A New Nonlinear Reinforcement Scheme for Stochastic Learning Automata

DANA SIMIAN, FLORIN STOICA
Department of Informatics
“Lucian Blaga” University of Sibiu
Str. Dr. Ion Ratiu 5-7, 550012, Sibiu
ROMANIA
dana.simian@ulbsibiu.ro , florin.stoica@ulbsibiu.ro

Abstract: Reinforcement schemes represent the basis of the learning process for stochastic learning automata, generating their learning behavior. An automaton using a reinforcement scheme can decide the best action, based on past actions and environment responses. The aim of this paper is to introduce a new reinforcement scheme for stochastic learning automata. We test our schema and compare with other nonlinear reinforcement schemes. The results reveal a faster convergence of the new schema to the „optimal” action.

Key-Words: Stochastic Learning Automata, Reinforcement Scheme

1 Introduction

An stochastic learning automaton is a control mechanism which adapts to changes in its environment, based of a learning process. Rule-based systems may perform well in many control problems, but they require modifications for any change in the problem space. Narendra, in [5], emphasizes that the major advantage of reinforcement learning compared with other learning approaches is that it needs no information about the environment except that for the reinforcement signal. A stochastic learning automaton attempts a solution of the problem without any information on the optimal action and learns based on the environment response. Initially equal probabilities are attached to all actions and then the probabilities are updated based on environment response. There are three models for representation the response values. In the P-model, the response values are either 0 or 1, in the S-model the response values are continuous in the range (0, 1) and in the Q-model the values belong to a finite set of discrete values in the range (0, 1). The algorithm behind the desired learning process is named a reinforcement scheme.

Stochastic learning automata have many applications, one of them being the Intelligent Vehicle Control System. A detailed presentation about stochastic learning automaton and reinforcement schemes as well as applications in Intelligent Navigation of Autonomous Vehicles in an Automated Highway System can be found in [10].

The aim of this article is to build a new reinforcement scheme with better performances that other existing scheme ([7] – [11]).

The remainder of this paper is organized as follows. In section 2 we present the theoretical basis of our main results, the new reinforcement scheme is presented in section 3. Experimental results are presented in section 4. In section 5 are presented conclusions and further directions of study.

2 Mathematical model

Mathematical model of a stochastic automaton is defined by a triple \(\{\alpha, c, \beta\}\). \(\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}\) is a finite set of actions and defines the input of the environment. \(\beta = \{\beta_1, \beta_2\}\) represents a binary response set and defines the environment response (the outcomes). We will next consider only the P-model. We set \(\beta(n) = 0\) for a favorable outcome and \(\beta(n) = 1\) for an unfavorable outcome at the moment \(n\) \((n=1,2,\ldots)\). \(c = \{c_1, c_2, \ldots, c_r\}\) is a set of penalty probabilities, where \(c_i\) is the probability that action \(\alpha_i\) will result in an unfavorable response:

\[c_i = P(\beta(n) = 1|\alpha(n) = \alpha_i)\quad i = 1, 2, \ldots, r\]

Taking into account the changing of penalty probabilities over time, we classified the environment in stationary and nonstationary. In a stationary environment the penalty probabilities will never change while in a nonstationary environment the penalties will change over time.

In the following we will consider only the case of stationary random environments. Each action \(\alpha_i\), has associated, at time instant \(n\), a probability

\[\]
\[ p_i(n) = P(\alpha(n) = \alpha_j), \quad i, j = 1, \ldots, r \]

We denote by \( p(n) \) the vector of action probabilities:
\[ p(n) = (p_i(n), i = 1, \ldots, r). \]

The next action probability \( p(n+1) \), depends on the current probability \( p(n) \), the input from the environment, \( \alpha(n) \) and the resulting action.

Updating action probabilities is given by
\[ p(n+1) = T(p(n), \alpha(n), \beta(n)) \]

The type of mapping \( T \) gives the type of reinforcement scheme, which can be linear or nonlinear. There exist also hybrid schemes (see [5]).

