Reinforcement Learning in Non-Stationary Continuous Time and Space Scenarios

Eduardo W. Basso[1], **Paulo M. Engel**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{ewbasso,engel}@inf.ufrgs.br

Abstract. *In this paper we propose a neural architecture for solving continuous time and space reinforcement learning problems in non-stationary environments. The method is based on a mechanism for creating, updating and selecting partial models of the environment. The partial models are incrementally estimated using linear approximation functions and are built according to the system's capability of making predictions regarding a given sequence of observations. We propose, formalize and show the efficiency of this method in the non-stationary pendulum task. We show that the neural architecture with context detection performs better than a model-based RL algorithm and that it performs almost as well as the optimum, that is, a hypothetical system with extended sensor capabilities in a way that the environment effectively appears to be stationary. Finally, we present known limitations of the method and future works.*

1. Introduction

When implementing learning algorithms, one often faces the difficult problem of dealing with environments whose dynamics might change due to some unknown or not directly perceivable cause. Non-stationary environments affect standard reinforcement learning (RL) methods in a way that forces them to continuously relearn the policy from scratch. In this paper we describe a method for complementing RL algorithms so that they perform well in a specific class of non-stationary environments.

It is important to emphasize that majority of RL approaches were designed to work in stationary environments. When dealing with non-stationary environments, they usually have to continually readapt themselves to the changing dynamics of the environment. This causes two problems: **1)** the time for relearning how to behave makes the performance drop during the readjustment phase; and **2)** the system, when learning a new optimal policy, forgets the old one, and consequently makes the relearning process necessary even for dynamics which have already been experienced.

The non-stationary environments in that we are interested in this paper consist on those whose behavior is given by one among several different stationary dynamics. Each type of dynamics can be called a *mode* of the environment dynamics [Choi et al. 2001], or simply a *context* [Silva et al.]. We assume that the context can only be estimated by observing the transitions and rewards, and that the maintenance of multiple models of the environment (and their respective policies) is a good solution to this learning problem.

Partial models have been used for the purpose of dealing with non-stationarity in other works, such as Multiple Model-based Reinforcement Learning (MMRL)

[Doya et al. 2002] and Reinforcement Learning with Context Detection (RL-CD) [Silva et al.]. MMRL is able to deal with continuous time and space scenarios, but requires a fixed number of models. It implicitly assumes that the approximate number of different environment dynamics is known *a priori*, which is not always realistic. RL-CD assumes the number of contexts is unknown and creates new modules on demand, but is designed to deal with finite states and actions.

In order to overcome those restrictions, we designed a neural architecture which incrementally builds new modules. Our main hypothesis is similar to RL-CD's, in which the use of multiple partial models makes the learning system capable of partitioning the knowledge into models, each of which automatically assuming for itself the responsibility for "understanding" one kind of environment behavior. Based on this hypothesis, we propose, formalize and show the efficiency of our neural architecture for reinforcement learning, which performs well in non-stationary continuous time and space environments by continuously evaluating prediction errors generated by each partial model.

This paper is organized as follows. In section 2 we present some concepts about reinforcement learning in continuous time and space. In section 3 we discuss how to measure the relative quality of each partial model and how to use it to detect context changes. In section 4 we present a validation scenario which empirically shows that our method performs similarly to a hypothetical system which knows the context of the environment. Some concluding remarks, known limitations and future work are discussed in section 5.

2. RL in Continuous Time and Space

Reinforcement learning was built over Markov decision processes, and usually the states of the process are represented as a set of disconnected symbols. However, when dealing with real-life situated scenarios, such as robotics, one usually has to model the problem considering that the world states are not enumerable, and that actions and time must be measured continuously.

One of the most difficult problems in dealing with continuous space scenarios is that the representation of the value function can no longer be made via tabular methods. Thus, the state values must be approximated by a function. In order to avoid known convergence problems, linear approximation must be used [Santamaría et al. 1997].

