Car race controller based on evolutionnary trained radial basis function neural networks

**March, 05, 2015**

# Aims

We will presents a controller for the Simulated Car Racing.
We will start by training a neural based controller to drive alone in the track ( time trial).
The second step is to learn to compete with other ddrivers.

# Description of the available sensors

| Name | Range(unit) | Description |
|---|---|---|
| angle | $[-\pi, +\pi]$ (rad) | Angle between the car direction and the direction of the track axis. |
| curLapTime | $[0, +\infty]$ (s) | Time elapsed during current lap. |
| damage | $[0, +\infty]$ (point) | Current damage of the car (the higher is the value the higher is the damage). |
| distFromStart | $[0, +\infty]$ (m) | Distance of the car from the start line along the track line. |
| distRaced | $[0, +\infty]$ (m) | Distance covered by the car from the beginning of the race |
| fuel | $[0, +\infty]$ (l) | Current fuel level. |
| gear | -1,0,1,0..6 g | Current gear: -1 is reverse, 0 is neutral and the gear from 1 to 6. |
| lastLapTime | $[0, +\infty]$ (s) | Time to complete the last lap |
| opponents | [0,200] (m) | Vector of 36 opponent sensors: each sensor covers a span of 10 degrees within a range of 200 meters and returns the distance of the closest opponent in the covered area. |
| racePos | 1,2,..N | Position in the race with respect to other cars. |
| rpm | $[0, +\infty]$ (rpm) | Number of rotation per minute of the car engine. |
| speedX | $[-\infty, +\infty]$ (km/h) | Speed of the car along the longitudinal axis of the car. |
| speedY | $[-\infty, +\infty]$ (km/h) | Speed of the car along the transverse axis of the car. |
| speedZ | $[-\infty, +\infty]$ (km/h) | Speed of the car along the Z axis of the car. |
| track | [0,200] (m) | Vector of 19 range finder sensors: each sensors returns the distance between the track edge and the car within a range of 200 meters. |
| trackPos | $[-\infty, +\infty]$ | Distance between the car and the track axis. The value is normalized w.r.t to the track width: it is 0 when car is on the axis, -1 when the car is on the right edge of the track and +1 when it is on the left edge of the car. Values greater than 1 or smaller than -1 mean that the car is outside of the track. |

# Description of the available effectors

| Name | Range | Description |
|---|---|---|
| accel | [0,1] | Virtual gas pedal (0 means no gas, 1 full gas). |
| brake | [0,1] | Virtual brake pedal (0 means no brake, 1 full brake). |
| clutch | [0,1] | Virtual clutch pedal (0 means no clutch, 1 full clutch). |
| gear | -1,0,1,0..6 | Gear value. |
| steering | [-1,1] | Steering value: -1 and +1 means respectively full right and left, that corresponds to an angle of 0.366519 rad. |
| focus | [-90,90] | Focus direction in degrees. |
| meta | 0,1 | This is meta-control command: 0 do nothing, 1 ask competition server to restart the race. |

# Designing TORCS AI based controller

- High level control
  use information from the sensors to predict the trajectory in the next segment of the track ( with scripted driving policy to compute next gear , accel cluch and steering). This is could be done either by collecting data from the warm lap with low speed or adding a new sensor to get ahead information about the track as in **L. Cardamone, D. Loiacono, and P. Lanzi, ?Learning drivers for TORCS through imitation using supervised methods,? in Proceedings of the 5th international conference on Computational Intelligence and Games. IEEE Press, 2009, pp. 148?155.**
- Low level control : predict actions of the car directly(ifficult in case of noise).

# Designing indirect human-like controller

## RBF neural network Inputs:

- angle
- gear
- distRaced
- lastLapTime
- speedX
- speedY
- focus

## RBF neural network Outputs

- targetspeed
- targetposition

# Designing indirect human-like controller

- Train the RBFNN with GA.
- Hidden neurons number: Trial /error.
- Driving policy is used to compute the desired steer, gear acel and cluch to move the car from the currentpos and currspeed to target position and speed.
- $Fitness = \sum (targetpos - currpos)^2$
- Another way is to collect data from some tracks in the TORCS and train the RBFNN with BP or another training algorithm.

# Designing direct human-like controller

## RBF neural network Inputs:

- angle
- focus
- damage
- distRaced
- fuel
- gear
- lastLapTime
- rpm
- speedX
- speedY
- track
- wheelSpinVel

## RBF neural network Outputs

- accel
- brake
- clutch
- gear
- steering
- focus

# Designing direct human-like controller

- Hidden neurons number: Trial /error.
- Train the RBFNN by a genetic algorithm.
- Chromosome=[angle, currlapt, distFromStart,distRaced, fuel, gear, lastLapTime, rpm ]

1. Fitness1:

$$f_g = a*damage+b*fuel+c*lastLapTime+d*currlapt+e*angle \quad (1)$$

2. Fitness2:

$$f_g = (1-e^{\beta})(currlapt+angle)+e^{\beta}(damage+fuel) \quad (2)$$

3. Fitness3: a fitness for each 20ms of the race ( well position the car in the track) and a global function ( sum of ladtlaps time or points accorded to the controller)

$$f_g = (1-e^{\beta})(currlapt+angle)+e^{\beta}(damage+fuel) \quad (3)$$