Sequential Decision-Making under Non-stationary Environments via Sequential Change-point Detection

Emmanuel Hadoux, Aurélie Beynier, and Paul Weng

Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, Paris, France
firstname.surname@lip6.fr

Abstract. Reinforcement Learning (RL) has been mainly interested in computing an optimal policy for an agent acting in a stationary environment. However, in many real world decision problems the assumption on the stationarity does not hold. One can view a non-stationary environment as a set of contexts (also called modes or modules) where a context corresponds to a possible stationary dynamics of the environment. Even most approaches assume that the number of modes is known, a RL method - Reinforcement Learning with Context Detection (RLCD)- has been recently proposed to learn an *a pirori* unknown set of contexts and detect context changes. In this paper, we propose a new approach by adapting the tools developed in statistics and more precisely in sequential analysis for detecting an environmental change. Our approach is thus more theoretically founded and necessitates less parameters than RLCD. We also show that our parameters are easier to interpret and therefore easier to tune. Finally, we show experimentally that our approach outperforms the current methods on several application problems.

1 Introduction

Reinforcement learning (RL) [15] is a powerful framework for sequential decision-making. It can be seen as an extension of the Markov Decision Process (MDP) model [13] when the environment (i.e., transitions and rewards) is not known. The aim in RL is for an agent to learn an optimal policy (e.g., how to act in order to maximize the expected discounted sum of rewards) while interacting with the unknown environment.

In RL, the environment is generally assumed to be stationary (e.g., transitions and rewards do not evolve with time). Without this assumption, the standard RL algorithms may not perform very well. However, one may argue that in practice, the encountered problems are often characterized by non-stationary environments.

As non-stationary environments come in many flavors, tackling the general case is very difficult, if not impossible. In this paper, we restrict ourselves to the situation where the environment changes rarely and somewhat abruptly. The slow rate of change allows learning and the abrupt change allows easier

detection. A typical example of such non-stationary problem is that of control of traffic lights. Depending on the time of the day, the traffic follows different "patterns" (e.g., rush hours v.s. non-rush hours).

In the literature, such kind of non-stationary environments has already been actively investigated [4, 6, 14] in the RL setting. Depending on authors and domains, the pattern followed by the environment, which constitutes a temporary stationary environment, is called mode, context, module or model. [4] proposed Hidden-Mode MDPs (HM-MDPs) as an extension of MDPs where the non-stationarity is modeled as a hidden Markov chain, assumed to be known. [5] learns the HM-MDP in a RL setting using the Baum-Welch algorithm. The drawback of this approach is that it assumes the number of modes is known. [6] applies ideas from adaptive control [11] to RL, which consists in learning multiple models, computing a "responsibility signal" to evaluate the goodness of each models and averaging the models using this signal. Here, again, the number of models is a priori fixed. More recently, [14] also proposed to learn several models in RL where an error score is computed for all models in order to select the best current model as the one minimizing this error score. Interestingly, their work allows tackling the case where the number of models is not a priori known. When all learned models has a poor error score, a new model is learned. However, their approach requires multiple parameters to be tuned, depending on the problem at hand. This may be a difficult task as they are not always easy to interpret and their interplay can be subtle and difficult to predict. Besides, their error score seems to be ad-hoc and not very theoretically founded.

In this paper, we propose a new approach for solving this sequential decision-making problem under non-stationary environments. Our main idea is to adapt the tools developped in statistics and more precisely in sequential analysis [7] for detecting an environmental change [1]. This research domain started with the seminal work of [16] that has been actively developed [10] ever since. One of the main problems studied in sequential analysis is that of change point detection, which consists in detecting a change in the statistical property of a random variable that is repeatedly observed. This research has many applications (seismic detection, industrial quality control, signal segmentation...). Although the works of [14] and [6] could somehow be reinterpreted in the sequential analysis framework, to the best of our knowledge, our paper presents the first work that explicitly exploits those statistical tools. In doing so, our approach is more theoretically founded and necessitates less parameters than that of [14]. We argue that those parameters are easier to interpret and therefore easier to set a priori for solving new problems. We show experimentally that our approach outperforms the current methods.