For a given action probability vector, \( p(n) \), we define the average penalty, \( \mu(n) \),
\[ \mu(n) = P(\beta(n) = 1 \mid p(n)) = \sum_{i=1}^{r} P(\beta(n) = 1 \mid \alpha(n) = \alpha_i) \cdot P(\alpha(n) = \alpha_i) = \sum_{i=1}^{r} c_i p_i(n) \]

An important case of automata is the case of absolutely expedient automata. For absolutely expedient learning schemes are necessary and sufficient conditions of design. An automaton is absolutely expedient if (see [3]):
\[ M(n+1) < M(n), \quad \forall n \in N \]

The evaluation of performances of a learning automaton requires a quantitative basis for assessing the learning behavior. The problem of establishing “norms of behavior” is quite complex, even in the simplest P-model and stationary random environments. We will consider only the notions we need for introducing our new reinforcement scheme. Many details about mathematical model of stochastic learning automaton and the evaluation of its performance can be found in [10].

The general solution for absolutely expedient schemes was found by Lakshmivarahan in 1973 ([3]).

Our new reinforcement scheme starts from a nonlinear absolutely expedient reinforcement scheme, for a stationary N-teacher P-model environment, presented by Únsal ([10] – [11]). The action of the automaton results in a vector of responses from environments (or “teachers”). If the automaton perform, at the moment \( n \), the action \( \alpha_k \) and the environment responses are \( \beta_i^j \), \( j=1,\ldots,N \) then the vector of action probabilities, \( p(n) \), is updated as follows ([11]):
\[
\begin{align*}
\hat{p}_i(n+1) &= \hat{p}_i(n) + \frac{1}{N} \sum_{k=1}^{N} \beta_i^k \cdot \sum_{j \neq i} \phi_j(p(n)) - \\
&- \left[ 1 - \frac{1}{N} \sum_{k=1}^{N} \beta_i^k \right] \cdot \sum_{j \neq i} \psi_j(p(n))
\end{align*}
\]

(1)

The functions \( \phi_i \) and \( \psi_i \) satisfy the following conditions \( \forall j \in \{1,\ldots,r\} \setminus \{i\} \):
\[
\begin{align*}
\frac{\phi_i(p(n))}{p_i(n)} &= \ldots = \frac{\phi_j(p(n))}{p_j(n)} = \lambda(p(n)) \\
\frac{\psi_i(p(n))}{p_i(n)} &= \ldots = \frac{\psi_j(p(n))}{p_j(n)} = \mu(p(n))
\end{align*}
\]

(2)

(3)

(4)

(5)

(6)

(7)

(8)

The conditions (5)-(8) ensure that
\[ 0 < p_k < 1, \quad k = 1, \ldots, r. \]

If the functions \( \lambda(p(n)) \) and \( \mu(p(n)) \) satisfy the conditions:
\[
\begin{align*}
\lambda(p(n)) &\leq 0 \\
\mu(p(n)) &\leq 0
\end{align*}
\]

(9)

then the automaton with the reinforcement scheme given in (1) – (2) is absolutely expedient in a stationary environment ([1], [10]).

3 Main results. A new nonlinear reinforcement scheme

Because the theorem from [1], which defines the conditions for absolutely expediency of a reinforcement scheme in a stationary environment, is also valid for a single-teacher model, we can define a single environment response that is a function \( f \) which combines many teacher outputs. Thus, we can update the Únsal’s algorithm. We have the following update rule:
\[
\begin{align*}
p_i(n+1) &= p_i(n) + f \cdot (\delta \cdot (1 - \theta) \cdot H(n)) \cdot \\
&\cdot [1 - p_i(n)] \cdot (1 - f) \cdot (\theta) \cdot (1 - \delta) \cdot [1 - p_i(n)]
\end{align*}
\]

(10)
\[ p_j(n+1) = p_j(n) - f(-\delta (1-\theta) H(n)) \]

\[ * p_j(n) + (1-f) (-\delta (1-\theta) p_j(n)) \]

\[ \forall j \neq i \]

(11)

Hence, our scheme is derived from the reinforcement scheme given in (1) - (2), setting

\[ \psi_j(p(n)) = -\delta (1-\theta) p_j(n) \]

\[ \phi_j(p(n)) = -\delta (1-\theta) H(n) * p_j(n) \]

where learning parameters \( \theta \) and \( \delta \) are real values which satisfy the conditions \( 0<\theta<1, 0<\delta<1 \).

