Application of Reinforcement Learning to Control System Design

Mahdieh Shadi[1], S. D. Katebi[2]
Department of Computer Science
School of Engineering
Shiraz University
shadi@cse.shirazu.ac.ir

Abstract

This paper is concerned with the development of an online Reinforcement Learning (RL) technique that significantly improves the control systems behavior. The reinforcement learner is based on Q-learning and the final controller is an artificial neural network whose weights are tuned by on line learning. In order to speed up the learning processes and prevent the plant from the instability, initially a PID is utilized as an augmented controller until the reinforcement learning becomes capable of keep the system stable and prevent the system from undesirable behavior. Example of use is presented and the effectiveness of the proposed approach is shown.

1- Introduction

Most control design techniques require a very accurate mathematical model of the plant, which is not available in many real problems. Reinforcement learning (RL) algorithms, on the other hand, are able to explore and learn to improve control performance without the knowledge of the analytical system model.

It is well known that RL agents can learn optimal behaviors in an unknown environment through online, trial and error exploration and exploitation, when certain conditions are satisfied [1]. Therefore, using reinforcement learning techniques as part of a controller could be an approach for regaining the loss of performance that may be caused by many factors such as disturbances and excessive noise.

One approach is to replace the traditional controller completely with a reinforcement learner. Kavehercy [2] applied this approach to a water vessel process control problem. Another approach is to use RL to tune the parameters of a traditional controller aiming at the best performance [3]. In a paper by Anderson [4] new reinforcement learning

1-M.Sc. Student
2- Professor

architectures were proposed and studied. Development in this paper follows this latest report. Firstly, in our work all state variables are used by the RL, providing more degree of freedom, while in the above mentioned report only the output or tracking error are used. Secondly, previous researches demonstrate RL technique on a first order system with a random input signal, however, in this work a second order system with step input is considered.

2- Problem statement

The role of the agent as a reinforcement learner and as the controller is depicted in Figure (1). Figure (1.a) is the canonical representation of a specific type of reinforcement learner: a Q-learning agent [5]. A state/action pair is input to the Q-learner to produce the value function representative of the input pair. This value function is commonly referred to as a Q-value. The Q-learning agent specifically implements the reinforcement learner by storing value functions .Via these value functions, the Q-learner can perform policy evaluation and policy improvement. Sutton, Barto and Kaelbling [1, 6] provide detailed description of Q-learners and value functions .The diagram in Figure (1.b) shows the agent in the role of a controller. Input to this agent is the current stats of the system the output is the agent's control signal. This agent implements a control policy.

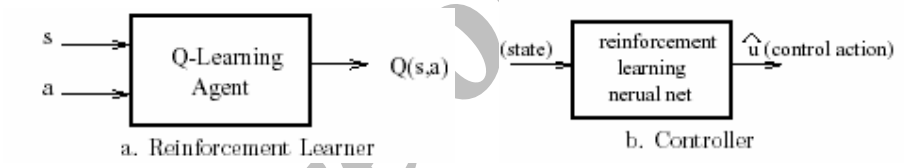

Figure1: Reinforcement Learning and Control Agents

Earlier efforts in reinforcement learning architecture utilized the actor-critic design. Actor-critic architectures are treated in [1] and only a brief description is given here. One of the Actor-critic architectures is called TD method that has a separate memory structure to explicitly represent the policy independent of the value function. The policy structure is known as the *actor*, because it is used to select actions, and the estimated value function is known as the *critic*, because it criticizes the actions made by the actor. Learning is always on-policy: the critic must learn about and critique whatever policy is currently being followed by the actor. The critique takes the form of a TD error. This scalar signal is the sole output of the critic and drives all learning in both actor and critic, as suggested by Figure 2:

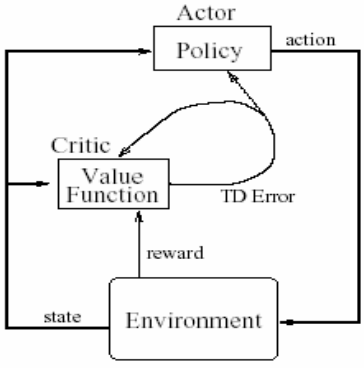

Figure 2: Actor-Critic Agent

Typically, the critic is a state-value function. After each action selection, the critic evaluates the new state to determine whether things have gone better or worse than expected. That evaluation is the TD error:

$$d_t = r_{t+1} + gQ(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \qquad (1)$$

Where Q is the current value function implemented by the critic. This TD error can be used to evaluate the action just selected, the action a_t taken in state s_t. If the TD error is positive, it suggests that the tendency to select a_t should be strengthened for the future, whereas if the TD error is negative it suggests the tendency should be weakened.