2 Background

2.1 MDP and RL

A *Markov Decision Process* (MDP) is a common model to represent sequential decision-making problems. It is characterized by a tuple $\langle S, A, T, R \rangle$ with S the

set of states, A the set of actions, T the transition function $S \times A \rightarrow \Pr(S)$ and R the reward function $S \times A \rightarrow \mathbb{R}$. Solving a problem modeled by an MDP consists in finding a *policy* π that is a mapping of A over S.

When all the components of the MDP are known, one can use algorithms such as *value iteration* [2] or *policy iteration* [9]. However, when T and/or R are unknown, *reinforcement learning* (RL) methods can be used. Indeed, the optimal policy is learned using reward feedbacks from the environment. The policy learned by RL is guaranteed to converge towards an optimal policy if the environment is stationary and the number of experiences is infinite. In practice, the policy learned is ϵ-optimal if the number of experiences is high enough.

The problems we try to solve in this paper are non-stationary. Therefore, we cannot directly use either MDP model or RL methods. However, as we restrict ourselves to the situation where the environment changes rarely, we can make the hypothesis that the environment is stationary during a timeframe and changes at the end of the current timeframe.

2.2 HM-MDP

Recent works propose to formalize non-stationary environments as a set of contexts (also called modes or modules) where a context corresponds to a possible stationary dynamics of the environments [4, 6, 14]. Hidden-Mode Markov Decision Processes have been proposed by Choi et al. [4] to formalize this idea.

An HM-MDP is characterized by a pair $\langle \mathbf{M}, C \rangle$ where $\mathbf{M}$ is a set of modes $\{m_1, \ldots, m_n\}$ with n the number of possible modes and $C : \mathbf{M} \times \mathbf{M} \rightarrow [0, 1]$ is a transition function over modes. For all $i \in \{1, \ldots, n\}$, $m_i = \langle \mathbf{S}, \mathbf{A}, T_i, R_i \rangle$ is an MDP. Note that $\mathbf{S}$ and $\mathbf{A}$ are shared by all m_i's and that an HM-MDP with $n = 1$ is a standard MDP. In HM-MDPs, the only observable information is the current state $s \in \mathbf{S}$, the current mode $m \in \mathbf{M}$ is not observable.

Recently this model has also been described in the context of MOMDPs [12, 3].

Choi et al. proposed a variant of the Baum-Welch algorithm to learn an HM-MDP. However, one major limitation of this approach is that the number of contexts (or modes) is assumed to be known in advance. Da Silva et al. [14] have thus proposed a reinforcement learning method for context detection to incrementally build possible models. Their approach allows for an automatic definition and detection of models. da Silva et al. define quality signals for each partial model. When the quality of the current model becomes worse than the quality of another model (already defined), the system detects a context change and switches to the other partial model. If the quality of all the models becomes too poor, a new model is built. However, this approach requires several parameters to be tuned for each decision-problem. These parameters are not easy to interpreted and their interactions are difficult to predict thus leading to an even more difficult tuning.

3 Detecting an Environmental Change

Let $M_0 = (S, A, T_0, R_0)$ and $M_1 = (S, A, T_1, R_1)$ be two modes or MDPs that are both assumed to be known. We assume the environment is currently represented by M_0 and at some unknown time step, the environment changes from mode M_0 to mode M_1. The problem we want to tackle here is that of detecting as soon as possible this environmental change. To that aim, a natural idea is to use statistical hypothesis tests for such detections, i.e., given an observed history, a null hypothesis "the current mode is M_0" is tested against an alternative hypothesis "the current mode is M_1". When performing such tests, one wants to minimize the probabilities of two contradictory errors:

- type I error: reject the null hypothesis when it is true,
- type II error: accept the null hypothesis when it is false.

In the online setting, sequential statistical tests are preferred: they perform repeated tests as observations becomes available and permit detections with smaller size samples in expectation [16] compared to standard statistical tests. Viewing detections as statistical tests highlights the contradiction between fast detection (type I error) and false detection (type II error).

