Computational Intelligence assignment report

Sardana Nazarova[1]

University of Amsterdam, 1012 WX Amsterdam, The Netherlands,
sardanazar@mail.ru

Abstract. This paper is a report of an assignment of a course Computational Intelligence. The main goal of the assignment is to apply computational intelligence techniques in a practical setting - building a controller for a race car in The Open Racing Car Simulator (TORCS) using artificial neural network and evolving that network with evolutionary algorithm techniques.

Keywords: computational intelligence, car controller, artificial neural network

1 A basic controller

A basic controller was constructed using artificial neural network. Firstly, the data to train neural network was collected and then suitable neural network was selected and trained to be able to keep the car on the track for an arbitrary circuit.

1.1 Collecting training set

To train the neural network off-line supervised learning approach was selected. There are two options for acquiring data: driving the car on the track yourself and driving the car by pre-defined controller. Since driving without joystick for racing games cause a lot of errors in data, the second option was chosen. As controller I select the default controller SimpleDriver in Java patch champ2010client-java, because it uses the same sensors and action models as provided in competition driver template, in contradistinction to controllers in TORCS itself, written in C++.

The competition car controller uses automated clutch, automated gearbox, automated ABS and automated recovering, that means we need to control only acceleration, steering and brake. The sensors used for calculating acceleration, steering and brake in SimpleDriver are following:

1. speed
2. angle to track axis
3. track position
4. track edge sensor parallel to car axis
5. track edge sensor +5 degree w.r.t. car axis

6. track edge sensor -5 degree w.r.t. car axis

The default driver was modified to record listed above sensors and required actuators. The modified controller was launched on a track Alpine 2 at two laps. The choice is substantiated by the fact that the track has a lot of both left and right turns of various shapes, hence there is no need for collecting data from other tracks.

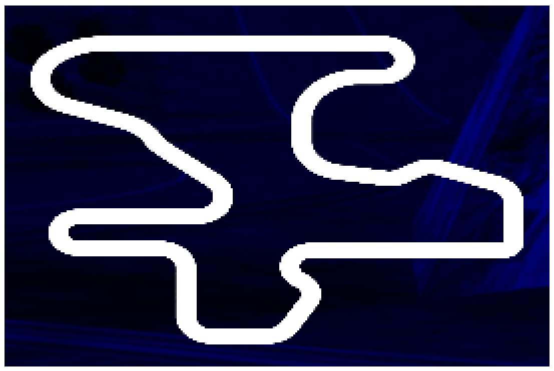


Fig. 1. Aline 2 (Source: TORCS)

As we can see from the Figure 1, the straight parts of road are long, which will cause the neural network to be over trained for driving directly and as a result being unable to turn. To avoid that, data should be cleaned use no more than 1000 rows of training set with steering < 0.01.

1.2 Neural network

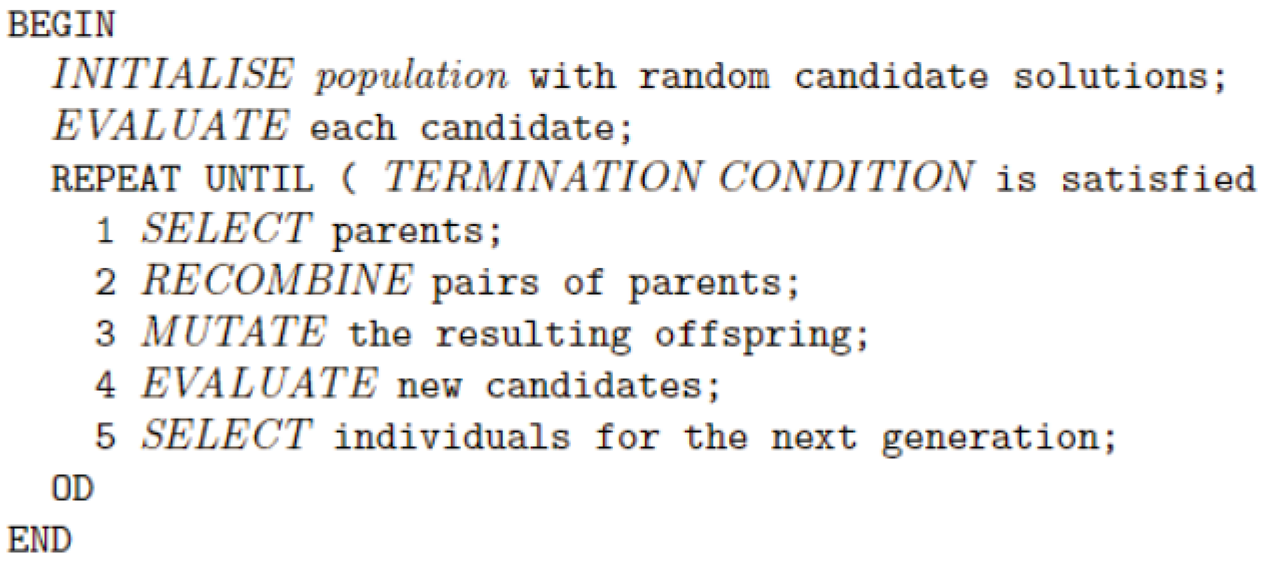
From previous sub-chapter we know that there are 6 inputs and 3 outputs. Since there are few input and output neurons, Multilayer perceptron with one hidden layer of 6 neurons and with transfer function *tanh* is selected as the neural network. Learning algorithm is Back-propagation. Expected performance is controller to drive the same as default controller SimpleDriver, which has maximum speed of $150km/h$ and rides in the middle of the road. However, the trained controller was unable to turn on a high speeds, that's why the maximum speed is limited to $60km/h$. On the Table 1 there are results of the default driver and the trained driver on laps without sharp turns, the trained driver is got stucked on sharp turns and can't continue ride. As result, the trained driver is able to complete only a half of available in TORCS road tracks. As we may see, the trained driver completes laps slower than the default driver, because the speed limit was added.

Table 1. Comparison of the drivers

Track	Default driver	Trained driver
Alpine 2	3:48:84	4:29:03
E-Track 4	3:57:87	7:01:33
E-Road	3:00:15	4:47:77
CG track 3	2:41:41	5:11:32

2 Evolving trained neural network

To make the trained neural network drives better, the general scheme of Evolution Algorithm is used (see the Figure 2).



```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Fig. 2. EA scheme in pseudo-code (Source: CI course slides)

The size of population is 5. To initialize each candidate the initial neural network is mutated. Then all 5 drivers are launched to drive on selected track and to evaluate candidates the fitness function is defined as a lap time. As parents two fastest drivers are chosen and the slowest driver is replaced by the offspring. Termination condition is maximum generations are 5. The procedure runs on different tracks to avoid over fitting for just one track. Unfortunately, no significant improvement is observed after evolution algorithm applied. As mutation rate is chosen small, probably, recombination is implemented wrong.

References

1. Kruse, Rudolf., Borgelt, C., Held, P., Klawonn, F., Moewes, C., Steinbrecher, M.: Computational Intelligence: A Methodological Introduction. Springer London, (2013)