gives the confidence probability, i.e. *p*-value. Figure 6 shows the confidence in the online kernel-based LSPI performance is very high, where the *p*-value converged quickly to 1.

5 CONCLUSIONS AND FUTURE WORK

We presented Markov decision process which is a mathematical model for the reinforcement learning. We introduced online and offline least squares policy iteration (LSPI) that find the optimal policy in an unknown environment. We presented knowledge gradient KG policy to be used as an exploration policy in the online learning algorithm. We introduced offline kernel-based LSPI (KBLSPI). We also introduced approximate linear dependency (ALD) method to select feature automatically and get rid of tuning empirically the center nodes. We proposed online-KBLSPI which uses KG exploration policy and ALD method. Finally, we compared online-KBLSPI and online-LSPI and concluded that the average frequency of optimal policy performance is improved by using online-KBLSPI. Future work must compare the performance



(*a*) *p*-value of the 50-chain domain



(*b*) *p*-value of the grid domain

Figure 6: The confidence probability *p*-value that the average frequency of optimal policy performance of online-KBLSPI performs better than online-LSPI. Subfigure (*a*) shows the *p*-value of the 50-chain using standard deviation of reward $\sigma_a = 0.1$. Subfigure (*b*) shows the *p*-value of the grid domain using standard deviation of reward $\sigma_a = 0.1$.

of online-LSPI and online-KBLSPI using other types of basis functions, e.g. the hybrid shortest path basis functions (Yahyaa and Manderick, 2012), must compare the performance using continuous MDP domain, e.g. Interval pendulum and must prove a convergence analysis of the online-KBLSPI.

REFERENCES

Engel, Y. and Meir, R. (2005). Algorithms and representations for reinforcement learning. Technical report, Ph.D. thesis, Senate of the Hebrew.

I.O. Ryzhov, W. P. and Frazier, P. (2012). The knowledge-gradient policy for a general class of online learning problems. *Operation Research*, 60(1):180–195.

Koller, D. and Parr, R. (2000). Policy iteration for factored mdps. In *Proceedings of the 16th Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

L. Buşoniu, D. Ernst, B. D. S. and Babuška, R. (2010). Online least-squares policy iteration for reinforcement learning control. In *Proceedings of the 2010 American Control Conference*.

Lagoudakis, M. G. and Parr, R. (2003). Model-free least squares policy iteration. Technical report, Ph.D. thesis, Duke University.

M. Sugiyama, H. Hachiya, C. T. and Vijayakumar, S. (2008). Geodesic gaussian kernels for value function approximation. *Journal of Autonomous Robots*, 25(3):287–304.

Mahadevan, S. (2008). *Representation Discovery Using Harmonic Analysis*. Morgan and Claypool Publishers.

Powell, W. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley and Sons, New York, USA.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, Inc., New York, USA.

Scholkopf, B. and Smola, A. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, Cambridge, MA, 1st edition.

Vapnik, V. (1998). *The Grid: Statistical Learning Theory*. Wiley Press, New York, United State of America.

X. Xu, D. H. and Lu, X. (2007). Kernel-based least squares policy iteration for reinforcement learning. *Journal of IEEE Transactions on Neural Network*, 18(4):973–992.

Y. Engel, S. M. and Meir, R. (2004). The kernel recursive least-squares algorithm. *Journal of IEEE Transactions on Signal Processing*, 52(8):2275–2285.

Y. Engel, S. M. and Meir, R. (2005). Reinforcement learning with gaussian processes. In *Proceedings of the 22nd International Conference on Machine learning (ICML)*, New York, NY, USA. ACM.

Yahyaa, S. Q. and Manderick, B. (2012). Shortest path gaussian kernels for state action graphs: An empirical study. In *The 24th Benelux Conference on Artificial Intelligence (BNAIC)*, Maastricht, The Netherlands.

Yahyaa, S. Q. and Manderick, B. (2013). Knowledge gradient exploration in online least squares policy iteration. In *The 5th International Conference on Agents and Artificial Intelligence (ICAART)*, Barcelona, Spain.