A simple approach to implement those sequential statistical tests for change point detection is to recourse to cumulative sums (CUSUM) [1]. We present the CUSUM approach adapted for our purposes below.

3.1 Detecting a Change in Transition Distributions

In our setting, CUSUM can be specified as follows for detecting a change in the transition distributions. Let $(s_0, a_1, s_1, a_2, s_2, \ldots, s_{t-1}, a_t, s_t, \ldots)$ denote the observed history and define $S_0^T = 0$. At each time step $t \geq 1$, compute:

$$S_t^T = \max(0, S_{t-1}^T + \ln(\frac{T_1(s_t, a_t, s_{t+1})}{T_0(s_t, a_t, s_{t+1})})) \tag{1}$$

and compare S_t^T to a threshold $c^T > 0$. If $S_t^T \geq c^T$, then a change in the transition function is detected. The intuitive idea of CUSUM is quite simple: If M_1 is more likely than M_0 to have generated the recent history, then decide that the environment has changed.

3.2 Detecting a Change in Reward Distributions

As in the general case, the observed rewards are stochastic, we assume that R_0 and R_1 are functions from $S \times A$ to probability distributions over numerical values (actual obtained rewards). We denote $R_i(s, a, r)$ the probability of obtaining numerical reward r when choosing action a in state s in mode M_i. Moreover, to simplify the presentation, we assume that the possible numerical rewards are finite and known. This is generally not a very restrictive assumption as we are considering finite-state MDPs.

To detect a change in the reward function, the same procedure as for the transitions can then be applied. Let $(r_1, r_2, \ldots, r_t, \ldots)$ be the sequence of obtained rewards and $S_0^R = 0$. At each time step $t \geq 1$, compute:

$$S_t^R = \max(0, S_{t-1}^R + \ln(\frac{R_1(s_t, a_t, r_t)}{R_0(s_t, a_t, r_t)})) \tag{2}$$

If S_t^R is greater than a threshold c^R, then a change of the reward function is detected.

3.3 Joint Detection

The two previous sums can be combined by computing at each time step $t \geq 1$:

$$S_t = \max(0, S_{t-1} + \ln(\frac{T_1(s_t, a_t, s_{t+1})R_1(s_t, a_t, r_t)}{T_0(s_t, a_t, s_{t+1})R_0(s_t, a_t, r_t)})) \tag{3}$$

with $S_0 = 0$. Sum S_t is to be compared with a threshold c to detect a change of mode.

Computing S_t^T and S_t^R separately can be advantageous in some situations as this makes it possible to detect a change in the transition function or in the reward function alone. Indeed, in some domains, the non-stationarity is only limited to one of the functions and/or they can evolve in a non synchronous manner. The advantage of using S_t^{TR} is that it may allow a faster detection of the environmental change because of the combined effects of the simultaneous change of the transition function and the reward function. For ease of exposition, we will focus in the remainder of the paper on the joint detection of changes in the reward and transition functions.

3.4 Detecting Changes with Multiples Models

In the case where there are many possible models $M_0, M_1, \ldots, M_k$ (with $k \geq 2$), all assumed to be known, the previous procedures can be adapted as follows. We assume that M_0 is the current model and when a change occurs at an unknown time step, the new model can be any of M_i with $1 \geq i \geq k$. Now, we need to compute k scores at each time step $t \geq 1$:

$$S_{i,t} = \max(0, S_{i,t-1} + \ln(\frac{T_i(s_t, a_t, s_{t+1})R_i(s_t, a_t, r_t)}{T_0(s_t, a_t, s_{t+1})R_0(s_t, a_t, r_t)})) \tag{4}$$

with $S_{i,0} = 0$ and $i = 1, \ldots, k$. Then an environmental change is detected if $\max_{i=1,\ldots,k} S_{i,t} \geq c$ and the current environment M_j is chosen to be $S_{j,t} = \max_{i=1,\ldots,k} S_{i,t}$.