The function \( H \) is defined as:

\[ H(n) = \min_1: \max_{1} \left[ \frac{p_j(n)}{\delta (1-\theta) (1-p_j(n))} - \epsilon, \right] \]

\[ \left( \frac{1-p_j(n)}{\delta (1-\theta) p_j(n)} - \epsilon \right] \]

where \( \epsilon \) is an arbitrarily small positive real number.

Our reinforcement scheme differs from schemes given in [10] - [11] and [9] by the definition of the functions: \( H \), \( \Psi_k \), and \( \phi_k \).

We prove next that our scheme verifies all the conditions of the reinforcement scheme defined in (1) - (2).

\[ \phi_j(p(n)) = -\delta (1-\theta) H(n) * p_j(n) \]

\[ \psi_j(p(n)) = -\delta (1-\theta) p_j(n) \]

\[ \lambda(p(n)) = \lambda(p(n)) \]

Condition (5) is satisfied by the definition of function \( H(n) \):

\[ p_j(n) + \sum_{j \neq i} \phi_j(p(n)) > 0 \Leftrightarrow \]

\[ p_j(n) - \delta (1-\theta) H(n) * (1-p_j(n)) > 0 \Leftrightarrow \]

\[ \delta (1-\theta) H(n) * (1-p_j(n)) < p_j(n) \Leftrightarrow \]

\[ H(n) < \frac{p_j(n)}{\delta (1-\theta) (1-p_j(n))} \]

Condition (6) is equivalent with:

\[ p_j(n) - \theta (1-\delta) p_j(n) < p_j(n) + \theta (1-\delta) (1-p_j(n)) < 1 \]

It is satisfied because, for \( 0<\theta<1, 0<\delta<1 \), we have

\[ p_j(n) + \theta (1-\delta) * (1-p_j(n)) < p_j(n) + (1-p_j(n)) = 1 \]

Condition (7) is equivalent with:

\[ p_j(n) + \psi_j(p(n)) > 0 \Leftrightarrow p_j(n) - \theta (1-\delta) p_j(n) > 0 \]

for all \( j \in \{1,...,r\} \setminus \{i\} \)

It holds because

\[ p_j(n) - \theta (1-\delta) p_j(n) = p_j(n) * (1-\theta * (1-\delta)) > 0 \]

for \( 0<\theta<1, 0<\delta<1 \) and \( 0<p_j(n)<1 \), for all \( j \in \{1,...,r\} \setminus \{i\} \)

Condition (8) is equivalent with:

\[ p_j(n) - \phi_j(p(n)) < 1 \Leftrightarrow p_j(n) + \delta (1-\theta) H(n) * p_j(n) < 1 \Leftrightarrow \]

\[ H(n) < \frac{1-p_j(n)}{\delta (1-\theta) p_j(n)}, \forall j \in \{1,...,r\} \setminus \{i\} \]

This condition is satisfied by the definition of the function \( H(n) \).

All conditions of the equations (5) - (8) being satisfied, our reinforcement scheme is a candidate for absolute expediency.

Furthermore, the functions \( \lambda \) and \( \mu \) for our nonlinear scheme satisfy the following:

\[ \lambda(p(n)) = -\delta (1-\theta) H(n) \leq 0 \]

\[ \mu(p(n)) = -\theta (1-\delta) < 0 \leq 0 \]

\[ \lambda(p(n)) + \mu(p(n)) < 0 \]

In conclusion, we state the algorithm given in equations (10) - (11) is absolutely expedient in a stationary environment

4 Experimental results

Reinforcement learning is justified if it is easier to implement the reinforcement function than the desired behavior, or if the behavior generated by an agent presents properties which cannot be directly built.

