ECAI 92

10th European Conference on Artificial Intelligence
August 3 – 7, 1992, Vienna, Austria
Proceedings

Edited by
Bernd Neumann
Organised by the European Committee for Artificial Intelligence (ECCAI)
Hosted by the Austrian Society for Artificial Intelligence

Bernd Neumann
University of Hamburg
Germany
Programme Chairperson

Werner Horn
Austrian Research Institute for Artificial Intelligence, Vienna
Austria
Local Arrangements Chairperson

JOHN WILEY & SONS
Chichester · New York · Brisbane · Toronto · Singapore

Collision-Free Movement of an Autonomous Vehicle Using Reinforcement Learning

D. Kontoravdis, A. Likas and A. Stafylopatis *

Computer Science Division, Department of Electrical and Computer Engineering
National Technical University of Athens, 157 73 Zographou, Athens, Greece

Abstract. This paper is aimed to explore the potential use of reinforcement learning in control applications. Reinforcement learning systems are of particular interest because they require as training feedback only a scalar signal provided to the entire neural network and they admit a simple on-line implementation. In this paper we demonstrate the ability of a reinforcement learning adaptive controller to drive an autonomous vehicle through simulated paths comprising left and right turns. The neural network is responsible for providing the proper control commands so that the vehicle stays on the road and avoids collision. Keywords: adaptive control, neural networks, reinforcement learning, autonomous vehicle.

1 Introduction

There has been a great interest in the use of neural networks in the area of adaptive control. Techniques for training and utilizing neural network models effectively as components of the overall control system have been investigated by many researchers [3, 7, 9].

Typically a trainable adaptive controller architecture facing supervised learning consists of a teacher, the trainable controller and the plant to be controlled. The teacher may be automated as a linear or nonlinear control law, or it may be a human expert. The controller consists of a neural network architecture that is suitable for supervised learning, while the teacher constitutes the source that provides examples of desired behavior. Through these examples the neural network learns to imitate the behavior of the expert, thus becoming able to function autonomously. The ability to train neural networks in this fashion is extremely useful when the classic AI approach of rule based inferences is not applicable because such rules are not clear and well organized.

However, in some learning tasks arising in control neither a designed nor a human expert is available. In such cases we wish to adjust a control rule in order to improve the performance of a plant as measured by a performance measure that in some way evaluates the overall behavior of the plant. In these situations it may be possible to improve plant performance over time by means of on-line learning methods performing what is called *reinforcement learning* [3, 10]. Reinforcement learning has been effectively applied to control tasks. In [1, 2], neural nets learning via reinforcement learning and temporal differences methods were used in controlling the motion of an inverted pendulum. The applicability of the approach has been also demonstrated by Helferty et al. [5] for a one legged hopping machine and by Hoskins and Himmelblau [6] in a chemical process control situation.

Our concern in this paper is the potential use of a reinforcement learning strategy to drive an autonomous vehicle through simulated paths made of straight segments as well as left and right turns. The neural network is assigned the task of providing the proper control commands so that the vehicle stays on the road and avoids collision. The system developed demonstrates a type of adaptive probabilistic search, allowing the quick selection of an action due to parallelism inherent in connectionist networks.

Section 2 of the paper gives an overview of the reinforcement learning algorithm making a contrast to the supervised learning paradigm, while Section 3 presents in detail the autonomous vehicle navigation task. In Section 4 the reinforcement learning algorithm used to train the neural adaptive controller is introduced, while the experimental results concerning the performance of the proposed approach are discussed in Section 5. Finally, the main conclusions are summarized in Section 6.

2 Reinforcement Learning in Control

Reinforcement learning addresses the problem of improving performance as evaluated by any measure whose values are supplied to the learning network. In tasks of this kind desired control signals are not

*Part of this work was supported by ESPRIT Project ANNIE.