The Actor-Critic is a model-free algorithm which deals with the continuous input space by transforming it into a set of separate states through *boxes*. Those boxes are arbitrarily placed in the input space and perform the binary activation of the states. The continuous version of Actor-Critic [Doya 1996] uses a *Normalized Gaussian Network* (NGN) to extract continuous features of the input space with fuzzy activation. The gaussians are uniformly distributed in the input space with fixed parameter, which guarantees linear training of the free parameters. Doya states that less continuous features than binary states are necessary to solve a task.

The reinforcement learning with *Value-Gradient based Policy* [Doya 2000] is a model-based algorithm which uses the gradient of the value function in respect to the control signals as an estimate of the greedy action. It is composed by a forward model and a critic. The forward model represents the dynamics of the environment, and the

critic estimates the state values. Both use Normalized Gaussian Networks to deal with the respective continuous input spaces.

Actor-Critic and Value-Gradient methods deal properly with continuous time and space scenarios. However, since none has been built regarding the non-stationarity of the environment, both suffer with decrease of performance due changes on the environment's dynamics.

3. Context Detection

First of all, we assume that the system contains several modules, and each module has one partial forward model of the environment dynamics and one local controller. Each partial model is specialized in a different environment dynamics and the respective controller is locally optimal. Only one module is active at a time. The instantaneous *quality* of a module is a value inversely proportional to the prediction error of its model. The *quality trace* integrates the instantaneous quality over the time and, at each moment, the module with the highest quality trace is chosen as the current active model. If the quality traces of all modules are worse than the minimum allowed, the system assumes that the environment is in a new context. Thus, a new module is created from scratch, which will learn both a dynamics partial model and the corresponding locally optimal policy to the new context.

The class of non-stationary environments that we are interested in is similar to the one studied by Hidden-Mode MDPs researchers [Choi et al. 2001], except that, in our case, time and space are continuous. We assume that the following properties hold: **1)** environmental changes are confined to a small number of contexts, which are stationary environments with distinct dynamics; **2)** the current context cannot be directly observed, but can be estimated according to the types of transitions and rewards observed; **3)** environmental context changes are independent of the agent's actions; and **4)** context changes are relatively infrequent. These assumptions are considered plausible for a broad number of real applications [Choi et al. 2001], but in case they are not met scalability problems should be carefully considered and studied.

3.1. Modules of the architecture

The modules of the proposed neural architecture are implemented as independent mechanisms of Reinforcement Learning with Value-Gradient based Policy [**?**]. Since the modules are independent, at an instant, only the active module contributes to control and only that module is adjusted. Each module m has two main approximation functions: the partial model $f_m(\cdot)$, and the local critic $V_m(\cdot)$.

The partial model $f_m(\cdot)$ of the environment dynamics estimates the change on state variables $\hat{\dot{\mathbf{x}}}_m(t)$ by the equation 1, where $\hat{\dot{x}}_{mi}$ is the prediction made by model m regarding the expected change on state variable x_i given the current state $\mathbf{x}(t)$ and action $\mathbf{u}(t)$; $b^f(\cdot)$ are the normalized gaussian functions uniformly distributed to cover the state-action space; B^f is the number of gaussian functions; and $\mathbf{w}_m^f$ are the free parameters of the model, which weight the activation of the state-action features.

$$\hat{\dot{x}}_{mi}(t) = \sum_{j=1}^{B^f} w_{mij}^f \, b_j^f \left(\mathbf{x}(t), \mathbf{u}(t)\right) \tag{1}$$

The estimation of the state value is approximated by the critic $V_m(\cdot)$ according to equation 2, where v_m is the estimation of model m for the value of state $\mathbf{x}(t)$ at the instant t, given by the function $V_m(\cdot)$ and its free parameters $\mathbf{w}_m^V$; $b^V(\cdot)$ are the normalized gaussian functions uniformly distributed to cover the state space; B^V is the number of gaussian functions for the critic; and $\mathbf{w}_m^V$ are the free parameters of the critic, which weight the activation of the state features.

$$v_m(t) = \sum_{j=1}^{B^V} w_{mj}^V \, b_j^V \left(\mathbf{x}(t)\right) \tag{2}$$

Notice that the gaussian functions $b^f(\cdot)$, used by the models, extract features of the state-action input space, and the gaussian functions $b^V(\cdot)$, used by the critic, extract features of the state input space.