3.5 Detecting Changes in practice

In practice, in the RL setting, the number of models and their specifications are generally unknown. In that case, we simply use in the CUSUM procedure the empirical estimates learned from the observed history in place of the unknown models $M_0, M_1, \ldots, M_k$. We always add among the estimated models, a "uniform" model where all transition and reward probabilities are uniform. This "uniform" model allows new models to be learned. The exact method is explained in details in Section 4.

Concerning the choice of the threshold c in the CUSUM procedure, one possibility is to use the heuristic proposed by [16] (although in a different simpler setting):

$$c = \ln \frac{1-\beta}{\alpha} \tag{5}$$

where β is the probability of a type II error and α is that of a type I error. Although this choice of the threshold value may not be optimal, this heuristics allows for some interpretation of the parameter. Besides, in our experiments, this choice seems to be reasonable and leads to good performance.

4 RL with Context Detection

Da Silva et al. developed the *Reinforcement Learning with Context Detection* algorithm (RLCD) to simultaneously learn and act in a non-stationnary environment. At each time step, a quality measure of each already learned model is calculated, depending on the last seen transition and reward. The model maximizing the measure is chosen as the next current model and is updated. However, if the maximum quality is below a given threshold, a new model is added to the list of known models, uniformly initialized and selected as the next current model. With this method, RLCD is able to tackle problems without the prior knowledge of the number of models to learn. Unfortunately, RLCD requires a set of parameters to be tuned accordingly to the problem. Moreover, this quality mesure seems to be ad-hoc and also depend on a hand-tuned threshold.

We propose an adaptation of RLCD, replacing the quality measure by the approach previously presented. Algorithm 1 presents our adaptation of RLCD using the detection on the transitions distributions. The solving part (given by $\pi_{m_{cur}}(s)$) and the learning part are exactly the same as in the original RLCD algorithm.

In this version of RLCD, we calculate S_m for each model and detect a change if this value is above c. Moreover, a new model is created if the model maximizing the value is the uniform model.

This detection method is not only more theoretically founded, but is also more efficient.

Algorithm 1 RLCD with Sequential Change-Point Detection

$m_{cur} \leftarrow newmodel()$
$M \leftarrow mcur$
$s \leftarrow s_0$, any starting state
loop
  Let a be the action indicated by $\pi_{m_{cur}}(s)$
  Observe next state s' and reward r
  for all $m \in M \setminus m_{cur}$ **do**
    $S_m \leftarrow \max(0, S_m + \ln \frac{T_m(s,a,s')}{T_{m_{cur}(s,a,s')}})$
  end for
  $m_{max} \leftarrow \arg\max_m S_m$
  if $S_{m_{max}} > c$ **then**
    $S_{uniform} \leftarrow \max(0, S_{uniform} + \ln \frac{1/|S|}{T_{m_{cur}}(s,a,s')})$
    if $S_{uniform} > c$ and $S_{uniform} > S_{m_{max}}$ **then**
      $m_{cur} \leftarrow newmodel()$
      $M \leftarrow m_{cur}$
    else
      $m_{cur} \leftarrow m_{max}$
    end if
  end if
  Update m_{cur} with original RLCD equations
end loop

5 Experimental Results

5.1 Ball Catching

The first problem is taken from [14]. In this environment, a cat has to catch a ball moving on a toroidal grid. The direction towards which the ball moves is given by the context of the environment. This problem has 15×15 states (the size of the grid), 5 actions (4 possible direction for the cat with a no move choice) and 4 contexts (one for each possible direction for the ball). The reward is set to -1 for each move and 10 when the cat catches the ball. We compare our method to the classic RLCD algorithm, using the same algorithm (Prioritized-Sweeping) to calculate the optimal policy for the currently learned policy. In that way, the differences in the results can only be explained by the efficiency of the context switching detection. As we said, the original RLCD algorithm needs some extra parameters to be set. We used those involving the best results we could find, which were equals to those given in [14]. Figure 1 shows the results obtained using the following protocol:

- a step is the minimum between the number of movements the cat needs to catch the ball and 100
- an episode is 100 steps
- we choose a starting context and run 5 episodes in this context and calculate the mean step for each episode

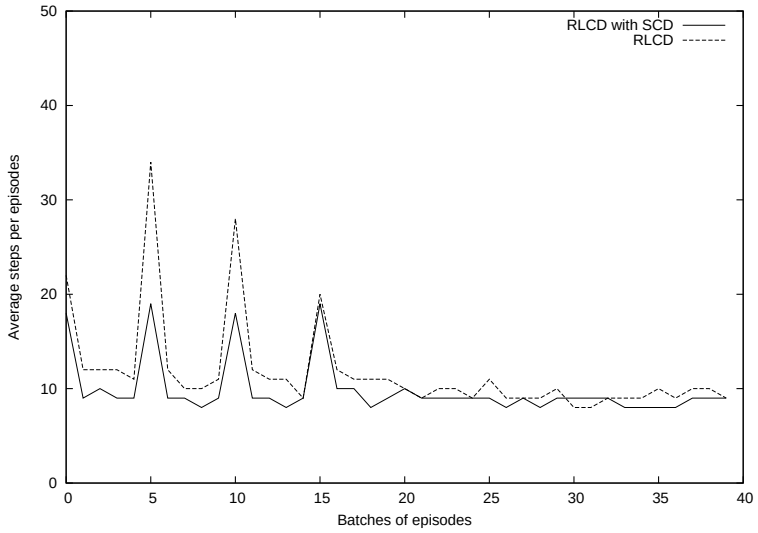

Fig. 1. Results for the ballcatching problem

– we switch the context for the next one and start 5 more episodes

The purpose of this protocol is to study the behaviour of the algorithms with switching to either a learned and an unknown context. Figure 1 shows that for each episode where the context is already known (the 4 last of each 5 episodes), RLCD and our adaptation perform equally. The difference is on the episodes where the context has been switched (the first of each 5 episodes). It takes less movements in mean to catch the ball, meaning the switching has been detected earlier.

5.2 Traffic

This problem is composed by 9 independent traffic lights (nodes) controlling the passage of cars one a 3×3 grid. The nodes on the edges of the grid are linked to 6 sources (3 at the north and 3 at the east) and 6 sinks (3 at the south and 3 at the west). Cars enter the grid by the sources and go in a straight line to the corresponding sink. Each traffic light can select a plan amongst 3, defining the amount of time it lets pass the cars at the north and the east: the first signal plan gives equal green times for both vertical and horizontal directions, the second signal plan gives priority to the vertical direction and the third signal plan gives priority to the horizontal direction. The environment can be in 3 different contexts conditioning the rate of arrival of the cars at each sources

(north or east): low insertion rates from north and east, high insertion rate from north sources and average insertion rate from east, high insertion rates from east sources and average insertion rate from north. The context defined by high insertion rates from both north and east sources is not considered since even an optimal policy does not prevent from saturating the network.

