Anticipating Rewards in Continuous Time and Space with Echo State Networks and Actor-Critic Design

Mohamed Oubbati[1] and Markus Kächele[1] and Petia Koprinkova-Hristova[2] Günther Palm[1]

1- Institute of Neural Information Processing, University of Ulm. Germany.

2- Institute of Control and System Research, Bulgarian Academy of Sciences. Bulgaria.

Abstract. In this paper we implement an echo state network within the concept of actor-critic design to obtain optimal control policy for a mobile robot. The robot is asked to anticipate future rewards/punishments and react accordingly. Experimental results show that the proposed approach is simple and effective.

1   Introduction

Dynamic programming is one of the most general approaches to obtain optimal control policies for autonomous systems. However, due to the so-called 'curse of dimensionality' [1], this approach can only be applied to simple and small-scale problems. Over the years, many efforts have been done to circumvent this problem by building a system, called 'critic', to approximate the cost function in dynamic programming [2, 3]. The idea is to use function approximation structures such as neural networks to approximate value functions (i.e. mapping state-action pairs to a value estimate) and therefore addressing large-scale problems. Once the critic network is converged, the control policy improvement step is carried out to adapt the parameters of the controller. However, implementing this framework in real applications needs fast online training algorithms as well as robust modelisation of the robot-environment interaction. Recurrent neural networks (RNNs) are powerful tools to learn such systems, for two main reasons. First, they are universal approximators of dynamical systems [4]. Second, they can exhibit continuous dynamics; a suitable property to model robot-environment interaction. To overcome the training difficulties of RNNs, concept of "reservoir computing" such as echo state networks (ESN) [5] has been proposed. The core idea of reservoir computing consists to use a large RNN as a "pool" of excitable complex dynamics, from which readout neurons can learn to extract the current state. This reduces the complexity of training to simple linear regression while preserving the recurrent property of the network. In this paper we implement the ESN as critic network within the concept of adaptive critic design (ACD) [3] to obtain optimal control policy for a real mobile robot interacting with its environment. The ESN critic has to consider continuous state/action space, deal with uncertainties, and be computationally cheap in order to deal with real-world constraints. Recently, we implemented ESNs within the framework of Actor-Critic reinforcement learning to generate behavior of obstacle avoidance for a real robot [6]. In this work, we train an ESN-critic to estimate the long-term cost by robot's sensations associated with future higher rewards. Anticipating future negative rewards enables the robot to react earlier in order to maximize future rewards by trying other movement directions.

2   Actor-Critic Design with ESN

We assume that the world (robot-environment) consists of states S and actions A, with the mapping between states from which actions being defined by a probabilistic transition function $F(\acute{s}|s,a) : S \times A \times S \to [0,1]$. Suppose that one associates with this system the performance index (cost-to-go) in the Bellman equation of dynamic programming

$$J(k) = \sum_{i=0}^{\infty} \gamma^i U(k+i) \tag{1}$$

where U is called the utility function and γ is the discount factor with $0 < \gamma \leq 1$. In order to predict future rewards the ESN critic is asked to estimate J by minimizing the following error measure over time

$$\|E\| = \sum_k E_k = \sum_k [\hat{J}(k) - U(k) - \gamma\hat{J}(k+1)]^2 \tag{2}$$

where $\hat{J}(k) = \hat{J}[s(k), a(k), k, \theta_c]$ and θ_c represents the parameters of the ESN critic. An ESN (Fig. 1. a) is formed by a so-called "dynamic reservoir", which contains a large number of sparsely interconnected neurons with non-trainable weights. The activation of internal neurons is updated according to

$$X(k) = f(W_{in}V(k) + WX(k-1)) \tag{3}$$

$$\hat{J}(k) = f^{out}(W^{out}X(k)) \tag{4}$$

where $f = tanh()$, and f^{out} is chosen as a linear activation function (identity). The input W_{in} and internal W weight matrices are generated randomly. The input vector V is a concatenation of S and A, and X represents the state vector of internal neurons. An essential condition for successful using of ESN is the "echo state" property, where the internal state is required to be an "echo" of its history. In practice it was found that when the spectral radius of W is $|\lambda_{max}| < 1$, we do have an echo state network [1]. If this condition is met, only weights connections from internal neurons to the output (W^{out}) are to be trained. This could be done by any suitable off-/on line training methods, such as (recursive) least squares [7]. When the overall cost (expressed as a sum of all $U(k)$ in the near future) is estimated, the objective is then to choose a control sequence $a(k)$ ($k = 1, 2, ...$), so that J is minimized (Fig. 1. b). The control action can be computed as follows

$$a(k+1) = a(k) - \delta\frac{\partial\hat{J}(k)}{\partial a(k)} \tag{5}$$

where δ is a learning rate. The gradient of $\hat{J}$ with respect to $a(k)$ can be computed using the chaine rules

$$\frac{\partial\hat{J}(k)}{\partial a(k)} = \frac{\partial\hat{J}(k)}{\partial X(k)}\frac{\partial X(k)}{\partial a(k)} \tag{6}$$

---

[1]During the remainder of this paper the spectral radius will be dsigned by the parameter α.
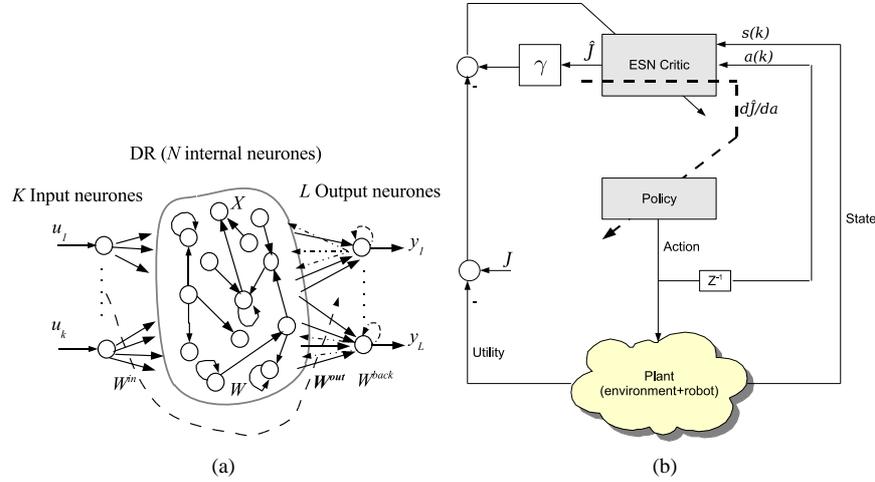
Fig. 1: **Actor-Critic Design with ESN**. (a) Basic architecture of ESN: Dotted arrows indicate connections that are possible but not required. (b) ESN as a critic network for ACD design.

where

$$\frac{\partial \hat{J}(k)}{\partial X(k)} = W^{out} \tag{7}$$

Assuming that $W_{in} = [W_a W_s]$ concatenation of action and state input weights we obtain

$$\frac{\partial X(k)}{\partial a(k)} = (I - X^2(k))W_a^T \tag{8}$$

Hence

$$\frac{\partial \hat{J}(k)}{\partial a(k)} = W^{out}(I - X^2(k))W_a^T \tag{9}$$

3   Implementation

We consider the scenario presented in Fig. 2. The mobile robot has a differential-drive, and its geometric configuration is described by $q = [x, y, \phi]^T$ where (x, y) are its coordinates, and ϕ its heading angle. It is equiped with 8 infrared (IR) proximity sensors and two encoders to provide q at any time. We consider a continuous state space $S(k) = \{x(k), y(k), d_i(k) | i = 1, 2, \ldots, 8\}$ (odometrie and distances to obstacles), and continuous action space in a form of desired speeds for the right and left wheels $a(k) = \{v_r, v_l\}$. The reinforcement signal varies continuously according to the variations of the sensory inputs as follows

$$r_i(k) = \begin{cases} d_i(k) - \lambda_{th} & \text{if } d_i(k) \leq \lambda_{th} \\ 0 & \text{if } d_i(k) > \lambda_{th} \end{cases} \tag{10}$$
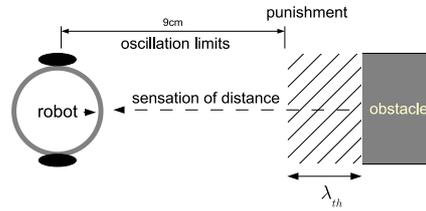
Fig. 2: **Scenario**: The robot is moved periodically forward and backward at the front of the obstacle, and learning system must be able to predict future negative rewards before reaching the distance threshold λ_{th}.

where λ_{th} defines the threshold, above which the given sensor's reading can be considered as approaching obstacle (producing of negative rewards). The utility function is chosen as

$$U(k) = \sum_{i=1}^{8} r_i(k) \tag{11}$$

The first experiment tests the learning performance of ESN-critic in predicting future rewards before reaching the distance threshold λ_{th}. We started the experiment with an untrained ESN-critic, and we initialized the discount factor $\gamma = 0$. The robot is moved periodically forward and backward at the front of the obstacle, and after each period (episode) we increased γ with a step 0.05 in order to expand the time horizon and thus enabling the ESN critic to learn sensations associated with future negative rewards. Fig. 3 shows the anticipated rewards vs. sensory input, where we can observe how reward anticipation evolve with number of episodes.

In the second experiment the robot is asked to generate "well timed" actions in order to avoid future negative rewards and to maximize the reward in other directions of movement. To learn this behaviour we let the robot moving in the direction of the obstacle in order to acquire a negative reward, and react according to the control policy (5). After repeating this process several times, the robot showed the desired behavior by changing the movement direction earlier before its IR-sensors detected the obstacle (Fig. 4. a). In both experiments we set the ESN spectral radius $\alpha = 0.6$, and the number of neurons in the reservoir $N = 6$. The input and the internal synaptic connections weights were randomly initialized from a uniform distribution over $[-1, +1]$, and the internal weight matrix W has a sparse connectivity of 20%.

4   Conclusion

We trained ESNs within the frame of adaptive critic design to adress reinforcement learning problems with continuous time and space. In a partially observable environment the ESN-critic was asked to estimate the long-term cost by robot's sensations associated with future higher rewards. The first results have shown that a small ESN (6 internal neurons) could generalize successfully in presence of new environmental perceptions (but similar to those seen during training) and anticipate the expected negative
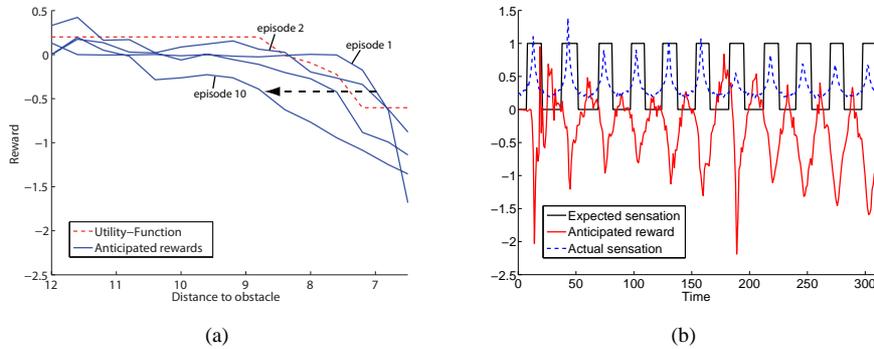
Fig. 3: **Anticipated rewards vs. sensory inputs**. (a) At each time step ESN-critic is trained with new data (arround 30 input/ouput data per one episode) in order to memorize sensations associated with future negative rewards. After some episodes the ESN-critic began to anticipate negative rewards according to the actual state. (b) The expected and actual robot's sensation are compared with the anticipated reward over time. With time, we can see how the negative reward is produced before crossing the distance threshold λ_{th} which is represented by the peaks of the actual sensation.
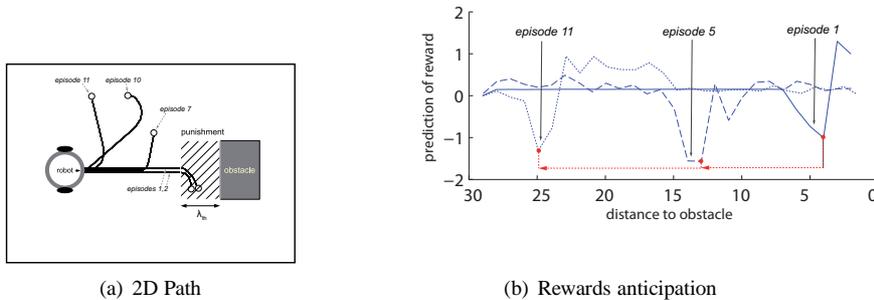

Fig. 4: **Reward and Action Anticipation**: In this experiment the robot should anticipate future rewards and react accordingly. During first episodes negative rewards with action are generated when reaching the distance threshold λ_{th}. After many episodes, the timing of anticipating the negative reward (negative peak) moved slowly towards the left (b) enabling to choose desired actions earlier in order to maximize the reward in other directions (a).

rewards. It was also shown how the robot uses reward anticipation to react earlier in order to maximize future rewards by trying other movement directions.

During training it was not easy to find optimal values for the learning parameters. With a "relatively" large reservoir dimension ($N > 15$ neurons) the ESN-critic lost stability at many times. The same observation was made when using a small $\alpha \in [0.1, 0.5]$.

Our next step is to study the effect of these parameters on the overall learning performance. We will also consider time/space varying reward signals, so that the robot must be able to adapt its behavior when the environment changes its topology.

References

[1] Stuart E. Dreyfus and Averill M. Law. *Art and Theory of Dynamic Programming*. Academic Press, Inc., Orlando, FL, USA, 1977.

[2] A. G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. pages 81–93, 1990.

[3] D. Prokhorov and D. Wunsch. Adaptive critic designs. *IEEE Transactions on Neural Networks*, 8:997–1007, 1997.

[4] K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Network*, 6(6):801–806, 1993.

[5] Jaeger H. The 'echo state' approach to analysing and training recurrent neural networks. Technical Report 148, AIS Fraunhofer, St. Augustin, Germany, 2001.

[6] H. P. Koprinkova, M. Oubbati, and G. Palm. Adaptive critic design with echo state network. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 1010–1015, 2010.

[7] H. Jaeger. Tutorial on training RNNs, covering BPPT, RTRL, EKF and the echo state network approach. Technical Report 159, AIS Fraunhofer, Germany, 2002.