Given the model of the dynamics and the value function, the action $\mathbf{u}_m(t)$ to be taken at instant t is given by the gradient of the value function with respect to an action $\mathbf{u}_0$, defined as the joint action that assigns zero to all agent's actuators. The action is computed as follows:

$$\mathbf{u}_m(t) = \tanh\left(\frac{1}{c}\frac{\partial V_m\left(\mathbf{x}(t), \mathbf{w}_m^V\right)}{\partial \mathbf{x}(t)}\frac{\partial f_m\left(\mathbf{x}(t), \mathbf{u}_0, \mathbf{w}_m^f\right)}{\partial \mathbf{u}_0} + \epsilon \mathbf{n}(t)\right) \tag{3}$$

where $\mathbf{n}$ is a vector with random values in $[-1, 1]$, and ϵ is the exploration coefficient which, if zero, makes the action $u_m(t)$ greedy. Finally, c is a gain coefficient.

The active partial model is adjusted by back-propagating the prediction error, according to equation 4, where $\eta_m^f(t)$ is the learning rate for that instant. The model's learning rate is computed by $\eta_m^f(t) = (1 - \tau_m(t))\,\eta^{f_0}$, based on the local stability of the model $\tau_m(t)$, which is explained in section 3.3.

$$\dot{\mathbf{w}}_m^f(t) = \eta_m^f(t)\frac{\partial \hat{\dot{\mathbf{x}}}_m(t)}{\partial \mathbf{w}_m^f(t)}^T \left(\dot{\mathbf{x}}(t) - \hat{\dot{\mathbf{x}}}_m(t)\right) \tag{4}$$

The active critic is adjusted by temporal difference with exponential eligibility trace. The temporal difference error is computed by $\delta(t) = r(t) + T^r \dot{v}(t) - v(t)$, where $r(t)$ is the current reward and T^r is the discount step of rewards. Free parameters of the critic are adjusted by equation 5, where η^V is the learning rate and e_{mi} is the eligibility for the weight w_{mi}^V. Finally, the eligibility values are updated according to equation 6, where T^e is the discount step of eligibility. For further details on the value gradient method, as well as on adjusting the forward model and the critic, refer to [Doya 2000].

$$\dot{w}_{mi}^V = -\eta^V \delta(t) e_{mi}(t) \tag{5}$$

$$\dot{e}_{mi} = \frac{1}{T^e}\left(\frac{\partial v(t)}{\partial w_{mi}} - e_{mi}(t)\right) \tag{6}$$

3.2. Detecting the context

In order to detect context changes, the system must be able to evaluate how well all partial models can predict the environment. For each model m, the instantaneous quality is computed under the assumption that the prediction error of state changes are approximately normal. In equation 7, σ_m is the mean square error of the state prediction, and τ_m indicates the local stability for model m, which prevents a new model from being ignored just for being incomplete. These are explained further in subsection 3.3.

$$\lambda_m(t) = \exp\left(-\frac{1}{2}\left(\sum_i \frac{\dot{x}_i(t) - \hat{\dot{x}}_{mi}(t)}{\sigma_{mi}(t)}\right)^{2\tau_m(t)}\right) \tag{7}$$

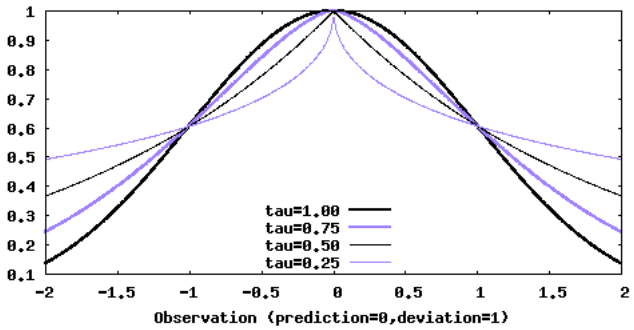

Figure 1. Effect of the stability on the instantaneous quality function.

Considering the instantaneous quality λ_m of all models, we can compute the probability of each model being correct given recent observations. This value is called quality trace of the model $\bar{\lambda}_m$, and summarizes the temporal behavior of the instantaneous qualities considering other models. Since λ_m is proportional to the probability of a model being correct, we can use this value to activate the best available model.

The problem of using only the relative probabilities is that we do not have a global indication of how good a model is independently of other models. In other words, even if all models are bad, one of them would be activated. In order to solve this problem we define a constant instantaneous quality λ_0 which represents a critical value for the hypothesis that no model is correct. When all models have a quality trace lower than $\bar{\lambda}_0$, then we have an indication that no model is sufficiently good and it is necessary to create a new model to well-represent the non-stationary environment.

The quality trace of each partial model m is computed by equation 8, where α is a parameter that controls the memory of the trace, that is, past prediction quality will not be considered if $\alpha = 0$, and $\bar{\lambda}$ will be a complete temporal quality trace if $\alpha = 1$. Since we are dealing with non-stationary environments, considering all past prediction errors might cause a delay on detecting context changes. In the other hand, since two context may overlap, considering no past predictions at all might be bad choice.

$$\bar{\lambda}_m(t) = \frac{\lambda_m(t)\bar{\lambda}_m(t-\Delta t)^\alpha}{\sum_j \lambda_j(t)\bar{\lambda}_j(t-\Delta t)^\alpha} \tag{8}$$

The overall algorithm for the context detection is given by algorithm 1. After the observation of state change $\dot{x}$ and the reward r, the instantaneous quality is computed for all models and all quality traces are updated, including $\bar{\lambda}_0$. Afterwards, we activate the model with highest quality trace. If the highest quality trace found is the trace of the critic quality, then a new model is created. After the model selection/creation is performed, the current model and critic are adjusted.

Algorithm 1 Context detection algorithm

$m_{cur} \leftarrow newModule()$
$\mathcal{M} \leftarrow \{m_{cur}\}$
$\bar{\lambda}_k(t) \leftarrow \frac{1}{|\mathcal{M}|+1}, \forall k \in [0, |\mathcal{M}|]$
loop
  Observe state $\mathbf{x}(t)$
  Perform action $\mathbf{u}_m(t)$
  Observe effect $\dot{\mathbf{x}}(t)$ and reward $r(t)$
  Compute $\lambda_k(t) \forall k \in [1, |\mathcal{M}|]$ by equation 7
  Update $\bar{\lambda}_k(t) \forall k \in [0, |\mathcal{M}|]$ by equation 8
  $m_{cur} \leftarrow \arg\max_k \left(\bar{\lambda}(t)_k\right)$
  if $\bar{\lambda}_{cur}(t) = \bar{\lambda}_0(t)$ **then**
    $m_{cur} \leftarrow newModule()$
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{m_{cur}\}$
    $\bar{\lambda}_k(t) \leftarrow \frac{1}{|\mathcal{M}+1|}, \forall k \in [0, |\mathcal{M}|$
  end if
  $adjust(m_{cur}, \dot{\mathbf{x}}(t), r(t))$
end loop

3.3. Local stability and deviation

In order to compute the prediction quality of models, it was necessary to extend the value-gradient mechanism by adding two new neural structures to each module. These structures are responsible for estimating local experience $\chi_m(\cdot)$ and local deviation $\sigma_m(\cdot)$. Notice that the model's normalized gaussian functions $b^f(\cdot)$ are shared by $\chi_m(\cdot)$ and $\sigma_m(\cdot)$.

The evaluation of the instantaneous quality assumes that the state changes follow an approximately normal multivariate distribution with mean at $\hat{\dot{\mathbf{x}}}_m$ and standard deviation given by σ_m. However, when the model is incomplete or new, deviations from the expected state are not so relevant since the model is unstable. For these reasons, we use a stability term τ that prevents the algorithm from creating new models when bad predictions come from incomplete models. If $\tau_m = 1$, which occurs when the model m has been updated with sufficient experience samples, then λ_m becomes gaussian-shaped in respect to $\dot{\mathbf{x}}(t)$.

Since the model can be incomplete in some regions of the input space $\langle \mathbf{x}, \mathbf{u} \rangle$, we say that the stability $\tau_m(t)$ is *local*, and is computed according to the equation 9, where $\chi_m(t)$ is the local experience of the model, and T^τ is a fixed parameter which represents the amount of local experience necessary to consider that the model is locally stable.

$$\tau_m(t) = 1 - \frac{1}{1 + 2\frac{\chi_m(t)}{T^\tau}} \tag{9}$$

The stability is monotonic in respect to local experience $\chi_m(\cdot)$, which is the integration of all past feature activations for the same state-action pair. The local experience $\chi(t)$ is computed by the equation 10, where w^χ_{mi} is the accumulator of experience associated to model m and gaussian function i. Accumulators are updated by equation 11.

$$\chi_m(t) = \sum_{i=1}^{B^f} b_i^f\left(\mathbf{x}(t), \mathbf{u}(t)\right)\, w^\chi_{mi} \tag{10}$$

$$\dot{w}^\chi_{mi} = b_i^f\left(\mathbf{x}(t), \mathbf{u}(t)\right) \tag{11}$$

The local deviation of the model m at state-action $(\mathbf{x}(t), \mathbf{u}(t))$ for each observation signal i is estimated by equation 12, where w^σ_{mij} represents the average quadratic error of prediction variable $\dot{x}_j$ associated with gaussian function b_i^f. Free parameters are updated by equation 13, which implements a the moving average of the quadratic prediction error weighted by the respective gaussian function activation.

$$\sigma_{mj}(t) = \sum_{i=1}^{B^f} \sqrt{w^\sigma_{mij}}\, b_i^f\left(\mathbf{x}(t), \mathbf{u}(t)\right) \tag{12}$$

$$\dot{w}^\sigma_{mij} = \left(\left(\dot{x}_j(t) - \hat{\dot{x}}_{mj}(t)\right)^2 - w^\sigma_{mij}\right) \frac{b_i^f\left(\mathbf{x}(t), \mathbf{u}(t)\right)}{\int_0^t b_i^f\left(\mathbf{x}(s), \mathbf{u}(s)\right)} \tag{13}$$

4. Empirical Results

In order to evaluate the performance of our algorithm, we test it in a non-stationary scenario with continuous time and space. The validation scenario consists in a RL agent trying to swing up a pendulum exposed to wind. The direction of the wind changes along the time, pushing the pendulum either to the right or to the left, but it keeps blowing in a specific direction during many episodes. Since our agent cannot directly perceive the direction of the wind, it *perceives* the environment as being non-stationary. We compare our method with the single model-based algorithm with Value Gradient Policy [Doya et al. 2002], and with a full-aware multiple-model based algorithm which receives the context information from an oracle. The oracle approach can be thought of as an algorithm with extended sensors capable of perceiving the wind direction, which makes the environment look stationary.

The effect of the wind on the pendulum is illustrated in figure 2 and its dynamics is given by equation 14, where the constants μ, m, l and g are the friction, mass, length and gravity coefficients, respectively, θ is the pendulum angle, u is the torque applied to the pendulum, and W is the wind force. The task is difficult if the maximum torque is lower than then the maximum load ml^2.

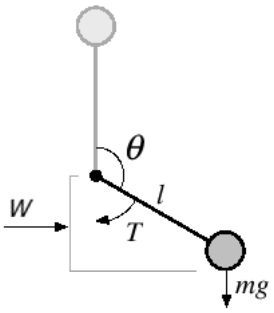

Figure 2. Non-stationary pendulum scenario.

$$\ddot{\theta} = \frac{-\mu\dot{\theta} + mglsin(\theta) + u + Wcos(\theta)}{ml^2} \tag{14}$$

In our experiment, we measured average reward received by episode. An episode takes up to 20 seconds and starts with the pendulum in its rest position π. The episode ends with success whenever the agent successfully balances the pendulum a position in which its is higher than $\cos\left(\frac{\pi}{4}\right)$ for at least 10 seconds. Otherwise, the agent fails when the pendulum is over rotated (out of $[-\pi,3\pi]$). The reward function is given by the pendulum height $r(t) = \cos(\theta(t))$ at each time step, and $r(t) = -1$ when the agent fails.

In figure 3 we present the results of our performance comparison. We explicitly change the wind direction each 100 episodes. Since the modules of the context detection and the oracle approaches are the same, the difference of performance depends on the model activation. If the context signal is perfectly detected, our approach will perform as well as the oracle.

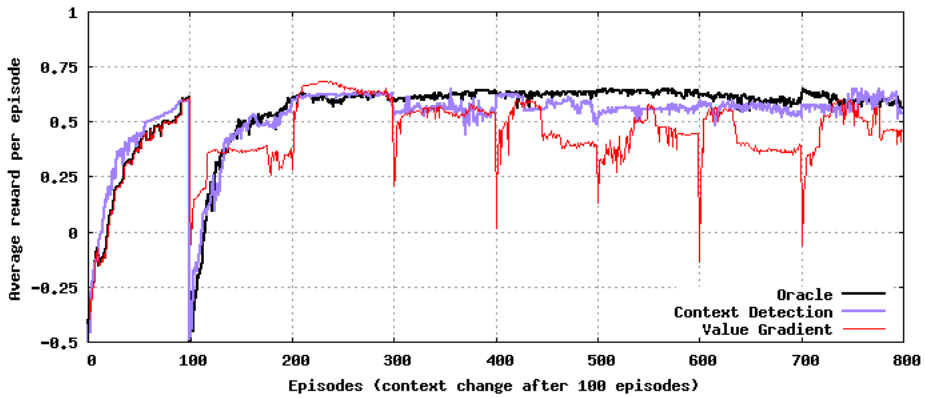

Figure 3. Performance for Value Gradient, Oracle and Context Detection.

The critic's NGN was designed with 9x9 gaussian functions uniformly distributed to cover the space $[-\pi,\pi][-\frac{5\pi}{2},\frac{5\pi}{2}]$, while the model's NGN was designed with 9x9x2 gaussian functions to cover the space $[-\pi,\pi]\,[-\frac{5\pi}{2},\frac{5\pi}{2}]\,[-u^{max},u^{max}]$. Parameters related to the independent modules were $T^r = 1, T^e = 0.1, \eta^V = 5, \epsilon = 0.5, T^n = 1$ and $c = 0.1$. Parameters related to learning the partial models were $\eta_0^f = 20$ and $T^\tau = 100$. Parameters related to module activation were $\alpha = 0.9$ and $\lambda_0 = 1\%$.

Since the context detection approach starts with only one model, and the oracle activates only one model until the first context change, both perform the same as the single model based value gradient algorithm in the beginning. At episode 100, wind stops blowing to left and starts blowing to right. After that first context change, the oracle activates a second model and the context detection approach creates a new one. At that time, the single model mechanism takes advantage of past knowledge to deal with overlapped regions of both contexts, while the oracle and the context detection methods need to learn the new context from scratch. However, around episode 150, both methods with new models learned to solve the task, while the single model mechanism is *confused* by former information where contexts are different.

At following context changes, the context detection architecture performs almost the same as the oracle approach, which indicates success in context detection. The single model mechanism, however, presents several losses on performance after context changes, which demonstrate that some context detection is necessary for that non-stationary task.

Figure 4 shows the instantaneous quality and the quality trace of each model of the context detection mechanism around the first context change, step by step. Notice that only *model 1* exists at the beginning and its quality trace becomes worse than the trace of critical value a few steps after the context change, although its instantaneous quality drops immediately.

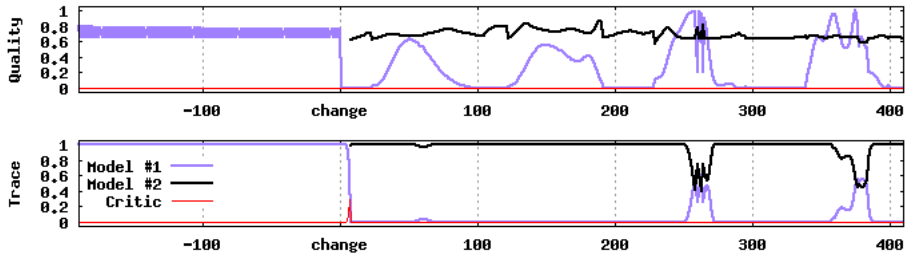

Figure 4. Instantaneous quality and quality trace around the first context change.

The second context change is shown in figure 5, where it is possible to see that, as the context turns back to the first one, *model 1* is activated after few observation steps.

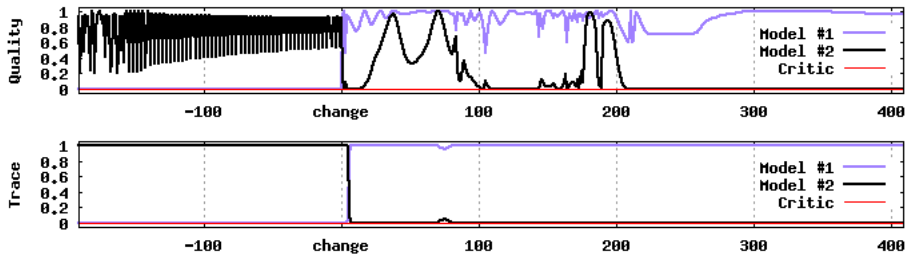

Figure 5. Instantaneous quality and trace around the second context change.

5. Conclusions and Future Work

In this paper we have formalized a neural architecture for solving reinforcement learning problems in non-stationary continuous time and state environments through context

detection. Our method was validated in a stochastic and noisy scenario. We have empirically shown that it has advantages over the single model-based approach and that its performance approaches the optimum (ie, the performance achieved with an oracle).

Since we do not assume that the number of models necessary to well-describe the environment is given a priori, some parameters are necessary to enable the system to deal with the bias-variance tradeoff. Our algorithm requires that two main parameters be adjusted: the critical quality λ_0, which represents the minimum instantaneous quality that will be accepted; and T^τ, which is related to the time necessary for the model to gain confidence regarding its predictions in a region of the input space. It is also necessary to adjust α, which specifies the time window considered in the calculation of the trace of quality. Although $\alpha = 0.9$ was shown as a good choice in all experiments, problems whose state transition dynamics change very abruptly may require lower values of α.

The neural architecture might have problems to deal with environments that change contexts too fast and have high deviation of observation. In that situations, the model would not experience sufficiently a context before creating a new one. We think it might be possible to overcome this limitation by, instead of creating models for the whole input space, to create sub-models for partial areas of the input space.

We feel that it is still necessary to adjust the method so that it can perceive non-stationarity in the reward function, and not only in the transition function. This way, we would be able to deal not only with scenarios with multiple dynamics, but effectively multiple tasks. Moreover, we think it is possible to relate the estimated current context signal (that is, the indication of the current model) to construct a hierarchy in the sense of creating a more abstract state space.

References

Choi, S. P. M., Yan-Yeung, D., and Zhang, N. L. (2001). Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning - Paradigms, Algorithms, and Applications*, pages 264–287, London, UK. Springer-Verlag.

Doya, K. (1996). Temporal difference learning in continuous time and space. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Proceedings of the conference on Advances in Neural Information Processing Systems*, volume 8, pages 1073–1079. The MIT Press.

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation*, 12(1):219–245.

Doya, K., Samejima, K., Katagiri, K.-I., and Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural Computation*, 14(6):1347–1369.

Santamaría, J. C., Sutton, R. S., and Ram, A. (1997). Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, 6(2):163–217.

Silva, B. C., Basso, E. W., Bazzan, A. L., and Engel, P. M. Dealing with non-stationary environments using context detection. In *23th International Conference on Machine Learning - (ICML 2006)*, pages 217–224.