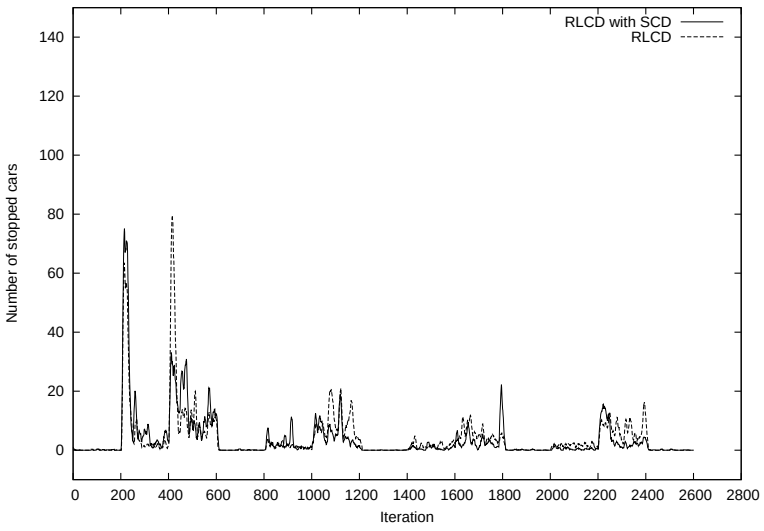

Fig. 2. Result for the traffic problem

Figure 2 presents the results for the traffic problem. It shows the number of stopped cars in the grid, depending on the iteration. The environment starts with a low rate of insertion and changes to the next context every 200 iterations. We can see that our method performs better than the original RLCD, especially when the environment is running an unknown model (e.g., iteration 400). This means that our adaptation is able to detect the change, create a new model and thus adapt the policy earlier, involving better results.

6 Conclusion

In this paper we presented a preliminary work using sequential change-point detection to detect context switching and learn the underlying model. We adapted the RLCD algorithm which is able to learn a model without knowing a priori

the number of contexts. We showed that this adaptation is able to enhance the detection ability of the original RLCD, thus involving better results.

We will extend this method in a future work in order to learn the transitions between the contexts in a stochastic setting. We will then be able to learn HM-MDPs without fixing the number of modes and thus bypass a requirement of the Baum-Welch adaptation proposed by Choi et al.

An other interesting application of this work is learning HS3MDPs, an extension of the HM-MDP where the transitions between the modes are semi-markovian [8]. It will requires to not only learn the transitions between the contexts but also the time the environment stays in each context.

References

1. Basseville, M., Nikiforov, I.V.: Detection of Abrupt Changes: Theory and Application. Prentice-Hall (1993)
2. Bellman, R.: Dynamic programming. Princeton university press, Princeton (1957)
3. Chadès, I., Carwardine, J., Martin, T.G., Nicol, S., Sabbadin, R., Buffet, O., et al.: MOMDPs: A solution for modelling adaptive management problems. In: AAAI (2012)
4. Choi, S.P.M., Yeung, D.Y., Zhang, N.L.: Hidden-mode markov decision processes for nonstationary sequential decision making. In: Sequence Learning - Paradigms, Algorithms, and Applications. Springer-Verlag (2001)
5. Choi, S., Yeung, D.Y., Zhang, N.: An environment model for nonstationary reinforcement learning. In: Advances in Neural Information Processing Systems. pp. 994–1000 (2000)
6. Doya, K., Samejima, K., ichi Katagiri, K., Kawato, M.: Multiple model-based reinforcement learning. Neural computation 14, 1347–1369 (2002)
7. Ghosh, B., Sen, P.: Handbook of Sequential Analysis. CRC Press (1991)
8. Hadoux, E., Beynier, A., Weng, P.: Solving hidden-semi-markov-mode markov decision problems. In: 8th Int. Conf. on Scalable Uncertainty Management (SUM) (2014)
9. Howard, R.: Dynamic Programming and Markov Processes. The M.I.T. Press (1960)
10. Lai, T.L.: Sequential analysis: Some classical problems and new challenges. Statistica Sinica 11, 303–408 (2001)
11. Narendra, K., Balakrishnan, J., Ciliz, M.: Adaptation and learning using multiple models, switching, and tuning. Control Systems, IEEE 15(3), 37–51 (1995)
12. Ong, S.C., Png, S.W., Hsu, D., Lee, W.S.: Planning under uncertainty for robotic tasks with mixed observability. The International Journal of Robotics Research 29(8), 1053–1068 (2010)
13. Puterman, M.: Markov decision processes: discrete stochastic dynamic programming. Wiley (1994)
14. da Silva, B.C., Basso, D.W., Bazzan, A.L., Engel, P.M.: Dealing with non-stationary environments using context detection. In: Proceedings of the 23rd International Conference on Machine Learning (2006)
15. Sutton, R., Barto, A.: Reinforcement learning: An introduction. MIT Press (1998)
16. Wald, A.: Sequential tests of statistical hypotheses. The Annals of Mathematical Statistics 16(2), 117–186 (1945)