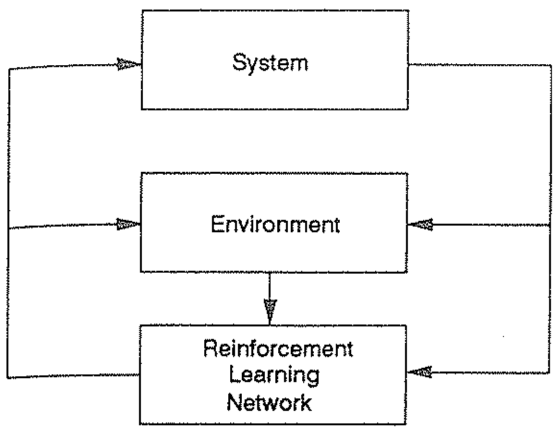

Figure 1: A reinforcement learning system

supplied to the learning network, but instead the latter receives from the external environment a scalar reinforcement signal which indicates the appropriateness of its response. Based on that evaluation signal the network has to determine the necessary changes that would lead to increases in the measure of system performance. The situation is illustrated in Figure 1 which shows a reinforcement learning network interacting with a system and receiving an evaluation signal generated by the environment. Thus, it can be said that in a supervised learning task the feedback provided to the system by the teacher is purely 'instructive', whereas in a reinforcement learning task, feedback can be thought of as 'evaluative' [10].

A reinforcement learning task mainly involves two issues [7]. The first issue is to specify a critic capable of evaluating system performance in a way that is both appropriate to the actual learning objective and informative enough to allow learning. This critic may be either an external agent providing an immediate reinforcement signal or another network that adapts over time (called 'adaptive critic') and learns to predict future consequences of an action. Sutton [8] has developed a general class of algortihms for the training of such networks called *temporal difference algorithms*. The second issue concerns the determination of how to teach the learning network to provide outputs that improve performance as it is measured by the critic.

A direct approach to adjusting control actions in order to improve system performance is to actively explore the space of network outputs. Specifically, modifications to actions are performed and the resulting changes to performance evaluation are observed. Changes leading to improved performance are incorporated into the control rule. Since in this approach there is a need for active exploration among possible network outcomes, there must be a source of variation within the network, to allow exploration of alternative actions. To this end, it is generally assumed that at least some of the units in the network compute their outputs as a stochastic function of their inputs. Such networks actively search for the optimal output pattern to associate with each input pattern. Williams [10] analyzed a general class of learning algorithms for networks of neuron-like stochastic computing elements facing associative reinforcement learning tasks. Moreover, he called the algorithms belonging to this class, REINFORCE algorithms.

In this paper we have implemented a reinforcement learning system adopting the direct approach mentioned above. The objective of the system is to achieve a collision-free movement of an autonomous vehicle using only little knowledge about the task. The autonomous vehicle driving task is well adapted to reinforcement learning techniques, since it allows for a meaningful reinforcement signal providing valuable information at each step of the learning process. On the contrary, most work reported in the literature mainly concerns applications in which an informative reinforcement signal evaluating a given action cannot be obtained readily but only with a considerable delay.

3 Driving an Autonomous Vehicle

To investigate the utility of reinforcement learning systems in the control area, we taught a neural network to drive a robot vehicle along a simulated path comprising left and right turns. The purpose of the network is to give the proper driving commands as a response to the current state of the vehicle, so that the car moves in a course without collisions. This task represents a sensor-motor association problem in the control area. Conventional approaches to solve the problem use artificial intelligence methods (expert systems), which are time-consuming and require much knowledge of the environment in which the vehicle is moving. In our approach we assume that very little is known about the vehicle's environment.

The vehicle recognizes its environment through the use of sensors. The number and configuration of the ultrasonic sensors plays an important role in the control task. On the one hand the number of sensors can not be arbitrarily large, on the other hand there must be a sufficient number of sensors to acquire the necessary information about the vehicle's environment. For our purposes nine sensors seemed to be adequate. Four of them were placed at the front side of the vehicle, two on both the left and right side and finally one sensor was situated at the back.

Each sensor can detect the presence of an obstacle situated within a conic space in front of it and provides a measure of the distance of the obstacle. To this end, every sensor cone was partitioned logarithmically into 7 areas, thus making possible the distinction of 8 different distances, each one binary coded in 3 bits. The coding was made in such a way that two neighboring areas differ in only one bit. Furthermore, the larger the number of 1's in a vector the closer to the vehicle is the detected obstacle. It should be noted that the logarithmic partitioning of the sensors results in more areas in the near range of the vehicle than there are far away. This is desirable since information concerning the environment close to the vehicle has more influence on driving decisions. From the above it is obvious that the sensors are supplying a 27 bit vector describing the environmental state of the vehicle.

The network is able to give eight different driving commands in response to the current state of the vehicle, each one coded as a 3 bit vector: forward, 90 degrees left, 60 degrees left, 30 degrees left, 90 degrees right, 60 degrees right, 30 degrees right and backward. Therefore, the learning network has 3 binary valued outputs.

In what follows we will describe in more detail the neural adaptive controller and the algorithm used to train it.

4 Specification of the Reinforcement Learning System

In order to construct our learning system we should effectively deal with the two issues involved in a reinforcement learning task. As far as our application is concerned, the first issue relates to the performance evaluation of the vehicle in a way that is appropriate to the learning objective, i.e. the movement of the car along a path without collision. Therefore, the evaluation of the vehicle's state should be based on how probable is for the car to collide and go off the road. This probability can be measured by the distances of the obstacles detected by the sensors. The farther are the obstacles the better is the state of the system. The performance evaluation is supplied to every unit of the neural controller by means of a scalar signal called *reinforcement*. In our system we assume that the reinforcement signal is provided by an agent external to the network. The latter is not aware of the mechanism used to compute it. Moreover, the reinforcement is real-valued ranging over [0,1], thus indicating a graded degree of success. One can think of $r = 1$ as 'success' and $r = 0$ as 'failure'. In our control task, failure corresponds to the situation where at least one of the nine sensors detects an obstacle at its closest distance. Moreover, success relates to the situation where there is no obstacle in the range covered by any of the front sensors. In general, the reinforcement signal r supplied to the network controller is obtained as an average of the partial reinforcements r_i, $i = 1 \dots 4$, computed by the four front sensors according to the following rule:

$$r_i = (1.0 - \frac{\xi}{7}) \tag{1}$$

where ξ takes on values from the set $\{0, 1, 2, 3, 4, 5, 6\}$. The value $\xi = 0$ indicates that there is no obstacle in the range covered by a specific sensor, while larger values of ξ correspond to obstacles at a closer distance. Exception to the aforementioned rule constitutes the case of failure which is determined by examining the values of *all* sensors to meet the punishment condition ($\xi = 7$) referred to above. It is obvious from the above description of the evaluation mechanism that large reinforcements correspond to actions leading to improved performance of the controlled system.

Another point that should be noted is that the more complex is a control task the more informative should be the evaluative signal, in order to help the system avoid undesired states, thus leading to faster learning. This is necessary in order to compensate as much as possible for the lack of an explicit signal specifying the desired response of the controller or the system to be controlled. In the autonomous vehicle navigation task, apart from the evaluation strategy presented above we used a heuristic based on the notion of *favourite direction*. The role of the favourite direction is to help the vehicle decide in situations where different actions (e.g. a left or right turn) can be chosen, all resulting in rewarding evaluations by the environment. If in such cases the vehicle does not follow the favourite direction, then its action is evaluated as a failure. In fact, favourite direction serves as a means to help the vehicle avoid driving in a course without objective e.g. going forward and backward along the same segment of the path.

We now turn to the second issue encountered in a reinforcement learning task. This concerns the construction of a real-time neural controller which learns to adapt action probabilities in order to maximize some function of the reinforcement signal, specifically the expectation of its value on the upcoming trial. The controller decides which action to apply given the state of the vehicle. By correlating the actions chosen and the received reinforcement signal, the controller adapts the probability of choosing an action by modifying the connection weights. Thus, rewarded actions become more probable while the probability of penalized actions is decreased.

The neural controller was implemented as a connectionist network consisting of Bernoulli logistic units [10]. There are 27 input and 3 output units in the network whereas there are no hidden layers. The network receives the reinforcement signal r from the external environment and updates the weights using the following modified version of the REINFORCE rule:

$$\Delta w_{ij}(t) = \alpha(r(t) - \overline{r}(t-1))\overline{e}_{ij}(t) \qquad (2)$$

where α is a learning rate factor, $r(t)$ is the real valued punish/reward signal at time t, $\overline{r}(t-1)$ is an exponentially weighted average, or trace, of prior reinforcement signals and $\overline{e}_{ij}(t)$ is an exponentially weighted average of recent values of the eligibility e_{ij} of w_{ij}, called *eligibility trace.*

The eligibility trace is computed by

$$\overline{e}_{ij}(t) = \delta\overline{e}_{ij}(t-1) + (1-\delta)e_{ij}(t) \qquad (3)$$

where δ is a decay rate positive and less than 1, while the eligibility $e_{ij}(t)$ of the weight w_{ij} at time t is given by

$$e_{ij}(t) = (y_i(t) - p_i(t))x_j(t) \qquad (4)$$

where y_i denotes the output of the i^{th} unit, p_i denotes the probability that the i^{th} unit produces a 1 and x^j denotes the input to the i^{th} unit from the j^{th} unit.

The trace of prior reinforcement values is given by

$$\overline{r}(t) = \gamma\overline{r}(t-1) + (1-\gamma)r(t) \qquad (5)$$

where γ is a decay rate positive and less than 1.

Qualitatively, the weight update rule in equation (2) has the effect of rewarding actions which lead to better than usual reinforcement and penalizing actions leading to worse than usual reinforcement. In fact, the use of *reinforcement comparison* leads to faster and more reliable learning. Specifically, an a priori prediction of what reinforcement value to expect on a particular trial is used as the basis for comparison. This prediction computed as an exponentially weighted average of past reinforcement values, is itself adaptive. It should be pointed out that the reinforcement trace $\overline{r}$ is computed and stored for each state of the vehicle. This can be implemented using a lookup table or any other function approximation technique such as connectionist networks. Another point that should be mentioned is the use of the eligibility trace in the weight update rule. Eligibility trace is a quantity indicating how much responsibility (or credit) an earlier state, as well as the taken action, has for the current system status. The decay rate indicates that this responsibility (or credit) decreases with time. By ascribing the credit/blame of the current state to earlier actions, we perform a kind of temporal credit assignment: the consequences of an action unfold over time. Moreover, this causes the weights to change 'smoothly'. Thus, the performance of the learning algorithm is improved.

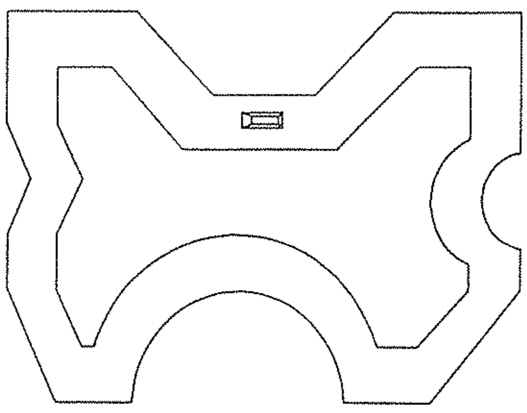

Figure 2: A typical ground

5 Results

Each experiment we performed consisted of a number of runs that differed only in the seed values for the random number generator. Each run consisted of a sequence of cycles, where each cycle began with the vehicle at the same initial state and ended with a failure signal. At the start of each run the vehicle was placed to a randomly chosen state. Except for the random initial state, identical parameter values were used for all runs. Statistical results of the effectiveness of learning during each run were obtained as follows. For smoothing purposes, at the end of each cycle an average value of the number of steps per cycle was computed by averaging over all cycles from the beginning of the run up to that point. Finally, curves were plotted at the end of each experiment by averaging over all runs. This representation aims at giving an overall view of the progress of learning without being affected by random fluctuations. Of course, under this style of presentation, the contribution of high scores is not readily visualized, since it slowly affects the average value.

The learning system was 'naive' at the start of each run, i.e., all the weights w_{ij} were set to zero. This made all the actions equally probable. Initially, we tried to optimize the parameters of the weight modification equations by manually searching for values resulting in best performance. For this purpose, various

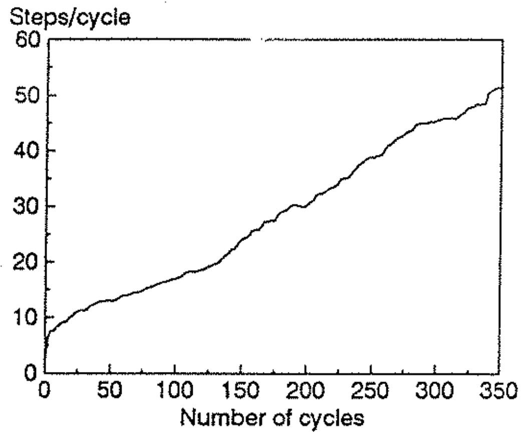

Figure 3: Performance of the vehicle

types of grounds were tested and for each ground we simulated a series of runs. After testing approximately 10 sets of values, we found that the best parameter values are: $\alpha = 0.7$, $\gamma = 0.9$ and $\delta = 0.65$.

After having selected the values of the parameters we investigated the behavior of the vehicle under the commands of the reinforcement learning controller. To this end, a large number of grounds were tested. Although the search was initially uniformly random over the possible actions, it gradually gained direction as the action probabilities became biased in favor of more successful actions. As a result, in the initial stages, when the network was 'naive', the behavior of the vehicle was very unstable. The vehicle could not stay on the road before significant training was accomplished. However, at the end the vehicle exhibited very good behavior, i.e. very long cycles were accomplished.

Among the grounds used in our experiments a typical one is shown in Figure 2. Figure 3 shows the performance of the system during learning to move on the above ground, by presenting the average number of steps per cycle as a function of the number of cycles. As can be seen the vehicle attained excellent performance. Analogous was the behavior of the system in all experiments with similar grounds.

6 Conclusions

The objective of this paper has been the achievement of a greater understanding of the role of reinforcement learning in control. Our experiments revealed that we can take advantage of the exploratory nature of reinforcement learning to directly modify control rules in order to obtain improvements in system performance.

Several extensions or alterations to the presented approach may be considered in the future in order to gain a deeper understanding of the reinforcement learning strategy. Among these we can mention the use of sensors with continuous-valued areas and incorporation to the system of the ability of the vehicle for analog movements, the employment of temporal difference methods to effectively deal with combined structural and temporal credit assignment and the adoption of a hybrid approach, e.g. combination with an expert system to achieve a more effective evaluation of the performance of the vehicle.

Acknowledgement

We wish to thank IBP Pietsch, Ettlingen, Germany, for our cooperation within the ANNIE Project, that allowed us to gain insight into the problems of autonomous vehicle navigation.

References

[1] Anderson, C.W., *Learning to Control an Inverted Pendulum Using Neural Networks*, IEEE Control Sys. Mag., vol.9, num.3, April 1989.

[2] Barto, A.G., Sutton, R.S. and Anderson, C.W., *Neuronlike Elements that Can Solve Difficult Learning Control Problems*, IEEE Transactions on Systems, Man, and Cybernetics, 13, 835-846, 1983.

[3] Barto, A.G., Bradtke, S.J. and Singh, S.P., *Real-time Learning and Control Using Asynchronous Dynamic Programming*, Technical Report 91-57, University of Massachusetts, Amherst, August 1991.

[4] Guez, A., and Selinsky, J., *A Neuromorphic Controller with a Human Teacher*, Proc. IEEE Intern. Conf. on Neural Networks, San Diego, CA, pp. II-595 - II-602, July 1988.

[5] Helferty, J.J., Collins, J.B. and Kam, M., *A Learning Strategy for the Control of a Mobile Robot that Hops and Runs*, Proc. IASTED-88, Galveston, pp. 7-11, 1988.

[6] Hoskins, J.C. and Himmelblau, D.M., *Automatic Chemical Process Control Using Reinforcement Learning in Artificial Neural Networks*, First Annual Int. Neural Network Society Meeting, Boston, MA, Sept. 1988.

[7] Miller, W.T., Sutton, R.S. and Werbos, P.J. (eds), *Neural Networks for Control*, MIT Press, 1990.

[8] Sutton, R.S., *Learning to Predict by the Methods of Temporal Differences*, Machine Learning, vol.3, pp.9-44, 1988.

[9] Werbos, P.J., *Backpropagation and Neurocontrol: A Review and Prospectus*, Proceedings of the IEEE IJCNN, New York, IEEE Press, I:209-216, 1989.

[10] Williams, R.J., *Toward a Theory of Reinforcement Learning Connectionist Systems*, Technical Report NU-CCS-88-3, College of Computer Science, Northeastern University, 1988.