The actor network can be thought of as the control agent, because it implements a policy. This is a part of the dynamic system as it interacts directly with the system by providing control signals for the plant. The critic network implements the reinforcement learning part of the agent as it provides policy evaluation and can be used to perform policy improvement. This learning agent architecture has the advantage of implementing both a reinforcement learner and a controller.

First, a topology for a Neural Network (NN) representing the actor (the controller) is selected. This is usually a two layer feed forward multilayer perceptron, as shown in Figure (3). The NN architecture with *tanh* activation function in hidden layer explicitly implements a policy as a mathematical function and is therefore amenable to the stability analysis [7].

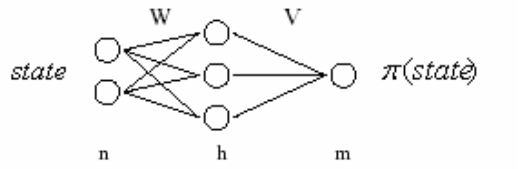

Figure3: Actor

Next attention is turned to the critic network (the reinforcement learner). As mentioned earlier, the critic accepts a set of state and action as inputs and produces the value function for the state/action pair, as shown in Figure(4). A so called Cerebellar Model Articulation Controller (CMAC) network is a more sophisticated variant of table lookup methods [8, 9] which features an improved ability to generalize on learning examples. Often CMAC networks require more training time than table lookup methods to reach their final weight values.

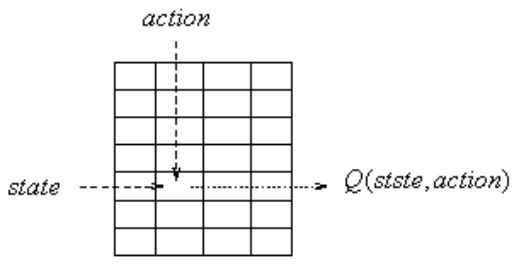

Figure4: Critic

In this work architecture is proposed that utilizes the internal states of the plant more effectively. Firstly, an approximate PID controller which ensures the system stability is designed. The online Reinforcement Learner, as outlined below is invoked in parallel with PID controller to learn those control actions which improve the system performance that is, either reducing the tracking error or regulating correctly as the case may be. The complete design procedures and the main computational step of the proposed method are outlined below.

3- Implementation

The proposed Q-learning parameters and algorithms are described below.

a)- Actor/critic configurations

(i) *Actor Net:*

- feed-forward ,two-layer ,neural network
- parameterized by input and output weights
- n (number of inputs) includes the tracking error, plant states and bias.
- m (number of outputs) here is only one, the control action.
- h (number of hidden units) can be small for faster learning.
- *tanh* is hidden unit activation function and activation function of output unit is linear.
- Trained via back propagation (gradient descent).

(ii) *Critic Net:*

- table look-up
- parameterized by table ,Q
- n-1+m input signal.
- a single output Q (s,a)
- Trained via Q-learning.

b) Reward punishment and learning parameters

- Punishment: abs(error)
- State: plant state ,tracking error and bias(only for actor net)

c) Learning Algorithm:

1. Initialize State, weights, learning rates
2. Send State to actor and calculate action u

If stopping criteria is satisfied then stop

3. Send State and action, u to critic and calculate Q_ value of current State.
4. Update critic for Q_ value of old State using Q_ learning algorithm

$$Q(oldstate) = Q(oldstate) + a\left[error + g \min_{action} Q(currentstate) - Q(oldstate)\right]$$

5. Find the optimal action u^* ($\hat{U}_{\ln}$ =small local neighborhood of u

$$u^* = \min_{u \in \hat{U}_{\ln}} Q(sate_{\ln}, u)\ )$$

6. Update actor weights for current State by using u^* .
7. Use u^* and states to calculate new action.
8. Sum action u with PID output as control action to plant.
9. Calculate new plant states, tracking error from the simulation program.
10. If time is equal End Time, end this episode.
11. Go to 2

The main purpose of the RL is to tune the weights of the actor network, and critic output from the CMAC (look up table) representing the stat action value. These parameters are updated in an iterative manner until halting criteria are satisfied .The input output training pair for the actor network are current tracking error and current states and the control action respectively. The control action acts on the system to produce new states. These new states and the error are sent to the critic as input for the evaluation of the action values based on the Q-learning.
The critic network table stores the action values. The table is indexed by state/action pairs. Each entry in the table refers to the value of a particular state/action pair. Recall that value refers to the sum of the future tracking errors over time. In reinforcement learning, we compute the temporal difference error and then perform gradient descent to update the table entries so as to minimize the temporal difference error. In the next step, we use the back propagation algorithm to train the actor network and use the information in the critic network to compute the training exemplar for actor.
The information stored about the state/action values by the critic network are used to find the optimal control action u . For this purpose, first we define an interval $\hat{U}_{\ln}$ as a local neighborhood around the actor network's current output. Since it is desired that the actor net make small incremental adjustments to its control action, it is not necessary to select a large neighborhood for u . Therefore, a grid of control actions is mapped out within the small neighborhood $\hat{U}_{\ln}$ to find the control action with the smallest value function according to the critic network. We use this value as the training exemplar for the actor network. Next the state information is updated, and the algorithm repeated for a set of new training sample.

4- Example

The system under consideration is described by the following open loop transfer function:

$$P(s) = \frac{5}{s^2 + 2s + 4} \qquad (2)$$

The reinforcement learning proceeds according to the followings.

Actor Net:

- n(number of inputs) = 4.(plant states: x_1, x_2 and error and bias)
- m =1 (action policy).
- h =3 (number of hidden units).

Critic Net:

- number of inputs = 4 (plant states: x_1, x_2 and error and action u)

- output=1 (Q-value)

A trail and error and approximate parameters for the PID controller are chosen as $k_p = 25, k_i = .1, k_d = 5$. The response of the plant to a step input with this setting of the PID controller as the only control action is shown in Figure (5) as the solid curve. It is seen that this control configuration is not capable of eliminating the steady state error.

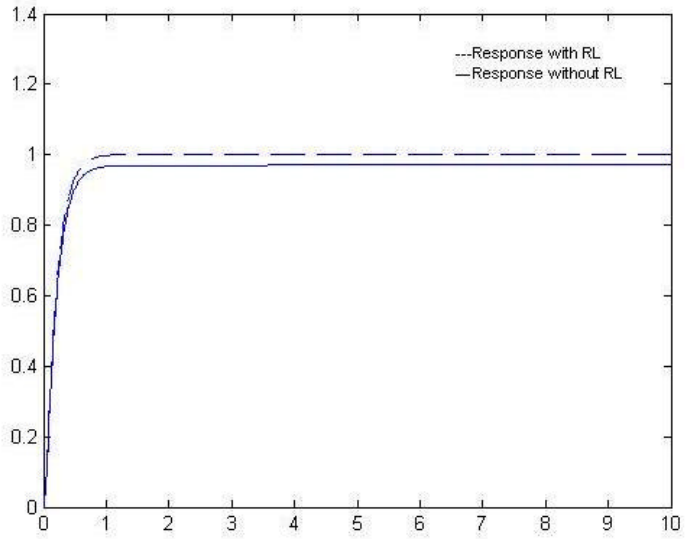

Figure5: Response of system

The internal state of the plant and tracking error are used as input to critic. As explained above Actor Network is a two layer feed forward network. The state space representation of the system is as follows:

$$\begin{aligned} \dot{x}_1 &= -2x_1 - 4x_2 + u \\ \dot{x}_2 &= x_1 \\ y &= x_2 \end{aligned} \qquad (3)$$

The Simulink implementation of the above state space model is shown in Figure (6).

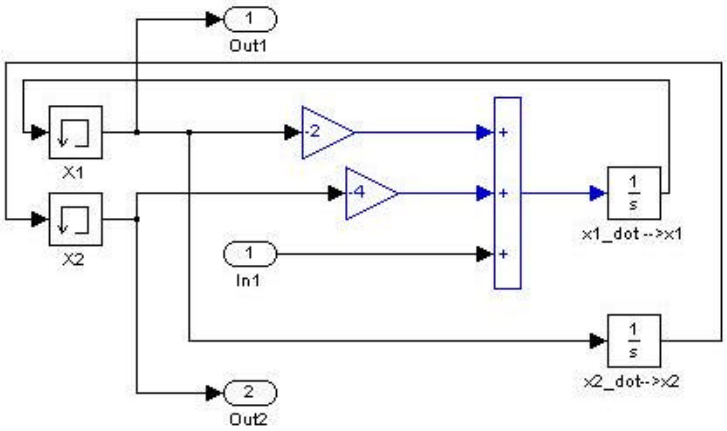

Figure6: Implementation of State Space

The complete reinforcement learning configuration is shown in Figure (7). For a unit step input, after 22 iterations the steady state error has been eliminated as shown in Figure (5) the dashed curve.