For this second reason, reinforcement learning is used in autonomous robotics ([2], [10]).

To prove the performances of our new scheme we show that our algorithm converges to a solution faster than the one given in [11] and our algorithm presented in [9]. In order to do this we will use the example from [9]. Figure 1 illustrates a grid world in which a robot navigates. Shaded cells represent barriers. The current position of the robot is marked by a circle. Navigation is done using four actions \( \alpha = \{N, S, E, W\} \), the actions denoting the four possible movements along the coordinate directions.

![Figure 1 - experimental problem for testing](image-url)

The algorithm used is:
While the probability of the optimal action is lower than a certain value (0.9999) do
Step 1. Choose an action, \(a(n)=a_i\), based on the action probability vector \(p(n)\).
Step 2. Read environment responses \(\beta_i^j, j=1,...,4\) from the sensor modules.
Step 3. Compute the combined environment responses \(f\).
Step 4. Update action probabilities \(p(n)\) according to the predefined reinforcement scheme.

End while

Table 1 presents the results for different values of \(\theta\) and \(\delta\) and for two different initial conditions, where in the first case all probabilities are initially the same and in second case the optimal action initially has a small probability value (0.0005), with only one action receiving reward (only one optimal action).

<table>
<thead>
<tr>
<th>(\theta)</th>
<th>(\delta)</th>
<th>New alg.</th>
<th>New alg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.33</td>
<td>0.50</td>
<td>31.02</td>
<td>48.09</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>30.94</td>
<td>61.05</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>30.03</td>
<td>67.99</td>
</tr>
<tr>
<td></td>
<td>0.15</td>
<td>29.29</td>
<td>77.86</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>28.80</td>
<td>94.28</td>
</tr>
<tr>
<td>0.22</td>
<td>0.50</td>
<td>33.12</td>
<td>52.40</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>39.31</td>
<td>66.90</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>40.26</td>
<td>73.26</td>
</tr>
<tr>
<td></td>
<td>0.15</td>
<td>41.98</td>
<td>84.28</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>42.26</td>
<td>100.68</td>
</tr>
<tr>
<td>0.11</td>
<td>0.50</td>
<td>53.22</td>
<td>66.54</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>56.49</td>
<td>74.19</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>53.18</td>
<td>84.91</td>
</tr>
<tr>
<td></td>
<td>0.15</td>
<td>66.31</td>
<td>100.82</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>73.73</td>
<td>124.60</td>
</tr>
</tbody>
</table>

Table 1: Convergence rates for a single optimal action of a 4-actions automaton (200 runs for each parameter set)

Comparing values from corresponding columns, we conclude that our algorithm converges to a solution faster than the one obtained in [9] when the optimal action has a small probability value assigned at startup.

5 Conclusions

Reinforcement learning is a great interest topic for machine learning and artificial intelligence communities. It offers a way of programming agents by reward and punishment without needing to specify how the task (i.e., behavior) is to be achieved.

In this article we built a new reinforcement scheme. We proved that our new scheme satisfies all necessary and sufficient conditions for absolute expediency in a stationary environment and the nonlinear algorithm based on this scheme is found to converge to the “optimal” action faster than nonlinear schemes previously defined in ([11], [9]). The values of both learning parameters are situated in the interval \((0,1)\), giving more stability to our scheme. Using this new reinforcement scheme was developed a simulator for an Intelligent Vehicle Control System, in a multi-agent approach. The entire system was implemented in Java, and is based on JADE platform. The simulator will be presented and analyzed in other article.

As a further improvement, we will use a genetic algorithm for finding the optimal values of learning parameters from our reinforcement scheme. The fitness function for evaluation of chromosomes will take into account the number of steps necessary for learning process to accomplish the stop condition.

References:


