A Neural Network for Automatic Vehicles Guidance.

Alessandro Ghidotti Piovan

Università di Bologna, Italy
alessandro.ghidotti@studio.unibo.it

Abstract. The purpose of this work involves the application and the evaluation of a Reinforcement Learning (RL) based approach to address the problem of controlling the steering of a vehicle. The car is required to drive autonomously on unknown tracks without never going out of the way. The problem has been solved by using a simple neural network with two neurons, and according to the ASE-ACE algorithm. The evaluation has been carried out with referring to The Open Racing Car Simulator (TORCS).

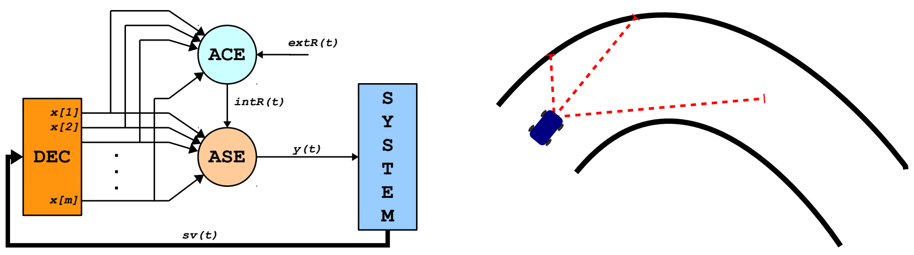
Keywords: reinforcement learning, neural network, car simulator, automatic vehicle guidance

1 Introduction

Reinforcement Learning (RL) is a learning paradigm for neural networks based on the idea of an agent which is placed into a certain environment and it is required it learns to take proper actions without having any knowledge about the environment itself. The agent learns by trial and error: an entity called *critic* observes the agent's behaviour and punishes or rewards the agent if it executes respectively wrong or proper actions; these kinds of judgements on the action taken are said *reinforcements* [1]. Thus, the learning process takes place with the agent beginning to operate into a certain environment with mostly wrong actions, but taking into account reinforcements provided by the critic, and eventually changing its behaviour in order to operate correctly.

A simple neural model for RL implementation is shown in Fig.1(a) and was proposed in [2]. A certain number of state variables $sv(t)$ represent the system state in some time instant. This state variables array is provided as input for a decoder which couples it with the corresponding sensorial stimulus x_i. The decoded stimuli are given as input for (i.e. connected to the synapses of) the two artificial neurons named Associative Search Element (ASE) and Adaptive Critic Element (ACE). Any agent's action depends on the control signal $y(t)$ generated by the ASE. This model distinguishes explicitly *external* from *internal* reinforcement concepts: the former, indicated with $extR(t)$, is the reinforcement signal effectively coming from the critic entity; the latter, indicated with $intR(t)$, is a further and more informative reinforcement coming from the ACE. Notice that this model is based *only* on punishments: the critic never rewards the agent. As the agent's neural network learns the proper behaviour, more and more wrong actions decrease, and the learning curve's convergence slows down; this happens because failures grow away more and more from the taken decision over time. Therefore, the ACE's role is to *predict* whether there will be a failure as a result of an action taken in a certain agent-environment context, not providing to the ASE a "boolean" reinforcement signal, but a finer-graned one which represents the failure probability in that state. The probability is calculated according to the $extR(t)$ value (and some other parameters too [2]), which on the contrary here is a boolean one. The more the state "dangerous" is, the lower the $intR(t)$ value is, and vice versa. Thus, the ASE associates a sensorial stimulus with a control action, and the ACE associates a sensorial stimulus with a failure probability. Finally, ASE-ACE model is really simple to implement and suitable for the automatic vehicles guidance problem too.

Experiments and evaluations have been carried out by using The Open Racing Car Simulator (TORCS) [3]. With respect to this development environment, the system entity shown in Fig.1(a) is the so said *robot*, that is the vehicle chosen for the race.



(a) (b)

Fig. 1. (a) ASE-ACE Neural model. (b) Proximity Sensors model

2 ASE-ACE Applied to the Automatic Vehicles Guidance

2.1 The Problem Model

It's required to provide an intelligent controller for the steering actuator of a vehicle, in such a way that it is able to drive autonomously on any track within a certain category (i.e., road, dirt, snow, etc.). The problem has been modelled in a very simple way which can be schematized as follows:

- The car speed depends on an auxiliary control system, here modeled as a *black-box* on which it is not actually possible to act or get information.

- A set of proximity sensors (i.e. laser range-finders) are installed on the vehicle and each of them provides the distance from the sensor itself and the nearest track boundary into its range [5].
- The steering control output is given by a real value in the range $[-1, 1]$; positive values produce a steering left rotation, a right rotation the negative ones.
- The steering control system does not have any kind of knowledge on the surrounding environment or the physical laws that regulate the car motion.

The aforesaid elements lead to the exclusion of learning paradigms based on training-set or needing a knowledge-base, since the only information available according to the model are the proximity sensors output signals. Therefore, a suitable approach for this problem can be the RL paradigm, just because it does not need other information than the signals coming from the proximity sensors to perform both the training and validation phases.

2.2 Sensors and State Decoding

Since the tracks can have an infinite number of different geometrical characteristics, the choice on the proximity sensors number and their orientation, range, and resolution must be chosen here in an experimental way. The simple and quite good configuration used in this work uses only three sensors as shown in Fig.1(b): the front sensor is oriented as the major axis of the car, while the side ones have an orientation of respectively $\pm 45°$ relative to the major axis of the car. The range of the front sensor is up to 100mt and that of the side sensors is up to 50mt. Notice that range and decoding resolution are key parameters since the algorithm complexity is proportional to the number of possible states (i.e. sensorial stimuli) decoded from the sensors input.

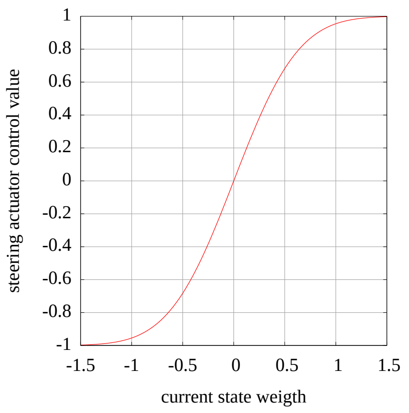
Two non-linear staircase quantization function for the sensors signals were used, each of them with 5 thresholds: the former with ”large” thresholds for the front sensor, the latter with an higher resolution for the side sensors. The rationale is the most the car is in the middle of the track and far away from a turn, the least the state should change; if the car reaches a curve or starts getting close to the track boundary, the state should quickly change to allow to the control system to *quickly* carry out the proper control action.

2.3 Neural Network Output Function

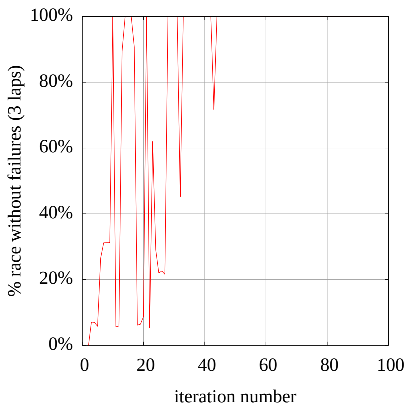
According to the originally proposed model, ASE used the following non linear output function to generate the control action.

$$y(t) = f\left[\sum_{i=1}^{n} w_i(t) + noise(t)\right] \qquad (1)$$

where $f(x) = sign(x)$. The aforesaid output function is not suitable to the considered problem because it produces fast commands for the robot steering actuator which correspond to its limit points. This results in a very nervous "driving style", and the car motion assumes a continuous oscillatory trend even on straight sections of the track. Furthermore, because of the acceleration directly handled by the said "black-box", this kind of steering style causes frequent car spinning, especially outgoing from fast curves. For the vehicle guidance problem this output function must be replaced with some kind of "smoother" one. The function chosen here is a sigmoidal function, obtained vertically shifting a zero-mean gaussian cdf with σ^2_{out} variance, as shown in Fig.2(a). This choice has



(a) Output Function.



(b) Learning Curve.

Fig. 2. (a) Output function with $\sigma^2_{out} = 0.5$. (b) Learning curve obtained after nearly 100 iterations, each of three laps.

been tested experimentally and it completely eliminated the spinning situations and reduced a lot the vehicle oscillatory trend due to the control noise. Definitely, a set of parameters for the ASE-ACE algorithm which work well in this case are: $\alpha = 10$, $\beta = 0.5$, $\gamma = 0.9$, $\delta = 0.95$, $\lambda = 0.8$, $\sigma_{out} = 0.5$, $\sigma_{noise} = 0.001$; the decoder thresholds are: $4, 6, 8, 12, 16$mt and $4, 6, 7, 10, 12$mt. The aforesaid parameters are defined as follows:

- α and β are positive constants which determine the rate of change of the weights associated with the input pathways for the ASE and the ACE respectively;
- $0 \leq \delta < 1$ and $0 \leq \lambda < 1$ are constants which take part in the computation of the eligibility traces decay rate for the ASE and the ACE respectively;
- $0 < \gamma \leq 1$ is a constant which provides for eventual predictions decay in absence of external reinforcement during the internal reinforcement calculation.

Such parameters have slightly different values if compared with the originally ones used by the authors for the pole-balancing experiment [2], except for the so-said *learning rate* (α). In fact, in this steering control application the number of useful time steps to carry out a trajectory correction should be considerably more than the pole-balancing case, and therefore it's reasonable to choose a lower learning rate.

3 Results and Conclusions

The evaluation of the neural network has been carried out in two phases. Firstly a learning phase in which a totally untrained vehicle starts driving on a set of "easy" learning tracks. Then, a validation phase where the trained vehicle tries driving on a set of unknown tracks. The tracks chosen for the first phase have many different track angles and widths. Cyclically running all the learning tracks, the neural network learns quite well how to control the steering actuator of the vehicle, with respect to the different possible curves properties of a generic track. Each iteration refers to the attempt to perform three complete track's laps. Performing more than one lap leads to speed up the learning curve because the car always begins a race with a standing start from the middle of the track. If the first lap is successfully completed, the car can pass over the start line with very different speed and position. Therefore, after having completed the first lap, the car often goes out of the way at the beginning of the second one. The learning curve obtained during this phase is shown in Fig.2(b), and proofs the learning phase was really fast through this approach, nearly 80 iterations. As shown by the simulation results for the ASE-ACE application to the cart-pole balancing problem [2], after nearly 100 iterations the neural network was able to keep the pole balanced for over 500.000 time-steps (i.e. approximately 2.8 hours of simulated real time), which indicated the learning process completion. Therefore, it is possible to state that the obtained performances are substantially consistent with the authors' experiment ones.

An interesting point is that the learnt driving style favours a central track position on in the width sense. This behaviour is due to the side proximity sensors orientation with respect to the major axis of the car and their range: if the range is smaller, the neural network's state should remain the same within a certain distance from the middle track line. If the side sensors' orientation relative angle is smaller, the vehicle tends to carry out some kind of more sporting trajectories, but with some difficulties on sharp curves; on the contrary, if said angle is greater the vehicle still tends to follow the middle track line, but carrying out forceful steering where it is useless too. A better driving style could be obtained by using seven proximity sensors, where the side ones are oriented with a 30° step from the central one [4, 5]. In this case, however, the learning curve convergence speed should be lower because of the much greater number of states of the neural network. After the learning phase, the evaluations on the validation tracks were really good: the vehicle never went out of the way and you could see a slight improvement of lap times too, as long as it converges to a substantially constant time. This improvement of the lap times depends on the track chosen, and although for some tracks there have not been significant improvements, the greatest ones obtained during the experiments have been of nearly 1 second.

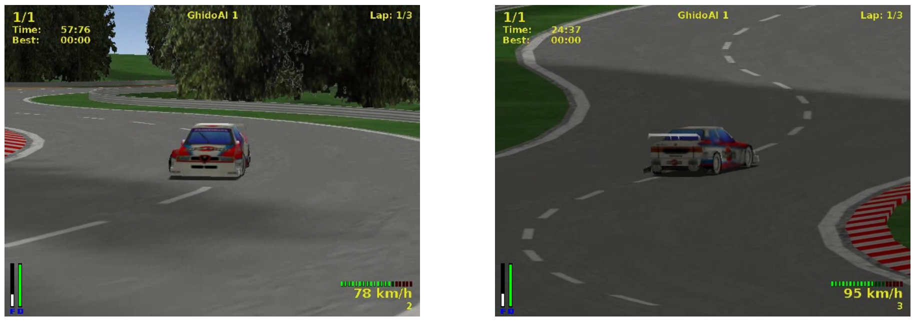


Fig. 3. The robot learnt to drive autonomously on an unknown track with many curves.

References

1. Michie, D., Chambers, RA.: BOXES: An experiment in adaptive control Machine intelligence Vol. 2-2, 137–152 (1968)
2. Barto, A.G., Sutton, R.S., Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, & Cybernetics, (1983)
3. The open racing car simulator website, `http://torcs.sourceforge.net/`
4. Loiacono, D., Togelius, J., Lanzi, P.L., Kinnaird-Heether, L., Lucas, S.M., Simmerson, M., Perez, D., Reynolds, R.G., Saez, Y.: The wcci 2008 simulated car racing competition. Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On, 119–126 (2008)
5. Cardamone, L.: On-line and Off-line Learning of Driving Tasks for The Open Racing Car Simulator (TORCS) Using Neuroevolution. Politecnico di Milano, 2008