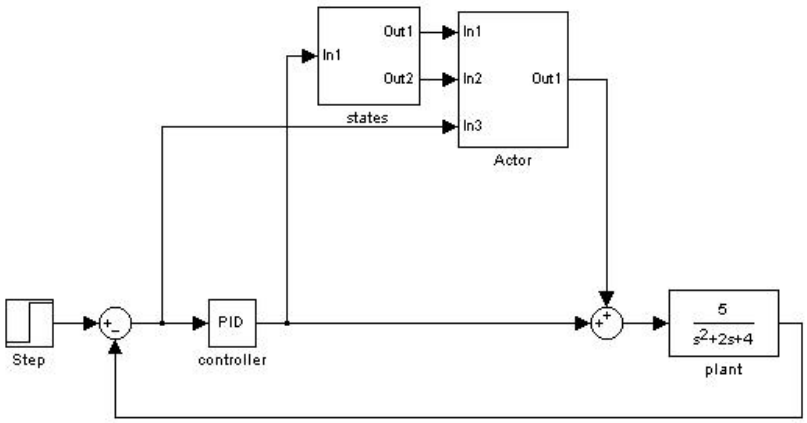

Figure 7: Combined PID controller and reinforcement learner

5- Conclusion

A control deign procedure based on reinforcement learning is developed. Although there are several different techniques for implementation of RL, a modified version of Q-learning which utilizes Temporal Difference has been implemented in this work. The actor which evaluates the learning policy function is implemented as a simple feed forward neural network trained by the back propagation algorithms using the systems parameters such as the internal states and systems tracking error for the training data. The critic that evaluated action values is in the form of a simple look up table being updated and refined based on TD learning. The procedure has been tested on a second order system. The result

shows a significant improvement over the classical PID controller. The main advantage of RL is that no knowledge of the model of the systems is required and the learning is solely based on the trial and error and particularly interaction with the environment.

Reference:

[1] Sutton, R. S. and Barto, A.G. *Reinforcement Learning: An Introduction.* The MIT Press, (1998).

[2] Kavehercy, S,*Reinforcement learning and Artificial Neural Networks: The optimal control of water vessel process*, M.Sc. thesis, University of Amsterdam, (1996).

[3] M.N. Howell, T.J. Gordon, *Continuous action reinforcement learning automata and their application to adaptive digital filter design*, *Engineering Applications of Artificial Intelligence* pp. 549-561, (2001).

[4] Anderson, C., Hittle, D., Katz, A., and Kretchmar, R. *Reinforcement learning combined with PI control for the control of a heating coil*. Journal of Artificial Intelligence in Engineering, (1996).

[5] Watkins, C. J. *Learning from delayed rewards*. PhD thesis, Cambridge University,(1989).

[6] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. *Reinforcement learning: A survey*.Journal of Artificial Intelligence, (1996).

[7] Kretchmar, R. M *A synthesis of reinforcement learning and robust control theory*, (2000).

[8] Miller, W. T., Glanz, F. H., and Kraft, L. G. *CMAC: An associative neural network alternative to back propagation*. Proceedings of the IEEE, 78:1561-1567, (1990).

[9] Sutton, R. S. *Generalization in reinforcement learning: Successful examples using sparse course coding.* In Advances in Neural Information Processing Systems 8, (1996).

[10] Kretchmar, R. M. and Anderson, C. W. *Comparison of CMACS and radial basis functions for local function approximation in reinforcement learning*. In ICNN'97: Proceedings of the International Conference on Neural Networks. ICNN, (1997).

[11] Kretchmar, R. M. and Anderson, C. W. *Using temporal neighborhoods to adapt function approximator in reinforcement learning*. In IWANN'99: International Workshop on Artificial Neural Networks. IWANN, (1999).